Deutsch

FUJITSU Software

BeanConnect V3.0B

Benutzerhandbuch

Ausgabe August 2015

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Der Adapter BeanConnect™ gehört zur Produkt-Suite openSEAS (open Suite for Enterprise Application Server). BeanConnect realisiert die Verknüpfung zwischen klassischen Transaktionsmonitoren und modernen Application Servern und ermöglicht damit die effiziente Integration von Legacy-Anwendungen in moderne Java-Anwendungen.

Dieses Handbuch beschreibt das Produkt BeanConnect. BeanConnect bietet einen JCA 1.6-konformen Adapter, der openUTM- und CICS™-Anwendungen mit Anwendungen auf Basis von Java™ EE (Java Platform, Enterprise Edition), z.B. dem Oracle™ WebLogic Server, verbindet. In den folgenden Abschnitten bezieht sich der Begriff EIS (Enterprise Information System) auf die openUTM- oder CICS-Anwendung. Mit openUTM-Anwendung (kurz UTM-Anwendung) ist sowohl eine stand-alone UTM-Anwendung als auch eine UTM-Cluster-Anwendung gemeint.

BeanConnect unterstützt verschiedene Kommunikationsrichtungen und -modelle, so ermöglicht es Outbound- und Inbound-Kommunikation, transaktionale und nicht-transaktionale Kommunikation sowie asynchrone und dialogbasierte Kommunikation.

BeanConnect-Komponenten

BeanConnect besteht aus folgenden Komponenten:

- Der **BeanConnect Resource Adapter** implementiert die JCA 1.6-Schnittstellen. Als konformer JCA-Adapter wird er in den Application Server deployt und läuft innerhalb des Application Servers.
- Der **BeanConnect Proxy** stellt die Funktionalität einer Protokollmaschine sowie Funktionen zur Transaktionssteuerung und Transaktionssicherung bereit. Er kann als intelligentes Gateway betrachtet werden. Er kommuniziert mit dem Resource Adapter, der im Application Server läuft, auf der einen Seite und mit dem EIS auf der anderen.
- Die **BeanConnect Management Console** (MC) ist eine Java-basierte grafische Benutzeroberfläche für die Konfiguration und Administration von BeanConnect Proxys. Mit Hilfe der Management Console können Sie mehrere Proxys auf demselben System oder auf anderen Systemen verwalten.
- Das **Management Console Command Line Interface** (im Folgenden kurz mit MC-CLI bezeichnet) ist ein Paket von Jython-Funktionen, mit dem Sie Funktionen der BeanConnect Management Console aus einem Jython-Skript heraus ausführen können.

- Die **BeanConnect Tools** sind Tools, die man bei bestimmten Einsatzszenarien von BeanConnect benötigt. Dazu gehören Cobol2Java und der MC-CmdHandler.

1.1 Zielgruppe

Dieses Handbuch richtet sich an folgende Zielgruppen:

- BeanConnect Administratoren
- Administratoren von Application Servern wie z.B. Oracle WebLogic Server
- Deployer
- EJB-Entwickler (Enterprise Java Beans)
- openUTM- und CICS-Administratoren

Es wird davon ausgegangen, dass Sie mit Java und der JCA-Spezifikation V1.6 vertraut sind.

1.2 Aufbau der Dokumentation zu BeanConnect

Die BeanConnect Dokumentation besteht aus folgenden Komponenten:

- Das Handbuch **BeanConnect** (dieses Dokument)
- Das **Hilfesystem** für die Management Console. Dieses bietet online schnelle und kontextbezogene Unterstützung bei der Konfiguration und Administration von BeanConnect Proxys.
- Die **JavaDoc** zu BeanConnect, die mit der JAR-Datei des Resource Adapters `BC30B00_RA.jar` ausgeliefert wird. Diese Dokumentation steht nach der Installation des Resource Adapters zur Verfügung.
- Die **JavaDoc** zum Command Line Interface der BeanConnect Management Console. Diese Dokumentation steht nach der Installation der BeanConnect Management Console zur Verfügung.

Einzelheiten zum Oracle WebLogic Server und zu weiteren Softwareprodukten, die in diesem Handbuch erwähnt werden, finden Sie in den zugehörigen Dokumenten.

1.3 Struktur dieses Handbuchs

Kapitel „JCA-Adapter-Integration – Überblick“ auf Seite 27 vermittelt einen Überblick über das Oracle-Konzept für die Adapterintegration. Es beschreibt die Funktionen von BeanConnect und zeigt, wie dieses Produkt in die Umgebung des Oracle WebLogic Server eingebunden wird.

Kapitel „BeanConnect installieren“ auf Seite 49 beschreibt Installation, Update-Installation und Deinstallation von BeanConnect.

Kapitel „Konfiguration im Application Server“ auf Seite 91 bietet Informationen zum Konfigurieren der Outbound- und Inbound-Kommunikation über das OSI TP / LU6.2 Protokoll und der Outbound-Kommunikation über das UPIC-Protokoll. Es beschreibt das Deployment des Resource Adapters, einer OLTP Message-Driven Bean und von Enterprise Java Beans.

Kapitel „BeanConnect Management Console - Überblick“ auf Seite 157 vermittelt einen Überblick über die BeanConnect Management Console. Die Management Console wird zum Konfigurieren und Administrieren eines oder mehrerer BeanConnect Proxys verwendet.

Kapitel „BeanConnect konfigurieren“ auf Seite 179 beschreibt die erforderlichen Schritte zum Konfigurieren eines BeanConnect Proxys und seiner Komponenten mit der Management Console.

Kapitel 7, „Konfiguration im EIS Partner anpassen“ beschreibt die erforderlichen Konfigurationsaktivitäten im EIS (Enterprise Information System), um eine Kommunikation zwischen dem Application Server und dem EIS aufzubauen.

Kapitel 8, „BeanConnect administrieren“ beschreibt die Administrationsaufgaben beim Betrieb der BeanConnect Proxys.

Kapitel 9, „Command Line Interface der BeanConnect Management Console (MC-CLI)“ beschreibt das Command Line Interface der Management Console, mit dem Sie die Funktionen der Management Console über Jython Skripts ausführen können.

Kapitel 10, „Interfaces und Programmierung“ beschreibt die Programmierung der Kommunikation zwischen dem Application Server und dem EIS.

Kapitel 11, „Zeichensatz-Konvertierung und Sprachunterstützung“ beschreibt die Code-Konvertierung zwischen der spezifischen Codierung des EIS Partners und Unicode. Zusätzlich informiert dieses Kapitel über die Sprachenunterstützung bei der Ausgabe von Meldungen.

Kapitel 12, „Hoch-Verfügbarkeit und Skalierbarkeit“ beschreibt die Konfigurationsänderungen, die für den Hochlast-Betrieb in BeanConnect notwendig werden können.

Kapitel 13, „Logging, Diagnose und Fehlerbehebung“ beschreibt die Vielzahl verschiedener Diagnose- und Trace-Funktionen.

Kapitel 14, „Cobol2Java“ beschreibt die Integration von COBOL-Anwendungen in BS2000-Systemen und BeanConnect Clients. Dabei wird auch das Tool Cobol2Java beschrieben, mit dem die COBOL-Datentypen auf Java-Klassen abgebildet und konvertiert werden.

1.4 Änderungen gegenüber der Vorgängerversion

Im Folgenden werden die wichtigsten Änderungen von BeanConnect V3.0A und V3.0B gegenüber BeanConnect V2.1A aufgelistet. Eine vollständige Beschreibung - inbesondere zur Software-Konfiguration - finden Sie in der Freigabemitteilung.

Neue Funktionen in V3.0A

- Der BeanConnect Resource Adapter unterstützt die Spezifikation JCA 1.6 und damit die neuen Contracts Generic Work Context und Security Work Context.
- BeanConnect V3.0 kann mit Application Servern zusammenarbeiten, die die Spezifikation JCA 1.5 oder JCA 1.6 unterstützen.

  Für die Version 3.0 von BeanConect wurde der Oracle WebLogic Server als der Standard Application Server verwendet. Alle Beispiele im Handbuch beziehen sich auf diesen Application Server.
- Command Line Interface der Management Console zur Konfiguration und Administration der BeanConnect Komponenten per Skript.

  BeanConnect bietet ein Command Line Interface auf Basis der Skriptsprache Jython. Damit lassen sich folgende Administrationsfunktionen aus einem Jython-Skript starten:

  – Alle Proxy-Funktionen und alle Proxy Cluster-Funktionen, d.h. die Konfiguration und Administration von Resource Adaptern, EIS Partnern, Inbound Services, Outbound Services, Inbound Communication Endpoints und Outbound Communication Endpoints.

    Sie können die hier aufgeführten Objekte erzeugen, die Eigenschaften der Objekte lesen und ändern und die Objekte aus der Konfiguration entfernen. Außerdem können Sie Administrationsfunktionen ausführen wie z.B. das Prüfen der Verfügbarkeit oder Starten und Beenden eines BeanConnect Proxys.

  – Die Verarbeitung von Todo-TopicsSie können die Eigenschaften der Todo-Topics lesen und Todo-Topics löschen.
- Funktionserweiterungen in der Management Console

  – BeanConnect unterstützt UTM-Cluster-Anwendungen bei Outbound-Kommunikation, indem die Knoten-Anwendungen einzeln konfiguriert und angesprochen werden können.

  – Im Eigenschaftsdialog eines Outbound Communication Endpoint können Sie den EIS Partner jetzt direkt ändern.

Neue Funktionen in V3.0B

- Unterstützung langer Rechnernamen

  BeanConnect unterstützt das Hostname Mapping für Rechnernamen, die länger als 8 Zeichen sind. Dies gilt sowohl für einen Rechernamen eines Proxys als auch für Rechnernamen von EIS Partnern.

  Das Mapping für einen Proxy wird bei der Installation des Proxys festgelegt, das Mapping für den EIS Partner beim Eintragen eines neuen EIS Partners.

- MC-CLI Recording

  Alle Aktionen der Management Console, zu denen es Funktionen in der MC-CLI gibt, werden in internen Puffern der Management Console mitgeschnitten. Diese Mitschnitte lassen sich in einem internen Editor ansehen oder auf Datei ausgeben. Die Mitschnitte können zur Protokollierung dienen oder als Vorlage für ein MC-CLI-Skript verwendet werden.

- Erweiterung des MC-CLI
  - Kommunikation mit CICS-Partnern:
    Das MC-CLI bietet neue Module und Funktionen, mit denen die Kommunikation mit CICS-Partnern durchgeführt werden kann.
  - Die Ausgabe von asynchronen Meldungen kann beim `Init()` eingestellt werden.
  - Die mitgelieferten Jython-Skript-Beispiele wurden erweitert und lassen sich dadurch einfacher anwenden.

Sonstige Änderungen

- Administrationsaufträge der Management Console werden vom BeanConnect Proxy bevorzugt gegenüber Outbound- oder Inbound-Aufträgen behandelt.

1.5 Hinweise zu Fremdprodukten und Fremdliteratur

In diesem Handbuch wird u.a. Bezug genommen auf folgende Fremdprodukte, die zusammen mit BeanConnect eingesetzt werden können:

- Oracle WebLogic Server
- IBM™ Communications Server für Windows-Systeme und IBM Communications Server für Linux-Systeme
- SNAP-IX™ von Metaswitch Networks Ltd. für Solaris-Systeme
- CICS (Customer Information Control System) von IBM

Dieses Handbuch nimmt an einigen Stellen konkreten Bezug auf Parameter, die im Oracle WebLogic Server angegeben werden müssen. Für die vorliegende Beschreibung wird Oracle WebLogic Server 12 vorausgesetzt, d.h. die Angaben gelten nur für genau diese Version. Wenn eine andere Version eingesetzt wird, sind eventuell andere Angaben nötig. Details finden Sie in der einschlägigen Dokumentation von Oracle.

1.6 Darstellungsmittel

In diesem Handbuch werden folgende Darstellungsmittel verwendet:

Darstellungsmittel	Bedeutung
. . .	Vertikale Auslassungspunkte in einem Beispiel weisen darauf hin, dass Informationen, die sich nicht direkt auf das Beispiel beziehen, weggelassen wurden.
. . .	Horizontale Auslassungspunkte in Anweisungen oder Befehlen weisen darauf hin, dass Teile der Anweisung oder des Befehls, die sich nicht direkt auf das Beispiel beziehen, weggelassen wurden.
Halbfette Schrift	Halbfette Schrift im Fließtext kennzeichnet alle Zitate der Benutzeroberfläche (Menüpunkte, Feldnamen, Optionen, etc.)
`Schreibmaschinen-schrift`	Schrift mit festen Zeichenabstand kennzeichnet Systemeingaben, Systemausgaben und Dateinamen.
< >	Spitze Klammern enthalten Namen, die vom Benutzer eingegeben werden müssen. In den Beispielen mit XML kennzeichnen spitze Klammern XML-Anweisungen.
[ ]	Eckige Klammern enthalten optionale Angaben, von denen Sie eine auswählen können, aber nicht müssen.
{ }	Geschwungene Klammern enthalten alternative Angaben, von denen Sie genau eine auswählen müssen.
|	Eine vertikale Linie trennt die alternativen oder optionalen Angaben.

Dieses Symbol kennzeichnet wichtige Hinweise und weitere Informationen.

Dieses Symbol kennzeichnet eine Warnung.

2 JCA-Adapter-Integration – Überblick

Dieses Dokument beschreibt das Produkt BeanConnect. BeanConnect stellt einen JCA 1.6-konformen Adapter zur Verfügung, der openUTM-Anwendungen (Universeller Transaktions-Monitor) und CICS-Anwendungen (Customer Information Control System) mit Java EE basierten Anwendungen verbindet, wie z.B. dem Oracle WebLogic Server.

Dieses Kapitel bietet einen Überblick über die JCA-Adapter-Integration in eine Java EE Application Server-Umgebung. Die Java EE Connector Architecture (JCA) ist eine Software-Architektur zur Integration von heterogenen Anwendungen in die Java EE Plattform.

Das Kapitel beschreibt die Funktionen von BeanConnect und zeigt, wie dieses Produkt in die Umgebung des Oracle WebLogic Servers eingebunden wird. Dieses Kapitel enthält Informationen zu folgenden Themen:

- Die JCA-Adapter-Versionen
- BeanConnect-Architektur
- BeanConnect als JCA-konformer Resource Adapter
- BeanConnect im Cluster-Betrieb

2.1 Die JCA-Adapter-Versionen

Im Java Community Process (JCP) werden Java Specification Requests (JSRs) erarbeitet und in einem formalen, allgemein zugänglichen Prozess verabschiedet. Die JSR-Dokumente werden über eine Nummer identifiziert und beschreiben Spezifikationen und Technologien, die zur Java-Plattform hinzugefügt werden können. Z.B. definiert das Dokument "JSR 322: Java EE Connector Architecture 1.6" Standard-Java-Interfaces für die einfache Integration von Anwendungen zur Kommunikation mit einem EIS. Die Einhaltung dieser Spezifikation wird in diesem Dokument verkürzt mit " JCA 1.6 Konformität" bezeichnet. Vorgängerdokument war das Dokument "JSR 112: J2EE™ Connector Architecture 1.5". Die Konformität mit diesem Dokument wird kurz als "JCA 1.5 Konformität" bezeichnet.

2.1.1 JCA-Adapter-Integration

Ein Resource Adapter ist speziell an das Enterprise Information System (EIS) angepasst, für das er entwickelt wurde. Er stellt die Operationen auf Systemebene bereit, die für den Betrieb und die Kommunikation mit dem EIS benötigt werden. Ein Resource Adapter, der die JCA-Schnittstellen unterstützt, kann mit jedem Application Server verwendet werden, der ebenfalls diese Schnittstellen unterstützt. Der Resource Adapter stellt dem Application Server seine Fähigkeiten über ein JCA-definiertes SPI zur Verfügung. Durch die Verwendung des definierten SPI kann der Application Server die Dienste des Resource Adapters wirksam in seine Operationen integrieren, während er die Anwendungen selbst von der darunterliegenden Implementierung des EIS isoliert. Wichtige Voraussetzungen für eine wirksame und skalierbare Integration für die Kommunikation mit den EIS-Systemen sind Dienste wie z.B.

- Connection Management und Pooling,
- Transaction Management zur Unterstützung globaler, d.h. Application Server und EIS umfassender Transaktionen,
- Logging und Tracing
- sowie ein Sicherheits-Framework, das sowohl container- als auch anwendungsgesteuerte Anmeldung ermöglicht.

2.1.2 JCA 1.6 System Contracts

Es werden alle Contracts der JCA 1.6-Spezifikation unterstützt:

- Connection Management Contract

  Der Connection Management Contract ermöglicht den Anwendungskomponenten, eine Verbindung mit einem EIS aufzubauen und jedes vom Application Server bereitgestellte Connection Pooling zu nutzen.

- Transaction Management Contract

  Der Transaction Management Contract ermöglicht einem Application Server, Transaktionen mit einem Transaktionsmanager zwischen mehreren Resource Managern zu verwalten.

- Security Contract

  Der Security Contract bietet Zugangs- und Zugriffskontrolle sowie sichere Kommunikation zwischen dem Java EE Server und dem EIS.

- Lifecycle Management Contract

  Der Lifecycle Management Contract ermöglicht dem Application Server die Aktivitätsphasen des Resource Adapters, d.h. Funktionen im Zusammenhang mit dem Starten und Beenden, zu verwalten.

- Work Management Contract

  Der Work Management Contract ermöglicht dem Resource Adapter, Aufträge zur Ausführung an einen Application Server zu schicken. Da der Application Server die Arbeit für den Resource Adapter übernimmt, braucht sich der Resource Adapter nicht um das Thread Management zu kümmern. Stattdessen verwaltet der Application Server diesen Aspekt effizient und verwendet bei Bedarf das Thread Pooling. Der Work Management Contract ist zwar nicht erforderlich (der Resource Adapter kann für Arbeitsabläufe auch seinen eigenen Thread verwalten), wird aber dringend empfohlen.

- Message Inflow Contract

  Der Message Inflow Contract ermöglicht einem Resource Adapter die synchrone oder asynchrone Auslieferung von Nachrichten an die Endpunkte im Application Server unabhängig von Nachrichtenstil, Semantik und Infrastruktur.

- Generic Work Context Contract

  Der Generic Work Context Contract ermöglicht einem Resource Adapter, bei der Inbound-Kommunikation Kontextinformation, die der Resource Adapter vom EIS erhalten hat, an eine Work-Instanz weiterzugeben, mit deren Ausführung er den Application Server beauftragt hat.

- Transaction Inflow Contract

  Der Transaction Inflow Contract ermöglicht einem Resource Adapter eine importierte Transaktion an einen Application Server weiterzuleiten. Außerdem ermöglicht er es, den Abschluss einer Transaktion zu übermitteln, sowie die Aufrufe zum Wiederanlauf, die von einem EIS angestoßen wurden.

- Security Work Context

  Der Security Work Context ermöglicht einem Resource Adapter Security-Information, die der Resource Adapter vom EIS erhalten hat, an eine Work-Instanz weiterzugeben, mit deren Ausführung er den Application Server beauftragt hat. Diese Funktionalität wird als Security Inflow bezeichnet.

- Common Client Interface (CCI)

  Das CCI beschreibt einen Standard-API-Client und ist primär an den Anforderungen der Entwicklung von Tools zur Anwendungsentwicklung und EAI-Frameworks (Enterprise Application Integration) ausgerichtet. Verglichen mit dem BeanConnect-spezifischen API bietet das CCI einen eingeschränkten Funktionsumfang.

Detaillierte Informationen finden Sie außerdem in der JCA 1.6-Spezifikation.

2.1.3 SOA-Architektur

Der JCA-Adapter unterstützt das SOA Konzept (Service-oriented architecture).

SOA ist ein Konzept für eine Systemarchitektur, in der Funktionen in Form von wieder verwendbaren, technisch voneinander unabhängigen und lose gekoppelten **Services** implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über Schnittstellen aufgerufen werden, deren Spezifikationen öffentlich und damit vertrauenswürdig sein können. Service-Interaktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt.

Mit Hilfe von BeanConnect können Komponenten von openUTM- und CICS-Anwendungen als Services verfügbar gemacht werden. Umgekehrt können openUTM- und CICS-Anwendungen Services im Application Server ansprechen.

Weitere Informationen finden Sie z.B. unter
http://de.wikipedia.org/wiki/Serviceorientierte_Architektur

2.1.4 JCA-Adapter-Integration in Oracle WebLogic Server

Oracle WebLogic Server 12 ist eine Implementierung der Java EE 6-Spezifikation und bietet standardisierte APIs an, mit denen z.B. auch Verbindungen zu entfernten Enterprise Information Systems (EIS) verwendet werden können. Der Oracle WebLogic Server unterstützt als Bestandteil der Java EE 6 die Java EE Connector Architecture 1.6 gemäß JSR322-Spezifikation. Der BeanConnect JCA 1.6-Resource Adapter kann dadurch einfach im Oracle WebLogic Server 12 deployt werden.

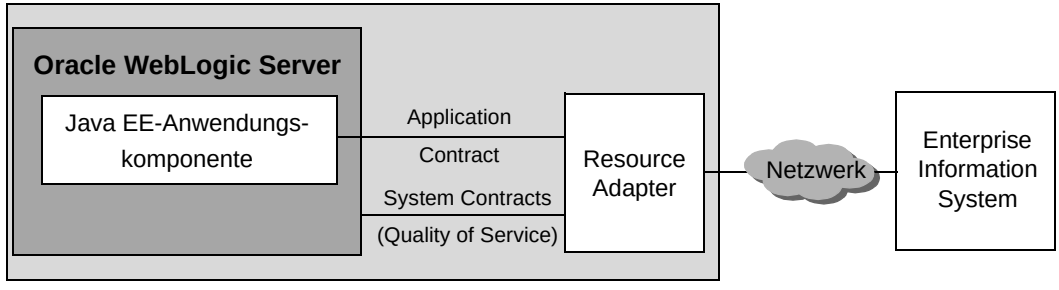


Bild 1: Oracle WebLogic Server – Java EE Connection Architecture

Das Deployment der Anwendungskomponenten und des Resource Adapters wird mit Hilfe von Deployment Descriptoren vorgenommen, mit deren Hilfe der Resource Adapter und die Anwendungskomponenten in den Application Server integriert werden:

- Standard Deployment Descriptor `ejb-jar.xml` für die Anwendungskomponente gemäß Java EE Spezifikation.

- Standard Deployment Descriptor `ra.xml` für den Resource Adapter.
- Application Server spezifische Deployment Descriptoren `weblogic-ra.xml` für den Resource Adapter und `weblogic-ejb-jar.xml` für die Anwendungskomponente.

Eine detaillierte Beschreibung dieser Deployment Descriptoren finden Sie in Abschnitt „Konfigurationsdateien im Application Server“ auf Seite 92.

2.2 BeanConnect-Architektur

BeanConnect ermöglicht die Kommunikation mit Enterprise Information Systems (kurz EIS Partner) vom Typ openUTM und vom Typ CICS. Dieses Kapitel beschreibt, welche Komponenten für diese Partner benötigt werden und welche Funktionen diese Komponenten besitzen.

2.2.1 BeanConnect Komponenten

BeanConnect besteht aus folgenden Komponenten:

- Der **BeanConnect Resource Adapter** implementiert die Interfaces von JCA 1.6. Als konformer JCA-Adapter wird er in den Application Server deployt und läuft innerhalb des Application Servers.

  Um auch mit älteren Application Servern, die JCA 1.6 noch nicht unterstützen, weiter zusammenarbeiten zu können, wird zusätzlich ein JCA 1.5-kompatibler Resource Adapter bereitgestellt, der nur die JCA 1.5-Funktionalität anbietet.

- Der **BeanConnect Proxy** stellt die Funktionalität einer Protokollmaschine sowie Funktionen zur Transaktionssteuerung bereit. Er kann als intelligentes Gateway betrachtet werden. Er kommuniziert auf der einen Seite mit dem Resource Adapter, der im Application Server läuft, und auf der anderen Seite mit dem EIS. Er kann sich auf demselben Rechner wie der Resource Adapter oder auf einem anderen Rechner befinden.

- Die **BeanConnect Management Console** (MC) ist eine Java-basierte grafische Benutzeroberfläche u.a. für die Konfiguration und Administration von Proxys. Die Management Console bietet auch ein Command Line Interface (MC-CLI) an.

  Mit Hilfe der Management Console können mehrere Proxys auf demselben Rechner wie die Management Console oder auf anderen Rechnern verwaltet werden. Für die Konfiguration von Outbound-Verbindungen über das UPIC-Protokoll wird die Management Console nicht benötigt.

- Die **BeanConnect Tools** können unabhängig vom Proxy-Container installiert und eingesetzt werden. Zu diesen Tools gehören der Management Console Command Handler (MC-CmdHandler) und Cobol2Java.

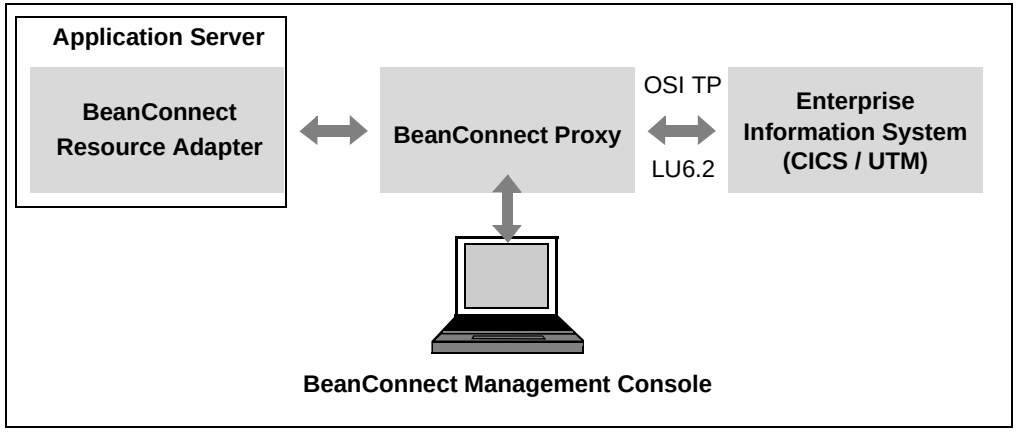


Bild 2: BeanConnect-Komponenten

Der BeanConnect Proxy verwendet je nach Typ des EIS Partners das passende Protokoll, d.h. OSI TP bei openUTM-Partnern oder LU6.2 bei CICS-Partnern.

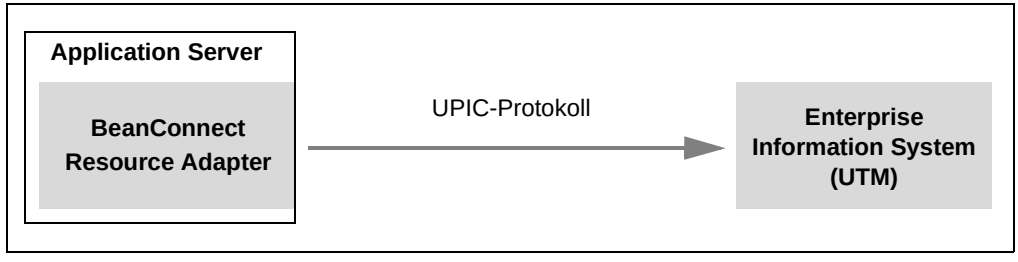


Bild 3: BeanConnect-Komponenten für die Outbound-Kommunikation mit openUTM-Partnern über das UPIC-Protokoll

In komplexeren Szenarien kann die Notwendigkeit bestehen, einen oder mehrere Application Server mit mehreren EIS zu verbinden. Beachten Sie in diesem Fall die folgenden Regeln:

- In einer Application Server Instanz darf BeanConnect nur einmal deployt werden.
- Ein Resource Adapter kommuniziert mit genau einem Proxy.
- Ein Proxy kann entweder mit einem Resource Adapter kommunizieren (Abschnitt „Standard-Betrieb mit einem Resource Adapter und einem Proxy“ auf Seite 39) oder mit mehreren Resource Adaptern (Abschnitt „Multi-Resource Adapter Betrieb“ auf Seite 40 oder Abschnitt „BeanConnect im Cluster-Betrieb“ auf Seite 47).
- Ein Proxy kann mit mehreren EIS Partnern kommunizieren.

- Eine Management Console kann mehrere Proxys verwalten (Konfiguration und Administration).

Außerdem kann BeanConnect in einer Cluster-Konfiguration mit mehreren Resource Adaptern und mehreren Proxys betrieben werden, siehe Abschnitt „BeanConnect im Cluster-Betrieb“ auf Seite 47.

2.2.1.1 BeanConnect Resource Adapter

BeanConnect ist ein Connector Resource Adapter gemäß der JCA 1.6-Spezifikation von Sun Microsystems™. Er ermöglicht standardisierte Verbindungen von openUTM- und CICS-Anwendungen zu Anwendungen, die auf Java EE basierten Application Servern laufen und spielt eine tragende Rolle für die Integration und Konnektivität zwischen einem EIS und einem Application Server. Der Resource Adapter dient als Kontaktstelle zwischen Anwendungskomponenten, Application Servern und Enterprise Information Systems.

Der Resource Adapter liegt als RAR-Archiv vor. Dieses Archiv muss im Application Server deployt werden.

Weitere Einzelheiten über die Konfiguration des Resource Adapters finden Sie in Kapitel „Konfiguration im Application Server“ auf Seite 91.

2.2.1.2 BeanConnect Proxy

Der Proxy verbindet Resource Adapter und EIS.

Er dient als Container für die Konfigurationseigenschaften der Kommunikationspartner und hält alle Informationen zu Services, Communication Endpoints und zu Message Endpoints bereit.

Der Proxy ist für das Weiterleiten von Nachrichten zuständig und sorgt dafür, dass diese zu den entsprechenden Partnern und Services zugeordnet werden. Er stellt Funktionen für die Transaktionssicherung bereit und für die Verifizierung von Zugriffsrechten (Benutzerkennung und Passwort) bei Anfragen an und von openUTM- oder CICS-Anwendungen. Der Proxy dient auch zur Sicherung von asynchronen Nachrichten (Inbound und Outbound) bis zu ihrer Zustellung an den EIS Partner oder die Message Endpoint Anwendung.

Die BeanConnect-Konfiguration besteht im Wesentlichen aus der Konfiguration des Proxys.

Der Proxy enthält den Proxy-Container, der auf dem Transaktionsmonitor openUTM V6.3 basiert.

Den Proxy können Sie mit der Management Console auf drei verschiedene Arten konfigurieren:

- als openUTM-Proxy, der ausschließlich mit EIS Partnern vom Typ openUTM kommuniziert.

  Ein openUTM-Proxy besteht nur aus dem Proxy-Container.

- als CICS-Proxy, der ausschließlich mit EIS Partnern vom Typ CICS kommuniziert.

  Ein CICS-Proxy benötigt zusätzliche Komponenten für die Kommunikation mit CICS-Anwendungen. Für die CICS-Kommunikation ist eine eigene Lizenz erforderlich.

- als kombinierter Proxy

  Ein kombinierter Proxy kommuniziert sowohl mit EIS Partnern vom Typ openUTM als auch mit EIS Partnern vom Typ CICS. Für die CICS-Kommunikation ist eine eigene, getrennte Lizenz erforderlich.

Komponenten eines CICS-Proxys

Bei der Kommunikation mit CICS-Anwendungen besteht der Proxy zusätzlich aus folgenden internen Komponenten:

- openUTM-LU62 Gateway, das den LU6.2-Protokoll-Stack für die Verbindung zu EIS Partnern implementiert, die das SNA-Protokoll LU6.2 unterstützen. Es werden sowohl transaktionale als auch nicht-transaktionale Verbindungen mit CICS-Anwendungen unterstützt.

- Communication Service:
  Dies ist ein Produkt eines Drittanbieters, das den SNA-Stack implementiert.

  Dabei handelt es sich um SNAP-IX von Metaswitch Networks Ltd für Solaris-Systeme oder um den Communications Server von IBM für Linux- und Windows-Systeme. Diese Produkte sind nicht im Lieferumfang von BeanConnect enthalten.

Die folgende Abbildung zeigt die Komponenten des Proxys für die Kommunikation mit CICS-Anwendungen:

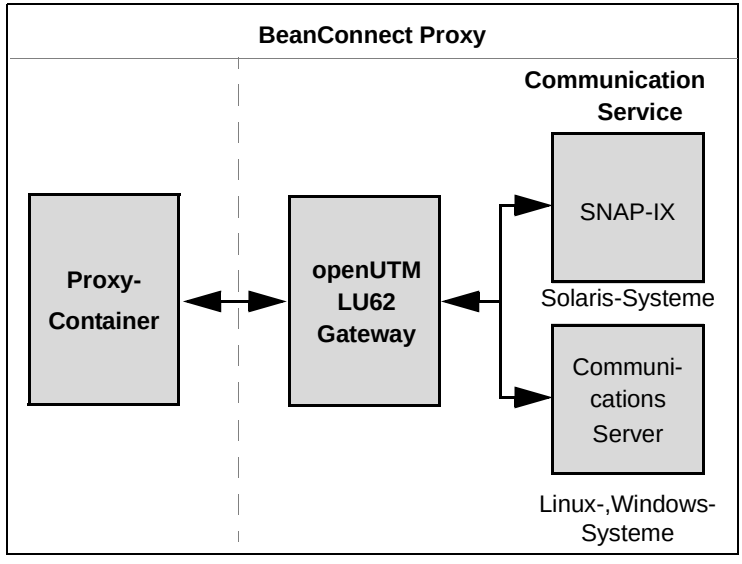


Bild 4: BeanConnect Proxy-Komponenten eines CICS-Proxys

Der Proxy und alle Proxy-Komponenten werden mit der Management Console verwaltet und administriert. Die Proxy-Komponenten openUTM-LU62 Gateway und Communication Service können wahlweise auf demselben Rechner wie der Proxy-Container oder gemeinsam auf einem anderen Rechner ablaufen.

Wenn eine zu administrierende Komponente (Proxy, Gateway, Communication Service) auf einem anderen Rechner läuft als die Management Console, dann muss auf dem anderen Rechner ein MCCmdHandler installiert sein, siehe auch Abschnitt „BeanConnect Tools“ auf Seite 38.

Es ist zu beachten, dass der Communication Service unter einer Benutzerkennung installiert sein muss, die SNA-Berechtigung hat.

Der BeanConnect Proxy wird für die Outbound-Kommunikation über das OSI TP und das LU6.2 Protokoll sowie für die Inbound-Kommunikation verwendet.

Bei Outbound-Kommunikation mit EIS Partnern von Typ openUTM über das UPIC-Protokoll wird der Proxy nicht verwendet.

2.2.1.3 BeanConnect Management Console

Die Management Console unterstützt die Konfiguration und Administration des Proxys und der Proxy-Komponenten. Sie stellt eine Java-basierte Bedienoberfläche bereit, mit der sich die notwendigen Konfigurationsschritte einfach ausführen lassen.

Die Management Console kann mit mehr als einem lokalen oder entfernten Proxy arbeiten.

- **Lokal** bedeutet, dass sich der Proxy auf demselben Rechner befindet wie die Management Console, und dass der Proxy unter der gleichen Benutzerkennung installiert ist, oder dass die Zugriffsrechte auf die Benutzerkennung, unter der der Proxy installiert ist, entsprechend gesetzt sind.
- **Entfernt** bedeutet, dass der Proxy auf einem anderen Rechner als die Management Console installiert ist oder dass der Proxy zwar auf dem gleichen Rechner, aber unter einer anderen Benutzerkennung installiert ist, auf die die Management Console keinen Zugriff hat.

Der Zugriff auf ein entferntes System wird mit dem Management Console Command Handler (**MC-CmdHandler**) ausgeführt. Der MC-CmdHandler ist ein stand-alone Java-Programm, das der Management Console die Verwaltung von entfernten Proxys ermöglicht.

Der Administrator definiert mit der Management Console alle für die Proxy-Konfiguration notwendigen Daten. Folgende Informationen können festgelegt werden:

- Allgemeine Angaben über den Proxy bzw. den Proxy Cluster
- Verwaltung der Properties von Resource Adaptern und Zugriff auf die Deployment Descriptoren der Resource Adapter.
- Beschreibung der EIS Partner
- Services und Communication Endpoints für Outbound-Kommunikation (siehe Abschnitt „Outbound-Kommunikation“ auf Seite 42)
- Services und Message Endpoints für Inbound-Kommunikation (siehe Abschnitt „Inbound-Kommunikation“ auf Seite 42)
- Verbindung zum JMX-Server für den Zugriff auf MBeans und die von diesen angebotenen Informationen und Funktionen

Zusätzlich bei CICS-Partnern:

- Verbindung zum openUTM-LU62 Gateway
- Verbindung zwischen dem Communication Service und dem EIS Partner

Die Konfigurationsdaten werden von der Management Console gespeichert, so dass sie in späteren Management Console-Sitzungen zur Verfügung stehen und geändert oder vervollständigt werden können. Die Management Console erstellt Konfigurationsdateien, die vom Proxy, den Proxy-Komponenten und dem EIS verwendet werden können.

Außerdem können Sie die Konfigurationsdaten der BeanConnect-Konfiguration mit dem Command Line Interface (MC-CLI) lesen und bearbeiten, siehe Kapitel „Command Line Interface der BeanConnect Management Console (MC-CLI)“ auf Seite 309.

2.2.1.4 BeanConnect Tools

BeanConnect bietet bestimmte Komponenten als Tools an, die unabhängig vom Proxy-Container installiert und eingesetzt werden können. Diese Tools sind unabhängig davon, mit welchem EIS Partner kommuniziert wird.

BeanConnect stellt folgende Tools zur Verfügung:

- MC-CmdHandler (Management Console Command Handler)

  Mit Hilfe von separat installierten MC-CmdHandlern ist es z.B. möglich, ferne Resource Adapter oder für CICS-Partner benötigte Proxy-Komponenten über die Management Console zu administrieren, ohne den Proxy-Container installieren zu müssen.

- Cobol2Java

  Cobol2Java ermöglicht die objektorientierte Abbildung von COBOL-Datenstrukturen auf Java-Klassen.

2.2.2 Standard-Betrieb mit einem Resource Adapter und einem Proxy

Im Standard-Betrieb arbeitet ein Proxy mit genau einem Resource Adapter zusammen, es können jedoch mehrere EIS Partner angesprochen werden.

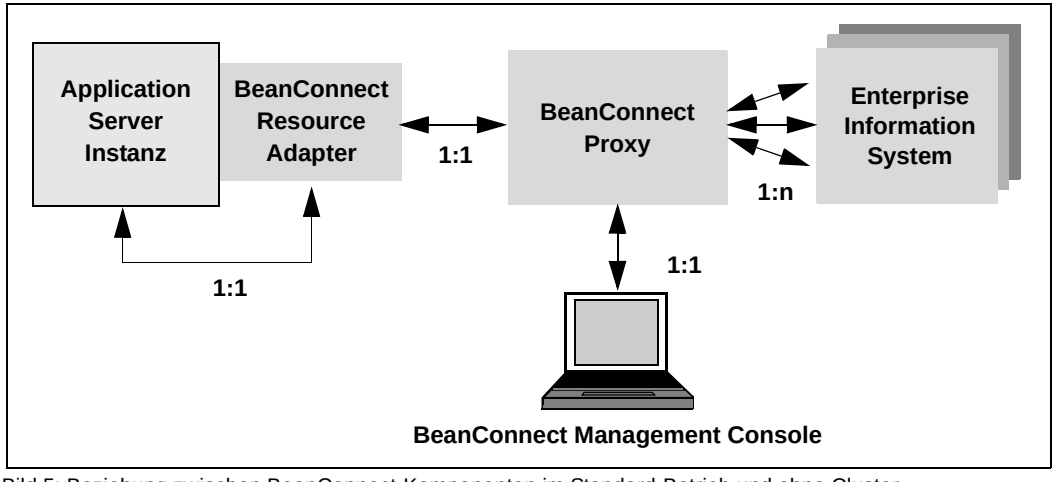


Bild 5: Beziehung zwischen BeanConnect-Komponenten im Standard-Betrieb und ohne Cluster

Der BeanConnect Resource Adapter kann nicht mehrmals in derselben Application Server-Instanz deployt werden.

2.2.3 Multi-Resource Adapter Betrieb

Im Multi-Resource Adapter Betrieb arbeitet ein Proxy mit mehreren Resource Adaptern zusammen, die unabhängig voneinander agieren und auf unterschiedlichen Application Server Instanzen laufen. Der BeanConnect Resource Adapter kann nicht mehrmals auf derselben Application Server Instanz laufen.

Cluster- und Multi-Resource Adapter Betrieb schließen sich gegenseitig aus.

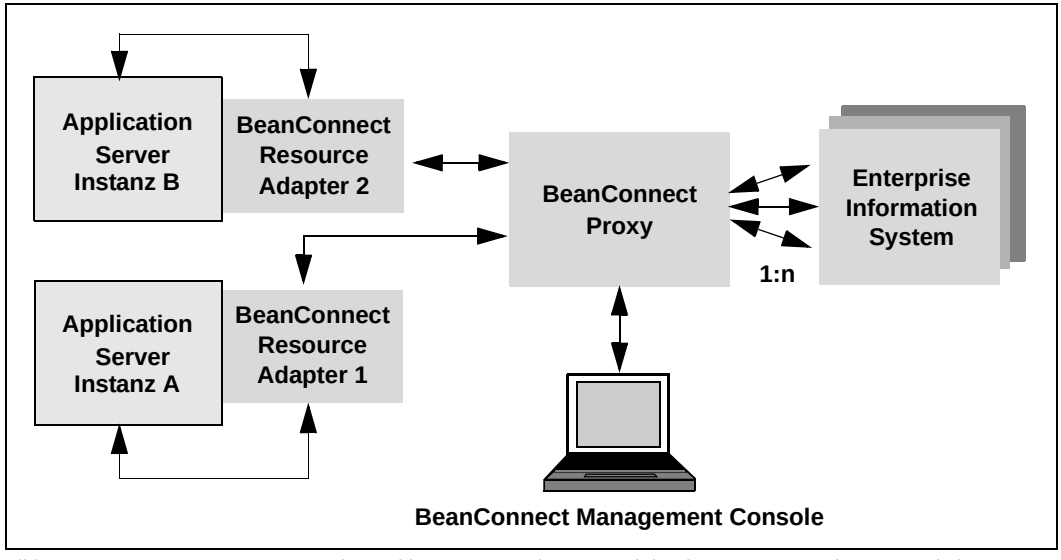


Bild 6: BeanConnect-Komponenten im Multi-Resource Adapter Betrieb mit 2 Resource Adaptern und einem Proxy

Ein Proxy kann bis zu 32 Resource Adapter auf unterschiedlichen Application Server Instanzen verwalten. Der Multi-Resource Adapter Betrieb wird wie beim Standard-Betrieb über die Management Console konfiguriert und administriert.

2.3 BeanConnect als JCA-konformer Resource Adapter

BeanConnect unterstützt verschiedene Kommunikationsrichtungen und -modelle. BeanConnect ermöglicht Outbound- und Inbound-Kommunikation, transaktionale und nicht-transaktionale Kommunikation sowie asynchrone und dialogbasierte Kommunikation.

2.3.1 Outbound- und Inbound-Kommunikation

Bei **Outbound**-Kommunikation kommuniziert eine im Application Server deployte EJB mit einer Partneranwendung auf EIS-Seite.

Bei **Inbound**-Kommunikation sendet ein EIS Nachrichten an eine Message-Driven Bean-Anwendung in einem Java EE Application Server.

BeanConnect unterstützt für openUTM-Partner folgende Kommunikationsvarianten:

- bidirektionale (Outbound und Inbound) und transaktionale oder nicht-transaktionale Kommunikation mit OLTP-Anwendungen (On-Line Transaction Processing) über das OSI TP Protokoll.
- unidirektionaler (nur Outbound) und nicht-transaktionaler Zugriff auf openUTM-Anwendungen über das UPIC-Protokoll. Dieser Zugriff wird ohne Beteiligung des Proxys durchgeführt.
- Nicht-transaktionale asynchrone Inbound-Kommunikation über openUTM-Socket- und RFC1006-Protokoll (siehe Abschnitt „Inbound-Kommunikation“ auf Seite 42).

BeanConnect unterstützt für CICS-Partner folgende Kommunikationsvariante:

- bidirektionaler (Outbound und Inbound) und transaktionaler oder nicht-transaktionaler Zugriff auf/von OLTP-Anwendungen (On-Line Transaction Processing) über das LU6.2-Protokoll. LU6.2 ist ein SNA-Protokoll (Systems Network Architecture), das sowohl System-to-System-Kommunikation als auch System-to-Device-Kommunikation unterstützt.

Außerdem unterstützt BeanConnect über den Proxy nicht-transaktionale Inbound-Kommunikation mit beliebigen Anwendungen, die eines der Protokolle UPIC, openUTM-Socket oder RFC1006 unterstützen (siehe Abschnitt „Inbound-Kommunikation“ auf Seite 42).

2.3.1.1 Outbound-Kommunikation

Bei Outbound-Kommunikation initiiert eine im Application Server deployte EJB die Kommunikation mit einer Partneranwendung auf der EIS-Seite. Outbound-Kommunikation kann sowohl als asynchrone wie auch als dialogbasierte Kommunikation stattfinden.

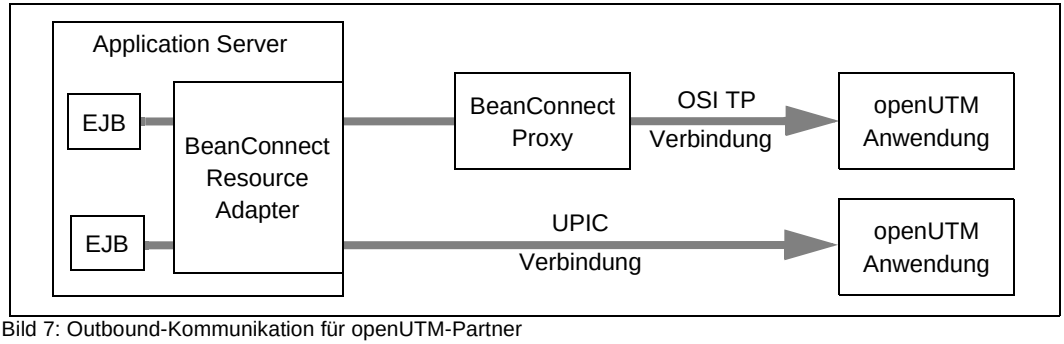


Bild 7: Outbound-Kommunikation für openUTM-Partner

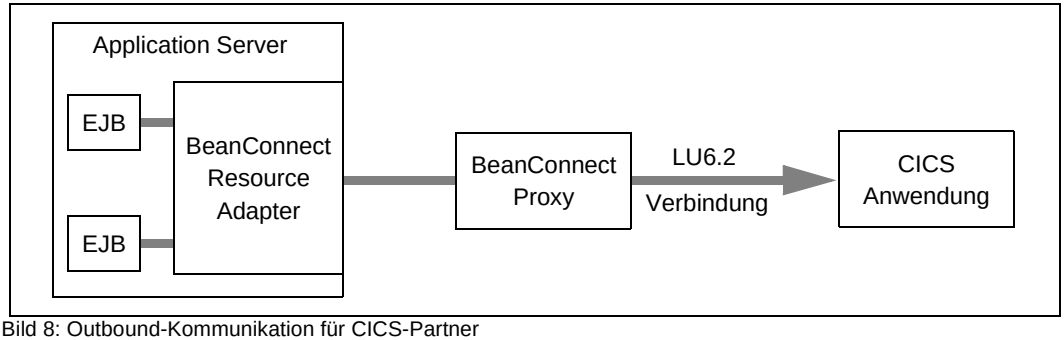


Bild 8: Outbound-Kommunikation für CICS-Partner

2.3.1.2 Inbound-Kommunikation

Bei Inbound-Kommunikation sendet ein EIS Nachrichten an eine Message-Driven Bean-Anwendung in einem Java EE Application Server. Die Message-Driven Bean muss ein Resource Adapter-spezifisches Message-Listener-Interface implementieren. BeanConnect unterstützt das Service Endpoint Message Listener Interface, das zu dem in der JCA-Spezifikation definierten CCI (Common Client Interface) gehört, und darüber hinaus jeweils ein eigenes Message-Listener-Interface für asynchrone und dialogbasierte Kommunikation (siehe Abschnitt „Interfaces“ auf Seite 45).

BeanConnect unterstützt folgende Kommunikationsarten für Inbound-Kommunikation:

- EIS ist eine openUTM-Anwendung
  - transaktionale und nicht-transaktionale Kommunikation über das OSI TP Protokoll (asynchrone und dialogbasierte Kommunikation)
  - nicht-transaktionale Kommunikation über das openUTM-Socket- oder das RFC1006-Protokoll (nur asynchron)
- EIS ist eine CICS-Anwendung
  - transaktionale und nicht-transaktionale Kommunikation über das LU6.2-Protokoll (asynchrone und dialogbasierte Kommunikation)
- EIS ist eine andere Anwendung
  - UPIC-Anwendung: nicht-transaktionale Kommunikation über das UPIC-Protokoll (dialogbasierte Kommunikation)
  - openUTM-Socket-Client oder RFC1006-Anwendung: nicht-transaktionale Kommunikation über das openUTM-Socket- oder das RFC1006-Protokoll (asynchron und dialogbasierte Kommunikation)

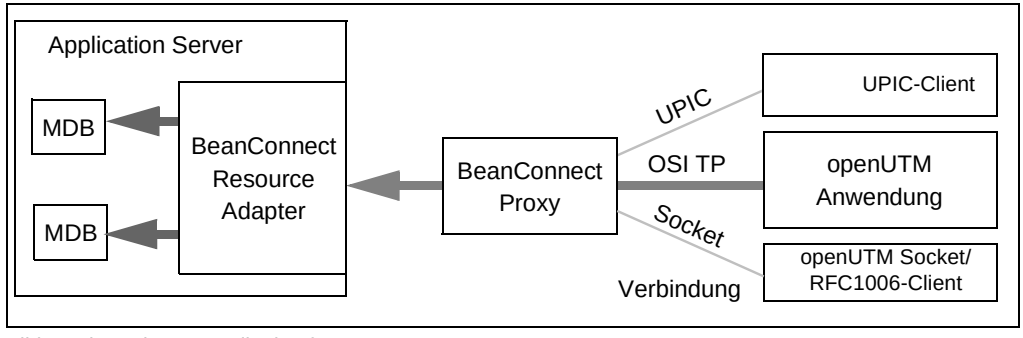


Bild 9: Inbound-Kommunikation für openUTM-Partner

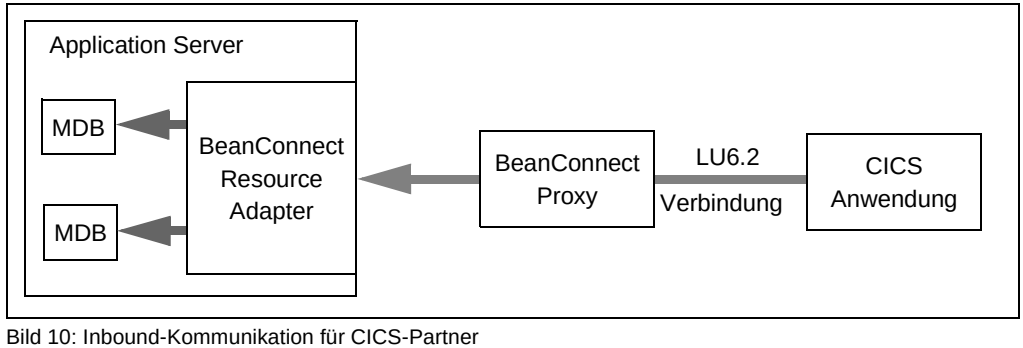


Bild 10: Inbound-Kommunikation für CICS-Partner

2.3.2 Asynchrone und dialogbasierte Kommunikation

Bei Inbound- wie Outbound-Kommunikation unterstützt BeanConnect sowohl asynchrone als auch dialogbasierte Kommunikation.

2.3.2.1 Dialogbasierte Kommunikation

Bei dialogbasierter Kommunikation wartet ein Kommunikationspartner auf eine Antwort des anderen Partners, bevor er die Verarbeitung fortsetzt, d.h.

- bei dialogbasierter Outbound-Kommunikation wartet die Anwendung (EJB) im Application Server auf die Antwort des EIS Partners..
- bei dialogbasierter Inbound-Kommunikation wartet der EIS Partner auf die Antwort der Message-Driven Bean Anwendung.

2.3.2.2 Asynchrone Kommunikation

Bei asynchroner Kommunikation wartet ein Kommunikationspartner nicht auf die Antwort des anderen Partners, d.h.

- bei asynchroner Outbound-Kommunikation sendet die Anwendung (EJB) im Application Server eine Nachricht an den EIS Partner, ohne eine Antwort zu erwarten,
- bei asynchroner Inbound-Kommunikation sendet der EIS Partner eine Nachricht an eine Message-Driven Bean Anwendung, ohne eine Antwort zu erwarten.

2.3.3 Transaktionale und nicht-transaktionale Kommunikation

BeanConnect unterstützt transaktionale und nicht-transaktionale Kommunikation.

Transaktionale Kommunikation ist nur für die Kommunikation mit OLTP-Anwendungen über das OSI TP und das LU6.2-Protokoll möglich.

2.3.3.1 Transaktionale Kommunikation

Im Falle von dialogbasierter Kommunikation ist die Verarbeitung in der entfernten Anwendung Teil der globalen Transaktion. Damit ist gewährleistet, dass die verteilten Ressourcen immer konsistent sind, auch über Anwendungen hinweg.

Im Falle von asynchroner Kommunikation besteht die Transaktion nur während der Übermittlung der Nachricht. Asynchron-Aufträge werden bei transaktionaler Kommunikation genau einmal übermittelt. Das bedeutet, dass bei einer Netzwerkstörung oder beim Ausfall einer Anwendung der Asynchron-Auftrag weder verloren geht, noch mehrfach übermittelt wird.

2.3.3.2 Nicht-transaktionale Kommunikation

Im Falle von nicht-transaktionaler Kommunikation ist die Verarbeitung in der entfernten Anwendung unabhängig von der lokalen Transaktion. Wenn zwei unabhängige Transaktionen miteinander kommunizieren, führen beide Anwendungen unabhängig voneinander ihre Transaktionen aus (Commits und Rollbacks). Dies kann z.B. im Falle einer Kommunikationsstörung zu Dateninkonsistenz in den verschiedenen Anwendungen führen. Diese Art der Kommunikation gewährleistet nicht, dass Asynchron-Aufträge nur einmal übermittelt werden.

2.3.4 Interfaces

BeanConnect unterstützt sowohl BeanConnect-spezifische Interfaces als auch Standard-Interfaces gemäß JCA-Spezifikation. Dieser Abschnitt gibt eine groben Überblick über die Interfaces für Outbound- und Inbound-Kommunikation.

Weitere Einzelheiten zu den Programmier-Interfaces finden Sie in Kapitel „Interfaces und Programmierung“ auf Seite 439.

Interfaces für Outbound-Kommunikation

Bei Outbound-Kommunikation kommuniziert eine im Application Server deployte EJB mit einer Partneranwendung auf EIS-Seite. Diese EJB kann mit EIS Partnern über die Interfaces in folgenden Packages kommunizieren:

- `net.fsc.jca.communication`

  Die im Package `net.fsc.jca.communication` zusammengefassten Interfaces definieren proprietäre BeanConnect-spezifische Kommunikations-Interfaces. Diese unterstützen unterschiedliche Programmier-Modi (wie Send/Receive und Call) und ermöglichen den Zugriff auf die Funktionen, die vom darunterliegenden Kommunikationsprotokoll unterstützt werden.

- `net.fsc.jca.communication.cci`

  Das Common Client Interface (CCI) ist in der JCA-Spezifikation definiert. Es beschreibt einen Standard-API-Client und ist primär an den Anforderungen der Entwicklung von Tools zur Anwendungsentwicklung und EAI-Frameworks (Enterprise Application Integration) ausgerichtet. Verglichen mit dem BeanConnect-spezifischen API bietet das CCI einen eingeschränkten Funktionsumfang.

Interfaces für Inbound-Kommunikation

Bei Inbound-Kommunikation sendet ein EIS Nachrichten an eine Message-Driven Bean-Anwendung in einem Java EE Application Server.

Die Message-Driven Bean muss ein Resource Adapter-spezifisches Message-Listener-Interface implementieren.

BeanConnect unterstützt folgende Message-Listener-Interfaces:

- `net.fsc.jca.communication.AsyncOltpMessageListener`

  BeanConnect-spezifisches Interface für asynchrone Kommunikation

- `net.fsc.jca.communication.OltpMessageListener`

  BeanConnect-spezifisches Interface für dialogbasierte Kommunikation

- `javax.resource.cci.MessageListener`

  Common Client Interface (CCI) für dialogbasierte Kommunikation

Eine Message-Driven Bean, die eines der beiden erst geannten Message-Listener-Interfaces implementiert, wird OLTP Message-Driven Bean genannt.

2.4 BeanConnect im Cluster-Betrieb

BeanConnect unterstützt sowohl Cluster im Application Server als auch Proxy Cluster. Damit können n Instanzen des Resource Adapters m Proxy Instanzen zugeordnet werden.

Der Cluster-Betrieb dient dazu, die Ausfallsicherheit zu erhöhen und eine Lastverteilung zwischen den Instanzen zu realisieren.

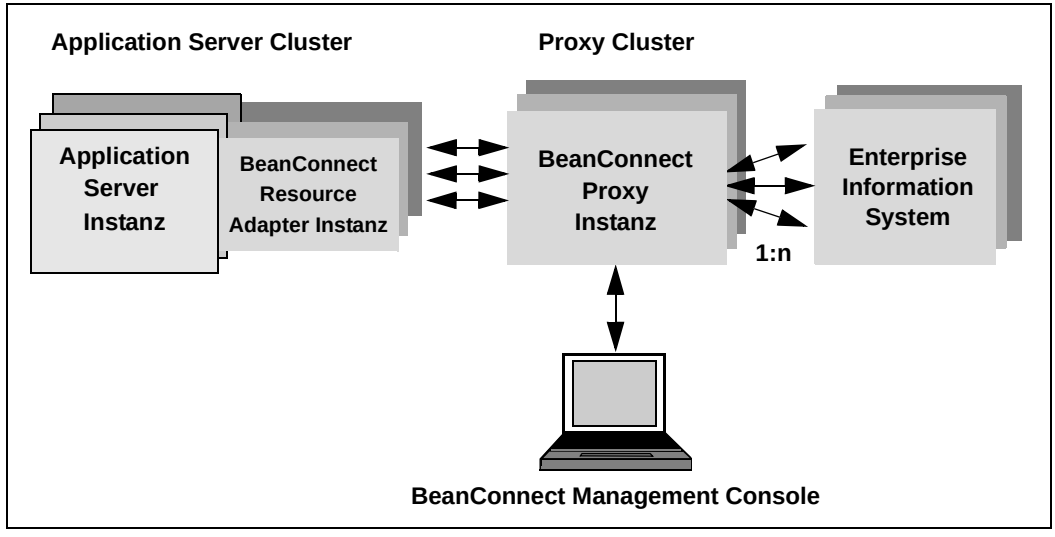


Bild 11: BeanConnect-Komponenten im Cluster-Betrieb

Für die Outbound-Kommunikation ist eine Resource Adapter Instanz zu einer Zeit immer genau einer Proxy Cluster Instanz zugeordnet. Fällt diese Instanz aus, dann sucht sich die Resource Adapter Instanz eine neue Proxy Cluster Instanz. Die Lastverteilung wird mit den Mechanismen des Application Servers realisiert.

Für die Inbound-Kommunikation sind einer Proxy Instanz immer alle Application Server Instanzen zugeordnet, die Lastverteilung übernimmt der Proxy-Container.

Der Proxy Cluster wird mit der Management Console konfiguriert. Eine der Proxy Instanzen ist als Master Instanz ausgezeichnet, mit deren Hilfe die anderen Proxy Instanzen synchronisiert werden, z.B. bei Änderungen in den Konfigurationsdaten.

3 BeanConnect installieren

Dieses Kapitel beschreibt die Installation, Update-Installation und Deinstallation von BeanConnect. Dabei werden folgende Themen behandelt:

- BeanConnect auf Solaris-Systemen installieren
- BeanConnect auf Linux-Systemen installieren
- BeanConnect auf Windows-Systemen installieren
- BeanConnect Resource Adapter installieren
- BeanConnect Tools installieren
- Update-Installation für den BeanConnect Proxy-Container und die Management Console
- BeanConnect deinstallieren
- BeanConnect Resource Adapter deinstallieren
- BeanConnect Tools deinstallieren

Installationsverzeichnisse

In diesem Handbuch werden die verschiedenen Installationsverzeichnisse wie folgt bezeichnet:

`<BC_inst_dir>`
: Installationsverzeichnis für die BeanConnect-Produktdateien.

  Standardeinstellung auf Unix- und Linux-Systemen: `opt/lib/bc30b00`.

`<BC_home>`
: Installationsverzeichnis für die BeanConnect-Dateien und -Verzeichnisse.

`<MC_home>`
: Installationsverzeichnis für die BeanConnect-Management Console. Standardmäßig ist `<MC_home>` ein Unterverzeichnis von `<BC_home>`.

`<Proxy_home>`
: Installationsverzeichnis für den BeanConnect Proxy Container. Standardmäßig ist ist `<Proxy_home>` ein Unterverzeichnis von `<BC_home>`.

Hinweis für Unix- und Linux-Plattformen

Soll die Installation von BeanConnect Resource Adapter bzw. BeanConnect Tools auf einer Unix- oder Linux-Plattform über eine grafische Benutzeroberfläche durchgeführt werden, so muss die Umgebungsvariable DISPLAY gesetzt sein.

Falls eine automated Installation (also ohne grafische Benutzeroberfläche) durchgeführt werden soll, so muss sie über die entsprechende auto-xml-Datei parametrisiert werden.

3.1 BeanConnect auf Solaris-Systemen installieren

BeanConnect unterstützt das Betriebssystem Sun™ Solaris (SPARC).

BeanConnect umfasst die folgenden Produktdateien:

- BeanConnect Proxy
- openUTM
- openUTM-LU62 Gateway (nur für CICS-Partner)
- PCMX

Die Installation von BeanConnect umfasst die folgenden Schritte:

1. Master-Installation
2. BeanConnect Proxy-Container und Management Console installieren

3.1.1 Master-Installation

Die Master-Installation muss unter der root-Kennung erfolgen.

Starten Sie die Master-Installation mit dem folgenden Befehl:

```
pkgadd -d MASTER_BC30B00.pkg
```

Bei der Master-Installation können Sie die folgenden Pakete installieren:

- BeanConnect
- PCMX (Communications Manager UNIX OS)
- openUTM-LU62 Gateway (für CICS-Partner)
- openUTM

Wählen Sie die Pakete, die Sie installieren möchten.

Für die Benutzung von BeanConnect müssen alle Pakete der Master-Installation installiert sein, Sie sollten jedoch nur die Pakete installieren, die noch nicht auf Ihrem System vorhanden sind. Achten Sie auch darauf, dass die Versionen der Pakete übereinstimmen.

Wenn Sie mehrere Pakete installieren wollen, müssen Sie das Master-Installationspaket nicht immer neu starten. Wenn Sie das Master-Paket starten, werden die einzelnen Pakete, die Sie installieren können, durchnummeriert aufgelistet.

Beispiel 1 Installation mehrerer Pakete

Sie möchten die Produktdateien von BeanConnect (Paket 1), PCMX (Paket 2) und openUTM (Paket 4) installieren.

```
The following packages are available:
 1 BC30B00             FUJITSU Software BeanConnect V3.0
                       (sparc) 3.0B00
 2 SMAWpcmx            Communications Manager UNIX OS
                       (sparc) 6.0B00
 3 SMAWutm6s           openUTM-LU62: openUTM LU6.2 comm. (using SNAP-IX)
                       (sparc) 5.1A40
 4 UTM63A00            FUJITSU Software openUTM Enterprise Edition V6.3
                       (sparc) 6.3A00

Select package(s) you wish to process (or 'all' to process all packages).
(default: all) [?,??,q]: 1 2 4
```

Die Installationen der Pakete 1, 2 und 4 werden nacheinander durchlaufen.

3.1.1.1 Produktdateien von BeanConnect installieren

Bevor Sie den BeanConnect Container und/oder die BeanConnect Management Console installieren können, müssen Sie zunächst die Produktdateien von BeanConnect installieren.

Gehen Sie wie folgt vor.

1. Wählen Sie das Paket BeanConnect aus der Master-Installation.
2. Geben Sie das Installationsverzeichnis von BeanConnect an, in das die Produktdateien von BeanConnect installiert werden sollen. Voreinstellung: `/opt/lib`

3.1.1.2 PCMX installieren

PCMX muss als Software-Voraussetzung installiert werden, wenn auf diesem Rechner auch ein BeanConnect Proxy installiert wird. Wenn auf diesem Rechner nur die BeanConnect Management Console oder andere BeanConnect Tools installiert werden, ist die Installation von PCMX nicht notwendig.

1. Wählen Sie das PCMX-Paket (Communications Manager UNIX OS) aus der Master-Installation.
2. PCMX (Communications Manager UNIX OS) wird automatisch installiert.

3.1.1.3 openUTM-LU62 Gateway installieren (für CICS-Partner)

1. Wählen Sie das Paket mit dem openUTM-LU62 Gateway aus der Master-Installation.
2. Das openUTM-LU62 Gateway wird automatisch installiert.

3.1.1.4 openUTM installieren

openUTM muss vor dem BeanConnect Proxy-Container installiert werden. D.h. die Installation von openUTM ist nicht notwendig, wenn Sie auf diesem Rechner nur die BeanConnect Management Console oder andere BeanConnect Tools betreiben wollen.

1. Wählen Sie das openUTM-Paket aus der Master-Installation.
2. Geben Sie das Home-Verzeichnis an, in das openUTM installiert werden soll. Voreinstellung: `/opt/lib`
3. Geben Sie das Basisverzeichnis `<basedir>` an. Wählen Sie hier noch einmal das Verzeichnis, das Sie in Schritt 2 angegeben haben.

3.1.1.5 Silent Installation

Für die Unterpakete openUTM und BeanConnect benötigt man eine sogenannte <response-file>, um die Fragen nach location, owner und group beantworten zu können. Zusätzlich benötigt man für beide Produkte eine modifizierte `<installation-administration-file>`, um die Sicherheitsabfrage nach root-Berechtigung zu unterdrücken.

Die Standard-Datei `default` befindet sich unter `/var/sadm/install/admin`.

1. Kopieren Sie die Standard-Datei.
2. Tragen Sie bei `action= nocheck` ein.
3. Geben Sie die Datei als `<installation-administration-file>` an.

Auf Solaris-Systemen rufen Sie die Silent Installation folgendermaßen auf:

```
pkgadd -r <response-file> -a <installation-administration-file>
-d MASTER_BC30B00.pkg
<<EOF
<Unterpakete Nummer mit Leerzeichen getrennt>
EOF
```

Beispiel für `<response-file>`:

```
LOC="/opt/lib"
OWNER="root"
GROUP="root"
```

3.1.2 BeanConnect Proxy-Container und Management Console installieren

Bevor Sie BeanConnect-Komponenten installieren können, müssen Sie zunächst JDK, PCMX, openUTM, das openUTM-LU62 Gateway (für CICS-Partner) und die Produktdateien von BeanConnect aus der Master-Installation installieren.

Das BeanConnect Installationsprogramm kann folgende Aktionen ausführen:

- Den BeanConnect Proxy-Container und den MC-CmdHandler installieren
- Die BeanConnect Management Console inklusive Command Line Interface (MC-CLI) installieren

Installationsprozedur starten

Nach der Installation der Produktdateien folgt als zweiter Schritt eine benutzerspezifische Installation der BeanConnect-Komponenten.

Melden Sie sich am System unter der Benutzerkennung an, unter der BeanConnect laufen soll. Für die Installation sind keine Root- oder Administrationsrechte erforderlich.

1. Wechseln Sie in das Verzeichnis, in das Sie den Proxy-Container / die Management Console installieren möchten.

   Beispiel:

   Wechseln Sie in das Verzeichnis `<home1/lib>`, wenn Sie den Proxy-Container nach `<home1/lib>/<proxy_cont_name>` und die Management Console nach `<home1/lib>/<console_name>` installieren wollen.

2. Starten Sie die Installation mit folgendem Befehl:

   ```
   <BC_inst_dir>/shsc/install.BeanConnect
   ```

3. Die Installationsprozedur zeigt ein Menü mit den Komponenten an, die Sie installieren können.

   - **1 BeanConnect Proxy-Container**

     Damit installieren Sie den BeanConnect Proxy-Container und den MC-CmdHandler.

   - **2 Management Console (Administration Tool)**

     Damit installieren Sie die BeanConnect Management Console inklusive Command Line Interface (MC-CLI).

   Geben Sie die Nummer(n) der Komponente(n) an, die Sie installieren möchten. Wenn Sie beide Komponenten installieren möchten, geben Sie `1 2` ein. Wenn Sie `q` eingeben, wird die Installationsprozedur beendet.

BeanConnect Proxy installieren

Die Installationsprozedur `<BC_inst_dir>/shsc/install.BeanConnect` benötigt folgenden Input:

1. Benutzer-Basisverzeichnis, in das der BeanConnect Proxy installiert werden soll

   Die Installationsprozedur schlägt das aktuelle Verzeichnis (aus dem die Prozedur gestartet wurde) als das BeanConnect Benutzer-Basisverzeichnis <`BC_home`> vor, in das der Proxy-Container installiert werden soll.

   Wenn Sie dieses Verzeichnis mit `y` bestätigen, wird die Installationsprozedur fortgesetzt.
   Wenn Sie `n` angeben, wird die Installation abgebrochen.

   Um den Proxy-Container in einem anderen Verzeichnis zu installieren, wechseln Sie in dieses Verzeichnis und starten Sie die Installation erneut.

2. Name des Proxy-Containers

   Geben Sie den Namen des Proxy-Containers an.

   Dieser Name muss in Ihrem System eindeutig sein.
   Der Name darf aus maximal acht Zeichen bestehen.
   Folgende Zeichen sind zulässig: A,...,Z, a,...,z, 0,...,9
   Kleinbuchstaben werden in Großbuchstaben umgewandelt.

   Die Voreinstellung lautet `BCCONT`.

   Ein Unterverzeichnis mit dem Namen des Proxy-Containers wird im BeanConnect Benutzer-Basisverzeichnis erstellt. Die Dateien werden in dieses Unterverzeichnis installiert. Daher lautet das Home-Verzeichnis des Proxy-Containers
   `<BC_home>/<proxy_cont_name>`
   (z.B. `/opt/lib/bc30b00/BCCONT`).

   Wenn Sie den Namen eines bereits installierten, unter `<BC_home>` liegenden Proxy-Containers angeben, werden Sie gefragt, ob Sie den Proxy-Container überschreiben (Neuinstallation) oder eine Update-Installation durchführen möchten (siehe Abschnitt „Update-Installation für den BeanConnect Proxy-Container und die Management Console“ auf Seite 80).

3. Convert Long Host Name

   Besitzt der Rechner einen langen Rechnernamen (> 8 Zeichen), so müssen Sie hier einen maximal 8 Zeichen langen „Mapped Hostnamen“ angeben.
   Standardeinstellung: Die letzten 8 Zeichen des Rechnernamens.

4. Home-Verzeichnis für openUTM

   Geben Sie das Home-Verzeichnis für openUTM an. Die Installationsprozedur sucht auch nach bereits auf dem Computer installierten passenden openUTM-Versionen und zeigt eine Liste der installierten UTM-Versionen ab einer Mindestversion an. Aus dieser Liste können Sie das gewünschte Home-Verzeichnis auswählen.

   Ist openUTM nicht im angegebenen Verzeichnis installiert, wird folgende Fehlermeldung angezeigt:
   `openUTM not found!`
   Prüfen Sie in diesem Fall das angegebene Home-Verzeichnis oder die Installation von openUTM (siehe Abschnitt „openUTM installieren“ auf Seite 53

5. Java-Home-Verzeichnis

   Voraussetzung: JDK muss auf Ihrem System installiert sein, bevor Sie den Proxy installieren können.

   Die Installationsprozedur fragt Sie nach dem Java-Home-Verzeichnis. Sie müssen das Verzeichnis explizit angeben, das System schlägt keinen Namen vor. Geben Sie einen vollständigen Verzeichnisnamen an. Die Prozedur prüft Ihre Eingabe.

   Ist das JDK nicht im angegebenen Verzeichnis installiert, wird folgende Fehlermeldung angezeigt:
   `JDK not found!`
   Prüfen Sie in diesem Fall das angegebene Home-Verzeichnis oder installieren Sie JDK zuerst.

6. Bestätigung

   Die Installationsprozedur zeigt den Namen des Archivs an, aus dem der Proxy installiert wird. Die Installation muss aus dem Archiv `<BC_inst_dir>/32/CPIO.BCCont` bzw. `<BC_inst_dir>/64/CPIO.BCCont` erfolgen.

   Bestätigen Sie dieses Archiv.

7. Portnummer des Proxy-Containers

   BeanConnect benötigt ein Intervall von einhundert Portnummern. Das System fordert Sie auf, die Portnummer einzugeben, die den Beginn dieses Intervalls markiert.

   Der Portnummernbereich darf nicht von anderen Proxy-Containern oder anderen Anwendungen verwendet werden.

   `Start Value of the Port Number Range for BeanConnect`

   Geben Sie die Nummer des ersten Ports des Portnummernbereichs ein. Die nächsten hundert Portnummern werden für BeanConnect reserviert. Die Portnummer darf im Bereich von 1025 bis 32667 liegen. Standard: 31000.

8. Administrationspasswort für den Proxy-Container

   Geben Sie das Passwort ein. Standard: admin

9. Sobald dieser Dialog abgeschlossen ist, wird der Proxy-Container installiert.

Wenn Sie mehrere Proxy-Container installieren möchten, können Sie die Installationsprozedur mehrmals wiederholen.

Management Console installieren

Die Installationsprozedur benötigt von Ihnen folgende Angaben:

1. Benutzer-Basisverzeichnis, in das die Management Console installiert werden soll (entfällt, wenn Sie Schritt 1 und 2 ausgewählt haben, d.h. zuvor den Proxy installiert haben)

   Die Installationsprozedur schlägt das aktuelle Verzeichnis (aus dem die Prozedur gestartet wurde) als das BeanConnect Benutzer-Basisverzeichnis <`BC_home`> vor, in das die Management Console installiert werden soll.

   Wenn Sie dieses Verzeichnis mit `y` bestätigen, wird die Installationsprozedur fortgesetzt. Wenn Sie `n` angeben, wird die Installation abgebrochen.

2. Verzeichnis für die Konfigurationsdateien der Management Console.

   Die Konfigurationsdateien werden in einem Unterverzeichnis des Benutzer-Basisverzeichnisses von BeanConnect <`BC_home`> gespeichert.

   Geben Sie das Unterverzeichnis <`console_name`> an. Standard: `console`

   Demzufolge lautet das Home-Verzeichnis für die Management Console `<BC_home>/<console_name>`.

3. Java-Home-Verzeichnis

   Voraussetzungen: JDK muss auf Ihrem System installiert sein, bevor Sie die Management Console installieren können.

   Die Installationsprozedur fragt Sie nach dem Java-Home-Verzeichnis. Geben Sie einen vollständigen Verzeichnisnamen an. Die Prozedur prüft Ihre Eingabe.

   Ist das JDK nicht im angegebenen Verzeichnis installiert, wird folgende Fehlermeldung angezeigt:
   `JDK not found!`
   Prüfen Sie das angegebene Java-Home-Verzeichnis oder installieren Sie JDK zuerst.

4. Bit-Architektur auswählen

   Die Installationsprozedur fragt Sie nach der Bit-Architektur (32/64), unter der die Management Console ablaufen soll. Damit können Sie erzwingen, dass die Management Console unter einer anderen Bit-Architektur abläuft als das JDK, das Sie beim Java-Home-Verzeichnis angegeben haben.

   Wenn Sie den Namen einer im <BC_home> bereits installierten Management Console angeben, werden Sie gefragt, ob Sie die Management Console überschreiben (Neuinstallation) oder eine Update-Installation durchführen möchten (siehe Abschnitt „Update-Installation für den BeanConnect Proxy-Container und die Management Console“ auf Seite 80).

5. Oracle WebLogic Server Konfiguration

   Sie erhalten folgende Abfrage:

   ```
   Configuration for Oracle WebLogic Server?([y]|n)
   ```

   Bei Eingabe von n fahren Sie mit Punkt 6 fort.

   Bei Eingabe von y kommt die Aufforderung:

   ```
   Please enter product installation directory of WebLogic Server or enter
   <RETURN> for remote machine
   ```

   Bei Eingabe von <RETURN> kommt die Meldung:

   ```
   Please copy "wlclient.jar" and "wljmxclient.jar" to "<console-Installations-
   verzeichnis>/bin/weblogic" after this installation
   ```

6. Bestätigung

   Die Installationsprozedur zeigt den Namen des Archivs an, aus dem die Management Console installiert wird. Die Installation muss aus dem Archiv <BC_inst_dir>/CPIO.console erfolgen.

   Bestätigen Sie dieses Archiv.

3.2 BeanConnect auf Linux-Systemen installieren

BeanConnect unterstützt das Betriebssystem Linux.

BeanConnect umfasst die folgenden Produktdateien:

- BeanConnect Proxy
- openUTM
- openUTM-LU62 Gateway (für CICS-Partner)
- PCMX

Die Installation von BeanConnect umfasst die folgenden Schritte:

1. Master-Installation
2. BeanConnect Proxy-Container und die Management Console installieren

3.2.1 Master-Installation

Die Master-Installation muss unter der root-Kennung erfolgen.

Starten Sie die Master-Installation mit dem folgenden Befehl:

```
rpm -i MASTER_BC30B00.rpm --ignorearch --prefix=<BC_install>
```

Geben Sie für `prefix=<BC_install>` das Installationsverzeichnis für BeanConnect an. Ist `prefix` nicht angegeben, wird das Standard-Installationsverzeichnis für BeanConnect `/opt/lib/` verwendet.

Bei der Master-Installation können Sie die folgenden Pakete installieren:

- PCMX (Communications Manager UNIX OS)
- openUTM-LU62 Gateway (für CICS-Partner)
- openUTM
- BeanConnect

Wählen Sie die Pakete, die Sie installieren möchten.

Für die Benutzung von BeanConnect müssen alle Pakete der Master-Installation installiert sein, die für die gewünschten Funktionen benötigt werden. Sie sollten jedoch nur die Pakete installieren, die noch nicht auf Ihrem System vorhanden sind. Achten Sie auch darauf, dass die Versionen der Pakete übereinstimmen.

Wenn Sie mehrere Pakete installieren wollen, müssen Sie das Master-Installationspaket nicht immer neu starten. Wenn Sie das Master-Paket starten, werden die einzelnen Pakete, die Sie installieren können, durchnummeriert aufgelistet.

Beispiel 2 Installation mehrerer Pakete

Sie möchten die Produktdateien von PCMX (Paket 1), openUTM (Paket 3) und BeanConnect (Paket 4) installieren:

```
The following packages are available:
  1  PCMX         Communications Manager LINUX
                 6.0B00
  2  UTMLU62      openUTM-LU62: openUTM LU6.2 communication
                 5.1A40
  3  UTM63A00     FUJITSU Software openUTM Enterprise Edition V6.3
                 6.3A00
  4  BC30B00      FUJITSU Software BeanConnect V3.0
                 3.0B00

Select package(s) you wish to process (or 'all' to process all packages).
(default: all) [?,??,q]: 1 3 4
```

Die Installationen der Pakete 1, 3 und 4 werden nacheinander durchlaufen.

Die eigentliche Installation läuft im Hintergrund ab. Deshalb müssen Sie auf die Abschlussmeldungen nach Ausgabe des shell-Prompts noch kurz warten, um den Erfolg der Installationen beurteilen zu können. Loggen Sie sich daher erst aus, nachdem die Meldungen ausgegeben wurden.

3.2.1.1 PCMX installieren

PCMX muss als Software-Voraussetzung installiert werden, wenn auf diesem Rechner auch ein BeanConnect Proxy installiert wird. Wenn auf diesem Rechner nur die BeanConnect Management Console oder andere BeanConnect Tools installiert werden, ist die Installation von PCMX nicht notwendig.

1. Wählen Sie das PCMX-Paket (Communications Manager UNIX OS) aus der Master-Installation.
2. PCMX wird automatisch in das Verzeichnis `/opt/lib/` installiert.

3.2.1.2 openUTM-LU62 Gateway installieren (für CICS-Partner)

1. Wählen Sie das Paket mit dem openUTM-LU62 Gateway aus der Master-Installation.
2. Das openUTM-LU62 Gateway wird automatisch in das Verzeichnis `/opt/lib/` installiert.

3.2.1.3 openUTM installieren

openUTM muss vor dem BeanConnect Proxy-Container installiert werden. D.h. die Installation von openUTM ist nicht notwendig, wenn Sie auf diesem Rechner nur die BeanConnect Management Console oder andere BeanConnect Tools betreiben wollen.

1. Wählen Sie das openUTM-Paket aus der Master-Installation.
2. openUTM wird automatisch in das Verzeichnis installiert, das Sie für `prefix` angegeben haben.

3.2.1.4 Produktdateien von BeanConnect installieren

Bevor Sie den BeanConnect Container und/oder die BeanConnect Management Console installieren können, müssen Sie zunächst die Produktdateien von BeanConnect installieren.

Gehen Sie wie folgt vor.

1. Wählen Sie das Paket BeanConnect aus der Master-Installation.
2. Die Produktdateien werden automatisch in das Verzeichnis installiert, das Sie für `prefix` angegeben haben.

3.2.1.5 Silent Installation

Die Silent Installation auf Linux rufen Sie genauso auf wie die Master-Installation. Zusätzlich müssen Sie die Nummern der zu installierenden Teilpakete durch Leerzeichen getrennt in die Datei `/response` eintragen.

Bitte beachten Sie Folgendes:
Wenn Sie die Datei `/response` leer lassen, dann werden **alle** Teilpakete installiert.

3.2.2 BeanConnect Proxy-Container und die Management Console installieren

Bevor Sie BeanConnect installieren können, müssen Sie zunächst JDK, PCMX, openUTM, das openUTM-LU62 Gateway (für CICS-Partner) und die Produktdateien von BeanConnect aus der Master-Installation installieren.

Das Installationsprogramm von BeanConnect kann folgende Funktionen ausführen:

- Den BeanConnect Proxy-Container und den MC-CmdHandler installieren
- Die BeanConnect Management Console installieren (inklusive Command Line Interface MC-CLI)

Installationsprozedur starten

Nach der Installation der Produktdateien folgt als zweiter Schritt eine benutzerspezifische Installation der BeanConnect-Komponenten.

Für die Installation muss die Korn-Shell (`ksh`) verwendet werden.

Melden Sie sich am System unter der Benutzerkennung an, unter der BeanConnect laufen soll. Für die Installation sind keine Root- oder Administrationsrechte erforderlich.

1. Wechseln Sie in das Verzeichnis, in das Sie den Proxy-Container / die Management Console installieren möchten.

   Beispiel:

   Wechseln Sie in das Verzeichnis `<home2/lib>`, wenn Sie den Proxy-Container nach `<home2/lib>/<proxy_cont_name>` und die Management Console nach `<home2/lib>/<console_name>` installieren wollen.

2. Starten Sie die Installation mit folgendem Befehl:

   ```
   <BC_inst_dir>/shsc/install.BeanConnect
   ```

3. Die Installationsprozedur zeigt ein Menü mit den Komponenten an, die Sie installieren können.

   - `1 BeanConnect Proxy Container`

     Damit installieren Sie den BeanConnect Proxy-Container und den MC-CmdHandler.

   - `2 Management Console (Administration Tool)`

     Damit installieren Sie die BeanConnect Management Console inklusive MC-CLI.

Geben Sie die Nummer(n) der Komponente(n) an, die Sie installieren möchten. Wenn Sie beide Komponenten installieren möchten, geben Sie `1 2` ein. Wenn Sie `q` eingeben, wird die Installationsprozedur beendet.

BeanConnect Proxy-Container installieren

Die Installationsprozedur `<BC_inst_dir>/shsc/install.BeanConnect` benötigt folgenden Input:

1. Benutzer-Basisverzeichnis, in das der BeanConnect Proxy installiert werden soll

   Die Installationsprozedur schlägt das aktuelle Verzeichnis, aus dem die Prozedur gestartet wurde, als das BeanConnect Benutzer-Basisverzeichnis <`BC_home`> vor, in das der Proxy-Container installiert werden soll.

   Wenn Sie dieses Verzeichnis mit `y` bestätigen, wird die Installationsprozedur fortgesetzt. Wenn Sie `n` angeben, wird die Installation abgebrochen.

   Um den Proxy-Container in einem anderen Verzeichnis zu installieren, wechseln Sie in dieses Verzeichnis und starten Sie die Installation erneut.

2. Name des Proxy-Containers

   Geben Sie den Namen des Proxy-Containers an.

   Dieser Name muss in Ihrem System eindeutig sein.
   Der Name darf aus maximal acht Zeichen bestehen.
   Folgende Zeichen sind zulässig: A,...,Z, a,...,z, 0,...,9
   Kleinbuchstaben werden in Großbuchstaben umgewandelt.

   Die Voreinstellung lautet `BCCONT`.

   Ein Unterverzeichnis mit dem Namen des Proxy-Containers wird im BeanConnect Benutzer-Basisverzeichnis erstellt. Die Dateien werden in dieses Unterverzeichnis installiert. Daher lautet das Home-Verzeichnis des Proxy-Containers
   `<BC_home>/<proxy_cont_name>`
   (z.B. `/opt/lib/bc30b00/BCCONT`).

   Wenn Sie den Namen eines bereits installierten, unter `<BC_home>` liegenden Proxy-Containers angeben, werden Sie gefragt, ob Sie den Proxy-Container überschreiben (Neuinstallation) oder eine Update-Installation durchführen möchten (siehe Abschnitt „Update-Installation für den BeanConnect Proxy-Container und die Management Console“ auf Seite 80).

3. Convert Long Host Name

   Besitzt der Rechner einen langen Rechnernamen (> 8 Zeichen), so müssen Sie hier einen maximal 8 Zeichen langen „Mapped Hostnamen“ angeben.
   Standardeinstellung: Die letzten 8 Zeichen des Rechnernamens.

4. Home-Verzeichnis für openUTM

   Geben Sie das Home-Verzeichnis für openUTM an. Die Installationsprozedur sucht auch nach bereits auf dem Computer installierten passenden openUTM-Versionen und zeigt eine Liste der installierten UTM-Versionen ab einer Mindestversion an. Aus dieser Liste können Sie das gewünschte Home-Verzeichnis auswählen.

   Ist openUTM nicht im angegebenen Verzeichnis installiert, wird folgende Fehlermeldung angezeigt:
   `openUTM not found!`
   Prüfen Sie in diesem Fall das angegebene Home-Verzeichnis oder die Installation von openUTM (siehe Abschnitt „openUTM installieren“ auf Seite 61).

5. Java-Home-Verzeichnis

   Voraussetzung: JDK muss auf Ihrem System installiert sein, bevor Sie den Proxy installieren können.

   Die Installationsprozedur fragt Sie nach dem Java-Home-Verzeichnis. Sie müssen das Verzeichnis explizit angeben, das System schlägt keinen Namen vor. Geben Sie einen vollständigen Verzeichnisnamen an. Die Prozedur prüft Ihre Eingabe.

   Ist das JDK nicht im angegebenen Verzeichnis installiert, wird folgende Fehlermeldung angezeigt:
   `JDK not found!`
   Prüfen Sie in diesem Fall das angegebene Home-Verzeichnis oder installieren Sie JDK zuerst.

   Bitte beachten Sie, dass das JDK auf Linux-Systemen die gleiche Bitausprägung haben muss wie der Proxy!

6. Bestätigung

   Die Installationsprozedur zeigt den Namen des Archivs an, aus dem der Proxy installiert wird. Die Installation muss aus dem Archiv `<BC_inst_dir>/32/CPIO.BCCont` bzw. `<BC_inst_dir>/64/CPIO.BCCont` erfolgen.

   Bestätigen Sie dieses Archiv.

7. Portnummer des Proxy-Containers

   BeanConnect benötigt ein Intervall von einhundert Portnummern. Das System fordert Sie auf, die Portnummer einzugeben, die den Beginn dieses Intervalls markiert.

   Der Portnummernbereich darf nicht von anderen Proxy-Containern oder anderen Anwendungen verwendet werden.

   `Start Value of the Port Number Range for BeanConnect`

Geben Sie die Nummer des ersten Ports des Portnummernbereichs ein. Die nächsten hundert Portnummern werden für BeanConnect reserviert. Die Portnummer darf im Bereich von 1025 bis 32667 liegen. Standard: 31000.

8. Administrationspasswort für den Proxy-Container

   Geben Sie das Passwort ein. Standard: admin

9. Sobald dieser Dialog abgeschlossen ist, wird der Proxy-Container installiert.

Wenn Sie mehrere Proxy-Container installieren möchten, können Sie die Installationsprozedur mehrmals wiederholen.

Management Console installieren

Die Installationsprozedur benötigt von Ihnen folgende Angaben:

1. Benutzer-Basisverzeichnis, in das die Management Console installiert werden soll (entfällt, wenn Sie Schritt 1 und 2 ausgewählt haben, d.h. zuvor den Proxy installiert haben)

   Die Installationsprozedur schlägt das aktuelle Verzeichnis (aus dem die Prozedur gestartet wurde) als das BeanConnect Benutzer-Basisverzeichnis <BC_home> vor, in das die Management Console installiert werden soll.

   Wenn Sie dieses Verzeichnis mit y bestätigen, wird die Installationsprozedur fortgesetzt. Wenn Sie n angeben, wird die Installation abgebrochen.

2. Verzeichnis für die Konfigurationsdateien der Management Console.

   Die Konfigurationsdateien werden in einem Unterverzeichnis des Benutzer-Basisverzeichnisses von BeanConnect <BC_home> gespeichert.

   Geben Sie das Unterverzeichnis <console_name> an. Standard: console

   Demzufolge lautet das Home-Verzeichnis für die Management Console <BC_home>/<console_name>.

3. Java-Home-Verzeichnis

   Voraussetzungen: JDK muss auf Ihrem System installiert sein, bevor Sie die Management Console installieren können.

   Die Installationsprozedur fragt Sie nach dem Java-Home-Verzeichnis. Geben Sie einen vollständigen Verzeichnisnamen an. Die Prozedur prüft Ihre Eingabe.

   Ist das JDK nicht im angegebenen Verzeichnis installiert, wird folgende Fehlermeldung angezeigt:
   JDK not found!
   Prüfen Sie das angegebene Java-Home-Verzeichnis oder installieren Sie JDK zuerst.

4. Oracle WebLogic Server Konfiguration

   Sie erhalten folgende Abfrage:

   ```
   Configuration for Oracle WebLogic Server?([y]|n)
   ```

   Bei Eingabe von `n` fahren Sie mit Punkt 5 fort.

   Bei Eingabe von `y` kommt die Aufforderung:

   ```
   Please enter product installation directory of WebLogic Server or enter
   <RETURN> for remote machine
   ```

   Bei Eingabe von <RETURN> kommt die Meldung:

   ```
   Please copy "wlclient.jar" and "wljmxclient.jar" to "<Console-Installations-
   verzeichnis>/bin/weblogic" after this installation
   ```

5. Bestätigung

   Die Installationsprozedur zeigt den Namen des Archivs an, aus dem die Management Console installiert wird. Die Installation muss aus dem Archiv `<BC_inst_dir>/CPIO.console` erfolgen.

   Bestätigen Sie dieses Archiv.

3.3 BeanConnect auf Windows-Systemen installieren

BeanConnect unterstützt das Betriebssystem Microsoft Windows.

3.3.1 BeanConnect-DVD

Für die Installation brauchen Sie die Administrationsberechtigung.

Folgende Pakete befinden sich auf der BeanConnect DVD:

- PCMX32 V5.0A80
- openUTM V6.3A00
- BeanConnect Proxy V3.0B00
- openUTM-LU62 Gateway V5.1A41 (für CICS-Partner)

Für die Benutzung von BeanConnect müssen alle Pakete installiert sein, die für die gewünschten Funktionen benötigt werden. Sie sollten jedoch nur die Pakete installieren, die noch nicht auf Ihrem System vorhanden sind. Achten Sie auch darauf, dass die Versionen der Pakete übereinstimmen.

3.3.1.1 PCMX installieren

Bevor Sie openUTM und BeanConnect installieren, müssen Sie PCMX installieren.

1. Öffnen Sie die DOS-Eingabeaufforderung unter Administrationsberechtigung und geben Sie folgenden Befehl ein:

   ```
   msiexec /i <pcmx-verzeichnis>\pcmx-32.msi
   ```

2. Klicken Sie im Fenster **PCMX-32 Installation** auf die Schaltfläche **Next >**, um folgende Dialogfelder zu öffnen:

   - Klicken Sie im Dialogfeld **Information** auf **Readme**, um die Readme-Datei zu öffnen, die detaillierte Informationen enthält. Klicken Sie auf **License**, um die Lizenzvereinbarung anzuzeigen.
   - Aktivieren Sie im Dialogfeld **... choose your preferred Installation method** die Option **Standard Installation** (Voreinstellung) und klicken Sie auf die Schaltfläche **Next >**.

     Daraufhin wird das Dialogfeld **Ready to install** angezeigt, das eine Liste der Installationsverzeichnisse enthält.
   - Klicken Sie auf **Next >** um fortzufahren.
   - Klicken Sie im Dialogfeld **Installation completed** auf **Finish**.

3.3.1.2 openUTM installieren

openUTM muss vor BeanConnect installiert werden.

1. Öffnen Sie die DOS-Eingabeaufforderung unter Administrationsberechtigung und geben Sie folgenden Befehl ein:

   ```
   msiexec /i <openutm-verzeichnis>\utm.msi
   ```

2. Befolgen Sie die Anweisungen des Installationsprogramms und wählen Sie die gewünschten Optionen aus.

   openUTM überprüft die Systemanforderungen und stellt sicher, dass ausreichend Plattenspeicherplatz zur Verfügung steht. Erfüllt das System diese Anforderungen nicht, wird die Installation nicht durchgeführt.

   Befindet sich auf Ihrem Computer eine ältere Version von openUTM und Sie installieren openUTM in dasselbe Verzeichnis, so wird die ältere Version automatisch überschrieben.

   Befindet sich auf Ihrem Computer eine ältere Version von openUTM und Sie installieren openUTM in ein anderes Verzeichnis, so ist die zuletzt installierte Version die gültige Version.

3. Starten Sie das System neu, wenn Sie von der Installationsroutine dazu aufgefordert werden.

3.3.1.3 BeanConnect installieren

Bevor Sie BeanConnect installieren, müssen Sie bereits JDK, PCMX und openUTM installiert haben.

Das BeanConnect Installationsprogramm führt folgende Aktionen aus:

- Es installiert den BeanConnect Proxy-Container und den MC-CmdHandler
- Es installiert die BeanConnect Management Console inklusive Command Line Interface MC-CLI

Falls Sie eine bestehende Installation aktualisieren möchten, finden Sie weitere Einzelheiten in Abschnitt „Update-Installation für den BeanConnect Proxy-Container und die Management Console“ auf Seite 80.

Installationsprozedur

Die Installation von BeanConnect umfasst die folgenden Schritte:

1. Starten Sie das Installationsprogramm `<beanconnect-verzeichnis>\setup.exe`. unter Administrationsberechtigung.

   Daraufhin wird das Dialogfeld **Installation** geöffnet. Mit den Schaltflächen **Next >** und **< Back** können Sie durch die Folge von Dialogfeldern navigieren, die jetzt angezeigt werden.

2. Dialogfeld **Welcome**
   Klicken Sie auf die Schaltfläche **Next**, um die Installation zu starten.

3. Dialogfeld **Information**
   Klicken Sie auf **Readme**, um die BeanConnect Readme-Datei anzuzeigen.

4. Dialogfeld **Choose Language**
   Wählen Sie die Sprache für das Online-Hilfesystem der Management Console aus.

5. Dialogfeld **Common Resources**
   Entscheiden Sie, ob Sie die Common Resources installieren möchten. Dies sind z.B. Archive, Bibliotheken und Dokumente etc.. Bei einer Neuinstallation müssen Sie immer auch die Common Resources mit installieren, siehe „Common Resources installieren“ auf Seite 70.

   Falls Sie die Common Resources nicht installieren wollen, werden Sie im Dialogfeld **Common Resources Directory** aufgefordert, das Verzeichnis anzugeben, wo die Common Resources installiert sind, die Sie verwenden wollen.

   Bei einer Update-Installation müssen Sie sicher stellen, dass die aktuellen Common Resources installiert werden.

6. Dialogfeld **Installation Option**
   Wählen Sie den Eintrag **Normal installation** aus, um eine Neuinstallation durchzuführen.

   Wählen Sie den Eintrag **Update installation**, um eine vorhandene BeanConnect-Installation zu aktualisieren. Informationen zur Update-Installation finden Sie im Abschnitt „Update-Installation für den BeanConnect Proxy-Container und die Management Console“ auf Seite 80.

   Wählen Sie den Eintrag **Update installation without deinstallation**, um eine vorhandene BeanConnect-Installation zu aktualisieren ohne die bestehende Installation zu deinstallieren. Informationen zur Update-Installation finden Sie im Abschnitt „Update-Installation für den BeanConnect Proxy-Container und die Management Console“ auf Seite 80.

7. Dialogfeld **Proxy Components**
   Wählen Sie die zu installierenden Komponenten aus, indem Sie die entsprechenden Optionen aktivieren.
   Sie können eine oder beide Komponenten auswählen. Der anschließende Installationsprozess hängt von den Komponenten ab, die Sie für die Installation ausgewählt haben.

   - **BeanConnect Proxy Container**

     Über diese Option installieren und initiieren Sie den BeanConnect Proxy-Container und den MC-CmdHandler. Dazu wird zunächst das Dialogfeld **openUTM Directory** und anschließend das Dialogfeld **Java Directory** geöffnet.

     Dialogfeld **openUTM Directory**
     Geben Sie das openUTM-Home-Verzeichnis an, das die Datei `ex\libwork.dll` enthält.

     Dialogfeld **JAVA Directory**
     Geben Sie das Java-Home-Verzeichnis an. Dies ist das Verzeichnis auf Ihrem System, das die JDK-Dateien `lib\tools.jar` und `bin\java.exe` enthält.

   - **Management Console (Administration Tool)**

     Über diese Optionen installieren Sie die BeanConnect Management Console, mit der Sie BeanConnect Proxys konfigurieren und administrieren können.

     Wenn Sie nur die Option **Management Console (Administration Tool)** aktivieren, wird anschließend das Dialogfeld **Java Directory** geöffnet.

     Dialogfeld **JAVA Directory**
     Geben Sie das Java-Home-Verzeichnis an. Dies ist das Verzeichnis auf Ihrem System, das die JDK-Dateien `lib\tools.jar` und `bin\java.exe` enthält.

Common Resources installieren

Wenn Sie im Dialogfeld **Common Resources** die Installation der Common Resources ausgewählt haben, dann werden folgende Dialogfelder angezeigt.

1. Dialogfeld **Target BeanConnect Common Resources Directory to install**

   Geben Sie das Home-Verzeichnis an, in das die BeanConnect Common Resources installiert werden sollen. Voreinstellung: `C:\BeanConnect\BC30B00`.

2. Dialogfeld **BeanConnect Common Resources Program Group**

   Wählen Sie die Programmgruppe, in die das Installationsprogramm die BeanConnect-Symbole für die Common Resources speichern soll. Sie können eine bereits vorhandene Programmgruppe wählen oder eine neue erstellen.

   Standardmäßig erstellt das Installationsprogramm folgende Programmgruppe:

   `FUJITSU Software BeanConnect V3.0B00`

BeanConnect Proxy Container installieren

Existiert bereits ein Proxy Container gleichen Namens, so muss dieser vor der Installation ordnungsgemäß beendet werden.

Wenn Sie im Dialogfeld **Proxy Components** die Komponente **BeanConnect Proxy Container** auswählen, wird folgender Installationsdialog angezeigt:

1. Dialogfeld **Proxy Container Options**

   Geben Sie den Namen, die Portnummer und das Passwort des Proxy an.

   - **Name for Proxy Container**

     Name des Proxy-Containers. Standard: `BCCont`

     Dieser Name muss in Ihrem System eindeutig sein.
     Der Name darf aus maximal acht Zeichen bestehen.
     Folgende Zeichen sind zulässig: A,...,Z, a,...,z, 0,...,9

   - **Begin of port interval for Proxy Container**

     BeanConnect benötigt ein Intervall von 100 Portnummern. Das System fordert Sie auf, die Portnummer einzugeben, die den Beginn dieses Intervalls markiert.

     Der Portnummernbereich darf nicht von anderen Proxy-Containern oder anderen Anwendungen verwendet werden.

     `Start Value of the Port Number Range for BeanConnect`

     Geben Sie die Nummer des ersten Ports des Portnummernbereichs ein. Die nächsten hundert Portnummern werden für BeanConnect reserviert. Die Portnummer darf im Bereich von 1025 bis 32667 liegen. Standard: 31000.

   - **Password for Proxy Container**

     Geben Sie das Administrations-Passwort ein. Standard: admin

2. Dialogfeld **Target Proxy Container Directory to install**
   Geben Sie das Home-Verzeichnis für den Proxy-Container an, in das der Proxy-Container installiert werden soll.

   Standardmäßig wird der Proxy-Container im BeanConnect Home-Verzeichnis `<BC_home>\<proxy_cont_name>` installiert.

   Klicken Sie auf die Schaltfläche **Browse**, um ein anderes Verzeichnis auszuwählen.

3. Dialogfeld **Proxy Container Program Group**
   Wählen Sie die Programmgruppe, in die das Installationsprogramm die BeanConnect Symbole für den Proxy-Container speichern soll. Sie können eine bereits vorhandene Programmgruppe wählen oder eine neue erstellen.

   Standardmäßig erstellt das Installationsprogramm folgende Programmgruppe:

   `FUJITSU Software BeanConnect V3.0B00\<proxy_cont_name>`

4. Dialogfeld **Convert Long Host Name**

   Besitzt der Rechner einen langen Rechnernamen (> 8 Zeichen), so müssen Sie hier einen maximal 8 Zeichen langen „Mapped Hostname“ angeben.
   Standardeinstellung: Die letzten 8 Zeichen des Rechnernamens.

5. Dialogfeld **Permit ports through the windows firewall**

   Yes (Standardwert) bewirkt, dass das Installationsprogramm die Ports zum Betrieb des Management Console Command Handlers (MC-CmdHandler) für die Nutzung der Firewall freischaltet.

BeanConnect Management Console installieren

Wenn Sie im Dialogfeld **Proxy Components** die Komponente **Management Console (Administration Tool)** auswählen, wird folgender Installationsdialog angezeigt:

1. Dialogfeld **Target BeanConnect Management Console Directory to install**
   Geben Sie das Home-Verzeichnis für die Management Console an, in das die Management Console installiert werden soll.

   Standardmäßig wird die Management Console im Home-Verzeichnis von BeanConnect unter `<BC_home>\Console` installiert.

   Klicken Sie auf die Schaltfläche **Browse**, um ein anderes Verzeichnis auszuwählen.

2. Dialogfeld **BeanConnect Management Console Program Group**
   Wählen Sie die Programmgruppe, in die das Installationsprogramm die BeanConnect-Symbole für die Management Console speichern soll. Sie können eine bereits vorhandene Programmgruppe wählen oder eine neue erstellen.

   Standardmäßig erstellt das Installationsprogramm folgende Programmgruppe:

   `FUJITSU Software BeanConnect V3.0B00\\Management Console`

BeanConnect Proxy-Container und Management Console installieren

Wenn Sie im Dialogfeld **Proxy Components** sowohl den Eintrag **BeanConnect Proxy Container** als auch **Management Console (Administration Tool)** auswählen, werden der Proxy-Container und die Management Console in einem Schritt installiert.

Eine detaillierte Beschreibung der Dialogfelder finden Sie weiter oben.

In diesem Fall werden die folgenden Dialogfelder nur einmal angezeigt:

- **JAVA Directory**
- **Target BeanConnect Common Resources Directory to install**
- **BeanConnect Proxy Program Group**

Installation abschließen

Nun haben Sie alle Einträge und Einstellungen vorgenommen, die für die Installation erforderlich sind.

1. Dialogfeld **Ready to Install!**
   Klicken Sie auf die Schaltfläche **Install**, um die Installation zu starten.
2. Dialogfeld **Installation Completed!**
   Klicken Sie auf die Schaltfläche **Finish**, um die Installation abzuschließen.

Je nachdem, welche Option(en) Sie ausgewählt haben, erstellt das Installationsprogramm einen Proxy-Container samt "internem" MC-CmdHandler und/oder die Management Console.

Die Produktdateien von BeanConnect werden im Laufe des Installationsprozesses automatisch installiert.

3.3.1.4 openUTM-LU62 Gateway installieren (für CICS-Partner)

Sie installieren das openUTM-LU62 Gateway wie folgt:

1. Wählen Sie im Dialogfeld **Choose Products** (Beschreibung siehe oben) die Installation des openUTM-LU62 Gateways aus.
2. Befolgen Sie die Anweisungen des Installationsprogramms und wählen Sie die gewünschten Optionen.
3. Wählen Sie das Home-Verzeichnis, in das das openUTM-LU62 Gateway installiert werden soll.
4. Klicken Sie im Dialogfeld **Installation completed** auf **Finish**.

3.3.2 Den BeanConnect Proxy-Container über die Befehlszeile installieren

Das BeanConnect-Installationsprogramm `setup.exe` verfügt auch über eine Befehlsschnittstelle, die nachfolgend beschrieben wird.

Um das Produkt mittels Befehlen zu installieren, öffnen Sie ein DOS-Eingabefenster. Wechseln Sie in folgendes Verzeichnis:

```
<DVD_drive>\BEANCONNECT-PROXY\Windows
```

Geben Sie anschließend folgenden Befehl ein:

```
Setup.exe [/S /M=paramfile] [/E=errorfile]
```

Mit der Option `/S` geben Sie an, dass Sie die Installation ohne eine grafische Bedienoberfläche durchführen möchten. Geben Sie in diesem Fall außerdem den Parameter `/M=paramfile` ein. Die Installationsparameter werden daraufhin der Datei `paramfile` entnommen, in der Sie die Installationsparameter und -verzeichnisse selbst angeben können. Die Datei `paramfile` muss dem nachfolgend beschriebenen Format entsprechen. Wenn Sie eine Parameterdatei verwenden möchten, sollten Sie in dieser Datei sämtliche Parameter angeben, denn es ist in diesem Fall nicht garantiert, dass die Standardwerte verwendet werden.

Wenn Sie die Option `/S` nicht angeben, wird die Installation über die grafische Benutzeroberfläche gestartet.

Mit der Option `/E=errorfile` können Sie eine Protokolldatei angeben, in der die verwendeten Parameter, der Fortschritt der Installation und etwaige Fehlermeldungen aufgezeichnet werden. Wenn Sie die Option `/S` angeben, sollten Sie auch die Option `/E` verwenden.
Wenn Sie die Option `/E` nicht angeben, könnten etwaige Fehlermeldungen verloren gehen. Falls die Datei `errorfile` noch nicht existiert, wird sie vom Installationsprogramm erstellt. Gibt es bereits eine solche Datei, wird sie überschrieben.

Format der Parameterdatei

Nachfolgend finden Sie ein Beispiel einer Parameterdatei. Diese Datei (`BeanConnect_Install.ini`) ist im Lieferumfang enthalten. Sie können Sie an Ihre Anforderungen anpassen.

;Online Help Language; A: English; B: German	
OHELP_CFG=A	Sprache für das Online-Hilfesystem
;Installation Option UPDATEINSTALL=N	Update-Installation?

;BeanConnect Proxy directory to update UPDBEANCONNECTDIR=	
INST_PROXY=Y	Proxy-Container erstellen
INST_CONSOLE=Y	Management Console erstellen
; openUTM Directory UTMPATH=c:\openUTM-Server	Home-Verzeichnis für openUTM
; JAVA Home Directory	Java-Home-Verzeichnis
JAVA_HOME=C:\jdk1.7	
;BeanConnect Common Resources Directory ROOTDIR=C:\BeanConnect	BeanConnect Home-Verzeichnis
; ---------- ;Proxy parameters. Only evaluated if INST_PROXY=Y and UPDATEINSTALL=N ;----------	
PROXY_NAME=BCCont	Name des Proxy-Containers
PORT_BEGIN=31000	Port für die Kommunikation mit dem BeanConnect Proxy-Container
PROXY_PASSWORD=admin	Administrations-Password des Proxy-Containers
; Target Proxy Directory	
PROXYDIR=C:\BeanConnect\BCCont	Verzeichnis für die Installation des Proxy-Containers
; ---------- ;Management Console. Only evaluated if INST_CONSOLE=Y ; ----------	
; Only for Update installation. ; Existing Console directory to update	
UPDCONSOLEDIR=C:\BeanConnect\Console	Verzeichnis der vorhandenen Management Console
; Target Console directory CONSOLEDIR=C:\BeanConnect\Console	Verzeichnis zum Installieren der Management Console

3.4 BeanConnect Resource Adapter installieren

Sie installieren den BeanConnect Resource Adapter mit der JAR-Datei `BC30B00_RA.jar`.

Diese Datei enthält die folgenden Ressourcen:

- die RAR-Datei `BC30B00.rar`, die die Resource Adapter Klassen für JCA 1.6 enthält. Weitere Informationen zur Verwendung der RAR-Datei finden Sie im Abschnitt „Überblick“ auf Seite 92.
- die RAR-Datei `BC30B004JCA15.rar`, die die Resource Adapter Klassen für JCA 1.5 enthält. Weitere Informationen zur Verwendung der RAR-Datei finden Sie im Abschnitt „Überblick“ auf Seite 92.
- die Datei `BeanConnectDev.jar` zum Kompilieren der EJBs
- die Datei `BeanConnectVerifyer.jar`, die zur BeanConnect RAR-Datei hinzugefügt werden muss, wenn man den Sun JEE Verifier (Teil des Sun Glassfish V2 JEE5 Application Servers) benutzen möchte.
- Verzeichnis `config`: log4j-Konfigurationsdateien für das Logging des Resource Adapters
- Verzeichnis `Docs`: Dokumente
- Verzeichnis `encoding`: Beispiel für das Encoding.
- Verzeichnis `JavaDoc`: JavaDoc des Benutzer-API.
- Verzeichnis `scid`: Diagnose-Tool

Resource Adapter auf Systemen mit grafischer Oberfläche installieren

Auf Systemen mit grafischen Oberfläche gehen Sie wie folgt vor:

1. Öffnen Sie im jeweiligen System ein Fenster für die Eingabe von Kommandos, z.B. Shell oder DOS-Eingabeaufforderung
2. Wechseln Sie in das Verzeichnis, in dem sich JAR-Datei befindet
3. Entpacken Sie die JAR-Datei mit dem Befehl
   `java –jar BC30B00_RA.jar`
4. Befolgen Sie die Anweisungen des grafischen Installationsprogramms und legen Sie dabei das Installationsverzeichnis des Resource Adapters fest

Resource Adapter auf Systemen ohne grafische Oberfläche installieren

Für die Installation auf Systemen ohne grafische Oberfläche wird die Datei `RA-auto.xml` mit ausgeliefert. Gehen Sie wie folgt vor:

1. Öffnen Sie die Datei `RA-auto.xml` mit einem Texteditor und tragen Sie im Tag `<installpath>` den gewünschten Installationspfad des Resource Adapters ein.
2. Öffnen Sie im jeweiligen System ein Fenster für die Eingabe von Kommandos, z.B. Shell oder DOS-Eingabeaufforderung.
3. Wechseln Sie in das Verzeichnis, in dem sich die JAR-Datei befindet.
4. Entpacken Sie die JAR-Datei mit dem Befehl
   `java -jar BC30B00_RA.jar RA-auto.xml`

   Der Resource Adapter wird dadurch automatisch installiert. Die Dateien werden im aktuellen Verzeichnis ausgepackt ohne `RA-auto.xml` zu ändern.

3.5 BeanConnect Tools installieren

BeanConnect stellt zusätzlich zum Proxy und Resource Adapter mehrere Tools zur Verfügung. Diese Tools können separat installiert werden und liegen in Form von JAR-Archiven vor. Die Tools stehen in folgenden Archiven zur Verfügung:

MC-CmdHandler	`BC30B00_MCCmdHandler.jar`
Cobol2Java	`BC30B00_Cobol2Java.jar`

BeanConnect Tools auf Systemen mit grafischer Oberfläche installieren

Auf Systemen mit grafischer Oberfläche gehen Sie wie folgt vor:

1. Öffnen Sie im jeweiligen System ein Fenster für die Eingabe von Kommandos, z.B. Shell oder DOS-Eingabeaufforderung.
2. Wechseln Sie in das Verzeichnis, in dem sich die JAR-Datei befindet.
3. Entpacken Sie die JAR-Datei mit folgendem Befehl
   `java -jar <jar-archiv>`

   `<jar-archiv>` ist der oben aufgelistete Name des jeweiligen JAR-Archivs.
4. Befolgen Sie die Anweisungen des grafischen Installationsprogramms und legen Sie dabei das Installationsverzeichnis des Tools und ggf. weitere Parameter fest.

BeanConnect Tools auf Systemen ohne grafische Oberfläche installieren

Für die Installation auf Systemen ohne grafische Oberfläche wird für jedes Tool eine Auto-xml-Datei mit ausgeliefert. Diese besitzt den Namen `<tool>-auto.xml`, dabei ist `<tool>` der Name des zu installierenden Tools (`MCCmdHandler`, `Cobol2Java`).

Gehen Sie wie folgt vor:

1. Öffnen Sie die Auto-xml-Datei mit einem Texteditor und tragen Sie im Tag `<installpath>` den gewünschten Installationspfad des Tools ein. Bei einigen Tools müssen Sie noch weitere auskommentierte Tags mit den Konfigurationsdaten versorgen wie z.B. Portnummer oder Passwort.
2. Öffnen Sie im jeweiligen System ein Fenster für die Eingabe von Kommandos, z.B. Shell oder DOS-Eingabeaufforderung.
3. Wechseln Sie in das Verzeichnis, in dem sich die JAR-Datei befindet.
4. Entpacken Sie die JAR-Datei mit folgendem Befehl
   `java -jar <jar-archiv> <tool>-auto.xml`

   Das Tool wird dadurch automatisch installiert.

5. Für das Tool MC-CmdHandler müssen Sie nach der Installation noch den Pfad des JDK in die Datei `javaenv.cmd` (Windows-Systeme) bzw. `javaenv.sh` (Linux-, Solaris-Systeme) eintragen. Dazu editieren Sie die Datei mit einem Texteditor.

3.6 Update-Installation für den BeanConnect Proxy-Container und die Management Console

BeanConnect ermöglicht es Ihnen, Update-Installationen durchzuführen. Bei einer Update-Installation können Sie eine neue BeanConnect-Version oder einen neuen Patch installieren und dabei die Konfigurationsdaten des installierten Proxy-Containers und der Management Console verwenden.

In diesem Abschnitt erhalten Sie einen Überblick über folgende Themen:

- Update-Installation auf Solaris-Systemen
- Update-Installation auf Linux-Systemen
- Update-Installation auf Windows-Systemen

3.6.1 Update-Installation auf Solaris-Systemen

In diesem Abschnitt erfahren Sie, wie Sie die Installation des Proxy-Containers und der Management Console auf Solaris-Systemen aktualisieren. Wenn Sie neue Software erhalten, starten Sie die Master-Installation und installieren Sie die neuen Produkte (siehe Abschnitt „BeanConnect auf Solaris-Systemen installieren“ auf Seite 51).

Update-Installation starten

Die Update-Installation erfolgt auf dieselbe Weise wie eine Neuinstallation.

Wechseln Sie in das gewünschte installationsverzeichnis und starten Sie die Installationsprozedur mit folgendem Befehl:

```
<BC_inst_dir>/shsc/install.BeanConnect
```

Wie bei einer Neuinstallation zeigt die Installationsprozedur ein Menü mit den beiden Komponenten an, die installiert werden können. Wählen Sie die Komponenten aus, für die Sie ein Update installieren möchten:

```
Configuration Options

1 BeanConnect Proxy Container
2 Management Console (Administration Tool)
```

Die Installationsprozedur erkennt, wenn Sie sich in einem Verzeichnis befinden, das bereits eine der Komponenten enthält, und prüft, ob diese Version aktualisiert werden kann. Ist ein Update möglich, werden Sie gefragt, ob Sie eine Update-Installation durchführen oder die bestehende Version überschreiben möchten (Neuinstallation).

Für die ausgewählten Komponenten wird eine Update-Installation ausgeführt.

BeanConnect Proxy-Container

1. Das Home-Verzeichnis des Proxy-Containers wird kopiert und in das Verzeichnis `<BC_home>/<proxy_cont_name>.save` gespeichert. Dieses Verzeichnis wird bei der Update-Installation nicht geändert. Alle alten Konfigurationsdaten bleiben erhalten.
2. Abhängig davon, ob altes und neues Verzeichnis identisch sind, gilt Folgendes:
   – Falls das alte und das neue Verzeichnis identisch sind, dann wird der alte Proxy Container umbenannt, ein neuer Proxy Container installiert und die Daten vom alten Proxy Container übernommen.
   – Falls das alte und das neue Verzeichnis verschieden sind, dann wird ein neuer Proxy Container installiert. Anschließend werden die Daten vom alten Proxy Container übernommen.
3. Die alte Konfiguration wird in den neuen Proxy-Container übernommen.

Management Console

1. Das Home-Verzeichnis der Management Console wird kopiert und in das Verzeichnis `<BC_home>/<console_name>.save` gespeichert. Dieses Verzeichnis wird bei der Update-Installation nicht geändert. Alle alten Konfigurationsdaten bleiben erhalten.
2. Abhängig davon, ob altes und neues Verzeichnis identisch sind, gilt Folgendes:
   – Falls das alte und das neue Verzeichnis identisch sind, dann wird die alte Management Console umbenannt, eine neue Management Console installiert und die Daten von der alten Management Console übernommen.
   – Falls das alte und das neue Verzeichnis verschieden sind, dann wird eine neue Management Console installiert. Anschließend werden die Daten von der alten Management Console übernommen.
3. Die alten Konfigurationsdateien werden aus `<console_name>.save` ausgelesen und in das Home-Verzeichnis der neuen Management Console kopiert.

3.6.2 Update-Installation auf Linux-Systemen

In diesem Abschnitt erfahren Sie, wie Sie die Installation des Proxy-Containers und der Management Console auf Linux-Systemen aktualisieren. Wenn Sie neue Software erhalten, starten Sie die Master-Installation und installieren Sie die neuen Produkte (siehe Abschnitt „BeanConnect auf Linux-Systemen installieren“ auf Seite 59).

Update-Installation starten

Die Update-Installation erfolgt auf dieselbe Weise wie eine Neuinstallation.

Wechseln Sie in das gewünschte installationsverzeichnis und starten Sie die Installationsprozedur mit folgendem Befehl:

```
<BC_inst_dir>/shsc/install.BeanConnect
```

Wie bei einer Neuinstallation zeigt die Installationsprozedur ein Menü mit den beiden Komponenten an, die installiert werden können. Wählen Sie die Komponenten aus, für die Sie ein Update installieren möchten:

```
Konfigurationsoptionen

1 BeanConnect Proxy Container
2 Management Console (Administration Tool)
```

Die Installationsprozedur erkennt es, wenn Sie ein Home-Verzeichnis angeben, das bereits eine der Komponenten enthält, und prüft, ob diese Version aktualisiert werden kann. Ist ein Update möglich, werden Sie gefragt, ob Sie eine Update-Installation durchführen oder die bestehende Version überschreiben möchten (Neuinstallation).

Für die ausgewählten Komponenten wird eine Update-Installation ausgeführt.

BeanConnect Proxy-Container

1. Das Home-Verzeichnis des Proxy-Containers wird kopiert und in das Verzeichnis `<BC_home>/<proxy_cont_name>.save` gespeichert. Dieses Verzeichnis wird bei der Update-Installation nicht geändert. Alle alten Konfigurationsdaten bleiben erhalten.
2. Abhängig davon, ob altes und neues Verzeichnis identisch sind, gilt Folgendes:
   - Falls das alte und das neue Verzeichnis identisch sind, dann wird der alte Proxy Container umbenannt, ein neuer Proxy Container installiert und die Daten vom alten Proxy Container übernommen.
   - Falls das alte und das neue Verzeichnis verschieden sind, dann wird ein neuer Proxy Container installiert. Anschließend werden die Daten vom alten Proxy Container übernommen.
3. Die alte Konfiguration wird in den neuen Proxy-Container übernommen.

Management Console

1. Das Home-Verzeichnis der Management Console wird kopiert und in das Verzeichnis `<BC_home>/<console_name>.save` gespeichert. Dieses Verzeichnis wird bei der Update-Installation nicht geändert. Alle alten Konfigurationsdaten bleiben erhalten.
2. Abhängig davon, ob altes und neues Verzeichnis identisch sind, gilt Folgendes:
   - Falls das alte und das neue Verzeichnis identisch sind, dann wird die alte Management Console umbenannt, eine neue Management Console installiert und die Daten von der alten Management Console übernommen.
   - Falls das alte und das neue Verzeichnis verschieden sind, dann wird eine neue Management Console installiert. Anschließend werden die Daten von der alten Management Console übernommen.
3. Die neue Management Console wird in das Home-Verzeichnis der Management Console installiert.
4. Die alten Konfigurationsdateien werden aus `<console_name>.save` ausgelesen und in das Home-Verzeichnis der neuen Management Console kopiert.

3.6.3 Update-Installation auf Windows-Systemen

In diesem Abschnitt erfahren Sie, wie Sie die Installation des Proxy-Containers und der Management Console auf Windows-Systemen aktualisieren. Wenn Sie neue Software erhalten, starten Sie die Master-Installation und installieren Sie die neuen Produkte (siehe Abschnitt „BeanConnect auf Windows-Systemen installieren“ auf Seite 67).

Update-Installation starten

Starten Sie die Installation in der üblichen Weise (siehe <Color>Abschnitt <$elemparanumonly<$elemtext). Klicken Sie im Setup-Menü auf den Eintrag **BeanConnect Proxy**.

Im Folgenden werden nur die Dialogfelder erläutert, die sich von denen bei der Neuinstallation unterscheiden. Bei allen anderen Dialogfeldern gehen Sie auf dieselbe Weise vor wie bei einer Neuinstallation (siehe Abschnitt „BeanConnect auf Windows-Systemen installieren“ auf Seite 67).

- Dialogfeld **Installation Option**

  Wenn Sie eine vorhandene BeanConnect-Installation aktualisieren möchten, wählen Sie die Option **Update installation**.

  Wenn Sie eine vorhandene BeanConnect-Installation aktualisieren möchten, ohne die bestehende Installation zu deinstallieren, wählen Sie die Option **Update installation without deinstallation**.

- Dialogfeld **Existing Proxy Container Directory to update**.

  Geben Sie das Home-Verzeichnis des Proxy-Containers an, das aktualisiert werden soll.

- Dialogfeld **Existing BeanConnect Management Console Directory to update**

  Geben Sie das Home-Verzeichnis der Management Console an, das aktualisiert werden soll.

  Die anschließende Vorgehensweise ist im Abschnitt „BeanConnect installieren“ auf Seite 68 erläutert.

Die Update-Installation wird für die BeanConnect-Komponenten durchgeführt, die Sie im Dialogfeld **Configuration Options** ausgewählt haben (siehe „Installationsprozedur“ auf Seite 69).

Update-Installation BeanConnect Proxy-Container

1. Das Home-Verzeichnis des Proxy-Containers wird kopiert und in das Verzeichnis `<BC_home>\<proxy_cont_name>.save` gespeichert. Dieses Verzeichnis wird bei der Update-Installation nicht geändert. Alle alten Konfigurationsdaten bleiben erhalten.
2. Der bestehende Proxy-Container wird auf Wunsch deinstalliert.
3. Im Home-Verzeichnis des Proxy-Containers wird ein neuer Proxy-Container installiert.
4. Die alte Konfiguration wird in den neuen Proxy-Container übernommen.

Update-Installation Management Console

1. Das Home-Verzeichnis der Management Console wird kopiert und in das Verzeichnis `<BC_home>\<console_name>.save` gespeichert. Dieses Verzeichnis wird bei der Update-Installation nicht geändert. Alle alten Konfigurationsdaten bleiben erhalten.
2. Die bestehende Management Console wird auf Wunsch deinstalliert.
3. Die neue Management Console wird in das Home-Verzeichnis der Management Console installiert.
4. Die alten Konfigurationsdateien werden aus `<console_name>.save` ausgelesen und in das Home-Verzeichnis der neuen Management Console kopiert.

3.7 BeanConnect deinstallieren

Dieser Abschnitt erläutert, wie Sie BeanConnect deinstallieren:

- BeanConnect auf Solaris-Systemen deinstallieren
- BeanConnect auf Linux-Systemen deinstallieren
- BeanConnect auf Windows-Systemen deinstallieren

3.7.1 BeanConnect auf Solaris-Systemen deinstallieren

Dieser Abschnitt erläutert, wie Sie BeanConnect auf Solaris-Systemen deinstallieren.

BeanConnect Proxy-Container deinstallieren

Den Proxy-Container deinstallieren Sie, indem Sie das Home-Verzeichnis des Proxy-Containers löschen.

BeanConnect Management Console deinstallieren

Die Management Console deinstallieren Sie, indem Sie das Home-Verzeichnis der Management Console löschen.

Produktdateien von BeanConnect deinstallieren

Deinstallieren Sie die BeanConnect Produktdateien mit dem folgenden Befehl:

```
pkgrm BC30B00
```

PCMX deinstallieren

Deinstallieren Sie das mit ausgelieferte PCMX mit dem folgenden Befehl:

```
pkgrm SMAWpcmx
```

openUTM deinstallieren

Deinstallieren Sie openUTM mit dem folgenden Befehl: `pkgrm UTM<version>`

openUTM-LU62 Gateway deinstallieren (für CICS-Partner)

Deinstallieren Sie das openUTM-LU62 Gateway mit dem folgenden Befehl:

```
pkgrm SMAWutm6s
```

3.7.2 BeanConnect auf Linux-Systemen deinstallieren

Dieser Abschnitt erläutert, wie Sie BeanConnect auf Linux-Systemen deinstallieren.

BeanConnect Proxy-Container deinstallieren

Den Proxy-Container deinstallieren Sie, indem Sie das Home-Verzeichnis des Proxy-Containers löschen.

BeanConnect Management Console deinstallieren

Die Management Console deinstallieren Sie, indem Sie das Home-Verzeichnis der Management Console löschen.

Produktdateien von BeanConnect deinstallieren

Deinstallieren Sie die Produktdateien von BeanConnect mit dem folgenden Befehl:

```
rpm -e BC30B00
```

PCMX deinstallieren

Deinstallieren Sie PCMX mit dem folgenden Befehl:

```
rpm -e --nodeps PCMX-<version>
```

Den genauen Paketnamen erhalten Sie, indem Sie `rpm -qa | grep PCMX` eingeben.

openUTM deinstallieren

Deinstallieren Sie openUTM mit dem folgenden Befehl: `rpm -e UTM<version>`

openUTM-LU62 Gateway deinstallieren

Deinstallieren Sie das openUTM-LU62 Gateway mit dem folgenden Befehl:

```
rpm -e UTMLU62-<version>
```

3.7.3 BeanConnect auf Windows-Systemen deinstallieren

Dieser Abschnitt erläutert, wie Sie BeanConnect auf Windows-Systemen deinstallieren.

BeanConnect Proxy-Container deinstallieren

Wählen Sie folgenden Befehl, um den Proxy-Container zu deinstallieren:

Start - Programme - FUJITSU Software BeanConnect V3.0B00 - Proxy <proxy_cont_name> - Uninstall

Alternativ können Sie auch folgende Befehle verwenden:

1. Starten Sie das Deinstallationsprogramm über den Pfad **Start - Einstellungen - Systemsteuerung - Software**.
2. Wählen Sie den Eintrag **FUJITSU Software BeanConnect V3.0B00 <proxy_cont_name>** und klicken Sie auf die Schaltfläche **Entfernen**.

Bei der Deinstallation des Proxy-Containers werden folgende Elemente gelöscht:

- Die Verzeichnisse: `<BC_home>\<proxy_cont_name>`, `<BC_home>\<proxy_cont_name>.SAVE`
- Das Verzeichnis `<BC_home>\MCINFO`, sofern es leer ist
- Die Produktdateien werden **nicht** gelöscht (`<BC_home>\lib`, `<BC_home>\Docs`)

BeanConnect Management Console deinstallieren

Wählen Sie folgenden Befehl, um die Management Console zu deinstallieren:

Start - Programme - FUJITSU Software BeanConnect V3.0B00 - Management Console - Uninstall

Alternativ können Sie auch folgende Befehle verwenden:

1. Starten Sie das Deinstallationsprogramm über den Pfad **Start - Einstellungen - Systemsteuerung - Software**.
2. Wählen Sie den Eintrag **FUJITSU Software BeanConnect V3.0B00 Management Console** und klicken Sie auf die Schaltfläche **Entfernen**.

Bei der Deinstallation der Management Console werden folgende Elemente gelöscht:

- Die Verzeichnisse: `<BC_home>\<console_name>`, `<BC_home>\<console_name>.SAVE`
- Die Produktdateien werden **nicht** gelöscht (`<BC_home>\lib`, `<BC_home>\Docs`)

Common Resources von BeanConnect deinstallieren

Wählen Sie folgenden Befehl, um die Common Resources zu deinstallieren:

Start - Programme - FUJITSU Software BeanConnect V3.0B00 - Uninstall Common Resources

Alternativ können Sie auch folgende Befehle verwenden:

1. Starten Sie das Deinstallationsprogramm über den Pfad **Start - Einstellungen - Systemsteuerung - Software**.
2. Wählen Sie den Eintrag **FUJITSU Software BeanConnect V3.0B00 Common Resources** und klicken Sie auf die Schaltfläche **Entfernen**.

Bei der Deinstallation der Produktdateien werden folgende Elemente gelöscht:

- Das Verzeichnis `<BC_home>\lib`, `<BC_home>\Docs`
- Das Verzeichnis `<BC_home>\MCINFO`, sofern es leer ist

Wenn Sie die Common Resources von BeanConnect deinstallieren, funktionieren die Proxy-Container und die Management Console, die in demselben Verzeichnis `<BC_home>` installiert sind, nicht mehr.

PCMX deinstallieren

1. Starten Sie das Deinstallationsprogramm über den Pfad **Start - Einstellungen - Systemsteuerung - Software**.
2. Wählen Sie **PCMX-32 <version>** und klicken Sie auf die Schaltfläche **Entfernen**.

openUTM deinstallieren

Wählen Sie folgenden Befehl, um openUTM zu deinstallieren:

Start - Programme - FUJITSU Software openUTM-Server - Uninstall openUTM-Server

Alternativ können Sie auch folgende Befehle verwenden:

1. Starten Sie das Deinstallationsprogramm über den Pfad **Start - Einstellungen - Systemsteuerung - Software**.
2. Wählen Sie den Eintrag **FUJITSU Software openUTM-Server <version>** und klicken Sie auf die Schaltfläche **Entfernen**.

openUTM-LU62-Gateway deinstallieren (für CICS-Partner)

Sie deinstallieren Sie das openUTM-LU62 Gateway wie folgt:

Start - Programme - openUTM-LU62 - Uninstall openUTM-LU62

Alternativ können Sie auch folgende Befehle verwenden:

1. Starten Sie das Deinstallationsprogramm über den Pfad **Start - Einstellungen - Systemsteuerung - Software**.
2. Wählen Sie den Eintrag **openUTM-LU62 <version>** und klicken Sie auf die Schaltfläche **Entfernen**.

3.8 BeanConnect Resource Adapter deinstallieren

Sie deinstallieren den Resource Adapter, indem Sie die kopierten oder extrahierten Dateien und Verzeichnisse löschen, die bei der Installation des Resource Adapters aus der JAR-Datei entpackt wurden.

3.9 BeanConnect Tools deinstallieren

Sie deinstallieren ein BeanConnect Tool, indem Sie das Installationsverzeichnis des Tools löschen.

Besonderheiten sind nur beim Tool MC-CmdHandler zu beachten. Falls der MC-CmdHandler als Dienst konfiguriert ist (siehe Abschnitt „MC-CmdHandler als Dienst konfigurieren“ auf Seite 253), dann müssen Sie diesen vorher entfernen. Dazu gehen so vor:

- Windows-Systeme:

  Rufen Sie das Skript `MCCmdHandler_UnInstSrv.cmd` auf.

- Linux- und Solaris-Systeme:

  Wenn Sie einen einzelnen Dienst deinstallieren möchten, dann löschen Sie die entsprechende Zeile aus der Datei `/etc/init.d/bcmccmdhandler.dat`.

  Wenn Sie den Dienst als Ganzes deinstallieren möchten, dann müssen Sie die Dateien `/etc/init.d/bcmccmdhandler.sh` (inkl. symbolische Links) und `/etc/init.d/bcmccmdhandler.dat` löschen.

  Zu diesen Tätigkeiten benötigen Sie Systemverwalterrechte. Daher müssen Sie ggf. den Systemverwalter beauftragen.

4 Konfiguration im Application Server

Sowohl beim Deployment des BeanConnect Resource Adapters im Application Server als auch beim Deployment der Enterprise Java Beans (EJB) im Application Server müssen Einstellungen für BeanConnect konfiguriert werden.

Dieses Kapitel enthält Informationen zum Konfigurieren der Outbound-Kommunikation über die verschiedenen Protokolle (OSI TP für openUTM-Partner, LU6.2-Protokoll für CICS-Partner, UPIC-Protokoll für openUTM-Partner) und der Inbound-Kommunikation.

Dieses Kapitel beschreibt außerdem, wie Sie das Logging im Resource Adapter vorbereiten und was Sie beim Betrieb mit Multi-Resource Adapter oder im Cluster beachten müssen.

Es enthält Informationen zu folgenden Themen:

- Überblick
- Allgemeine Eigenschaften des Resource Adapters konfigurieren
- Kommunikation für Outbound über OSI TP / LU6.2 konfigurieren
- Kommunikation für Outbound über UPIC konfigurieren
- Kommunikation für Inbound konfigurieren
- Logging für den Resource Adapter vorbereiten
- Besonderheiten im Multi-Resource Adapter Betrieb
- Besonderheiten im Cluster-Betrieb

4.1 Überblick

Der Resource Adapter wird als so genanntes BeanConnect-RAR-Archiv ausgeliefert.

Es darf nur ein BeanConnect Resource Adapter pro Instanz im Application Server deployt werden.

Wenn man mehrere Resource Adapter deployt, kann es zu unvorhersehbaren Fehlern kommen.

Das BeanConnect-RAR-Archiv für JCA 1.6 hat den Namen `BC30B00.rar`, das BeanConnect-RAR-Archiv für JCA 1.5 den Namen `BC30B004JCA15.rar`. Beide Archive enthalten u.a. den Deployment Descriptor `ra.xml`.

4.1.1 Konfigurationsdateien im Application Server

Die Konfigurationseigenschaften des Resource Adapters und der EJBs werden in folgenden Dateien definiert:

- `ra.xml`

  Standard Deployment Descriptor für den Resource Adapter.

  Diese Datei ist im BeanConnect-RAR-Archiv enthalten und definiert die allgemeinen Eigenschaften des Resource Adapters.

- `weblogic-ra.xml`

  WebLogic-spezifischer Deployment Descriptor für den Resource Adapter.

  Diese Datei beschreibt die WebLogic-spezifischen Einstellungen für den Resource Adapter. Zum Teil beziehen sich die Einträge in `weblogic-ra.xml` auf Einträge in `ra.xml`.

- `ejb-jar.xml`

  Standard Deployment Descriptor für EJBs.

  Diese Datei beschreibt u.a. Eigenschaften der EJBs, die für die Kommunikation relevant sind. Die Einträge verweisen auf Einträge in `weblogic-ra.xml` und in `weblogic-ejb-jar.xml`.

- `weblogic-ejb-jar.xml`

  WebLogic-spezifischer Deployment Descriptor für EJBs.

  Diese Datei benötigen Sie bei Inbound-Kommunikation, um ein Message-Driven Bean einem Resource Adapter zuzuordnen, und bei Outbound-Kommunikation, um den verwendeten ConnectionFactories einen JNDI-Namen zuzuordnen (JNDI = Java Naming and Directory Interface).

Die Einträge verweisen auf Einträge in `weblogic-ra.xml`.

Die Eigenschaften der EJBs können - anstatt in Deployment Descriptor Dateien - teilweise auch durch Annotations beschrieben werden. Im Folgenden wird nur die Beschreibung der Eigenschaften der EJBs mit Deployment Descriptor dargestellt.

Der folgende Abschnitt beschreibt, wie Sie bei der Konfiguration vorgehen können.

4.1.2 Konfigurationsschritte für Outbound- und Inbound-Kommunikation

Die Vorgehensweise hängt davon ab, ob Sie Outbound-Kommunikation über OSI TP / LU6.2, Outbound-Kommunikation über UPIC und/oder Inbound-Kommunikation betreiben möchten. Diese Möglichkeiten werden im Folgenden einzeln dargestellt, wobei für den Multi-Resource Adapter Betrieb und Cluster-Betrieb besondere Regeln gelten.

Für Outbound-Kommunikation (OSI TP / LU6.2 und UPIC) müssen Sie in der ejb-jar.xml Resource Referenzen festlegen.

Vorgehen bei Outbound-Kommunikation über OSI TP / LU6.2

Wenn Sie BeanConnect für die Outbound-Kommunikation über OSI TP / LU6.2 im Standardmodus nutzen möchten (ein Proxy, ein Resource Adapter, ohne Cluster), dann sind folgende Tätigkeiten notwendig:

- Allgemeine Eigenschaften in ra.xml festlegen
- Allgemeine Eigenschaften des Resource Adapters in weblogic-ra.xml festlegen
  und
  Allgemeine und verbindungsspezifische Eigenschaften für OSI TP / LU6.2 in weblogic-ra.xml definieren.
- Die Datei `weblogic-ra.xml` vor dem Deployen in das Unterverzeichnis META-INF des BeanConnect RAR-Archivs einfügen.
- Deployment des Resource Adapters
- Enterprise Java Beans für OSI TP / LU6.2 deployen
- Logging für den Resource Adapter vorbereiten

Vorgehen bei Outbound-Kommunikation über UPIC

Wenn Sie BeanConnect für die Kommunikation über UPIC nutzen möchten, dann sind folgende Tätigkeiten notwendig:

- Allgemeine Eigenschaften des Resource Adapters in weblogic-ra.xml festlegen
  und
  Allgemeine verbindungsspezifische Eigenschaften für UPIC in weblogic-ra.xml definieren.
- Die Datei `weblogic-ra.xml` vor dem Deployen in das Unterverzeichnis META-INF des BeanConnect RAR-Archivs einfügen.
- Deployment des Resource Adapters
- Logging für den Resource Adapter vorbereiten
- Logging für den Resource Adapter vorbereiten

Vorgehen bei Inbound-Kommunikation

Wenn Sie BeanConnect für die Inbound-Kommunikation im Standardmodus nutzen möchten (ein Proxy, ein Resource Adapter, ohne Cluster), dann sind folgende Tätigkeiten notwendig:

- Allgemeine Eigenschaften in ra.xml festlegen
- Allgemeine Eigenschaften des Resource Adapters in weblogic-ra.xml festlegen
- Die Datei `weblogic-ra.xml` vor dem Deployen in das Unterverzeichnis META-INF des BeanConnect RAR-Archivs einfügen.
- Deployment des Resource Adapters
- Kommunikation für Inbound konfigurieren
- Logging für den Resource Adapter vorbereiten

Vorgehen bei Multi-Resource Adapter und im Cluster

Beim Betrieb mit Multi-Resource Adapter oder im Cluster sind zusätzliche Einstellungen in der Datei `ra.xml` notwendig. Details dazu finden Sie in:

- Besonderheiten im Multi-Resource Adapter Betrieb
- Besonderheiten im Cluster-Betrieb

4.2 Allgemeine Eigenschaften des Resource Adapters konfigurieren

Die allgemeinen Eigenschaften sind im Standard Deployment Descriptor `ra.xml` und im WebLogic-spezifischen Deployment Descriptors `weblogic-ra.xml` beschrieben. Der Standard Deployment Descriptor des Resource Adapters `ra.xml` ist im BeanConnect-RAR-Archiv enthalten.

Die Datei `ra.xml` enthält:

- die allgemeinen Konfigurations-Properties für die Verbindung des Resource Adapters mit dem Proxy
- die Definition der von BeanConnect unterstützten ConnectionFactories
- und die von diesen ConnectionFactories unterstützten Eigenschaften einschließlich deren Standardwerte

Sie kann als Vorlage für die Konfiguration der Verbindung betrachtet werden.

Die Datei `weblogic-ra.xml` enthält:

- die WebLogic spezifischen Konfigurations-Properties für den Resource Adapter
- die WebLogic spezifischen Konfigurations-Properties für die ConnectionFactories

Bevor Sie das BeanConnect-RAR-Archiv deployen, müssen Sie die allgemeinen Konfigurations-Properties in der Datei `ra.xml` und `weblogic-ra.xml` an Ihre Gegebenheiten anpassen.

4.2.1 Allgemeine Eigenschaften in ra.xml festlegen

Die allgemeinen Konfigurations-Properties sind für die Outbound-Kommmunikation über OSI TP / LU6.2 und für die Inbound-Kommunikation relevant.

Sie können den Resource Adapter auch mit den Voreinstellungen deployen. Spezifizieren Sie die Werte anschließend über die grafische Web-Oberfläche des Application Servers. Bei einem Undeployment können diese Änderungen jedoch verloren gehen.

ra.xml anpassen

Die Datei `ra.xml` können Sie wie folgt anpassen:

- Über die BeanConnect Management Console, sofern sich BeanConnect-RAR-Archiv und Management Console auf demselben Rechner befinden oder auf dem Rechner des BeanConnect-RAR-Archivs ein MC-CmdHandler installiert ist und dieser in der Management Console konfiguriert ist.

  Dabei wird der Deployment Descriptor direkt im RAR-Archiv geändert. Sie können die Werte der Properties dabei entweder editieren (Befehl **Edit ra.xml of BeanConnect Resource Adapter RAR…**) oder anhand der Werte aktualisieren, die für einen definierten Proxy konfiguriert wurden (Befehl **Update ra.xml of BeanConnect Resource Adapter RAR**). Das Aktualisieren hat den Vorteil, dass die Management Console den richtigen Wert für die Property `proxyURL` ermittelt.

  Näheres siehe Abschnitt „BeanConnect Resource Adapter konfigurieren“ auf Seite 202.

- Manuell mit Hilfe eines Texteditors:

  1. Extrahieren Sie die Datei `ra.xml` aus dem BeanConnect-RAR-Archiv:

     ```
     jar xf BC30B00.rar META-INF/ra.xml
     ```

  2. Ändern Sie in der Datei `ra.xml` mit einem beliebigen Texteditor die folgenden allgemeinen Konfigurations-Properties:

     proxyURL (Outbound und Verfügbarkeitsprüfung Proxy)

     transactionLogging (nur Outbound)

     transactionLogDir (nur Outbound)

     inboundListenerPort (Inbound und Verfügbarkeitsprüfung Resource Adapter)

     revisionNumber

  3. Fügen Sie die Datei `ra.xml` wieder in das BeanConnect-RAR-Archiv in das Unterverzeichnis `META-INF` ein:

     ```
     jar uf BC30B00.rar META-INF/ra.xml
     ```

Bei mehreren Resource Adaptern oder im Cluster müssen Sie zusätzliche Einstellungen vornehmen, siehe Abschnitt „Besonderheiten im Multi-Resource Adapter Betrieb“ auf Seite 151 und Abschnitt „Besonderheiten im Cluster-Betrieb“ auf Seite 153.

proxyURL

Die `proxyURL` legt die Zuordnung des Resource Adapters zum Proxy fest.

`proxyURL` wird global für alle Verbindungen definiert.

Definition: `oltp://<host>:<port>/<name>`

Erklärung:

`<host>`	Rechner, auf dem der Proxy-Container installiert ist. `<host>` kann als symbolischer Name oder als IPv4-Adresse angegeben werden.
`<port>`	Portnummer des Proxy-Containers +4
`<name>`	Anwendungsname des Proxy-Containers (BCU<port>)

Standard: `oltp://localhost:31004/BCU31004`

Beispiel:

```
<config-property>
  <description>BeanConnect Proxy URL for OLTP outbound
        communication</description>
  <config-property-name>proxyURL</config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>oltp://proxyhost:31004/BCU31004
  </config-property-value>
</config-property>
```

Wenn Sie die Funktion **Update ra.xml of BeanConnect Resource Adapter RAR** der Management Console verwenden, so werden die richtigen Werte für `proxyURL` gesetzt.

Wenn Sie `ra.xml` manuell editieren, dann müssen Sie den Wert für `<port>` aus dem bei der Installation des Proxys angegebenen Wert ermitteln (Port +4). Den Wert für `<name>` müssen Sie zusammensetzen aus dem Präfix BCU und `<port>`.

Die Management Console zeigt die zu verwendende Proxy-URL im Eigenschaftsdialogfeld des Resource Adapters an. Die Proxy-URL kann nach der Installation nicht geändert werden.

Für die Konfiguration in einem Cluster siehe Abschnitt „Besonderheiten im Cluster-Betrieb“ auf Seite 153.

transactionLogging

Mit diesem Attribut wird festgelegt, ob BeanConnect bei der Outbound-Kommunikation für Transaktionen mit EIS Partnern persistente Transaktions-Logs schreiben soll oder nicht.

Definition: `[NONE | FILE]`

Erklärung: Aktiviert oder deaktiviert das persistente Transaktions-Logging

`NONE`: Es werden keine persistenten Transaktions-Logs geschrieben.

`FILE`: Es werden persistente Transaktions-Logs geschrieben. Das Verzeichnis, in das die Logs geschrieben werden, wird im Attribut `transactionLogDir` angegeben.

Standard: `NONE`

Beispiel:

```
<config-property>
  <description>BeanConnect transaction logging:
      possible values are NONE | FILE
  </description>
  <config-property-name>transactionLogging
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>FILE
  </config-property-value>
</config-property>
```

Persistente Transaktions-Logs ermöglichen es, dass BeanConnect bei der Transaktions-Recovery nach einem Programm- oder Systemabsturz Auskunft über den Status derjenigen Transaktionen geben kann, die zum Zeitpunkt des Absturzes in Bearbeitung waren. Das Einschalten dieser Option wirkt sich auf die Performance aus, denn es gibt dadurch für jede Transaktion, die mit two-phase-commit beendet wird, zwei Dateizugriffe.

Wird `FILE` angegeben, dann muss auch für das Attribut `transactionLogDir` ein Wert angegeben werden.

Bei konfiguriertem Transaktions-Logging schreibt der Resource Adapter für jede Transaktion eine eigene Log-Datei. Der Dateiname setzt sich zusammen aus dem Prefix `tx.` und einer Nummer.

Eine Transaktions-Log Datei wird bei Prepare geschrieben und bei Commit oder Rollback gelöscht, d.h. sie ist im Normalfall temporär. Es gibt jedoch Situationen, in denen sie erhalten bleibt:

- Wenn der Resource Adapter zwischen Prepare und Rollback oder Commit beendet wird. Eine solche Log-Datei bleibt bis zum Abschluss einer Recovery für diese Transaktion erhalten.
- Wenn für eine Transaktion eine heuristische Entscheidung getroffen wurde. Eine solche Log-Datei bleibt unbegrenzt lange erhalten.

Beim Start des Resource Adapters werden alle Transaktions-Log-Dateien eingelesen. Für Transaktionen, die sich nach dem Start im Zustand Prepared befinden oder für die eine heuristische Entscheidung getroffen wurde, werden neue Transaktions-Log-Dateien geschrieben.

Für Transaktionen mit heuristischen Entscheidungen bleiben die Transaktions-Log Dateien unbegrenzt lange erhalten. Daher sollten diese von Zeit zu Zeit gelöscht werden. Als Kriterium dafür, welche Log-Dateien gelöscht werden können, kann die Erzeugungszeit einer Log-Datei dienen. Dateien, deren Erzeugungsdatum und -uhrzeit mit dem letzten Start des Resource Adapters übereinstimmen, enthalten heuristische Logs und der Application Server hat für diese Transaktionen nicht forget() aufgerufen; diese Dateien können gelöscht werden. Zur Kennzeichnung des Anwendungsstarts schreibt BeanConnect nach dem Bearbeiten der Transaktions-Log Dateien in der Startphase eine Datei mit Namen `tx.startup-complete.<date>.<time>` in das Transaktions-Log-Verzeichnis.

Alte `startup-complete`-Dateien werden von BeanConnect beim nächsten Start gelöscht.

transactionLogDir

Mit diesem Attribut wird das Verzeichnis festgelegt, in das BeanConnect persistente Transaktions-Logs schreiben soll. Diesem Attribut muss ein Wert zugewiesen werden, wenn dem Attribut `transactionLogging` der Wert `FILE` zugeordnet ist.

Definition: Name eines Verzeichnisses

Standard: `persistence\BeanConnect`

Beispiel:

```
<config-property>
  <description>Directory where transaction log files are to be
               stored (only needed if transactionLogging=FILE)
  </description>
  <config-property-name>transactionLogDir
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>persistence/BeanConnect
  </config-property-value>
</config-property>
```

Das Verzeichnis kann mit absolutem oder mit relativen Pfadnamen angegeben werden. Eine relative Pfadangabe ist dabei relativ zu dem Home-Directory des Application Servers.

Wenn das bei `transactionLogDir` angegebene Directory nicht existiert, dann wird es von BeanConnect neu angelegt.

inboundListenerPort

Bei Inbound-Kommunikation wartet der Resource Adapter an einem Socket-Port auf Verbindungsaufträge, die vom Proxy initiiert werden. Die Portnummer dieses Socket-Ports geben Sie mit der globalen Konfigurations-Property `inboundListenerPort` an. Dieser Wert muss vor dem Deployment des Resource Adapters angepasst werden.

Definition: <portnummer>

Erklärung: Portnummer des Socket-Ports, an dem der Resource Adapter auf Aufträge für Inbound-Kommunikation wartet.
0 bedeutet, dass keine Inbound-Kommunikation möglich ist.

Standard: 31099

Beispiel:

```
<config-property>
  <description>Resource Adapter Listener Port for Inbound
     Communication
  </description>
  <config-property-name>inboundListenerPort
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>31099
  </config-property-value>
</config-property>
```

Auch wenn keine Inbound-Funktionalität verwendet wird, sollten Sie hier einen Wert angeben, wenn Sie die Verfügbarkeit des Resource Adapters aus Sicht des Proxys überprüfen wollen (z.B. mit Hilfe der Management Console). Nur wenn Sie auch diese Funktionalität nicht benötigen, sollten Sie hier den Wert 0 angeben.

Die Portnummer, die mit `inboundListenerPort` definiert wird, muss mit der Listener-Portnummer übereinstimmen, die Sie über die Management-Console bei dem Aufnehmen des Resource Adapters zu einem Proxy angegeben haben.

revisionNumber

Dieses Attribut bezeichnet den Änderungsstand der Datei `ra.xml`. Den Wert des Attributs `revisionNumber` sollten Sie bei jeder Änderung der Datei `ra.xml` erhöhen, um eventuelle Inkonsistenzen in der Konfiguration leichter erkennen zu können.

Beispiel:

```
<config-property>
  <description>Revision number of the ra.xml. This number
    should be incremented with each change of the resource
    adapter properties.
  </description>
  <config-property-name>revisionNumber
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>2
  </config-property-value>
</config-property>
```

4.2.2 Allgemeine Eigenschaften des Resource Adapters in weblogic-ra.xml festlegen

In der Deployment Descriptor Datei `weblogic-ra.xml` sind die WebLogic-spezifischen Eigenschaften des Resource Adapters beschrieben.

Folgende allgemeine Konfigurations-Properties sind relevant:

- Mit dem Element `<jndi-name>` wird der JNDI-Name des Resource Adapters festgelegt, wie er bei Inbound-Kommunikation in der Datei `weblogic-ejb-jar.xml` im Element `<resource-adapter-jndi-name>` für jedes OLTP Message-Driven Bean angegeben werden muss.
- Die Property `<enable-global-access-to-classes>` muss auf `true` gesetzt werden, um Fehler beim Deployment der Anwendungen zu vermeiden.
- Im Element `<connector-work-manager>` kann über das Tag `<max-concurrent-long-running-requests>` die Anzahl der lang laufenden Work- Instanzen festgelegt werden. Der Default-Wert beträgt 10.

  Setzen Sie diese Anzahl um eins höher als die Anzahl der Workprozesse des BeanConnect Proxy-Containers. Beachten Sie, dass Sie in einer Cluster-Konfiguration die Workprozesse aller Proxy-Container des Clusters berücksichtigen müssen.
- Im Block `<security>` können Sie im Element `<security-work-context>` mit dem Tag `<inbound-mapping-required>` auswählen, ob sie Fall 1 oder Fall 2 der in der JCA-Spezifikation beschriebenen Arten des Security Inflow nutzen möchten.

  Der Default-Wert ist `false` und steht für Fall 1. In diesem Fall muss jede Benutzerkennung, die vom EIS in den Application Server propagiert werden soll, auch im Application Server konfiguriert werden. Die zu den Benutzerkennungen festgelegten Passwörter im Application Server werden für die Inbound-Kommunikation nicht validiert.

  Im Fall 2 müssen Sie im Element `<security-work-context>` eine Abbildung der EIS Benutzerkennung auf eine Benutzerkennung des Application Server konfigurieren.

  Siehe auch „Konfigurations-Properties in der Datei weblogic-ra.xml“ auf Seite 107.

4.2.3 Deployment und Undeployment des Resource Adapters

Voraussetzung

Sie sollten den Resource Adapter erst deployen, nachdem Sie die allgemeinen Konfigurationseinstellungen in die Datei `ra.xml` und die WebLogic-spezifischen Konfigurationseinstellungen in die Datei `weblogic-ra.xml` eingetragen haben. Siehe auch

- Abschnitt „Allgemeine Eigenschaften in ra.xml festlegen“ auf Seite 96,
- Allgemeine Eigenschaften des Resource Adapters in weblogic-ra.xml festlegen,
- Allgemeine und verbindungsspezifische Eigenschaften für OSI TP / LU6.2 in weblogic-ra.xml definieren und
- Allgemeine verbindungsspezifische Eigenschaften für UPIC in weblogic-ra.xml definieren.

4.2.3.1 Deployment des Resource Adapters

Für das Deployment von Anwendungen unter Oracle WebLogic Server können Sie abhängig vom Startup-Modus des Servers folgende Methoden verwenden:

- Production-Modus als Startup-Modus

  Das automatische Deployment ist generell nicht aktiviert.

  Es werden folgende drei Deployment-Möglichkeiten angeboten:

  - Deployment im Browser mit der WebLogic Server Administration Console
  - Deployment mit dem `weblogic.Deployer` Tool
  - Deployment mit dem Oracle WebLogic Scripting Tool

- Development-Modus als Startup-Modus

  Die WebLogic Server-Instanzen können Anwendungen automatisch deployen und aktualisieren, wenn diese Dateien im Verzeichnis `domain_name/autodeploy` liegen. Oracle empfiehlt, diese Methode für Anwendungen nur in "single-server"-Konfigurationen zu nutzen.

  Diese Methode wird für den BeanConnect Resource Adapter generell nicht empfohlen.

  Für den BeanConnect Resource Adapter wird empfohlen, auch im Development-Modus eine der Deployment-Methoden zu verwenden, die auch im Production-Modus möglich sind, siehe unten.

Beispiel:

Um den Resource Adapter im Browser mit der WebLogic Server Administration Console zu deployen, gehen Sie wie folgt vor:

1. Erstellen Sie die Datei `weblogic-ra.xml`.
2. `weblogic-ra.xml`zum Unterverzeichnis `META-INF` des BeanConnect RAR-Archivs hinzufügen: `jar uf BC30B00.rar META-INF/ weblogic-ra.xml`
3. Web-Browser starten, bei WebLogic Server Administration Console anmelden und folgende Schritte durchführen (abweichende englische Oberflächenbegriffe in Klammern):
   a) Nach `<domainName>>Deployments` navigieren und die Schaltfläche `Installieren` (`install`) klicken
   b) Zum Verzeichnis navigieren, unter dem das BeanConnect RAR-Archiv steht, das Archiv auswählen und `<Weiter>` (`<Next>`) klicken.

      Die WebLogic Server Administration Console führt weitere Schritte durch.
   c) Empfehlung: Als "Deployment-Order" einen Wert kleiner als 100 (Standardwert) wählen, z.B. `98`.
      Eine Anwendung, die vom Resource Adapter abhängt, kann dadurch später per Autodeployment hinzugefügt werden.

Beachten Sie für die beiden anderen Deployment-Möglichkeiten (weblogic.Deployer Tool und WebLogic Scripting Tool) die Hinweise in der Dokumentation zum
Oracle WebLogic Server unter "Deploying Applications and Modules with weblogic.Deployer" bzw. "Oracle WebLogic Scripting Tool".

4.2.3.2 Update-Deployment des Resource Adapters

Vor einem Update-Deployment oder erneuten Deployment müssen alle OLTP Message-Driven Bean Anwendungen, die den Resource Adapter referenzieren, gestoppt werden.

- Um das Update-Deployment vorzunehmen, navigieren Sie in der WebLogic Server Administration Console nach `<domainName>>Deployments`, wählen den installierten BeanConnect Resource Adapter aus und klicken die Schaltfläche `<Updaten>`.
- Nach dem Update-Deployment können die OLTP Message-Driven Bean Anwendungen wieder gestartet werden.

4.2.3.3 Undeployment des Resource Adapters

Vor einem Undeployment müssen alle OLTP Message-Driven Bean Anwendungen, die den Resource Adapter referenzieren, gestoppt werden.

Um das Undeployment vorzunehmen, navigieren Sie in der WebLogic Server Administration Console nach `<domainName>>Deployments`, wählen den installierten BeanConnect Resource Adapter aus und klicken die Schaltfläche `<Löschen>` (`<delete>`).

4.2.4 Beispiel ra.xml

Beispiel 3 zeigt die Definition der allgemeinen Konfigurations-Properties in der Datei `ra.xml`.

Beispiel 3 Konfigurations-Properties in der Datei ra.xml

Der Abschnitt mit den allgemeinen Konfigurations-Properties in der Datei `ra.xml` hat folgende Struktur:

```
...
<resourceadapter>

<resourceadapter-class>
  net.fsc.jca.BeanConnect.ResourceAdapterJBImpl
</resourceadapter-class>
<config-property>
  <description>Resource Adapter Listener Port for Inbound Communication
  </description>
  <config-property-name>inboundListenerPort
  </config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>31099</config-property-value>
</config-property>
<config-property>
  <description>BeanConnect Proxy URL for OLTP Outbound
```

```
  Communication</description>
  <config-property-name>proxyURL</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value>oltp://localhost:31004/BCU31004</config-property-
value>
</config-property>
<config-property>
  <description>BeanConnect transaction logging: possible values are NONE |
FILE</description>
<!-- If NONE is set, no persistent transaction logging will be performed.
If FILE is set, transaction log files will be written to the directory
specified in property
transactionLogDir. -->
<config-property-name>transactionLogging</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>FILE</config-property-value>
</config-property>
<config-property>
<description>Directory where transaction log files are to be stored (only
needed if
transactionLogging=FILE)</description>
<!-- The path name of the dircetory may be specified as absolute or relative
path.
A relative path is relative to the home directory of the application server.
-->
<config-property-name>transactionLogDir</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>persistence/BeanConnect</config-property-value>
</config-property>
<config-property>
  <description>Revision number of the ra.xml. This number should be
incremented with each     change of the resource adapter properties.
  </description>
  <config-property-name>revisionNumber
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>2
  </config-property-value>
</config-property>
```

4.2.5 Beispiel weblogic-ra.xml

Beispiel 4 zeigt die Definition der allgemeinen Konfigurations-Properties in der Datei `weblogic-ra.xml`.

Beispiel 4 Konfigurations-Properties in der Datei weblogic-ra.xml

Der Abschnitt mit den allgemeinen Konfigurations-Properties in der Datei `weblogic-ra.xml` hat folgende Struktur:

```
<weblogic-connector
   xmlns="http://xmlns.oracle.com/weblogic/weblogic-connector"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-connector
   http://www.oracle.com/technology/weblogic/weblogic-connector/1.3/
   weblogic-connector.xsd">
   <jndi-name>BeanConnect</jndi-name>
   <enable-global-access-to-classes>true</enable-global-access-to-classes>   (1)
   <work-manager>
     <name>BeanConnect_WorkManager</name>
   </work-manager>
   <connector-work-manager>
     <!-- For inbound communication the BeanConnect resource adapter starts a
long-running
          work instance, which listens at the inbound listener port.
          Each BeanConnect proxy work process establishs a
          connection to the BeanConnect resource adapter. For each
          connection the BeanConnect resource adapter starts a long-running
          work instance.
          The maximum number of concurrent long-running Work instances will
          be specified with the following property.

         Default value: 10
     -->
     <max-concurrent-long-running-requests>12
     </max-concurrent-long-running-requests>
   </connector-work-manager>
<security>
   <security-work-context>
       <!-- Two choices related to establishing the caller identity for a
            work instance are described in
            JSR 322: Java EE Connector Architecture 1.6
            The default value is false, which means Case 1.
            All caller-principal-mapping and group-principal-mapping
            subelements are ignored.
            If set to true, it means Case 2.
            All caller-principal-mapping and group-principal-mapping
            elements are used to determine the correct mapping from the
            EIS security domain to the WebLogic domain.
```

```
        -->
        <inbound-mapping-required>false</inbound-mapping-required>
      <caller-principal-default-mapped>
         <!-- Used for Inbound Message Endpoint invocation if inbound mapping
              required and a user is inflowed who has no explicit mapped
              caller principal  -->
         <!--
           <use-anonymous-identity>true</use-anonymous-identity>
         -->
         <principal-name>BcDefaultMappedPrincipal1</principal-name>
       </caller-principal-default-mapped>
       <caller-principal-mapping>
         <eis-caller-principal>EISUSER1</eis-caller-principal>
         <mapped-caller-principal>
           <!--
             <use-anonymous-identity>true</use-anonymous-identity>
           -->
           <principal-name>AsPrincipal1</principal-name>
         </mapped-caller-principal>
       </caller-principal-mapping>
       <caller-principal-mapping>
         <eis-caller-principal>EISUSER2</eis-caller-principal>
         <mapped-caller-principal>
           <principal-name>AsPrincipal2</principal-name>
         </mapped-caller-principal>
       </caller-principal-mapping>
       <caller-principal-mapping>
         <eis-caller-principal>EISUSER3</eis-caller-principal>
         <mapped-caller-principal>
           <principal-name>AsPrincipal3</principal-name>
         </mapped-caller-principal>
       </caller-principal-mapping>
       <caller-principal-mapping>
         <eis-caller-principal>EISUSER4</eis-caller-principal>
         <mapped-caller-principal>
           <principal-name>AsPrincipal4</principal-name>
         </mapped-caller-principal>
       </caller-principal-mapping>
    </security-work-context>
</security>
</weblogic-connector>
```

(1) Diese Property ist für einen ordnungsgemäßen Ablauf unbedingt notwendig!

4.3 Kommunikation für Outbound über OSI TP / LU6.2 konfigurieren

Für Outbound-Kommunikation müssen folgende Eigenschaften konfiguriert sein:

- Die allgemeinen Konfigurations-Properties in der Datei `ra.xml` (siehe Seite 96) und der Datei `weblogic-ra.xml` (siehe Seite 102).
- Die verbindungsspezifischen Eigenschaften für OSI TP / LU6.2 in der Datei `weblogic-ra.xml` (siehe Seite 102).

Connection Factories

Die Outbound-Kommunikation wird über sogenannte Connection Factories abgewickelt.

Sie müssen pro EIS Partner mindestens eine Connection Factory konfigurieren. Es ist jedoch auch möglich, für einen EIS Partner mehrere Connection Factories mit unterschiedlichen Eigenschaften zu definieren.

Für jede benötigte Connection Factory legen Sie Folgendes fest:

- JNDI-Namen: Dieser wird auch beim Deployen der EJB benötigt und dient dazu, die Connection Factory im EJB anzusprechen.
- Konfigurations-Properties: Dies sind die Parameter, die für die Verbindung zu dem EIS Partner gelten.
- Connection-Pooling: Connection-Pooling ist eine Funktionalität des Application Servers. Sie dient dazu, die Performance bei häufig genutzten Connections zu verbessern.
- Sicherheitseinstellungen: Dies sind die Daten, die für das Anmelden beim EIS Partner benötigt werden, z.B. Benutzerkennwort und Passwort. Wenn diese Daten direkt in der EJB codiert sind, kann der entsprechende Eintrag im Deployment Descriptor entfallen.
- In der Management Console wird jeder Connection Factory ein Outbound Communication Endpoint zugeordnet.

4.3.1 Allgemeine und verbindungsspezifische Eigenschaften für OSI TP / LU6.2 in weblogic-ra.xml definieren

Die Datei `weblogic-ra.xml` ist der WebLogic-spezifische Deployment Descriptor für den Resource Adapter. Dieser Deployment Descriptor legt die Einstellungen fest, die für die Outbound-Kommunikation mit den EIS Partnern benötigt werden.

Allgemeine Einstellungen für Outbound-Kommunikation

- Die Einstellungen für Outbound-Kommunikation werden in dem Element `<outbound-resource-adapter>` festgelegt.
- Eigenschaften für Connection Factories können mit den Sub-Elementen `<default-connection-properties>` und `<connection-definition-group>` bestimmt werden.
- Für jede `<connection-definition>`, die in der `ra.xml` enthalten ist, muss in der `weblogic-ra.xml` eine `<connection-definition-group>` definiert werden.
- Standardeinstellungen, die für alle Connection Factories gelten sollen, können Sie im Sub-Element `<default-connection-properties>` des Elements `<outbound-resource-adapter>` festlegen.
- Standardeinstellungen, die für alle Connection Factories einer `<connection-definition-group>` gelten sollen, können Sie im Sub-Element `<default-connection-properties>` des Elements `<connection-definition-group>` festlegen.
- Für jede von den EJB zu verwendende Connection Factory muss in der korrespondierenden `<connection-definition-group>` eine `<connection-instance>` beschrieben werden.

4.3.1.1 Resource für OSI TP / LU6.2 festlegen

Der Resource-Typ einer Connection Factory wird im Tag `<connection-factory-interface>` des Sub-Elements `<connection-definition-group>` festgelegt.

Für die Kommunikation über OSI TP / LU6.2 sind die folgenden Resource-Typen relevant:

- `net.fsc.jca.communication.cci.BCOltpConnectionFactory`
- `net.fsc.jca.communication.EISOltpConnectionFactory`

Für die EJB, die diese Connection Factory verwendet, muss der passende Typ in den Deployment Descriptoren `ejb-jar.xml` und `weblogic-ra.xml` angegeben werden, siehe Abschnitt „Enterprise Java Beans für OSI TP / LU6.2 deployen“ auf Seite 123.

Ein Beispiel für die Datei `weblogic-ra.xml` finden Sie in Abschnitt „Beispiel: weblogic-ra.xml“ auf Seite 120.

4.3.1.2 JNDI-Namen für OSI TP / LU6.2 festlegen

Der JNDI-Name der Connection Factory wird im Sub-Element `<connection-instance>` einer `<connection-definition-group>` mit dem Tag `<jndi-name>` festgelegt.

Eine Enterprise Java Bean (EJB) im Application Server spricht eine Connection Factory über JNDI an. Die Connection Factory ordnet der Connection die konfigurierten Properties zu.

Ein Beispiel finden Sie in Abschnitt „Beispiel: weblogic-ra.xml“ auf Seite 120.

4.3.1.3 Konfigurations-Properties für OSI TP / LU6.2 definieren

Die Konfigurationseigenschaften einer ConnectionFactory werden über das Sub-Element `<connection-properties>` im Element `<connection-instance>` definiert. Diese Änderungen können Sie auch über die grafische Oberfläche der WebLogic Server Administration Console vornehmen.

BeanConnect unterstützt die folgenden verbindungsspezifischen Konfigurations-Properties für Outbound-Kommunikation über OSI TP / LU6.2:

- `bufferedIO`
- `connectionURL`
- `displayName`
- `encoding`
- `encodingActive`
- `logLevel`
- `timeout`
- `transactional`

Die Verbindung, die ein `ConnectionFactory.getConnection()`-Aufruf liefert, wird mit diesen Konfigurations-Properties initialisiert.

bufferedIO

Mit der Konfigurations-Property `bufferedIO` definieren Sie, ob eine IO-Pufferung zwischen Resource Adapter und Proxy stattfinden soll. Ist diese Property aktiviert, wird die Interaktion zwischen Resource Adapter und Proxy auf ein Minimum reduziert. Um in einer Produktionsumgebung eine maximale Leistung zu erreichen, sollten Sie diese Property auf `true` setzen. Sie können diesen Wert beim Deployment oder im Betrieb über die ConnectionFactory MBean auf `false` setzen, um im Test Benutzerfehler so schnell wie möglich zu erkennen.

Definition: `[true | false]`

Erklärung: Aktiviert oder deaktiviert IO-Pufferung zwischen Resource Adapter und Proxy.

`true`	IO-Pufferung wird verwendet.
`false`	IO-Pufferung wird nicht verwendet.

Standard: `true`

Beispiel:

```
<property>
   <name>bufferedIO</name>
   <value>true</value>
</property>
```

connectionURL

Für die Kommunikation über OSI TP / LU6.2 gibt die Konfigurations-Property `connectionURL` den Namen des Outbound Communication Endpoints an, der die Verbindung zu einem EIS Partner repräsentiert und mit Hilfe der Management Console im Proxy definiert wurde. Der Name beginnt mit einem Präfix, der den Typ des EIS Partners beschreibt, z.B. `utm` oder `cics`. Der in BeanConnect bis V2.0 verwendete Name `oltp` wird aus Kompatibilitätsgründen weiterhin unterstützt und synonym zu `utm` verwendet.

Der Name des Outbound Communication Endpoints ist eine frei wählbare Zeichenfolge. Er muss mit dem entsprechenden Namen des Outbound Communication Endpoints übereinstimmen, der im Proxy konfiguriert wurde, siehe Abschnitt „Outbound-Kommunikation konfigurieren“ auf Seite 235.

Definition: `<type>://<name>`

Erklärung:

`<type>`	Typ des EIS Partners, mögliche Werte:	
	`utm`	der EIS Partner ist vom Typ openUTM.
	`cics`	der EIS Partner ist vom Typ CICS.
	`xatmi-rr`	der EIS Partner ist vom Typ XATMI und die Kommunikation wird nach dem Request/Reply-Paradigma durchgeführt.
	`xatmi-cv`	der EIS Partner ist vom Typ XATMI und die Kommunikation wird nach dem Conversational-Paradigma durchgeführt.
`<name>`	Name eines Outbound Communication Endpoints, wie er in der Management Console definiert wurde.	

Standard: `utm://outboundCommunicationEndpoint`

Beispiel:

```
<property>
   <name>connectionURL</name>
   <value>utm://HELLO</value>
</property>
```

displayName

Mit diesem Attribut kann für eine ManagedConnectionFactory ein Name vergeben werden, der von BeanConnect bei der Ausgabe von Informationen zu dieser ManagedConnectionFactory verwendet werden soll, z.B. bei der Ausgabe von MBeans und von LogWriter-Sätzen.

Definition: `[<name>]`

Erklärung: Frei wählbarer Name einer ManagedConnectionFactory, wie er z.B. in MBean- und LogWriter-Ausgaben verwendet werden soll.

Standard: Kein Standardwert.
Wenn Sie keinen Namen angeben, dann wird der interne Name der ManagedConnectionFactory verwendet. Dieser besteht aus dem Präfix "MCF" und einer 5-stelligen Zahl.

Beispiel:

```
<property>
   <name>displayName</name>
   <value>sample application/test</value>
</property>
```

encoding

Die Konfigurations-Property `encoding` definiert eine Code-Tabelle für die Konvertierung von Byte-Code (zum Beispiel EBCDIC) in Unicode. Diese Code-Tabelle wird für die Konvertierung von Byte-Streams in Strings und umgekehrt verwendet. Diese Konvertierungen werden immer implizit aufgerufen, wenn Interaktionen (z.B. `sndString()`, `rcvString()`) ausgeführt werden, die Strings als I/O-Parameter enthalten.

Die mit der Konfigurations-Property `encoding` definierte Code-Tabelle wird als Standard für die entsprechende Verbindung verwendet. Der Bean-Programmierer kann eine andere Code-Tabelle für die Verbindung bestimmen, indem er die Methode `setEncoding(Encoding)` des Interfaces `EISConnection` oder `OltpMessageContext` explizit aktiviert (siehe Abschnitt „Zeichensatz-Konvertierung“ auf Seite 489).

Die Code-Konvertierung mit dieser Code-Tabelle wird nur ausgeführt, wenn `encodingActive` aktiviert ist. Dies können Sie über die Konfigurations-Property `encodingActive` einstellen oder indem Sie die Methode `setEncodingActive(true)` anwenden.

Die Definitionen für openUTM-Partner und CICS-Partner unterscheiden sich.

Für openUTM-Partner gilt:

Definition:
```
[<builtin_encoding_table> |
builtin:<builtin_encoding_table> |
jdk:<jdk_encoding_table> |
custom:<encoding_table>]
```

Erklärung: Name einer Code-Tabelle, die für die Code-Konvertierung verwendet wird.
Ist kein Präfix oder ist das Präfix `builtin:` angegeben, müssen Sie den Namen einer internen Code-Tabelle angeben, die von BeanConnect bereitgestellt wird.
Folgende interne Code-Tabellen stehen zur Verfügung:
`OSD_EBCDIC_DF03_IRV`, `OSD_EBCDIC_DF04_1`, `OSD_EBCDIC_DF04_15`, `OSD_EBCDIC_DF04_DRV`
Mit dem Präfix `jdk:` geben Sie eine Code-Tabelle an, die im JDK enthalten ist.
Mit dem Präfix `custom:` weisen Sie Ihre eigene Code-Tabelle zu. Dabei müssen Sie den vollständigen Klassennamen der Code-Tabelle angeben. Weitere Einzelheiten zur Verwendung von eigenen Code-Tabellen finden Sie in der JavaDoc zu BeanConnect.

Standard: `default`
Dieser Wert wird für openUTM-Partner auf "`OSD_EBCDIC_DF04_DRV`" gesetzt.

Beispiel:
```
<property>
   <name>encoding</name>
   <value>OSD_EBCDIC_DF03_IRV</value>
</property>
```

Für CICS-Partner gilt:

Definition:
```
[jdk:<jdk_encoding_table> |
custom:<encoding_table>]
```

Erklärung: Name einer Code-Tabelle, die für die Code-Konvertierung verwendet wird.
Mit dem Präfix `jdk:` geben Sie eine Code-Tabelle an, die im JDK enthalten ist.
Mit dem Präfix `custom:` weisen Sie Ihre eigene Code-Tabelle zu. Dabei müssen Sie den vollständigen Klassennamen der Code-Tabelle angeben. Weitere Einzelheiten zur Verwendung von eigenen Code-Tabellen finden Sie in der JavaDoc zu BeanConnect.

Standard: `jdk:Cp1047`

Beispiel:
```
<property>
   <name>encoding</name>
   <value>jdk:Cp1250</value>
</property>
```

encodingActive

Die Konfigurations-Property `encodingActive` gibt an, ob die Code-Konvertierung aktiviert werden soll oder nicht. Sie wird direkt auf die Methode `setEncodingActive(boolean activate)` des Interfaces `EISConnection` abgebildet.

Definition: `[true | false]`

Erklärung: Gibt an, ob die Code-Konvertierung aktiviert wird oder nicht.

`true`	Die Code-Konvertierung gemäß der Einstellungen der Konfigurations-Property `encoding` ist aktiviert.
`false`	Die Standard-Code-Tabelle des JDK wird für die Konvertierung von Byte-Streams in Strings verwendet.

Standard: `false`

Beispiel:

```
<property>
   <name>encodingActive</name>
   <value>true</value>
</property>
```

Die Deployment-Einstellungen können mit der Methode `setEncodingActive()`, die im Interface `EISConnection` definiert ist, überschrieben werden.

logLevel

Mit diesem Attribut kann für eine Connection Factory der Level für die Ausgabe von Log-Sätzen auf einen LogWriter eingestellt werden. Ein LogWriter für eine Connection Factory wird je nach dem verwendeten Application Server auf unterschiedliche Weise konfiguriert. Näheres zur Konfiguration und zu den Ausgaben auf LogWriter siehe Abschnitt „Logwriter für Connection Factory“ auf Seite 532.

Definition: `[NONE | ERROR | INFO | ALL]`

Erklärung:

`NONE`	Es werden keine Ausgaben auf den LogWriter geschrieben.
`ERROR`	Es werden nur Informationen zu Exceptions und zu Rollbacks von Transaktionen auf den LogWriter geschrieben.
`INFO`	Es werden zusätzlich zu den bei `ERROR` aufgeführten Informationen alle Ereignisse protokolliert, die Transaktionen betreffen, z.B. Beginn oder Commit einer Transaktion.
`ALL`	Es werden zusätzlich zu den bei `INFO` aufgeführten Informationen alle Ereignisse protokolliert, die den Lifecycle für die Connections betreffen. Dies sind z.B. das Anfordern und Freigeben von Connection Handles durch die Anwendung oder Ereignisse, die das Pooling betreffen.

Standard: `NONE`

Beispiel:
```
<property>
   <name>logLevel</name>
   <value>INFO</value>
</property>
```

timeout

Die Konfigurations-Property `timeout` gibt an, wie lange der Resource Adapter maximal auf eine Antwort vom Proxy wartet.

Der hier angegebene Wert muss größer sein als die Zeit, die das EIS-System maximal zur Bearbeitung eines Aufrufs benötigt. Bei Ablauf des Timers wird eine Exception an die Anwendung geworfen und die Verbindung zwischen Resource Adapter und Proxy wird neu initialisiert. Dies führt im Allgemeinen zum Rücksetzen der Transaktion beim EIS Partner.

Definition: Zeit in Millisekunden.

Erklärung:

> 0	Maximale Wartezeit des Resource Adapters in Millisekunden.
0	Der Resource Adapter wartet unbegrenzt.

Standard: 300000 (entspricht 5 Minuten)

Beispiel:
```
<property>
   <name>timeout</name>
   <value>30000</value>
</property>
```

transactional

Die Konfigurations-Property `transactional` gibt an, ob die Kommunikation zwischen Application Server und EIS transaktional sein soll. In diesem Fall wird eine Transaktion des EIS in eine Transaktion des Application Servers einbezogen.

Definition: `[true | false]`

Erklärung:

`true`	Die Beteiligung an der Transaktion des Application Servers ist aktiviert.
`false`	Die Beteiligung an der Transaktion des Application Servers ist deaktiviert.

Standard: `false`

Beispiel:
```
<property>
   <name>transactional</name>
   <value>true</value>
</property>
```

4.3.1.4 Connection-Pooling für OSI TP / LU6.2 anpassen

Sie können allgemein, für jeden Resource Type oder für jede Connection Factory, in der Datei `weblogic-ra.xml` angeben, wie das Connection Pooling ausgeführt werden soll.

Das Connection Pooling dient dazu, die Leistung zu verbessern. Mit `max-capacity` wird festgelegt, wie viele Verbindungen zu einer Zeit für eine Connection Factory aktiv sein dürfen; der Standardwert ist 10. Verbindungsanforderungen, die über die hier festgelegte Anzahl hinausgehen, weist der Oracle WebLogic Server mit einer `ResourceAllocationException` ab.

Für Verbindungen, die häufig und von vielen Clients verwendet werden, sollten für `max-capacity` hohe Werte definiert werden. Für selten verwendete Verbindungen brauchen Sie gar kein Connection Pooling zu definieren. Näheres zur Konfiguration des Connection Pooling finden Sie in der Dokumentation des Application Servers.

Beispiel 5 zeigt die Definition einer Connection Factory mit Beispielwerten.

Beispiel 5 Connection Pooling

```
<connection-instance>
   <jndi-name>eis/my_EIS</jndi-name>
   <connection-properties>
      <pool-params>
         <initial-capacity>2</initial-capacity>
         <max-capacity>10</max-capacity>
      </pool-params>
       ...
   </connection-properties>
</connection-instance>
```

Weitere Pool-Parameter können Sie der Schema-Beschreibung zur Datei `weblogic-ra.xml` entnehmen.

4.3.1.5 Sicherheitseinstellungen definieren (Anmeldung verwalten)

Fordert eine EJB mit dem Aufruf `ConnectionFactory.getConnection()` eine Verbindung zum EIS an, wird diese Verbindung im abgesicherten Umfeld von BeanConnect aufgebaut. Insbesondere werden die Authentisierungsdaten (Benutzername und Passwort), die für den Zugriff der EJB auf das EIS erforderlich sind, beim Verbindungsaufbau übergeben.

EJBs haben zwei Möglichkeiten, sich beim EIS zu authentisieren:

- Von der Anwendung gesteuerte Authentisierung
- Vom Container gesteuerte Authentisierung

Es wird die containergesteuerte Authentisierung empfohlen.

Nachfolgend wird die grundsätzliche Vorgehensweise bei anwendungs- und containergesteuerter Authentisierung erläutert.

Anwendungsgesteuerte Authentisierung (application-managed)

In diesem Fall müssen die Authentisierungsdaten über den Programmcode der EJB bereitgestellt werden (siehe Kapitel „Interfaces und Programmierung“ auf Seite 439). Bei EJBs, die die Authentisierung selbst durchführen, muss das `<res-auth>`-Tag des zugehörigen EJB Deployment Descriptors wie folgt angegeben werden:

```
<res-auth>Application</res-auth>
```

Beispiel für das Setzen durch die EJB:

```
getConnection(new PasswordCredential(user, password));
```

Containergesteuerte Authentisierung (container-managed)

In diesem Fall regelt der Application Server den Transfer der Authentisierungsdaten. Bei EJBs, die dem Application Server die Authentisierung ermöglichen, muss das `<res-auth>`-Tag des zugehörigen EJB Deployment Descriptors wie folgt angegeben werden:

```
<res-auth>Container</res-auth>
```

Die Konfiguration der containergesteuerten Authentisierung ist spezifisch für den jeweiligen Application Server.

Für Oracle WebLogic Server gilt Folgendes:

- Container-Managed Sign-On im Oracle WebLogic Server basiert auf **Outbound Credential Mapping**, bei dem WebLogic Credentials (normalerweise User Name und Passwort) auf User Name und Passwort des EIS Partners abgebildet werden.
- Allgemein ist zu beachten, dass Oracle WebLogic Server Outbound Credential Mapping nur für das Default Security Realm (normalerweise "myrealm") unterstützt.
- Die Abbildung eines WebLogic User Namens auf den User Namen des EIS-Systems kann dabei entweder spezifisch für eine einzelne ManagedConnectionFactory oder aber für den gesamten Resource Adapter vorgenommen werden.
- Desweiteren ermöglicht Oracle WebLogic Server die Definition eines Default Mappings für User Namen, für die keine explizite Abbildung definiert ist, sowie das Festlegen eines User Namens für nicht-authentisierte User (Anonymous Mapping).

Oracle WebLogic Server prüft beim Outbound Credential Mapping Folgendes in der angegebenen Reihenfolge:

1. Ist für den User Namen eine Abbildung für die aktuelle ManagedConnectionFactory definiert oder ist im Falle eines nicht-authentisierten Users ein Anonymous Mapping definiert?
2. Ist für den User Namen eine Abbildung für den Resource Adapter definiert oder ist im Falle eines nicht-authentisierten Users ein Anonymous Mapping definiert?
3. Ist für die aktuelle ManagedConnectionFactory ein Default Mapping definiert?
4. Ist für den Resource Adapter ein Default Mapping definiert?

Für die erste der aufgelisteten Bedingungen, die zutrifft, wird der User Name wie definiert abgebildet.

Wenn keine der Bedingungen zutrifft, übergibt Oracle WebLogic Server keine Authentisierungsdaten an den Resource Adapter. In diesem Fall führt BeanConnect eine anwendungsgesteuerte Authentisierung durch.

Das Outbound Credential Mapping führen Sie bei Oracle WebLogic Server mittels der WebLogic Server Administration Console folgendermaßen durch:

1. Wählen Sie auf der linken Seite des Ausgabeschirms der WebLogic Server Administration Console die Option **Deployments** aus.
2. Klicken Sie auf der **Deployments**-Seite den Namen des Resource Adapters an.
3. Wählen Sie auf der **Settings**-Seite des Resource Adapters nacheinander die Registerkarten **Security** und **Outbound Credential Mapping** aus.
4. Nehmen Sie die gewünschten Abbildungen vor.

Einzelheiten zum Outbound Credential Mapping bei Oracle WebLogic Server sind in der Dokumentation des Application Servers im Abschnitt "Outbound Credential Mappings" beschrieben.

4.3.1.6 Beispiel: weblogic-ra.xml

Beispiel 6 zeigt die Definition der verbindungsspezifischen Konfigurations-Properties in der Datei `weblogic-ra.xml`.

Beispiel 6 Konfigurations-Properties in der Datei weblogic-ra.xml (OSI TP)

Der Abschnitt mit den verbindungsspezifischen Konfigurations-Properties in der Datei `weblogic-ra.xml` hat folgende Struktur:

```
<weblogic-connector
   xmlns="http://xmlns.oracle.com/weblogic/weblogic-connector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-connector
http://www.oracle.com/technology/weblogic/weblogic-connector/1.3/weblogic-
connector.xsd">   …
   <outbound-resource-adapter>
     <default-connection-properties>
       <pool-params>
         <initial-capacity>0</initial-capacity>
         <max-capacity>10</max-capacity>
         </pool-params>
         <transaction-support>XATransaction</transaction-support>
      </default-connection-properties>
      <connection-definition-group>
         <connection-factory-
interface>net.fsc.jca.communication.EISOltpConnectionFactory
          </connection-factory-interface>
        <default-connection-properties>
           <pool-params>
              <initial-capacity> [1]
              </initial-capacity>
              <max-capacity>20</max-capacity>
           </pool-params>
        </default-connection-properties>
       <connection-instance>
         <jndi-name>eis/my_BC_Factory1</jndi-name>
         <connection-properties>
           <pool-params>
             <initial-capacity>2</initial-capacity>
          </pool-params>
          <properties>
            <property>
               <name>ConnectionURL</name>
               <value>utm://accountEIS</value> [1]
             </property>
```

[1] Für CICS-Partner: <value>cics://accountEIS</value>

```
            <property>
               <name>encoding</name>
               <value>OSD_EBCDIC_DF04_DRV</value> [1]
               </property>
              <property>
                <name>encodingActive</name>
                  <value>true</value>
              </property>
             <property>
               <name>transactional</name>
               <value>true</value>
             </property>
             <property>
               <name>timeout</name>
               <value>30000</value>
             </property>
             <property>
               <name>bufferedIO</name>
               <value>true</value>
             </property>
             <property>
               <name>logLevel</name>
               <value>ALL</value>
             </property>
             <property>
               <name>displayName</name>
               <value>accountEIS</value>
             </property>
           </properties>
       </connection-properties>
    </connection-instance>
  </connection-definition-group>
  <connection-definition-group>
    <connection-factory-interface>
     net.fsc.jca.communication.cci.BCOltpConnectionFactory
    </connection-factory-interface>
    <connection-instance>
      <jndi-name>eis/my_CCI_factory</jndi-name>
      <connection-properties>
        <pool-params>
          <initial-capacity>3</initial-capacity>
        </pool-params>
        <properties>
          <property>
            <name>ConnectionURL</name>
            <value>utm://accountEIS"</value> [1]
```

[1] Für CICS-Partner: <value>jdk:Cp1047</value>

```
            </property>
            <property>
              <name>encoding</name>
              <value>OSD_EBCDIC_DF04_DRV</value> [2]
            </property>
            <property>
              <name>encodingActive</name>
              <value>true</value>
            </property>
           <property>
              <name>transactional</name>
              <value>true</value>
            </property>
            <property>
              <name>timeout</name>
              <value>30000</value>
            </property>
            <property>
              <name>bufferedIO</name>
              <value>true</value>
            </property>
            <property>
              <name>logLevel</name>
              <value>NONE</value>
            </property>
            <property>
              <name>displayName</name>
              <value>cci/accountEIS</value>
            </property>
          </properties>
        </connection-properties>
      </connection-instance>
    </connection-definition-group>
  </outbound-resource-adapter>
</weblogic-connector>
```

4.3.2 Enterprise Java Beans für OSI TP / LU6.2 deployen

Beim Deployment einer EJB, die BeanConnect für Outbound-Kommunikation nutzen soll, müssen Sie die EJB mit dem BeanConnect Deployment verknüpfen. Die folgenden Dateien sind für das Deployment einer EJB relevant:

- Code-Datei der EJB (`.java`- oder `.class`-Datei)
- Standardisierter Deployment Descriptor der EJB (`ejb-jar.xml`) oder Java Annotations
- Application Server spezifischer Deployment Descriptor der EJB (bei Oracle WebLogic Server: `weblogic-ejb-jar.xml`)
- Application Server spezifischer Deployment Descriptor für den Resource Adapter (bei Oracle WebLogic Server: `weblogic-ra.xml`)

Beim Deployment einer EJB wird die vom Bean-Entwickler benutzte Ressourcenreferenz dem Application Server über den Deployment Descriptor der EJB bekannt gemacht. Zusätzlich wird der Ressourcenreferenz ein Ressourcentyp zugewiesen.

BeanConnect unterstützt die folgenden Ressourcentypen, welche die unterschiedlichen Verbindungsarten repräsentieren, die verwendet werden können:

- Bei OSI TP / LU6.2-Kommunikation über das BeanConnect-Interface:

  `net.fsc.jca.communication.EISOltpConnectionFactory`

- Bei OSI TP / LU6.2-Kommunikation über das CCI-Interface:

  `net.fsc.jca.communication.cci.BCOltpConnectionFactory`

Der Ressourcentyp muss in folgenden Dateien angegeben werden:

- `ejb-jar.xml` mit dem Tag `<res-type>`
- `weblogic-ra.xml` mit dem Tag `<connection-factory-interface>`

Die Abschnitte der Code-Datei der EJB sowie die der Dateien `ejb-jar.xml`, `weblogic-ejb-jar.xml` und `weblogic-ra.xml`, die für das Deployment der EJB relevant sind, werden nachfolgend ausführlich beschrieben. Die **fettgedruckten** (Teil-)Pfadnamen geben die Beziehungen zwischen den einzelnen Dateien an.

- Code-Datei der EJB (`.java`- oder `.class`-Datei)

  Hier findet der JNDI-Lookup für das `ConnectionFactory`-Objekt über eine Ressourcenreferenz (kodierter Name) statt. Im folgenden Beispiel wird die Ressourcenreferenz `eis/Part1Dial` verwendet.

  ```
  ...
  cf=(EISConnectionFactory)
           ic.lookup("java:comp/env/eis/Part1Dial")
  ...
  ```

- Deployment Descriptor der EJB (`ejb-jar.xml`)

  Hier wird die Ressourcenreferenz (`ConnectionFactory`-Objekt) angegeben, auf die die EJB zugreift. Zusätzlich wird der Ressourcenreferenz ein Ressourcentyp zugewiesen. Im folgenden Beispiel wird als Ressourcentyp `net.fsc.jca.communication.EISOltpConnectionFactory` verwendet.

```
<session>
  <ejb-name>SimpleBeanConnect</ejb-name>
  ...
  <resource-ref>
    <res-ref-name>eis/Part1Dial</res-ref-name>
    <res-type>
      net.fsc.jca.communication.EISOltpConnectionFactory
    </res-type>
    <res-sharing-scope>Unshareable</res-sharing-scope>
   ...
  </resource-ref>
</session>
```

  Beachten Sie, dass für `<res-sharing-scope>` immer `Unshareable` angegeben werden muss.

- Application Server spezifischer Deployment Descriptor der EJB (bei Oracle WebLogic Server `weblogic-ejb-jar.xml`):

  Hier werden dem EJB-Namen und den Ressource-Referenzen, die in der Datei `ejb-jar.xml` definiert sind, JNDI-Namen des Application Servers zugeordnet.

```
<weblogic-enterprise-bean>
      <ejb-name>SimpleBeanConnect</ejb-name>
      <jndi-name>ejb/SimpleBeanConnect</jndi-name>
     <resource-description>
         <res-ref-name>comp/env/eis/Part1Dial</res-ref-name>
         <jndi-name>java:comp/env/eis/Part1Dial</jndi-name>
      </resource-description>
   </weblogic-enterprise-bean>
```

- Deployment Descriptor für den Resource Adapter (bei Oracle WebLogic Server `weblogic-ra.xml`)

  Hier wird die Connection Factory für das Deployment in der Application Server Instanz konfiguriert und über den JNDI-Namen (hier: `partner1Dial`) mit der Ressourcenreferenz verknüpft. Über die Konfiguration der Connection Factory in der Application Server Instanz erhält der Resource Adapter die URL des Services (im folgenden Code-Fragment: `utm://echo` und des EIS Partners.

```
<outbound-resource-adapter>
   <connection-definition-group>
     <connection-factory-interface>
        net.fsc.jca.communication.EISOltpConnectionFactory
     </connection-factory-interface>
     <connection-instance>
       <jndi-name>eis/partner1Dial</jndi-name>
       <connection-properties>
       <properties>
        <property>
          <name>ConnectionURL</name>
          <value>utm://echo</value> [1]
        </property>
        ...
       </connection-properties>
     </connection-factory-interface>
    </connection-definition-group>
</outbound-resource-adapter>
```

Der für `<connection-factory-interface>` angegebene Wert muss mit dem Wert übereinstimmen, der in der Datei `ejb-jar.xml` mit dem Tag `<res-type>` angegeben wurde.

Auf dem Proxy ist ein zusätzlicher Konfigurationsschritt erforderlich. Für jeden Namen eines Outbound Communication Endpoints, der über die Konfigurations-Property `connectionURL` in der Datei `weblogic-ra.xml` angegeben ist, müssen Sie im Proxy einen entsprechenden Outbound Communication Endpoint mit demselben Namen konfigurieren. In der Definition des Outbound Communication Endpoints wird der symbolische Service-Name auf einen realen Service-Namen in der EIS Partneranwendung abgebildet. Die Konfiguration eines Outbound Communication Endpoints können Sie mit Hilfe der Management Console durchführen (siehe Abschnitt „Outbound Communication Endpoints konfigurieren“ auf Seite 238).

[1] Für CICS-Partner: <value>cics://echo</value>

4.4 Kommunikation für Outbound über UPIC konfigurieren

Für die Kommunikation über UPIC müssen Sie nur die verbindungsspezifischen Eigenschaften in der Datei `weblogic-ra.xml` festlegen, die Datei `ra.xml` ist in diesem Fall nicht relevant.

Connection Factories

Die Outbound-Kommunikation wird über so genannte Connection Factories abgewickelt.

Sie müssen pro EIS Partner mindestens eine Connection Factory konfigurieren. Es ist jedoch auch möglich, für einen EIS Partner mehrere Connection Factories mit unterschiedlichen Eigenschaften zu definieren.

Für jede benötigte Connection Factory legen Sie Folgendes fest:

- JNDI-Name: Dieser wird auch beim Deployen der EJB benötigt und dient dazu, die Connection Factory im EJB anzusprechen.
- Konfigurations-Properties: Dies sind die Parameter, die für die Verbindung zu dem EIS Partner gelten.
- Connection-Pooling: Connection-Pooling ist eine Funktionalität des Application Servers. Sie dient dazu, die Performance bei häufig genutzten Connections zu verbessern.
- Sicherheitseinstellungen: Dies sind die Daten, die für das Anmelden beim EIS Partner benötigt werden, z.B. Benutzerkennwort und Passwort. Wenn diese Daten direkt in der EJB codiert sind, kann der entsprechende Eintrag im Deployment Descriptor entfallen.

4.4.1 Allgemeine verbindungsspezifische Eigenschaften für UPIC in weblogic-ra.xml definieren

Die Datei `weblogic-ra.xml` ist der WebLogic-spezifische Deployment Descriptor für den Resource Adapter. Dieser Deployment Descriptor legt die Einstellungen fest, die für die Outbound-Kommunikation mit den EIS Partnern benötigt werden.

Allgemeine Einstellungen für Outbound-Kommunikation

Die Einstellungen für Outbound-Kommunikation werden in dem Element `<outbound-resource-adapter>` festgelegt. Dabei gilt:

- Eigenschaften für Connection Factories legen Sie in den Sub-Elementen `<default-connection-properties>` und `<connection-definition-group>` fest.
- Default-Einstellungen, die für alle Connection Factories gelten sollen, legen Sie im Sub-Element `<default-connection-properties>` des Elements `<outbound-resource-adapter>` fest.
- Default-Einstellungen, für alle Connection Factories einer `<connection-definition-group>` gelten sollen, legen Sie im Sub-Element `<default-connection-properties>` des Elements `<connection-definition-group>` fest.
- Für jede von den EJB zu verwendende Connection Factory muss in der korrespondierenden `<connection-definition-group>` eine `<connection-instance>` beschrieben werden.

4.4.1.1 Resource für UPIC festlegen

Der Resource-Typ einer Connection Factory wird im Tag `<connection-factory-interface>` des Sub-Elements `<connection-definition-group>` festgelegt.

Für die Kommunikation über UPIC sind die folgenden Ressource-Typen relevant:

- `net.fsc.jca.communication.cci.BCUpicConnectionFactory`
- `net.fsc.jca.communication.EISUpicConnectionFactory`

Für das EJB, das diese Connection Factory verwendet, muss dann genau dieser Typ in den Deployment Descriptoren `ejb-jar.xml` und `weblogic-ra.xml` angegeben werden, siehe Abschnitt „Enterprise Java Beans für UPIC deployen“ auf Seite 138. Ein Beispiel finden Sie in Abschnitt „Beispiel: weblogic-ra.xml“ auf Seite 120.

4.4.1.2 JNDI-Namen für UPIC festlegen

Der JNDI-Name der Connection Factory wird im Sub-Element `<connection-instance>` einer `<connection-definition-group>` mit dem Tag `<jndi-name>` festgelegt.

Eine Enterprise Java Bean (EJB) im Application Server spricht eine Connection Factory über JNDI (Java Naming and Directory Interface) an. Die Connection Factory ordnet der Connection die konfigurierten Properties zu.

4.4.1.3 Konfigurations-Properties für UPIC definieren

Die Konfigurationseigenschaften einer ConnectionFactory werden über das Sub-Element `<connection-properties>` im Element `<connection-instance>` definiert. Diese Änderungen können Sie auch über die grafische Oberfläche der WebLogic Server Administration Console vornehmen.

BeanConnect unterstützt die folgenden verbindungsspezifischen Konfigurations-Properties für Outbound-Kommunikation über UPIC:

- `connectionURL`
- `displayName`
- `encoding`
- `encodingActive`
- `logLevel`
- `timeout`
- `reconnectThreshold`

Die Verbindung, die ein `ConnectionFactory.getConnection()`-Aufruf liefert, wird mit diesen Konfigurations-Properties vorinitialisiert.

connectionURL

Die Property `connectionURL` definiert den EIS Partner und, falls erforderlich, den Service, der angesprochen werden soll. Für die Outbound-Kommunikation über UPIC sind nur EIS Partner vom Typ openUTM zulässig.

Für eine Verbindung mit einer UTM-Cluster-Anwendung kann bei `connectionURL` eine Liste von URLs angegeben werden, in der die einzelnen URLs durch Kommas getrennt werden.

Die Methode `getConnection()` einer Connection Factory stellt eine Verbindung mit den entsprechenden Konfigurations-Properties bereit. Die Methode `getConnection()` kann sowohl vom Interface `EISConnection` als auch vom Interface `EISUpicConnection` verwendet werden.

Definition: `upic://<host>[:<port>]/[<local>:]<remote>[/<tac>] [?tsel[;tsel]]`

Erklärung:

`host`	Rechner, auf dem die openUTM-Partneranwendung ausgeführt wird.
`port`	Portnummer des Ports, an dem die openUTM-Partneranwendung lauscht (optional). Standard: 102
`local`	Lokaler Name des Clients (PTERM). Standard: `JUPIC`
`remote`	Name der openUTM-Partneranwendung (BCAMAPPL oder APPLINAME).
`tac`	TAC (ServiceName), der in der openUTM-Partneranwendung aufgerufen wird (optional). Dieser Wert kann mit Methoden überschrieben werden, die im Interface `EISConnection` definiert sind.
`tsel`	TSEL-Format-Definition für die lokale Adresse (lt) oder Remote-Adressen (rt) in der folgenden Form (optional): `[lt={a|e|t}]|[rt={a|e|t}]` a =ASCII, e = EBCDIC, t = TRANSDATA Standard: `lt=t` falls `local` keine Kleinbuchstaben enthält, `a` sonst `rt=t` falls `remote` keine Kleinbuchstaben enthält, `a` sonst

Standard: `upic://<host>:<port>/application/service`

Beispiel: EIS Partner im BS2000-System:

```
<property>
    <name>connectionURL</name>
    <value>upic://BS2HOST/UTMAPP/INFO</value>
</property>
```

EIS Partner auf Solaris-/Linux-/Windows-Systemen:

```
<property>
    <name>connectionURL</name>
    <value>upic://unixhost:24000/UTMAPP/INFO?rt=a</value>
</property>
```

displayName

Mit diesem Attribut kann für eine ManagedConnectionFactory ein Name vergeben werden, der von BeanConnect bei der Ausgabe von Informationen zu dieser ManagedConnectionFactory verwendet werden soll, z.B. bei der Ausgabe von MBeans und von LogWriter-Sätzen.

Definition: `[<name>]`

Erklärung: Frei wählbarer Name einer ManagedConnectionFactory, wie er z.B. in MBean- und LogWriter-Ausgaben verwendet werden soll.

Standard: Kein Standardwert.
Wenn Sie keinen Namen angeben, dann wird der interne Name der ManagedConnectionFactory verwendet. Dieser besteht aus dem Präfix "MCF" und einer 5-stelligen Zahl.

Beispiel:

```
<property>
    <name>displayName</name>
    <value>sample application/test</value>
</property>
```

encoding

Die Konfigurations-Property `encoding` definiert eine Code-Tabelle für die Konvertierung von Byte-Code (zum Beispiel EBCDIC) in Unicode. Diese Code-Tabelle wird für die Konvertierung von Byte-Streams in Strings und umgekehrt verwendet. Diese Konvertierungen werden immer implizit aufgerufen, wenn Interaktionen (z.B. `sndString()`, `rcvString()`) ausgeführt werden, die Strings als I/O-Parameter enthalten.

Die mit der Konfigurations-Property `encoding` definierte Code-Tabelle wird als Standard für die entsprechende Verbindung verwendet. Der Bean-Programmierer kann eine andere Code-Tabelle für die Verbindung bestimmen, indem er die Methode `setEncoding(Encoding)` des Interfaces `EISConnection` explizit aktiviert (siehe Abschnitt „Zeichensatz-Konvertierung“ auf Seite 489).

Die Code-Konvertierung mit dieser Code-Tabelle wird nur ausgeführt, wenn `encodingActive` aktiviert ist. Dies können Sie über die Konfigurations-Property `encodingActive` einstellen oder indem Sie die Methode `setEncodingActive(true)` anwenden.

Definition:

```
[<builtin_encoding_table>|
builtin:<builtin_encoding_table>|
jdk:<jdk_encoding_table>|
custom:<encoding_table>]
```

Erklärung: Name einer Code-Tabelle, die für die Code-Konvertierung verwendet wird.
Ist kein Präfix oder ist das Präfix `builtin:` angegeben, müssen Sie den Namen einer internen Code-Tabelle angeben, die von BeanConnect bereitgestellt wird.
Folgende interne Code-Tabellen stehen zur Verfügung:
`OSD_EBCDIC_DF03_IRV`, `OSD_EBCDIC_DF04_1`, `OSD_EBCDIC_DF04_15`, `OSD_EBCDIC_DF04_DRV`
Mit dem Präfix `jdk:` geben Sie eine Code-Tabelle an, die im JDK enthalten ist.
Mit dem Präfix `custom:` weisen Sie Ihre eigene Code-Tabelle zu. Dabei müssen Sie den vollständigen Klassennamen der Code-Tabelle angeben. Weitere Einzelheiten zur Verwendung von eigenen Code-Tabellen finden Sie in der JavaDoc zu BeanConnect.

Standard: `OSD_EBCDIC_DF04_DRV`

Beispiel:
```
<property>
    <name>encoding</name>
    <value>OSD_EBCDIC_DF03_IRV</value>
</property>
```

encodingActive

Die Konfigurations-Property `encodingActive` gibt an, ob die Code-Konvertierung aktiviert werden soll oder nicht. Sie wird direkt auf die Methode `setEncodingActive(boolean activate)` des Interfaces `EISConnection` abgebildet.

Definition: `[true | false]`

Erklärung: Gibt an, ob die Code-Konvertierung aktiviert wird oder nicht.

`true`	Die Code-Konvertierung gemäß der Einstellungen der Konfigurations-property `encoding` ist aktiviert.
`false`	Die Standard-Code-Tabelle des JDK wird für die Konvertierung von Byte-Streams in Strings verwendet.

Standard: `false`

Beispiel:
```
<property>
    <name>encodingActive</name>
    <value>true</value>
</property>
```

Die Deployment-Einstellungen können mit der Methode `setEncodingActive()`, die im Interface `EISConnection` definiert ist, überschrieben werden. Es ist auch möglich, an diesem Interface kundenspezifische Klassen für die Code-Konvertierung (`setEncoding(Encoding)`) zu aktivieren.

logLevel

Mit diesem Attribut kann für eine Connection Factory der Level für die Ausgabe von Log-Sätzen auf einen LogWriter eingestellt werden. Ein LogWriter für eine Connection Factory wird je nach dem verwendeten Application Server auf unterschiedliche Weise konfiguriert. Näheres zur Konfiguration und zu den Ausgaben auf LogWriter siehe Abschnitt „Logwriter für Connection Factory“ auf Seite 532.

Definition: `[NONE | ERROR | INFO | ALL]`

Erklärung:

`NONE`	Es werden keine Ausgaben auf den LogWriter geschrieben.
`ERROR`	Es werden nur Informationen zu Exceptions auf den LogWriter geschrieben.
`INFO`	Gleiche Ausgaben wie bei `ERROR`.
`ALL`	Es werden zusätzlich zu den bei `INFO/ERROR` aufgeführten Informationen auch alle Ereignisse protokolliert, die den Lifecycle für die Connections betreffen. Dies sind z.B. das Anfordern und Freigeben von Connection Handles durch die Anwendung oder Ereignisse, die das Pooling betreffen.

Standard: `NONE`

Beispiel:

```
<property>
    <name>logLevel</name>
    <value>INFO</value>
</property>
```

timeout

Die Konfigurations-Property `timeout` definiert die maximale Wartezeit für `receive()`- oder `call()`-Aufrufe. Diese Property wird direkt auf den Socket-Timeout der Java-Socket-Implementierung abgebildet.

Der hier angegebene Wert muss größer sein als die Zeit, die das EIS-System maximal zur Bearbeitung eines Aufrufs benötigt. Bei Ablauf des Timers wird eine Exception an die Anwendung geworfen und die Verbindung zwischen Resource Adapter und EIS wird neu initialisiert.

Definition: Wert für Zeitüberschreitung (in Millisekunden)

Standard: 30000 (entspricht 30 Sekunden)

Beispiel:

```
<property>
    <name>timeout</name>
    <value>30000</value>
</property>
```

reconnectThreshold

Die Konfigurations-Property `reconnectThreshold` legt fest, wie oft eine (physikalische) Verbindung benutzt werden darf (Anzahl `getConnection()`-Aufrufe), bevor sie durch BeanConnect abgebaut und neu aufgebaut werden soll. Ein solcher erzwungener Verbindungsabbau kann sinnvoll sein, wenn der EIS Partner eine Cluster-Anwendung ist und von Zeit zu Zeit eine Neuverteilung der Verbindungen auf die Cluster-Knoten erfolgen soll.

Der Wert 0 bedeutet, dass BeanConnect für diese Verbindungen keinen erzwungenen Verbindungsabbau durchführt.

Definition: Obergrenze für die Anzahl Nutzungen einer physikalischen Verbindung

Standard: 0 d.h. kein erzwungener Verbindungsabau durch BeanConnect

Beispiel:

```
<property>
    <name>reconnectThreshold</name>
    <value>10</value>
</property>
```

4.4.1.4 Connection-Pooling für UPIC anpassen

Sie können allgemein für jeden Resource Type oder für jede Connection Factory in der Datei `weblogic-ra.xml` angeben, wie das Connection Pooling ausgeführt werden soll.

Das Connection Pooling dient dazu die Leistung zu verbessern. Mit `max-capacity` wird festgelegt, wie viele Verbindungen zu einer Zeit für eine Connection Factory aktiv sein dürfen; der Standardwert ist 10. Verbindungsanforderungen, die über die hier festgelegte Anzahl hinausgehen, weist der Oracle WebLogic Server mit einer `ResourceAllocationException` ab.

Für Verbindungen, die häufig und von vielen Clients verwendet werden, sollten für `max-capacity` hohe Werte definiert werden. Für selten verwendete Verbindungen brauchen Sie gar kein Connection Pooling zu definieren. Näheres zur Konfiguration des Connection Pooling finden Sie in der Dokumentation des Application Servers.

Im Beispiel 7 finden Sie die Definition einer Connection Factory mit Beispielwerten:

Beispiel 7 Connection Pooling

```
<connection-instance>
      <jndi-name>eis/my_EIS</jndi-name>
      <connection-properties>
          <pool-params>
             <initial-capacity>2</initial-capacity>
             <max-capacity>10</max-capacity>
          </pool-params>
           …
     </connection-properties>
</connection-instance>
```

Weitere Pool-Parameter können Sie der Schema-Beschreibung zu der Datei `weblogic-ra.xml` entnehmen.

4.4.1.5 Transaction Support für UPIC festlegen

Für den BeanConnect Resource Adapter ist in der Datei `weblogic-ra.xml` als Transaction Support Level der Wert `XATransaction` eingestellt. Da die Kommunikation über UPIC-Verbindungen nicht in eine verteilte Transaktion aufgenommen werden kann, wird bei der Nutzung von Oracle WebLogic Server als Application Server empfohlen, dieses Attribut für die Connection Factories für UPIC-Verbindungen in der Datei `weblogic-ra.xml` zu überschreiben.

transaction-support

Der Transaction Support Level einer Connection Factory wird in der Datei `weblogic-ra.xml` im Sub-Element `<transaction-support>` einer `<connection-instance>` festgelegt. Für UPIC-Verbindungen sollte hier der Wert `NoTransaction` angegeben werden.

Beispiel: `<transaction-support>NoTransaction</transaction-support>`

4.4.1.6 Beispiel: weblogic-ra.xml (UPIC)

Beispiel 8 zeigt die Definition der verbindungsspezifischen Konfigurations-Properties in der Datei `weblogic-ra.xml`.

Beispiel 8 Konfigurations-Properties in der Datei weblogic-ra.xml

Der Abschnitt mit den Konfigurations-Properties in der Datei `weblogic-ra.xml` hat folgende Struktur:

```
<weblogic-connector
   xmlns="http://xmlns.oracle.com/weblogic/weblogic-connector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-connector
http://www.oracle.com/technology/weblogic/weblogic-connector/1.3/weblogic-
connector.xsd">
   …
   <outbound-resource-adapter>
     <default-connection-properties>
       <pool-params>
         <initial-capacity>1</initial-capacity
        <max-capacity>15</max-capacity>
       </pool-params>
         <transaction-support>NoTransaction</transaction-support>
     </default-connection-properties>
     <connection-definition-group>
       <connection-factory-interface>
         net.fsc.jca.communication.EISUpicConnectionFactory
       </connection-factory-interface>
       <default-connection-properties>
         <pool-params>
           <initial-capacity>2</initial-capacity>
           <max-capacity>100</max-capacity>
         </pool-params>
       </default-connection-properties>
       <connection-instance>
         <jndi-name>eis/my_BC_factory</jndi-name>
         <connection-properties>
           <pool-params>
             <initial-capacity>5</initial-capacity>
           </pool-params>
           <properties>
             <property>
               <name>ConnectionURL</name>
               <value>value="upic://BS2HOST/UTMAPP/INFO"</value>
              </property>
```

```
          <property>
            <name>encoding</name>
            <value>OSD_EBCDIC_DF04_DRV</value>
          </property>
          <property>
            <name>encodingActive</name>
            <value>true</value>
          </property>
          <property>
            <name>timeout</name>
            <value>30000</value>
          </property>
          <property>
            <name>logLevel</name>
            <value>INFO</value>
          </property>
          <property>
            <name>displayName</name>
            <value>upic/UTMAPP/INFO</value>
          </property>
        </properties>
      </connection-properties>
    </connection-instance>
</connection-definition-group>
<connection-definition-group>
  <connection-factory-interface>
   net.fsc.jca.communication.cci.BCUpicConnectionFactory
  </connection-factory-interface>
  <default-connection-properties>
    <pool-params>
      <initial-capacity>0</initial-capacity>
    </pool-params>
  </default-connection-properties>
  <connection-instance>
    <jndi-name>eis/my_CCI_factory</jndi-name>
    <connection-properties>
      <pool-params>
       <max-capacity>100</max-capacity>
      </pool-params>
      <transaction-support>NoTransaction</transaction-support>
      <properties>
        <property>
          <name>ConnectionURL</name>
          <value>upic://BS2HOST/UTMAPP/INFO</value>
         </property>
```

```
            <property>
              <name>encoding</name>
              <value>OSD_EBCDIC_DF04_DRV</value>
             </property>
            <property>
              <name>encodingActive</name>
              <value>true</value>
            </property>
            <property>
              <name>timeout</name>
              <value>30000</value>
            </property>
            <property>
              <name>logLevel</name>
              <value>ERROR</value>
            </property>
            <property>
              <name>displayName</name>
              <value>upic/cci/UTMAPP/INFO</value>
            </property>
          </properties>
        </connection-properties>
      </connection-instance>
    </connection-definition-group>
  </outbound-resource-adapter>
</weblogic-connector>
```

4.4.2 Enterprise Java Beans für UPIC deployen

Beim Deployment einer EJB, die BeanConnect für Outbound-Kommunikation nutzen soll, müssen Sie die EJB mit dem BeanConnect Deployment verknüpfen. Die folgenden Dateien sind für das Deployment einer EJB relevant:

- Code-Datei der EJB (`.java`- oder `.class`-Datei)
- Standardisierter Deployment Descriptor der EJB (`ejb-jar.xml`)
- Application Server spezifischer Deployment Descriptor der EJB (bei Oracle WebLogic Server: `weblogic-ejb-jar.xml`)
- Application Server spezifischer Deployment Descriptor für den Resource Adapter (bei Oracle WebLogic Server: `weblogic-ra.xml`)

Beim Deployment einer EJB wird die vom Bean-Entwickler benutzte Ressourcenreferenz dem Application Server über den Deployment Descriptor der EJB bekannt gemacht. Zusätzlich wird der Ressourcenreferenz ein Ressourcentyp zugewiesen.

BeanConnect unterstützt die folgenden Ressourcentypen, welche die unterschiedlichen Verbindungsarten repräsentieren, die verwendet werden können:

- Bei UPIC-Kommunikation über das BeanConnect-Interface:

  `net.fsc.jca.communication.EISUpicConnectionFactory`

- Bei UPIC-Kommunikation über das CCI-Interface:

  `net.fsc.jca.communication.cci.BCUpicConnectionFactory`

Der Ressourcentyp muss in folgenden Dateien angegeben werden:

- `weblogic-ra.xml` mit dem Tag `<connection-factory-interface>`
- `ejb-jar.xml` mit dem Tag `<res-type>`

Die Abschnitte der Code-Datei der EJB sowie die der Dateien `ejb-jar.xml`, `weblogic-ejb-jar.xml` und `weblogic-ra.xml`, die für das Deployment der EJB relevant sind, werden nachfolgend ausführlich beschrieben. Die **fettgedruckten** (Teil-)Pfadnamen geben die Beziehungen zwischen den einzelnen Dateien an.

- Code-Datei der EJB (`.java`- oder `.class`-Datei)

  Hier findet der JNDI-Lookup für das `ConnectionFactory`-Objekt über eine Ressourcenreferenz (kodierter Name) statt. Im folgenden Beispiel wird die Ressourcenreferenz `eis/Part1Dial` verwendet.

```
...
cf=(EISConnectionFactory)
         ic.lookup("java:comp/env/eis/Part1Dial")
...
```

- Deployment Descriptor der EJB (`ejb-jar.xml`)

  Hier wird die Ressourcenreferenz (`ConnectionFactory`-Objekt) angegeben, auf die die EJB zugreift. Zusätzlich wird der Ressourcenreferenz ein Ressourcentyp zugewiesen. Im folgenden Beispiel wird als Ressourcentyp `net.fsc.jca.communication.EISUpicConnectionFactory` verwendet.

```
<session>
  <ejb-name>SimpleBeanConnect</ejb-name>
  ...
  <resource-ref>
    <res-ref-name>eis/Part1Dial</res-ref-name>
    <res-type>
      net.fsc.jca.communication.EISUpicConnectionFactory
    </res-type>
    <res-sharing-scope>Unshareable</res-sharing-scope>
   ...
  </resource-ref>
</session>
```

  Beachten Sie, dass für `<res-sharing-scope>` immer `Unshareable` angegeben werden muss.

- Application Server spezifischer Deployment Descriptor der EJB (bei Oracle WebLogic Server: `weblogic-ejb-jar.xml`)

  Hier werden den in der Datei ejb-jar.xml definierten EJB-Namen und Ressource-Referenzen JNDI-Namen des Application Servers zugeordnet.

```
<weblogic-enterprise-bean>
   <ejb-name>SimpleBeanConnect</ejb-name>
   <jndi-name>ejb/SimpleBeanConnect</jndi-name>
   <resource-description>
      <res-ref-name>comp/env/eis/Part1Dial</res-ref-name>
      <jndi-name>java:comp/env/eis/partner1Dial</jndi-name>
   </resource-description>
</weblogic-enterprise-bean>
```

- Deployment Descriptor für den Resource Adapter (Oracle WebLogic Server: `weblogic-ra.xml`):

  Hier wird die Connection Factory für das Deployment im Application Server (hier: Oracle WebLogic Server) konfiguriert und über den JNDI-Namen (hier: `partner1Dial`) mit der Ressourcenreferenz verknüpft. Die Konfiguration der Connection Factory im Application Server informiert den Resource Adapter der URL über den Service (im folgenden Code-Fragment: `upic://BS2HOST/UTMAPP/INFO`) und den EIS Partner.

```
<outbound-resource-adapter>
   <connection-definition-group>
     <connection-factory-interface>
        net.fsc.jca.communication.EISUpicConnectionFactory
     </connection-factory-interface>
     <connection-instance>
       <jndi-name>eis/partner1Dial</jndi-name>
       <connection-properties>
       <properties>
        <property>
          <name>ConnectionURL</name>
          <value>upic://BS2HOST/UTMAPP/INFO</value>
        </property>
        ...
       </connection-properties>
     </connection-factory-interface>
    </connection-definition-group>
 </outbound-resource-adapter
```

Der für `<connection-factory-interface>` angegebene Wert muss mit dem Wert übereinstimmen, der in der Datei `ejb-jar.xml` mit dem Tag `<res-type>` angegeben wurde.

4.5 Kommunikation für Inbound konfigurieren

Die Konfiguration der allgemeinen Eigenschaften für die Inbound-Kommunikation ist in Abschnitt „Allgemeine Eigenschaften in ra.xml festlegen“ auf Seite 96 und Abschnitt „Allgemeine Eigenschaften des Resource Adapters in weblogic-ra.xml festlegen“ auf Seite 102 beschrieben.

Bei der Verwendung von Inbound-Kommunikation muss mindestens eine OLTP Message-Driven Bean deployt werden. Die OLTP Message-Driven Bean muss eines der folgenden Message-Listener-Interfaces implementieren:

- `net.fsc.jca.communication.AsyncOltpMessageListener`
- `net.fsc.jca.communication.OltpMessageListener`
- `javax.resource.cci.MessageListener`

Für das Deployment einer OLTP Message-Driven Bean muss ein Deployment Descriptor in der Datei `ejb-jar.xml` erstellt werden. Wenn die Datei `ejb-jar.xml` bei der Bean-Entwicklung nicht automatisch von einer IDE (**I**ntegrated **D**evelopment **E**nviroment) erstellt wird, müssen Sie die Datei manuell anlegen. In den meisten Fällen erfordert das Deployment einer OLTP Message-Driven Bean auch einen Application Server spezifischen Deployment Descriptor. Beim Oracle WebLogic Server wird der Application Server spezifische Deployment Descriptor in der Datei `weblogic-ejb-jar.xml` gespeichert. Beispiele für die Dateien `ejb-jar.xml` und `weblogic-ejb-jar.xml` für eine OLTP Message-Driven Bean finden Sie auf Seite 146 und Seite 149.

4.5.1 Konfigurations-Properties für Inbound-Kommunikation in ejb-jar.xml definieren

Die folgenden Properties müssen im Deployment Descriptor (ejb-jar.xml) der OLTP Message-Driven Bean eingestellt werden:

- Die Property `messaging-type` gibt das Message-Listener-Interface an, das von der OLTP Message-Driven Bean verwendet wird.
- Die `activation-config` Properties beziehen sich auf das spezifizierte Message-Listener-Interface. Jede dieser Properties ist in einem eigenen `activation-config-property`-Element innerhalb des `activation-config`-Elements in der Datei `ejb-jar.xml` definiert.

  Folgende `activation-config` Properties sind verfügbar:

  `messageEndpoint`

  `encoding`

  `encodingActive`

  `redeliveryThreshold`

Die Properties für die Datei `ejb-jar.xml` sind nachfolgend ausführlich beschrieben.

messaging-type

Die Property `messaging-type` gibt das Message-Listener-Interface an, das von der OLTP Message-Driven Bean verwendet wird.

Definition: Message-Listener-Interface, das von der OLTP Message-Driven Bean verwendet wird

Standard: –

Beispiel:

```
<messaging-type>
  net.fsc.jca.communication.AsyncOltpMessageListener
</messaging-type>
```

messageEndpoint

Die `activation-config` Property `messageEndpoint` gibt den Namen des Message Endpoints an. Der hier angegebene Name des Message Endpoints muss mit dem Namen des Inbound Message Endpoints übereinstimmen, der beim Konfigurieren des Proxys mit der Management Console angegeben wurde (siehe Abschnitt „Inbound Message Endpoints konfigurieren“ auf Seite 241).

Definition: Name des Message Endpoints.

Standard: –

Beispiel:

```
<activation-config-property>
  <activation-config-property-name>messageEndpoint
  </activation-config-property-name>
  <activation-config-property-value>SampleAsynOltpMdb
  </activation-config-property-value>
</activation-config-property>
```

encoding

Die `activation-config` Property `encoding` definiert eine Code-Tabelle für die Konvertierung von Byte-Code (zum Beispiel EBCDIC) in Unicode.

Die hier angegebene Property wird überschrieben, wenn die Message-Driven Bean durch einen Inbound Service aufgerufen wird, dem in der Management Console bei der Konfiguration des Proxy ein `Partner Encoding` zugeordnet wurde.

Definition:

```
[<builtin_encoding_table> |
builtin:<builtin_encoding_table>|
jdk:<jdk_encoding_table> |
custom:<encoding_table>]
```

Erklärung: Name einer Code-Tabelle, die für die Code-Konvertierung verwendet wird.
Ist kein Präfix oder ist das Präfix `builtin:` angegeben, müssen Sie den Namen einer integrierten Code-Tabelle angeben, die von BeanConnect bereitgestellt wird.
Folgende Code-Tabellen stehen zur Verfügung:
`OSD_EBCDIC_DF03_IRV`, `OSD_EBCDIC_DF04_1`, `OSD_EBCDIC_DF04_15`, `OSD_EBCDIC_DF04_DRV`
Mit dem Präfix `jdk:` geben Sie eine Code-Tabelle an, die im JDK enthalten ist.
Mit dem Präfix `custom:` weisen Sie Ihre eigene Code-Tabelle zu. Dabei müssen Sie den vollständigen Klassennamen der Code-Tabelle angeben. Weitere Einzelheiten zur Verwendung von eigenen Code-Tabellen finden Sie in der JavaDoc zu BeanConnect.

Standard: OSD_EBCDIC_DF04_DRV

Beispiel: für openUTM-Partner:

```
<activation-config-property>
  <activation-config-property-name>encoding
  </activation-config-property-name>
  <activation-config-property-value>OSD_EBCDIC_DF04_15
  </activation-config-property-value>
</activation-config-property>
```

für CICS-Partner:

```
<activation-config-property>
  <activation-config-property-name>encoding
  </activation-config-property-name>
  <activation-config-property-value>jdk:Cp1047
  </activation-config-property-value>
</activation-config-property>
```

encodingActive

Die `activation-config` Property `encodingActive` gibt an, ob die Code-Konvertierung aktiviert werden soll oder nicht.

`encodingActive` wird unabhängig vom hier angegebenen Wert auf `true` gesetzt, wenn die Message-Driven Bean durch einen Inbound Service aufgerufen wird, dem in der Management Console bei der Konfiguration des Proxy ein `Partner Encoding` zugeordnet wurde.

Definition: `[true | false]`

Erklärung: Gibt an, ob die Code-Konvertierung aktiviert wird oder nicht.

`true`	Die Code-Konvertierung gemäß der Einstellungen der activation-config-Property `encoding` ist aktiviert.
`false`	Die Standard-Code-Tabelle des JDK wird für die Konvertierung von Byte-Streams in Strings verwendet.

Standard: false

Beispiel:

```
<activation-config-property>
   <activation-config-property-name>encodingActive
   </activation-config-property-name>
   <activation-config-property-value>true
   </activation-config-property-value>
</activation-config-property>
```

redeliveryThreshold

Die `activation-config` Property `redeliveryThreshold` definiert die Anzahl der zusätzlichen Versuche, die Nachricht zuzustellen, wenn die Transaktion zurückgesetzt wird. Diese Property kann nur für asynchrone OLTP Message-Driven Beans eingestellt werden, d.h. für OLTP Message-Driven Beans, die das Message-Listener-Interface `net.fsc.jca.communication.AsyncOltpMessageListener` implementieren. Das Message-Listener-Interface wird in der Property `messaging-type` angegeben.

Diese Property wird nur wirksam, wenn die OLTP Message-Driven Bean mit dem Transaktionsattribut `Required` deployt wurde. In diesem Fall wird die Methode `onMessage` innerhalb einer Transaktion aufgerufen, die vom Proxy gestartet wurde (niemals vom EIS). Wird die Transaktion zurückgesetzt, so wird die Nachricht noch einmal zugestellt, es sei denn, der generierte Schwellwert wurde überschritten.

Definition: Anzahl der zusätzlichen Zustellversuche im Fehlerfall.
Minimalwert: 0
Maximalwert: 254

Standard: 0, d.h. die Nachricht wird nicht noch einmal zugestellt.

Beispiel:

```
<activation-config-property>
  <activation-config-property-name>redeliveryThreshold
  </activation-config-property-name>
  <activation-config-property-value>1
  </activation-config-property-value>
</activation-config-property>
```

4.5.2 Konfigurations-Properties für Inbound-Kommunikation in weblogic-ejb-jar.xml definieren

Im Application Server-spezifischen Deployment Descriptor `weblogic-ejb-jar.xml` muss der Wert des Tags `<resource-adapter-jndi-name>` des Elements `<message-driven-descriptor>` mit dem Namen des Resource Adapters übereinstimmen, der im Element `<jndi-name>` in der Datei `weblogic-ra.xml` angegeben wurde.

Siehe auch Beispiel 10 .

4.5.3 Beispiele für ejb-jar.xml und weblogic-ejb-jar.xml

Beispiel 9 ejb-jar.xml

Der folgende Code-Auszug zeigt einen Deployment Descriptor `ejb-jar.xml` für eine JAR-Datei, die drei OLTP Message-Driven Beans beschreibt. Die OLTP Message-Driven Beans implementieren drei verschiedene Message-Listener-Interfaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar id="ejb-jar_ID" metadata-complete="false" version="3.1"
          xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                              http://java.sun.com/xml/ns/javaee/ejb-
jar_3_1.xsd">
  <description xml:lang="en">Code Samples for Inbound
Communication</description>
  <display-name xml:lang="en">SampleMessageDrivenBeans</display-name>
  <enterprise-beans>
    <message-driven>
      <description xml:lang="en">
           Code Sample for Dialog Inbound Communication
      </description>
      <ejb-name>SampleDialogOltpMdbBean</ejb-name>
      <ejb-class>net.fsc.jca.BeanConnect.oltpmdb.SampleDialogOltpMdbBean
                                                                  </ejb-class>
      <messaging-type>net.fsc.jca.communication.OltpMessageListener
      </messaging-type>
      <transaction-type>Container</transaction-type>
      <activation-config>
        <activation-config-property>
          <activation-config-property-name>messageEndpoint
          </activation-config-property-name>
          <activation-config-property-value>SampleDialogOltpMdb
          </activation-config-property-value>
        </activation-config-property>
        <activation-config-property>
          <activation-config-property-name>encodingActive
          </activation-config-property-name>
          <activation-config-property-value>true
           </activation-config-property-value>
        </activation-config-property>
        <activation-config-property>
          <activation-config-property-name>encoding
          </activation-config-property-name>
          <activation-config-property-value>OSD_EBCDIC_DF04_15 [1]
```

[1] Für CICS-Partner: <activation-config-property-value>jdk:Cp1047

```
            </activation-config-property-value>
          </activation-config-property>
        </activation-config>
      </message-driven>
      <message-driven>
        <description xml:lang="en">
             Code Sample for Asynchronous Inbound Communication
        </description>
        <ejb-name>SampleAsynOltpMdbBean</ejb-name>
        <ejb-class>net.fsc.jca.BeanConnect.oltpmdb.SampleAsynOltpMdbBean
                                                              </ejb-class>
        <messaging-type>net.fsc.jca.communication.AsyncOltpMessageListener
        </messaging-type>
        <transaction-type>Container</transaction-type>
        <activation-config>
          <activation-config-property>
            <activation-config-property-name>messageEndpoint
             </activation-config-property-name>
            <activation-config-property-value>SampleAsynOltpMdb
             </activation-config-property-value>
          </activation-config-property>
          <activation-config-property>
           <activation-config-property-name>encodingActive
            </activation-config-property-name>
           <activation-config-property-value>true
            </activation-config-property-value>
          </activation-config-property>
            <activation-config-property-name>encoding
            </activation-config-property-name>
            <activation-config-property-value>OSD_EBCDIC_DF04_15 [1]
            </activation-config-property-value>
          </activation-config-property>
          <activation-config-property>
           <activation-config-property-name>redeliveryThreshold
            </activation-config-property-name>
            <activation-config-property-value>1</activation-config-property-
value>
          </activation-config-property>
          <activation-config-property>
        </activation-config>
      </message-driven>
      <message-driven>
        <description xml:lang="en">
            Code Sample for CCI Inbound Communication</description>
        <ejb-name>SampleCciOltpMdbBean</ejb-name>
```

[1] Für CICS-Partner: <activation-config-property-value>jdk:Cp1047

```
      <ejb-class>net.fsc.jca.BeanConnect.oltpmdb.SampleCciOltpMdbBean
                                                              </ejb-class>
      <messaging-type>javax.resource.cci.MessageListener</messaging-type>
      <transaction-type>Bean</transaction-type>
      <activation-config>
        <activation-config-property>
          <activation-config-property-name>messageEndpoint
           </activation-config-property-name>
          <activation-config-property-value>SampleCciOltpMdbBean
          </activation-config-property-value>
        </activation-config-property>
      </activation-config>
    </message-driven>
  </enterprise-beans>
  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>SampleDialogOltpMdbBean</ejb-name>
        <method-name>onMessage</method-name>
        <method-params>
          <method-param>net.fsc.jca.communication.OltpMessage</method-param>
        </method-params>
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
    <container-transaction>
      <method>
        <ejb-name>SampleAsynOltpMdbBean</ejb-name>
        <method-name>onMessage</method-name>
        <method-params>
          <method-param>net.fsc.jca.communication.OltpMessage</method-param>
        </method-params>
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
    <container-transaction>
      <method>
        <ejb-name>SampleCciOltpMdbBean</ejb-name>
        <method-name>onMessage</method-name>
        <method-params>
          <method-param>net.fsc.jca.communication.OltpMessage</method-param>
        </method-params>
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```

Beispiel 10 weblogic-ejb-jar.xml

Der erste Code-Auszug zeigt den Teil des Application Server-spezifischen Deployment Descriptors `weblogic-ra.xml` für den Resource Adapter. In diesem Teil wird der Name des Resource Adapters festlegt. Der nachfolgende Application Server-spezifische Deployment Descriptor `weblogic-ejb-jar.xml` bezieht sich dann auf diesen Namen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<weblogic-connector
   xmlns="http://xmlns.oracle.com/weblogic/weblogic-connector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-connector
http://www.oracle.com/technology/weblogic/weblogic-connector/1.3/weblogic-
connector.xsd">
   <jndi-name>BeanConnect</jndi-name>
```

Nun folgt der Code-Auszug aus dem Application Server-spezifischen Deployment Descriptor `weblogic-ejb-jar.xml`, der sich auf das EJB für die drei OLTP Message-Driven Beans aus Beispiel 9 bezieht:

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-ejb-jar xmlns="http://xmlns.oracle.com/weblogic/weblogic-ejb-jar"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-ejb-jar
http://xmlns.oracle.com/weblogic/weblogic-ejb-jar/1.1/weblogic-ejb-jar.xsd">
    <weblogic-enterprise-bean>
     <ejb-name>SampleDialogOltpMdbBean</ejb-name>
     <message-driven-descriptor>
       <resource-adapter-jndi-name>BeanConnect</resource-adapter-jndi-name>
     </message-driven-descriptor>
   </weblogic-enterprise-bean>
   <weblogic-enterprise-bean>
     <ejb-name>SampleAsynOltpMdbBean</ejb-name>
     <message-driven-descriptor>
       <resource-adapter-jndi-name>BeanConnect</resource-adapter-jndi-name>
     </message-driven-descriptor>
   </weblogic-enterprise-bean>
   <weblogic-enterprise-bean>
     <ejb-name>SampleCciOltpMdbBean</ejb-name>
     <message-driven-descriptor>
       <resource-adapter-jndi-name>BeanConnect</resource-adapter-jndi-name>
     </message-driven-descriptor>
   </weblogic-enterprise-bean>
 </weblogic-ejb-jar>
```

4.6 Logging für den Resource Adapter vorbereiten

Die log4j-property-Dateien finden Sie nach der Installation im Installationsverzeichnis des Resource Adapters im Unterverzeichnis `config`.

In diesem Verzeichnis stehen folgende Dateien:

1. `BeanConnect.log4j.properties.xml`
2. `BeanConnect.log4j.properties_debug.xml`
3. `BeanConnect.log4j.properties_default.xml`
4. `BeanConnect.log4j.properties_error.xml`

Diese Dateien enthalten Einstellungen für das Logging des Resource Adapters. Die erste und die dritte Datei sind inhaltlich gleich.

Der BeanConnect Resource Adapter liest die Einstellungen für das Logging aus der Datei `BeanConnect.log4j.properties.xml`.

Kopieren Sie diese Datei in das Verzeichnis `<WebLogicServerDomainDirectory>/config` (Oracle WebLogic Server-spezifisch).

Im Normalfall müssen Sie die Default-Einstellungen für das Logging nicht ändern. Bei Bedarf (z.B. Diagnose) können Sie den Umfang des Loggings nachträglich erweitern oder reduzieren, siehe Abschnitt „Überblick zum Logging des BeanConnect Resource Adapters“ auf Seite 536.

4.7 Besonderheiten im Multi-Resource Adapter Betrieb

In einer Multi-Resource Adapter Konfiguration arbeiten mehrere Resource Adapter Instanzen mit einer Proxy Instanz zusammen. Pro Proxy Instanz sind maximal 32 Resource Adapter Instanzen möglich. Jeder Resource Adapter Instanz ist ein eindeutiger Index zugeordnet. Dieser Index muss in die Datei `ra.xml` eingetragen werden, siehe unten.

Der Multi-Resource Adapter Betrieb ist für Outbound-Kommunikation über OSI TP / LU6.2 und für Inbound-Kommunikation möglich.

Im Application Server müssen Sie bei Multi-Resource Adapter Betrieb für jeden Resource Adapter folgende Konfigurationsschritte durchführen:

- Definieren Sie in der Datei `ra.xml` den Index des Resource Adapters mit der zusätzlichen Konfigurations-Property `resourceAdapterIndex`. Dies ist wie beim Standard-Betrieb unter bestimmten Voraussetzungen auch über die Management Console möglich, siehe „ra.xml anpassen“ auf Seite 96.

  Wenn Sie den Index manuell vergeben, müssen Sie darauf achten, dass jeder Resource Adapter einen eindeutigen Index bekommt. Für Inbound-Kommunikation benötigen Sie diesen eindeutigen Index bei der Konfiguration der Inbound Message Endpoints mit der Management Console (siehe Abschnitt „Inbound Message Endpoints konfigurieren“ auf Seite 241).

- Führen Sie die anderen Konfigurationsschritte im Application Server genauso aus wie beim Standard Resource Adapter Betrieb, siehe Abschnitt „Konfigurationsschritte für Outbound- und Inbound-Kommunikation“ auf Seite 93.

resourceAdapterIndex

`resourceAdapterIndex` legt den Index dieser Resource Adapter Instanz in einer Multi-Resource Adapter Konfiguration fest.

Diese Property ist nur in einer Multi-Resource Adapter Konfiguration von Bedeutung und darf nicht zusammen mit der Property `resourceAdapterAddresses` angegeben werden (siehe Abschnitt „Besonderheiten im Cluster-Betrieb“).

Definition: `<index>`

Erklärung: `<index>` ist eine Zahl zwischen 1 und 32. Sie wird durch die Management Console festgelegt, wenn die `ra.xml` mit der Management Console konfiguriert wird.

Standard: kein Standardwert

Beispiel:

```
<config-property>
  <description>Index of this resource adapter instance in a
    multi-resource-adapter configuration.
  </description>
  <config-property-name>resourceAdapterIndex
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>5
  </config-property-value>
</config-property>
```

4.8 Besonderheiten im Cluster-Betrieb

In einer Cluster-Umgebung können sowohl mehrere Resource Adapter Instanzen als auch mehrere Proxy Instanzen zum Ablauf kommen. Die Anzahl der Proxy Instanzen muss nicht gleich der Anzahl der Resource Adapter Instanzen sein. Es sind maximal 32 Resource Adapter Instanzen und maximal 32 Proxy Instanzen möglich.

Alle beteiligten Instanzen sind identisch konfiguriert, insbesondere wird jede Resource Adapter Instanz mit dem gleichen BeanConnect-RAR-Archiv deployt und arbeitet daher auch mit den gleichen Konfigurationswerten aus `ra.xml`.

Cluster-Betrieb ist für Outbound-Kommunikation über OSI TP / LU6.2 und für Inbound-Kommunikation möglich.

Im Application Server sind bei Cluster-Betrieb folgende Konfigurationsschritte nötig:

- Definieren Sie in der Datei `ra.xml` die zusätzlichen Parameter und Properties für den Cluster-Betrieb:
  - In der Property `proxyURL` müssen Sie die Adressen aller Proxy Instanzen angeben.
  - In der zusätzlichen Property `resourceAdapterAddresses` müssen Sie die Adressen aller Resource Adapter Instanzen angeben.
  - In den Properties `proxyReconnectCount` und `proxyReconnectInterval` können Sie die Parameter für Neuzuordnung von Resource Adapter und Proxy ändern.

  Dies ist wie beim Standard-Betrieb über die Management Console möglich, siehe „ra.xml anpassen“ auf Seite 96. Sie können die Datei `ra.xml` aber auch manuell anpassen.
- Führen Sie die anderen Konfigurationsschritte im Application Server genauso aus wie beim Betrieb mit nur einem Resource Adapter, siehe Abschnitt „Konfigurationsschritte für Outbound- und Inbound-Kommunikation“ auf Seite 93.

proxyURL

Die `proxyURL` legt im Cluster-Betrieb die Zuordnung der Resource Adapter Instanzen zu den Proxy Instanzen fest. Wenn mit mehreren Proxy Instanzen gearbeitet wird, dann müssen die Adressen aller Proxys angegeben werden, jeweils getrennt durch ein Semikolon.

Definition: `oltp://<host>:<port>/<name>; ... ;oltp://<host>:<port>/<name>`

Erklärung:

`<host>`	Rechner, auf dem der betreffende Proxy-Container installiert ist. `<host>` kann als symbolischer Name oder als IPv4-Adresse angegeben werden.
`<port>`	Portnummer des betreffenden Proxy-Containers + 4

`<name>` Anwendungsname des betreffenden Proxy-Containers (BCU<port>)

Die einzelnen Einträge müssen durch Semikolon getrennt sein.

Standard: `oltp://localhost:31004/BCU31004`

Beispiel:

```
<config-property>
  <description>BeanConnect Proxy URLs for OLTP outbound
        communication with 2 Proxies</description>
  <config-property-name>proxyURL</config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>oltp://proxyhost1:31004/BCU31004;
        oltp://proxyhost2:31014/BCU31014
  </config-property-value>
</config-property>
```

resourceAdapterAddresses

Diese Property ist nur in einer Cluster-Konfiguration mit mehreren Resource Adapter Instanzen von Bedeutung und darf nicht zusammen mit der Property `resourceAdapter-Index` angegeben werden (siehe Abschnitt „Besonderheiten im Multi-Resource Adapter Betrieb“ auf Seite 151).

Mit dieser Property werden die Adressen aller Rechner definiert, auf denen Instanzen des BeanConnect Resource Adapters zum Ablauf kommen. Sie können maximal 32 Adressen angeben, die Einträge müssen durch ein Semikolon getrennt sein.

Die Adressen geben Sie in der Form `host[:port]` an. Wenn Sie keine Portnummer angeben, dann wird die Portnummer der Property `inboundListenerPort` als Listener Port für die Inbound-Kommunikation verwendet, diese muss dann größer 0 sein. Wenn Sie eine Portnummer angeben, dann wird diese als Listener Port für die Inbound-Kommunikation verwendet, sie muss größer 0 sein.

Wenn unter einer Host-Adresse mehrere Resource Adapter Instanzen ablaufen sollen, dann müssen Sie diese Host-Adresse entsprechend oft in der Liste angeben und mit unterschiedlichen Portnummern versehen.

Definition: `<host>[:<port>]; ... ;<host>[:<port>]`

Erklärung:

`<host>` Rechner, auf dem die betreffende Resource Adapter Instanz abläuft. `<host>` kann als symbolischer Name oder als IPv4-Adresse angegeben werden.

`<port>` Portnummer der betreffenden Resource Adapter Instanz für Inbound-Kommunikation.

Die einzelnen Einträge müssen durch Semikolon getrennt sein.

Standard: Es gibt keinen Standardwert.

Beispiel:
```
<config-property>
  <config-property-name>resourceAdapterAddresses
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>
     host1:31099;host2:31099;host3:31099
  </config-property-value>
</config-property>
```

proxyReconnectCount

Diese Property ist nur in einer Cluster-Konfiguration mit mehreren Resource Adapter Instanzen und mehreren Proxy Instanzen von Bedeutung. `proxyReconnectCount` regelt die nutzungsabhängige Neuzuordnung einer Resource Adapter Instanz zu einer Proxy-Anwendung. Dieser Mechanismus wird aktiviert, sobald einer Proxy-Anwendung mehrere Resource Adapter Instanzen zugeordnet sind.

Definition: `<number>`

Erklärung: `<number>` gibt an, nach wie vielen Verbindungsanforderungen (`getConnection()`-Aufrufen) eine Neuzuordnung zwischen Resource Adapter Instanz und Proxy-Anwendung initiiert werden soll.
Wird für `<number>` der Wert 0 angegeben, dann ist die nutzungsabhängige Neuzuordnung deaktiviert.

Standard: 100

Beispiel:
```
<config-property>
  <config-property-name>proxyReconnectCount
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>200
  </config-property-value>
</config-property>
```

proxyReconnectInterval

Diese Property ist nur in einer Cluster-Konfiguration mit mehreren Resource Adapter Instanzen und mehreren Proxy Instanzen von Bedeutung. `proxyReconnectInterval` regelt die zeitabhängige Neuzuordnung einer Resource Adapter Instanz zu einer Proxy-Anwendung. Dieser Mechanismus wird aktiviert, sobald einer Proxy-Anwendung mehrere Resource Adapter Instanzen zugeordnet sind.

Definition: `<minutes>`

Erklärung: `<minutes>` gibt an, nach wie vielen Minuten eine Neuzuordnung zwischen Resource Adapter Instanz und Proxy-Anwendung initiiert werden soll.
Wird für `<minutes>` der Wert 0 angegeben, dann ist die zeitabhängige Neuzuordnung deaktiviert.

Standard: 10

Beispiel:

```
<config-property>
  <config-property-name>proxyReconnectInterval
  </config-property-name>
  <config-property-type>java.lang.String
  </config-property-type>
  <config-property-value>5
  </config-property-value>
</config-property>
```

Wird eine Cluster-Konfiguration mit mehr Resource Adapter Instanzen als Proxy Instanzen betrieben (d.h. mindestens einer Proxy Instanz ist immer mehr als eine Resource Adapter Instanz zugeordnet), dann sollten aus Performance-Gründen die nutzungsabhängige und die zeitabhängige Neuzuordnung ausgeschaltet oder zumindest größere Werte als die Standardwerte eingestellt werden.

5 BeanConnect Management Console - Überblick

Die BeanConnect Management Console wird zum Konfigurieren und Administrieren eines oder mehrerer BeanConnect Proxys verwendet. Die Management Console unterscheidet zwischen lokalen und entfernten Proxys:

Lokale Proxys sind Proxys, die auf dem selben Rechner unter derselben Kennung laufen wie die Management Console, alle anderen Proxys werden als **entfernte Proxys** bezeichnet.

Außerdem ist die Management Console ein JMX-Client. Damit ist es z.B. möglich, über die BeanConnect MBeans Einstellungen im Resource Adapter zu ändern, Statistikwerte der Verbindungen über die BeanConnect MBeans abzufragen oder auf die MBeans des Application Servers zuzugreifen.

Darüber hinaus können Sie über die Management Console auch die Logging-Eigenschaften im Application Server einstellen.

Die Management Console ist ein Verwaltungswerkzeug mit einer grafischen Bedienoberfläche.

Command Line Interface (MC-CLI)

Zusätzlich zur graphischen Bedienoberfläche bietet die Management Console ein Command Line Interface an, mit dem Sie die Funktionen der Management Console auch per Skript durchführen können. Dieses Command Line Interface der Management Console (MC-CLI) verwendet als Skriptsprache Jython. Detaillierte Informationen dazu finden Sie in Kapitel „Command Line Interface der BeanConnect Management Console (MC-CLI)“ auf Seite 309.

MC-CLI-Recording

Alle Aktionen der Management Console, zu denen es Funktionen in der MC-CLI gibt, werden in internen Puffern der Management Console mitgeschnitten. Diese Mitschnitte können Sie in einem internen Editor ansehen oder auf Datei ausgeben lassen. Weitere Informationen finden Sie in Abschnitt „MC-CLI Recording: Aktionen der Management Console mitschneiden“ auf Seite 172.

Weitere Einzelheiten über die Management Console finden Sie in der Online-Hilfe der Management Console.

Dieses Kapitel vermittelt einen Überblick über folgende Themen:

- Management Console starten und beenden
- Bedienoberfläche - Management Console-Fenster
- Funktionen der BeanConnect Management Console
- Verwaltungsdaten der Management Console

5.1 Management Console starten und beenden

Dieser Abschnitt erläutert, wie Sie die

- Management Console starten
- Online-Hilfe der Management Console starten
- Management Console beenden

5.1.1 Management Console starten

Management Console auf Unix-/Linux-Systemen starten

Starten Sie die Management Console mit dem Shell-Skript `startconsole.sh`:

1. Öffnen Sie eine Shell.
2. Wechseln Sie in das Home-Verzeichnis der Management Console.
3. Führen Sie das Shell-Skript `startconsole.sh` aus.

Management Console auf Windows-Systemen starten

Starten Sie die Management Console über die Programmgruppe:

Start - Programme - FUJITSU Software BeanConnect V3.0B00 - Management Console - Management Console

5.1.2 Online-Hilfe der Management Console starten

Sie starten die Online-Hilfe für die Management Console wie folgt:

- Drücken Sie die Taste **F1**.
- Wählen Sie im Menü **Help** den Befehl **Content**.
- Wenn Sie kontextsensitive Hilfe anfordern möchten, klicken Sie in dem entsprechenden Dialogfeld oder Arbeitsbereichsfenster auf die Schaltfläche **Help**.

Alternativ können Sie die Online-Hilfe aufrufen, ohne zuvor die Management Console zu starten:

- Auf Unix-/Linux-Systemen:
  1. Öffnen Sie eine Shell.
  2. Wechseln Sie in das Home-Verzeichnis der Management Console.
  3. Rufen Sie das Skript `starthelp.sh` auf.
- Auf Windows-Systemen:

  Wählen Sie in der Programmgruppe **FUJITSU Software BeanConnect V3.0B00 Management Console** den Befehl **MC Help**.

Sprache der Online-Hilfe

Sie ändern die Spracheinstellungen für die Online-Hilfe der Management Console wie folgt:

- Auf Unix-/Linux-Systemen:

  Bei der Installation der Management Console wird sowohl die englische als auch die deutsche Version der Online-Hilfe installiert. Die Management Console verwendet standardmäßig die englische Sprachversion.

  Sie ändern die Spracheinstellung in Deutsch, indem Sie die Datei `ConsoleHelp_de.jar` unter dem Dateinamen `ConsoleHelp.jar` in das Unterverzeichnis `lib` des BeanConnect-Installationsverzeichnisses verschieben.
- Auf Windows-Systemen:

  Während der Installation von BeanConnect können Sie die Spracheinstellung für die Online-Hilfe der Management Console auswählen (siehe Abschnitt „BeanConnect auf Windows-Systemen installieren“ auf Seite 67). Standardmäßig ist als Sprache Englisch eingestellt.

5.1.3 Management Console beenden

Beenden Sie die Management Console, indem Sie im Menü **File** den Befehl **Exit** wählen.

5.2 Bedienoberfläche - Management Console-Fenster

Das Fenster der Management Console enthält die nachfolgend beschriebenen Komponenten:

- Am oberen Fensterrand der Management Console befindet sich eine Menüleiste. Sie enthält die Menüs **File**, **View**, **Extras**, **Window** und **Help**.
- Die Fenstermitte ist in folgende Bereiche aufgeteilt:
  - Navigationsbereich (links)
    Dieser Bereich zeigt in einer Baumstruktur (Navigationsbaum) die verwalteten Proxys samt der in ihnen enthaltenen Ressourcen und Einstellungen an.
  - Arbeitsbereich (oben rechts)
    Dieser Bereich zeigt die Konfigurationsdaten mit den zugehörigen Einträgen an, die Sie im Navigationsbereich ausgewählt haben.
  - Protokollfenster (unten rechts)
    Dieser Bereich enthält die Meldungen der aktuellen Session an der Management Console. Das Protokollfenster kann dauerhaft ein- oder ausgeblendet werden.
- Am unteren Fensterrand befindet sich die Statusleiste.
  In der Statusleiste wird der Verarbeitungsstatus von zeitaufwändigen Aufträgen in Form von Textmeldungen angezeigt.

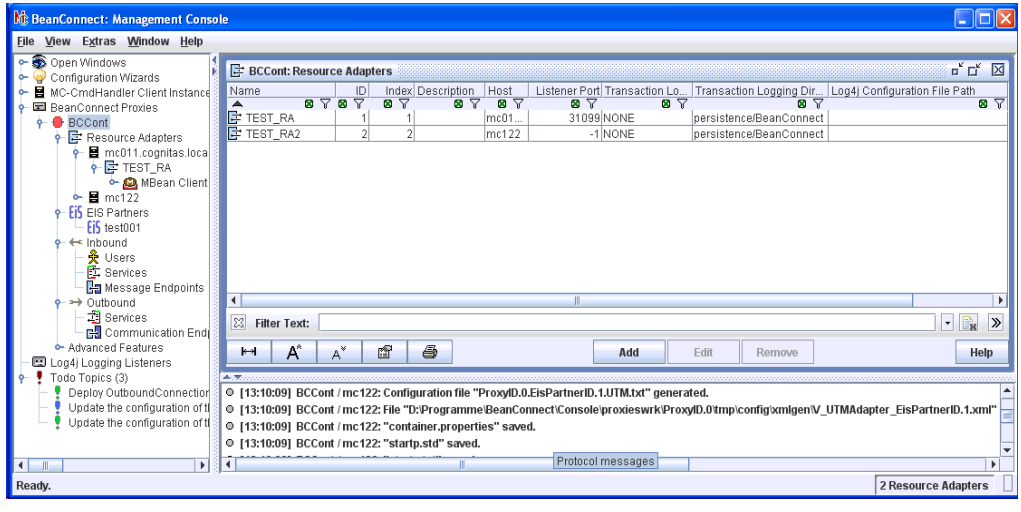


Bild 12: Bedienoberfläche der Management Console

5.2.1 Navigationsbereich in der Management Console

Die verwalteten BeanConnect Proxys werden im Navigationsbereich in einer Baumstruktur angezeigt (genau so wie im Windows Explorer die Laufwerke und Verzeichnisse). Unterhalb des BeanConnect Proxy-Knotens wird für jeden BeanConnect Proxy, der mit der Management Console verwaltet wird, ein eigener Teilbaum angezeigt (Proxy1 in Bild 13). Mit diesem Teilbaum können Sie die Konfigurationsdaten des BeanConnect Proxys anzeigen und bearbeiten. Wenn Sie im Teilbaum auf ein Element klicken, werden die mit dieser Komponente verbundenen Konfigurationsdaten im Arbeitsbereich in einem Arbeitsbereichsfenster angezeigt. Darüber hinaus enthält das Kontextmenü des entsprechenden Eintrags Befehle, mit denen Sie die Konfigurationsdaten ändern können.

Die Abbildung unten zeigt den Navigationsbereich der Management Console an:

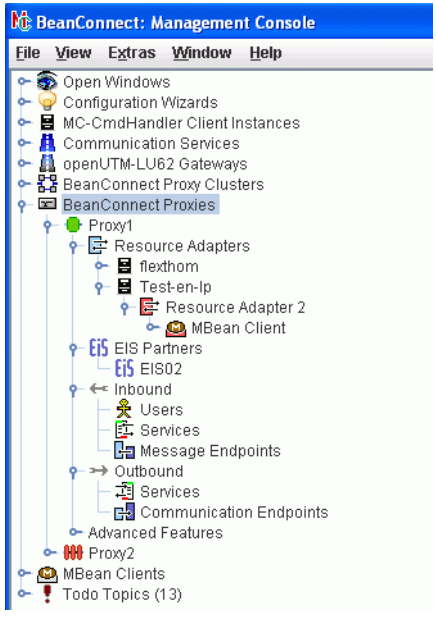


Bild 13: Navigationsbereich

5.2.2 Verwaltete Objekte

Für die Objektverwaltung muss kein Proxy-Container gestartet sein.

Es ist zwischen folgenden Objekten zu unterscheiden:

MC-CmdHandler Client Instances

Ein Objekt für jeden verwalteten MC-CmdHandler. Über einen MC-CmdHandler können Sie auf das Dateisystem des betreffenden Rechners zugreifen und dort Skripts ausführen.

Communication Services

Ein Objekt für jeden verwalteten Communication Service. Ein Communication Service wird für die Kommunikation mit CICS-Partnern benötigt, er kann sowohl lokal als auch auf einem fernen Rechner ablaufen.

openUTM-LU62 Gateways

Ein Objekt für jedes verwaltete openUTM-LU62 Gateway. Ein openUTM-LU62 Gateway wird zusammen mit dem Communication Service für die Kommunikation mit CICS-Partnern benötigt. Er läuft jeweils auf demselben Rechner wie der zugehörige Communication Service.

BeanConnect Proxy Clusters

Ein Objekt für jeden verwalteten BeanConnect Proxy Cluster. Ein Proxy Cluster besteht aus einem oder mehreren Proxys, die zuvor konfiguriert werden müssen.

BeanConnect Proxies

Ein Objekt für jeden verwalteten BeanConnect Proxy, der nicht in einem Proxy Cluster enthalten ist.

Nur „administrierbare“ BeanConnect Proxys können mit der Management Console verwaltet werden. Ein BeanConnect Proxy wird als administrierbar betrachtet, wenn einer der folgenden Punkte zutrifft:

- Bei dem Proxy handelt es sich um einen lokalen Proxy.
- Bei dem Proxy handelt es sich um einen entfernten Proxy, dessen zugehöriger MC-CmdHandler verfügbar und über die Management Console bedienbar ist.

Resource Adapters

Jedem Proxy ist ein oder mehrere Resource Adapter zugewiesen. Ein Resource Adapter läuft auf einer Instanz eines Application Servers ab.

MBean Client

Für jeden Resource Adapter kann ein MBean-Client definiert werden. Mit Hilfe des MBean-Clients kann die Management Console auf die MBeans des betreffenden JMX-Servers zugreifen (Attribute anzeigen und deren Werte ändern, Operationen ausführen und Notifications empfangen).

Es ist auch möglich, freie (stand-alone) MBean-Clients zu definieren, die keinem Resource Adapter zugeordnet sind. Freie MBean-Clients werden auf der obersten Ebene des Navigationsbaums dargestellt.

EIS Partners

Ein Objekt für jeden verwalteten EIS Partner.

Inbound Users

Der EIS Partner kann die Benutzerinformation (Benutzername und Passwort) über Inbound-Kommunikation weiterleiten. Diese Benutzerinformationen müssen dem Proxy bekannt sein.

Inbound Message Endpoints

Ein Inbound Message Endpoint stellt einen Kommunikationsendpunkt für Inbound-Kommunikation dar. Ein Proxy kann mehrere Inbound Message Endpoints verwalten.

Inbound Services

Ein Inbound Service stellt einen Service dar, den ein EIS Partner bei Inbound-Kommunikation adressiert. Ein Service ist immer genau einem Inbound Message Endpoint zugeordnet, es ist jedoch möglich, einem Inbound Message Endpoint mehrere Services zuzuordnen. Außerdem können Sie für einen Service Codierungseigenschaften definieren.

Outbound Services

Ein Outbound Service stellt einen vom EIS Partner zur Verfügung gestellten Service (Transaktionscode (TAC) oder CICS-Transaktion) dar.

Outbound Communication Endpoints

Für jeden (symbolischen) Service, der in der Konfigurations-Property `connectionURL` im Application Server angegeben ist, müssen Sie im Proxy einen entsprechenden Outbound Communication Endpoint mit dem gleichen Namen konfigurieren. In der Definition des Outbound Communication Endpoint wird der symbolische Service-Name auf einen realen Service-Namen in der EIS Partneranwendung abgebildet.

Outbound Communication Endpoints sind spezifisch für die EIS Partner. Einem EIS Partner können mehrere Outbound Communication Endpoints zugeordnet werden.

Näheres siehe Abschnitt „Outbound Communication Endpoints konfigurieren“ auf Seite 238.

5.2.3 Zusätzliche Funktionen und Informationen

Weitere Funktionen und Informationen finden Sie unter:

Advanced Functions

Die erweiterten Funktionen liefern statistische Daten und ermöglichen die Diagnosesteuerung. Weitere Informationen finden Sie in Kapitel „Logging, Diagnose und Fehlerbehebung“ auf Seite 517.

Todo-Aktionen (Todo Topics)

Die Management Console stellt eine Liste mit Todo-Aktionen bereit. Diese Liste enthält die wichtigsten Aktivitäten, die ausgeführt werden müssen, um veränderte Konfigurationsdaten der BeanConnect Proxys zu aktivieren. Sie können der Todo-Liste auch Ihre eigenen, persönlichen Aufgaben hinzufügen. Nähere Informationen hierzu finden Sie im Abschnitt „Todo-Aktionen“ auf Seite 171.

5.3 Funktionen der BeanConnect Management Console

Die Management Console unterstützt Sie mit folgenden Funktionen:

- Konfigurationsfunktionen
- Konfigurations-Wizards
- Proxys starten und beenden
- Verfügbarkeit von BeanConnect Komponenten und EIS Partnern überprüfen
- Diagnoseunterstützung
- Todo-Aktionen
- Management Console als JMX-Client

Vermeiden Sie es, einen BeanConnect Proxy von mehreren Management Console-Sitzungen aus gleichzeitig zu konfigurieren oder zu verwalten. Andernfalls könnten die Einstellungen einer Management Console-Sitzung von einer anderen überschrieben und Daten gelöscht werden.

5.3.1 Konfigurationsfunktionen

Die Konfiguration mit der Management Console umfasst die beiden folgenden Aktivitäten:

- BeanConnect Proxys konfigurieren
- EIS Partner konfigurieren

Mit dem Command Line Interface der Management Console können Sie die Konfigurationsfunktionen der Management Console auch per Skript durchführen. Detaillierte Informationen dazu finden Sie in Kapitel „Command Line Interface der BeanConnect Management Console (MC-CLI)“ auf Seite 309.

BeanConnect Proxys konfigurieren

Zum Konfigurieren eines Proxys können Sie die folgenden Aktivitäten von der Management Console aus starten:

1. Konfiguration festlegen und verändern
2. Konfiguration speichern
3. Proxy beenden
4. Proxy-Konfiguration aktualisieren
5. Proxy starten

Führen Sie diese Schritte in der angegebenen Reihenfolge aus, wenn Sie einen EIS Partner in die Management Console integrieren bzw. einen EIS Partner ändern oder entfernen möchten. Wenn Sie einen Outbound Communication Endpoint oder einen Inbound Message Endpoint generieren, ändern oder löschen möchten, können Sie die Konfigurationsdaten aktualisieren, während der Proxy läuft. In diesem Fall müssen Sie nur die Schritte 1 und 2 ausführen. Führen Sie anschließend einen Restart für den Proxy durch.

Die Management Console unterstützt Sie bei der Konfiguration eines Proxys, indem sie die erforderlichen Todo-Aktionen anzeigt (siehe Abschnitt „Todo-Aktionen“ auf Seite 171).

Weitere Einzelheiten zum Konfigurieren von Proxys finden Sie im Abschnitt „BeanConnect Proxy konfigurieren“ auf Seite 185.

EIS Partner konfigurieren

Die Management Console ermöglicht Ihnen, Konfigurationsfragmente für die generierten EIS Partner zu erstellen. Diese Konfigurationsfragmente werden bei der Konfiguration des BeanConnect Proxys erstellt und gespeichert. Es liegt jedoch in Ihrer Verantwortung, folgende Aufgaben auszuführen:

- Kopieren Sie die generierten Dateien, die die Konfigurationsanweisungen für die Anwendungen der EIS Partner enthalten, auf den Rechner, auf dem der EIS Partner läuft.
- Integrieren Sie diese Dateien in die Konfiguration des EIS Partners.

Nähere Informationen hierzu finden Sie in Kapitel „Konfiguration im EIS Partner anpassen“ auf Seite 263.

5.3.2 Konfigurations-Wizards

Die Management Console stellt Wizards für die Konfiguration von einzelnen Proxys und deren Komponententen zur Verfügung. Damit ist es auch für den weniger geübten Nutzer möglich, eine Konfiguration zu erstellen, ohne dass Parameter vergessen werden. Die

Konfigurations-Wizards starten Sie über das Kontextmenü, z.B. indem Sie im Navigationsbaum unter **Configurations Wizards** den Wizard-Teilbaum aufklappen und beim gewünschten Wizard den Kontext-Menübefehl **Start Configuration Wizard** auswählen.

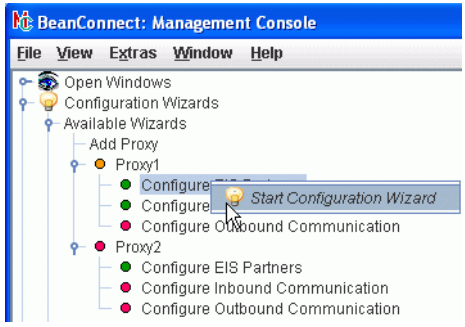


Bild 14: Configuration Wizard starten

Alternativ dazu können Sie auch im Menü **File** den Befehl **Configuration Wizards** wählen.

Der Wizard selber läuft in einem dreiteiligen Arbeitsfenster ab.

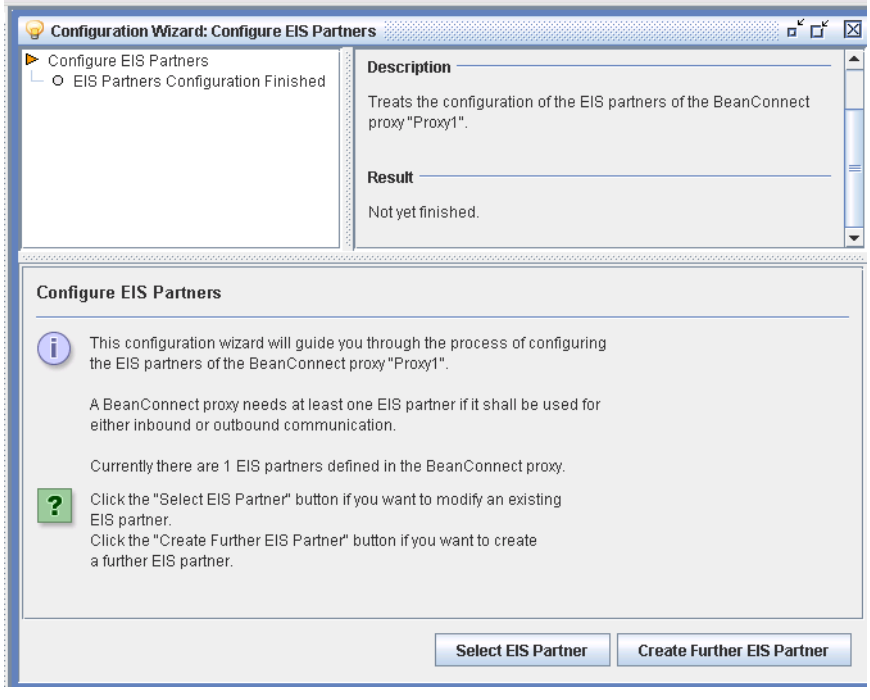


Bild 15: Arbeitsfenster eines Konfigurations-Wizards

Der linke obere Bereich des Wizards (im Beispiel **Configure EIS Partners**) gibt einen Überblick über alle Teilaufgaben des gesamten Konfigurations-Wizards in Form einer Baumstruktur.

Der Bereich rechts daneben beschreibt die Teilaufgabe, die in der Baumstruktur selektiert ist.

Der untere Bereich beschreibt detailliert die aktuelle Teilaufgabe. Sie können eventuell aus mehreren Optionen wählen, um den Konfigurationsprozess zu beeinflussen. Dazu dienen die Schaltflächen am unteren Rand, hier **Select EIS Partner** und **Create Further EIS Partner**.

Ein Konfigurations-Wizard kann vom Anwender jederzeit geschlossen werden, indem das Arbeitsbereichsfenster durch Klicken auf das sich rechts oben befindende Kreuz geschlossen wird.

Ist der Wizard noch nicht vollständig abgeschlossen, dann wird der Anwender darauf hingewiesen und er kann den momentanen Zustand des Wizards abspeichern lassen. Dadurch ist es möglich, den angefangenen Konfigurations-Wizard später (eventuell auch erst in einer weiteren Programmsitzung) fortzusetzen.

Weitere Details zum Arbeitsfenster des Wizards siehe Online-Hilfe.

5.3.3 Proxys starten und beenden

Mit der Management Console können Sie BeanConnect Proxys und Proxy-Komponenten sowohl auf lokalen als auch auf entfernten Rechnern starten und beenden oder einen Restart durchführen.

BeanConnect Proxy starten und beenden

Wählen Sie im Kontextmenü des BeanConnect Proxys den Befehl **Start Proxy** oder **Stop Proxy**, um den Proxy zu starten oder zu beenden.

Bei Proxys für CICS-Partner besteht der Proxy aus mehreren Komponenten. Mit der Management Console können Sie einzelne oder alle Proxy-Komponenten starten oder beenden. Ein eigenes Dialogfeld wird geöffnet, in dem Sie die Komponenten wählen können, die gestartet oder beendet werden sollen. Folgende Komponenten stehen zur Auswahl:

- Proxy-Container
- openUTM-LU62 Gateway (für CICS-Partner)
- SNAP-IX oder IBM Communications Server (für CICS-Partner)

Für CICS-Partner:

Das openUTM-LU62 Gateway und SNAP-IX oder der IBM Communications Server dürfen nicht beendet werden, solange andere Anwendungen gestartet sind, die diese Komponenten verwenden (wie z.B. Proxy-Container).

Die Management Console versucht die Proxy-Komponente nur dann zu starten, wenn die gewünschte Proxy-Komponente nicht bereits läuft.

Restart des BeanConnect Proxy

Wählen Sie im Kontextmenü des BeanConnect Proxys den Befehl **Save/Restart - Restart Proxy**.

Bei Proxys für CICS-Partner wird bei einem Restart, analog zum normalen Starten und Beenden, ein Dialogfeld geöffnet, in dem Sie die Komponenten wählen können, für die ein Restart durchgeführt werden soll.

Beim Restart eines Proxy-Containers werden die einzelnen Prozesse nacheinander beendet und neu gestartet. Während des Restarts bleibt der Proxy-Container verfügbar.

5.3.4 Verfügbarkeit von BeanConnect Komponenten und EIS Partnern überprüfen

Sie können mit der Management Console die Verfügbarkeit eines Proxys und aller Komponenten überprüfen, die diesem Proxy zugeordnet sind. Dazu gehören auch die EIS Partner. Die Prüfung können Sie für den Proxy oder einzeln für die bestimmte Komponenten durchführen.

Dazu wählen Sie im Kontextmenü des gewünschten Objekts den Befehl **Check Availabilty** aus. Dieser Befehl steht für folgende Objekte zur Verfügung:

- Proxy

  Überprüft die Verfügbarkeit des Proxy-Containers sowie aller Komponenten, die dem Proxy zugeordnet sind: alle Resource Adapter, alle MC-CmdHandler auf fernen Rechnern, wenn damit entfernte Proxys administriert werden sowie openUTM-LU62 Gateway und SNAP-IX bzw. IBM Communications Server, falls der Proxy für CICS-Partner konfiguriert ist. Außerdem wird die Verfügbarkeit aller zugeordneten EIS Partner überprüft.

- Resource Adapter

  Überprüft nur die Verfügbarkeit eines bestimmten Resource Adapters. Der zugehörige Proxy-Container muss dazu laufen.

- openUTM-LU62 Gateway / Communication Service

  Überprüft die Verfügbarkeit eines openUTM-LU62 Gateways bzw. eines Communications Services (SNAP-IX, IBM Communications Service). Die Objekte **openUTM-LU62 Gateways** bzw. **Communication Services** sind über die oberste Ebene im Navigationsbaum zugänglich. Sind die Komponenten auf einem fernen Rechner installiert, muss der zugehörige MC-CmdHandler laufen.

- EIS Partner

  Überprüft nur die Verfügbarkeit eines bestimmten EIS Partners. Der zugehörige Proxy-Container muss dazu laufen.

Weitere Einzelheiten finden Sie in Abschnitt „Verfügbarkeit von BeanConnect Proxys überprüfen“ auf Seite 287.

5.3.5 Diagnoseunterstützung

Die Management Console hilft Ihnen, Probleme zu diagnostizieren, die in der Umgebung eines BeanConnect Proxys auftreten. Die Management Console stellt einige Funktionen für folgende Aufgaben bereit:

- Logging und Traces konfigurieren
- Traces und Protokolle sammeln und anzeigen

Weitere Einzelheiten finden Sie in Kapitel „Logging, Diagnose und Fehlerbehebung“ auf Seite 517.

5.3.6 Todo-Aktionen

Die Management Console zeigt in einer Todo-Liste die Aktivitäten an, die ausgeführt werden müssen (z.B. zur Aktualisierung der Konfiguration).

Bei der Todo-Liste handelt es sich um einen Überblick über die durchzuführenden Aktionen. Bitte beachten Sie, dass die Liste auch Todo-Aktionen enthält, die sich auf Aufgaben beziehen, die auf entfernten Servern (Application oder EIS-Servern) ausgeführt werden müssen. Diese Aktionen können nicht vom lokalen Rechner aus gestartet werden. Sie müssen auf dem entfernten Rechner manuell gestartet werden.

Aufgrund bestimmter Aktionen fügt die Management Console der Todo-Liste automatisch Todo-Aktionen hinzu. Einige dieser Todo-Aktionen beziehen sich auf Aufgaben, die Sie mit der Management Console ausführen. In diesen Fällen werden die betreffenden Punkte automatisch aus der Liste entfernt, nachdem die damit verbundenen Aktionen erledigt wurden. Diese Aktionen können Sie auch direkt von der Todo-Liste aus ausführen.

Darüber hinaus kann die Todo-Liste Aktionen enthalten, die Sie nicht mit der Management Console ausführen können, wie z.B. die Generierungsanweisungen in die Konfiguration des EIS Partners einzubringen, die von der Management Console zur Konfiguration eines EIS Partners erstellt wurden. In diesen Fällen kann die Management Console nicht erkennen, ob Sie die Aktion durchgeführt haben oder nicht. Folglich müssen Sie nach Beendigung einer Aktion dieses Typs die entsprechende Todo-Aktion eigenhändig aus der Liste entfernen.

Sie können auch Ihre eigenen persönlichen Todo-Aktionen erstellen und sie der Liste hinzufügen. So müssen Sie diese nicht an anderer Stelle notieren.

5.3.7 MC-CLI Recording: Aktionen der Management Console mitschneiden

Unter MC-CLI Recording versteht man das Mitschneiden von Aktionen der Management Console. Die Mitschnitte werden in Form von MC-CLI-Funktionen gespeichert. Damit lassen sich Konfigurations- und Administrationsaufgaben aufzeichnen. Die aufgezeichneten Konfigurationsschritte können Sie für andere Proxys oder untergeordnete Objekte anpassen und die Konfiguration dieser Objekte in Jython-Skripten abwickeln. Die Mitschnitte können auch einfach nur zum Protokollieren der durchgeführten Aktionen genutzt werden.

Das MC-CLI Recording ist während einer Sitzung der Management Console immer eingeschaltet. Für alle Aktionen, die in der Management Console ausgeführt werden, werden die entsprechenden MC-CLI-Aufrufe in folgenden internen Puffern mitgeschnitten:

- ein Puffer „Console“ für alle Proxy-übergreifenden Aktionen
- je ein eigener Puffer pro Proxy bzw. Proxy-Cluster

Diese Puffer können Sie im internen Editor der Management Console ansehen und bei Bedarf auf Datei schreiben. Das Schreiben auf Datei ist in der Management Console konfigurierbar.

Es werden nur die Aktionen mitgeschnitten, für die es entsprechende Aufrufe im MC-CLI gibt. Wenn in einem Eigenschaftsdialog nur Eigenschaften geändert wurden, die nicht vom MC-CLI Recording unterstützt werden, wird beim Verlassen des Dialoges so verfahren, als ob nichts geändert wurde. Es wird ein Kommentar ausgegeben, dass nichts geändert wurde, und es wird kein `modifyProperties`-Aufruf mitgeschnitten.

Ausgabe des Mitschnitts manuell steuern

Sie können Mitschnitte für Aktionen eines Proxys, eines Proxy Clusters oder für Aktionen für die Console im Textfenster der Management Console ansehen, explizit auf Datei schreiben oder die angesammelten Mitschnitte löschen. Die Mitschnittausgabe steuern Sie wie folgt:

- ► Aktionen eines Proxys oder Proxy Clusters: Kontextmenü-Befehl **Mccli Recording of Proxy/Proxy Cluster** (+ Unterbefehl **Show/Save/Clear ...**) des entsprechenden Proxy bzw. ProxyClusters.
- ► Aktionen für die Console: Menü **Extra**, Befehl **Show/Save/Clear Mccli Recording of Console**.

Ausgabe des Mitschnitts auf Datei konfigurieren

Sie können einstellen, ob und ggf. wann ein Mitschnitt auf Datei gesichert werden soll: nur bei Bedarf (**on demand**), immer nach Sitzungsende (**at end**) oder nach jeder Aktion (**unbuffered**). Gehen Sie wie folgt vor:

- ► Wählen Sie im Menü **Extra** den Befehl **Settings - Logging/Traces** und aktivieren Sie im Bereich **Mccli Recording** den gewünschten Modus

Bitte beachten Sie Folgendes:

- Beim Löschen eines Proxys oder Proxy Clusters wird auch der entsprechende interne MC-CLI Mitschnitt gelöscht. Wenn der Modus **on demand** eingestellt ist (Standardeinstellung) und Sie die Mitschnitte sichern möchten, müssen Sie die Mitschnitte explizit vor dem Löschen auf Datei schreiben.
- Wenn Sie sehr umfangreiche Sitzungen protokollieren möchten, dann sollten Sie den Modus **unbuffered** einstellen.
  Der Grund: BeanConnect arbeitet mit zwei temporären Puffern (aktueller Puffer und Sicherungspuffer) mit einer Kapazität von jeweils ca. 1000 Zeilen. Werden mehr Daten mitgeschnitten als diese beiden Puffer zusammen aufnehmen können, können Daten verloren gehen, wenn nicht zwischendurch auf Datei gesichert wird.

Name der Mitschnitt-Datei

Eine Mitschnitt-Datei wird im Verzeichnis `<MC_home>/cli-rec` unter folgenden Namen abgelegt:

`<time>.Console.py`

für den Mitschnitt der Console-Aktionen

`<time>.<proxy_id>.py`

für den Mitschnitt der Aktionen des Proxys mit der ID `<proxy_id>`

`<time>.<cluster_id>.py`

für den Mitschnitt der Aktionen des Proxy-Clusters mit der ID `<cluster_id>`

Dabei ist `<proxy_id>` die eindeutige ID des Proxys, `<cluster_id>` die eindeutige ID des Clusters und `<time>` der aktuelle Zeitstempel beim Erstellen der Mitschnitt-Datei.

Falls schon eine Datei mit dem Namen `<time>.<id>.py` vorhanden ist, wird diese umbenannt in `<time>.<id>.py.old`. Eine Datei mit dem Namen `<time>.<id>.py.old` wird ohne Warnung überschrieben.

Weitere Informationen zum Format einer Mitschnitt-Datei finden Sie im Abschnitt „Erstellen von Jython Skripts aus MC-CLI-Mitschnitten“ auf Seite 437.

BeanConnect löscht die Mitschnitt-Dateien nicht. Überprüfen Sie daher regelmäßig, welche Dateien Sie noch benötigen. Nicht mehr benötigte Mitschnitt-Dateien müssen Sie manuell löschen.

5.3.8 Cluster-Unterstützung

Ein Proxy Cluster besteht aus einem oder mehreren Proxys. Sie können über die Management Console einen Proxy Cluster konfigurieren und administrieren. Dabei gehen Sie ähnlich vor wie beim Konfigurieren und Administrieren eines einzelnen Proxys.

Viele Menübefehle und Tätigkeiten sind gleich, denn der Proxy Cluster verhält sich in den meisten Fällen wie ein einzelner Proxy.

Im Proxy Cluster werden keine Konfigurations-Wizards unterstützt.

BeanConnect Proxy Cluster konfigurieren

Sie konfigurieren einen Proxy Cluster mit mehreren Proxys in folgenden Schritten:

1. Konfigurieren Sie einen einzelnen Proxy, der als Basis für den Proxy Cluster dient.
2. Im Kontextmenü dieses Proxys wählen Sie den Befehl **Define Proxy Cluster**. Damit definieren Sie den Proxy Cluster und nehmen gleichzeitig den Proxy in diesen Cluster auf. Dieser Proxy wird durch diese Aktion automatisch zum Master-Proxy des neuen Clusters.
3. Nehmen Sie weitere Proxys in die Management Console auf. Diese müssen nicht konfiguriert werden, denn die Konfigurationseigenschaften werden beim Aufnehmen in den Cluster überschrieben, da sie mit dem Master-Proxy synchronisiert werden.
4. Nehmen Sie die anderen Proxys in den Cluster auf, indem Sie im Kontextmenü den Befehl **Add to Proxy Cluster** wählen.
5. Speichern Sie die Cluster-Konfiguration. Die Konfigurationsdaten der Proxys werden dabei synchronisiert.

Sie können die Konfigurationseigenschaften eines Proxy Clusters innerhalb des Cluster-Verbundes ändern, z.B. EIS Partner, Inbound Services oder Outbound Services ändern oder hinzufügen.

Für den Proxy Cluster bietet die Management Console zusätzlich eine manuelle Synchronisierungsfunktion (Befehl **Synchronize Proxy Cluster** im Kontextmenü des Proxy Clusters). Diese Funktion wird z.B. dann benötigt, wenn ein Cluster Proxy eine zeitlang nicht administrierbar war (weil z.B. der betreffende Rechner nicht gelaufen ist) und sich die Konfiguration des Proxy Clusters während dieser Zeit geändert hat. Durch das explizite Synchronisieren wird dann die Konfiguration des einen Cluster Proxys wieder an die Konfiguration des Proxy Clusters angeglichen.

Weitere Informationen finden Sie in Abschnitt „BeanConnect Proxy Cluster konfigurieren“ auf Seite 197.

BeanConnect Proxy Cluster starten und beenden

Das Starten, Beenden und der Restart eines Proxy Clusters verläuft entsprechend zum Starten, Beenden und Restart eines einzelnen Proxys. Wählen Sie im Kontextmenü des Proxy Cluster den entsprechenden Befehl aus:

Start Proxy Cluster zum Starten,
Stop Proxy Cluster zum Beenden
Save/Restart - Restart Proxy Cluster zum Restart

Verfügbarkeit überprüfen

Die Verfügbarkeit des Proxy Clusters oder seiner Komponenten überprüfen Sie wie beim einzelnen Proxy, indem Sie den Befehl **Check Availability** im jeweiligen Kontextmenü auswählen.

Im Proxy Cluster bewirkt der Befehl, dass alle Proxy-Container sowie alle zugehörigen Komponenten geprüft werden, d.h. Resource Adapter, EIS Partner, MC-CmdHandler auf fernen Rechnern sowie openUTM-LU62 Gateway und Communication Service bei Anschluss von CICS-Partnern.

Bei der einzelnen Komponente bewirkt der Befehl immer nur die Prüfung dieser Komponente. Das openUTM-LU62 Gateway und der Communication Service wird wie beim Einzel-Proxy auf der obersten Ebene aufgeführt.

5.3.9 Management Console als JMX-Client

Die Management Console enthält auch einen JMX-Client, über den Sie einen Resource Adapter überwachen können. Es können mehrere JMX-Clients konfiguriert werden.

Ein JMX-Client kann einem Resource Adapter zugeordnet werden oder er kann als freier JMX-Client konfiguriert werden. Für einen Resource Adapter definieren Sie den JMX-Client im Kontextmenü des Resource Adapters per Befehl **Define MBean Client** und legen die Eigenschaften in einem Folgedialog fest, siehe Abschnitt „JMX-Client in der Management Console einrichten“ auf Seite 258.

Ein JMX-Client kommuniziert mit dem JMX-Server. Dieser läuft auf dem Application Server und stellt die Daten durch so genannte MBeans zur Verfügung, weshalb der JMX-Client auch MBean-Client genannt wird. Damit können Sie folgende Objekte überwachen:

- Resource Adapter
- Connection Factories
- Inbound-Verbindungen
- Message Endpoints

Der MBean-Client kann die Daten und Attribute lesen, die von den MBeans zur Verfügung gestellt werden; auf einige Attribute hat er auch schreibenden Zugriff. Es werden folgende Überwachungsfunktionen angeboten:

- Attribute der MBeans anzeigen

  Attribute sind z.B. die Konfigurationseigenschaften oder bestimmte statistische Werte der jeweiligen MBeans.

- Notifications lesen, die die MBeans ausgeben

  Notifications müssen zuerst über die MBeans abonniert (subscribe) werden.

  Notifications sind Meldungen, die der Application Server bei bestimmten Ereignissen erzeugt. Notifications werden außerdem vom Resource Adapter und von den Komponenten des Application Server erzeugt, die die MBeans zur Verfügung stellen.

- Statistikwerte sammeln und ausgeben

  Attributwerte können in festen Zeitintervallen eingesammelt und als Statistiken ausgegeben werden. Dazu gehören z.B. die Anzahl aktiven Verbindungen oder die Anzahl der zurückgesetzten Transaktionen.

- Operationen ausführen

  Operationen sind spezifische von den MBeans implementierte und über die Oberfläche zugängliche Funktionen; für Connection Factories sind das z.B. die Funktionen `cleanupPool` und `resetStatisticValues`.

Sie können für jeden MBean-Client per Administration einstellen, welche Überwachungsfunktionen Sie nutzen möchten (Notifications, Statistiken etc.). Dazu muss die Verbindung zwischen Management Console und dem JMX-Server auf dem Application Server bestehen. Außerdem können Sie Favoriten definieren, um auf häufig genutzte MBeans einfach zugreifen zu können.

Näheres darüber, wie Sie die Überwachungsfunktionen definieren und aktivieren, finden Sie im Abschnitt „Überwachen des Resource Adapters mit der Management Console“ auf Seite 296.

5.4 Verwaltungsdaten der Management Console

Die Management Console verwendet die nachfolgend aufgelisteten Dateien:

- `console.properties.xml`
- `log4j.properties.xml`

console.properties.xml

Diese XML-Datei befindet sich im Unterverzeichnis `config` der Management Console. Die Datei `console.properties.xml` enthält die Verwaltungsdaten aller der Management Console bekannten BeanConnect Proxys. Darüber hinaus werden in dieser Datei zusätzliche Einstellungen für die Management Console gespeichert.

Bei Bedarf wird die Datei `console.properties.xml` automatisch von der Management Console aktualisiert oder erweitert. Es ist nicht notwendig, die aktualisierten Daten explizit zu speichern.

log4j.properties.xml

Diese XML-Datei befindet sich im Unterverzeichnis `config` der Management Console. Bei der Datei `log4j.properties.xml` handelt es sich um die von der Management Console verwendete Log4j-Konfigurationsdatei. Sie können die Datei in der Management Console im Dialogfeld **Settings** aktualisieren. Wählen Sie **Extras** - **Settings...**, um dieses Dialogfeld zu öffnen.

6 BeanConnect konfigurieren

Der BeanConnect Proxy kommuniziert auf der einen Seite mit dem BeanConnect Resource Adapter, der im Application Server läuft, und auf der anderen Seite mit dem EIS Partner. Der Proxy wird nicht für Outbound-Kommunikation über UPIC verwendet.

Damit eine Kommunikation zwischen einer im Application Server deployten EJB und einer Partneranwendung gewährleistet ist, müssen der Proxy und die Proxy-Komponenten ordnungsgemäß konfiguriert werden.

Mit der BeanConnect Management Console können Sie den BeanConnect Proxy-Container und die Proxy-Komponenten konfigurieren. Darüber hinaus können Sie die Konfiguration von Proxy Clustern, EIS Partnern, Outbound Services, Outbound Communication Endpoints, Inbound Message Endpoints, Inbound Services und Management Console Command Handlern (MC-CmdHandler) vornehmen. Die Management Console kann auch als JMX-Client zur Anzeige von MBeans konfiguriert werden.

Sie können neue Objekte hinzufügen und bereits bestehende Objekte ändern oder löschen.

Dieses Kapitel enthält Informationen zu folgenden Themen:

Weitere Einzelheiten zur Verwendung der BeanConnect Management Console finden Sie in Kapitel „BeanConnect Management Console - Überblick“ auf Seite 157 sowie in der Online-Hilfe.

6.1 Konfigurationsschritte

Die folgenden Schritte müssen für die Konfiguration eines Proxys mit Hilfe der Management Console ausgeführt werden:

1. Proxy zu den Konfigurationsdaten der Management Console hinzufügen

   Jeder Proxy muss der Management Console bekannt gemacht werden, bevor er administriert werden kann. Wie Sie dabei vorgehen, ist im Abschnitt „BeanConnect Proxy zur Management Console hinzufügen“ auf Seite 181 beschrieben.

2. Proxy und seine Komponenten konfigurieren

   Beim Konfigurieren des Proxys und der Proxy-Komponenten müssen Sie alle Festlegungen treffen, damit die Partner miteinander kommunizieren können. Dazu gehören:

   – die Identität des Proxys (Abschnitt „Allgemeine Proxy-Daten“ auf Seite 186).

   – bei CICS-Partnern zusätzlich die Einstellungen für die Kommunikation mit dem openUTM-LU62 Gateway und dem Communication Service (siehe Abschnitt „Proxy-Komponenten CICS-Partner“ auf Seite 189), sowie Angaben zur Netzanbindung des CICS-Partners, siehe „Registerkarte CICS Partner“ auf Seite 229.

   – die Einstellungen für die Kommunikation mit dem Resource Adapter, der dem Proxy zugewiesen ist (siehe Abschnitt „BeanConnect Resource Adapter konfigurieren“ auf Seite 202).

3. EIS Partner und Kommunikationsobjekte konfigurieren

   Mit diesem Konfigurationsschritt definieren Sie die Kommunikationsbeziehungen zwischen dem Proxy und den EIS Partnern für Inbound- und Outbound-Kommunikation. Folgende Objekte müssen erstellt werden:

   – EIS Partner (siehe Abschnitt „EIS Partner konfigurieren“ auf Seite 212)

   – Outbound Services und Outbound Communication Endpoints (siehe Abschnitt „Outbound-Kommunikation konfigurieren“ auf Seite 235)

   – Inbound Users, Inbound Services und Inbound Message Endpoints (siehe Abschnitt „Inbound-Kommunikation konfigurieren“ auf Seite 240)

4. Konfiguration speichern und aktivieren

   Nachdem alle notwendigen Informationen angegeben wurden, müssen Sie die Konfiguration speichern. Die Properties der Konfigurationsobjekte werden von der Management Console gespeichert, so dass sie in späteren Management Console-Sitzungen zur Verfügung stehen. Anschließend erstellt die Management Console die Konfigurationsdateien, die vom Proxy, den Proxy-Komponenten und dem EIS Partner verwendet werden sollen. Abschließend muss die neue Konfiguration aktiviert werden.

   Diese Aufgaben werden im Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249 beschrieben.

   Wenn Sie einen Proxy Cluster bilden möchten, finden Sie die zugehörigen Informationen im Abschnitt „BeanConnect Proxy Cluster konfigurieren“ auf Seite 197.

   Zum Konfigurieren der Management Console als JMX-Client siehe Abschnitt „Management Console als JMX-Client konfigurieren“ auf Seite 255.

6.2 BeanConnect Proxy zur Management Console hinzufügen

Alle der Management Console bekannten Proxys, die keinem Proxy Cluster zugeordnet sind, werden mit den konfigurierten EIS Partnern und Kommunikationsobjekten im Navigationsbaum der Management Console unterhalb des Knotens **BeanConnect Proxies** angezeigt.

Mit einer Management Console können mehrere Proxys verwaltet werden.

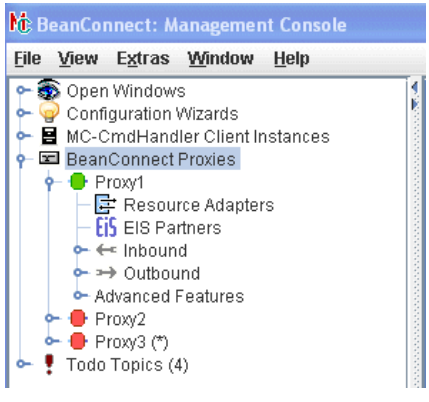


Bild 16: Proxys innerhalb der Management Console

Um auf die Daten eines Proxys zugreifen zu können, müssen Sie das Administrationspasswort eingeben. Dieses lautet standardmäßig **admin.** Das Administrationspasswort können Sie ändern, siehe Abschnitt „Administration-Passwort ändern“ auf Seite 194.

6.2.1 Neuen Proxy hinzufügen

So fügen Sie dem Navigationsbaum einen neuen Proxy hinzu:

- Befindet sich der Proxy auf demselben Rechner und derselben Kennung wie die Management Console, öffnen Sie das Kontextmenü des Knotens **BeanConnect Proxies** und wählen Sie den Befehl **Add Local Proxy**. Die Verwendung derselben Kennung ist dann nicht notwendig, wenn die Zugriffrechte so gesetzt sind, dass die Management Console auf alle benötigten Dateien des Proxys in der Kennung, unter der der Proxy läuft, Zugiff hat.
- Befindet sich der Proxy auf einem anderen Rechner, öffnen Sie das Kontextmenü des Knotens **BeanConnect Proxies** und wählen Sie den Befehl **Add Remote Proxy**. Sie müssen den Namen des Rechners eingeben, auf dem der Proxy-Container läuft.

  Damit Sie einen Proxy auf einem entfernten Rechner hinzufügen und verwalten können, muss der MC-CmdHandler des Proxys auf dem Proxy-Rechner laufen (siehe Abschnitt „Management Console Command Handler (MC-CmdHandler) konfigurieren“ auf Seite 251). Das Passwort des verwendeten MC-CmdHandler und das Administrations-Passwort des hinzuzufügenden Proxys müssen identisch sein!

Geben Sie bei **Host** den Namen des entfernten Rechners und bei **MC-CmdHandler Listener Port** den Listener Port des MC-CmdHandlers auf dem fernen Rechner ein.

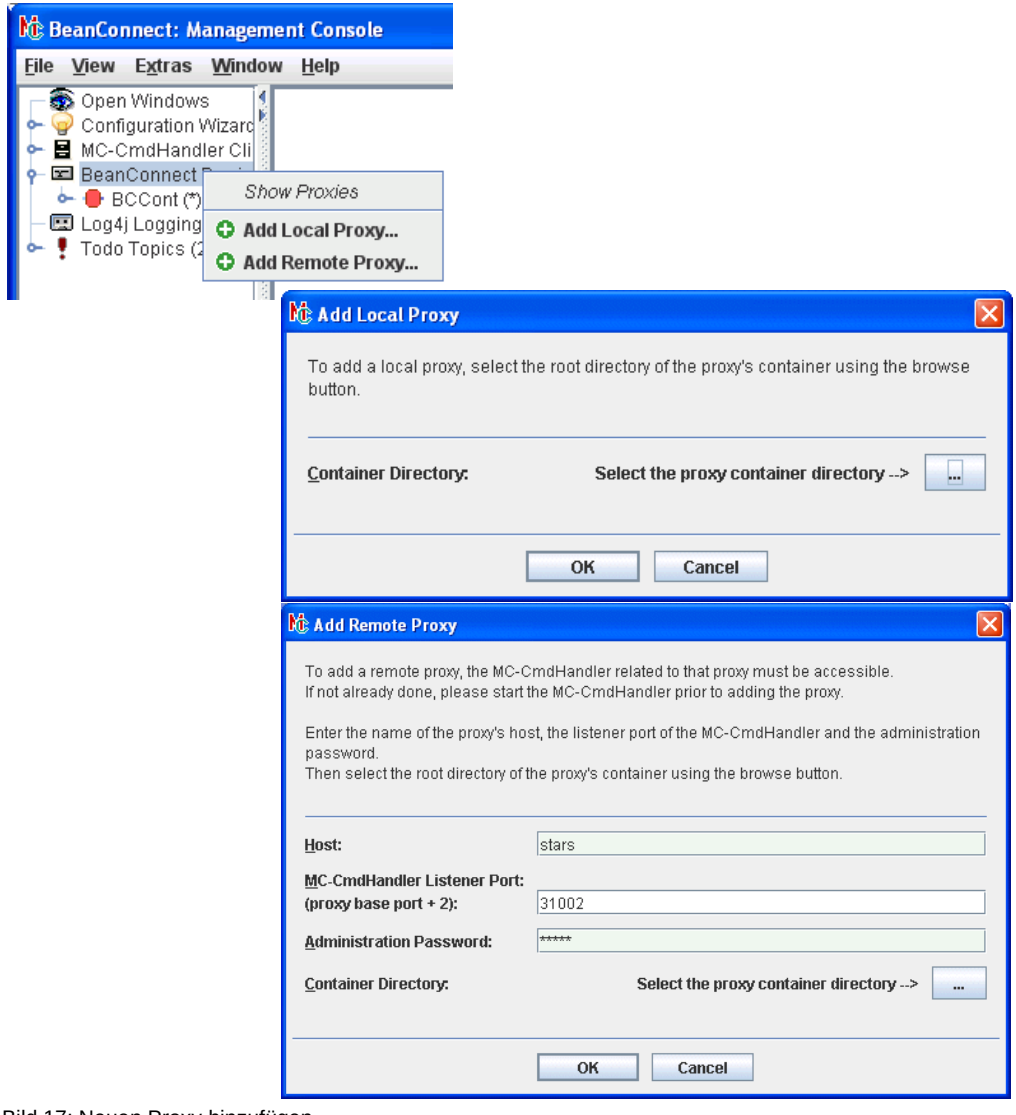


Bild 17: Neuen Proxy hinzufügen

Wenn Sie einen Proxy hinzufügen, wird ein Dialogfeld für die Auswahl des Proxy-Containers geöffnet. Durchsuchen Sie das Dateisystem nach dem Container-Verzeichnis.

Beide Menüeinträge befinden sich auch im Menü **File** und können auch von dort aus aktiviert werden.

Durch das Hinzufügen von Proxys in die Konfigurationsdaten der Management Console werden keine neuen Proxys installiert. Sie können nur Proxys hinzufügen, die bereits installiert sind.

Ein Proxy, der der Management Console hinzugefügt wurde, kann erst dann vollständig verwaltet werden, wenn alle für den Proxy und die Proxy-Komponenten notwendigen Parameter definiert wurden (siehe Abschnitt „BeanConnect Proxy konfigurieren“ auf Seite 185). Wenn Sie versuchen, eine Funktion aufzurufen, die nicht verfügbar ist, da die notwendige Konfiguration noch nicht abgeschlossen wurde, wird eine entsprechende Meldung ausgegeben.

Ein Proxy, der in demselben BeanConnect Home-Verzeichnis wie die Management Console installiert wurde, wird beim nächsten Start der Management Console erkannt und automatisch den Konfigurationsdaten hinzugefügt. Dies geschieht auch, wenn der Proxy nach der Management Console auf dem System installiert wurde.

6.2.2 Proxy entfernen

Einen Proxy entfernen Sie aus den Konfigurationsdaten der Management Console, indem Sie das Kontextmenü des Proxys öffnen und den Befehl **Remove Proxy** wählen.

Nachdem Sie die Eingabeaufforderung bestätigt haben, wird der Proxy aus den Konfigurationsdaten gelöscht. Der Proxy selbst wird jedoch nicht verändert und vor allem nicht deinstalliert.

6.3 BeanConnect Proxy konfigurieren

Wenn Sie der Management Console einen neuen Proxy hinzufügen, wird ein Eigenschaftsfeld angezeigt, in das Sie die Konfigurationsdaten für die Proxy-Komponenten eingeben. Die Konfigurationsdaten eines bereits bestehenden Proxys ändern Sie, indem Sie im Navigationsbaum das Kontextmenü des gewünschten Proxys öffnen und den Befehl **Edit Properties** wählen.

Das Eigenschaftsfeld besteht aus mehreren Seiten. Sie werden nachfolgend beschrieben.

6.3.1 Allgemeine Proxy-Daten

Die Registerkarte **General** enthält einige allgemeine Daten, die es der Management Console ermöglichen, den Proxy zu identifizieren und auf ihn zuzugreifen .

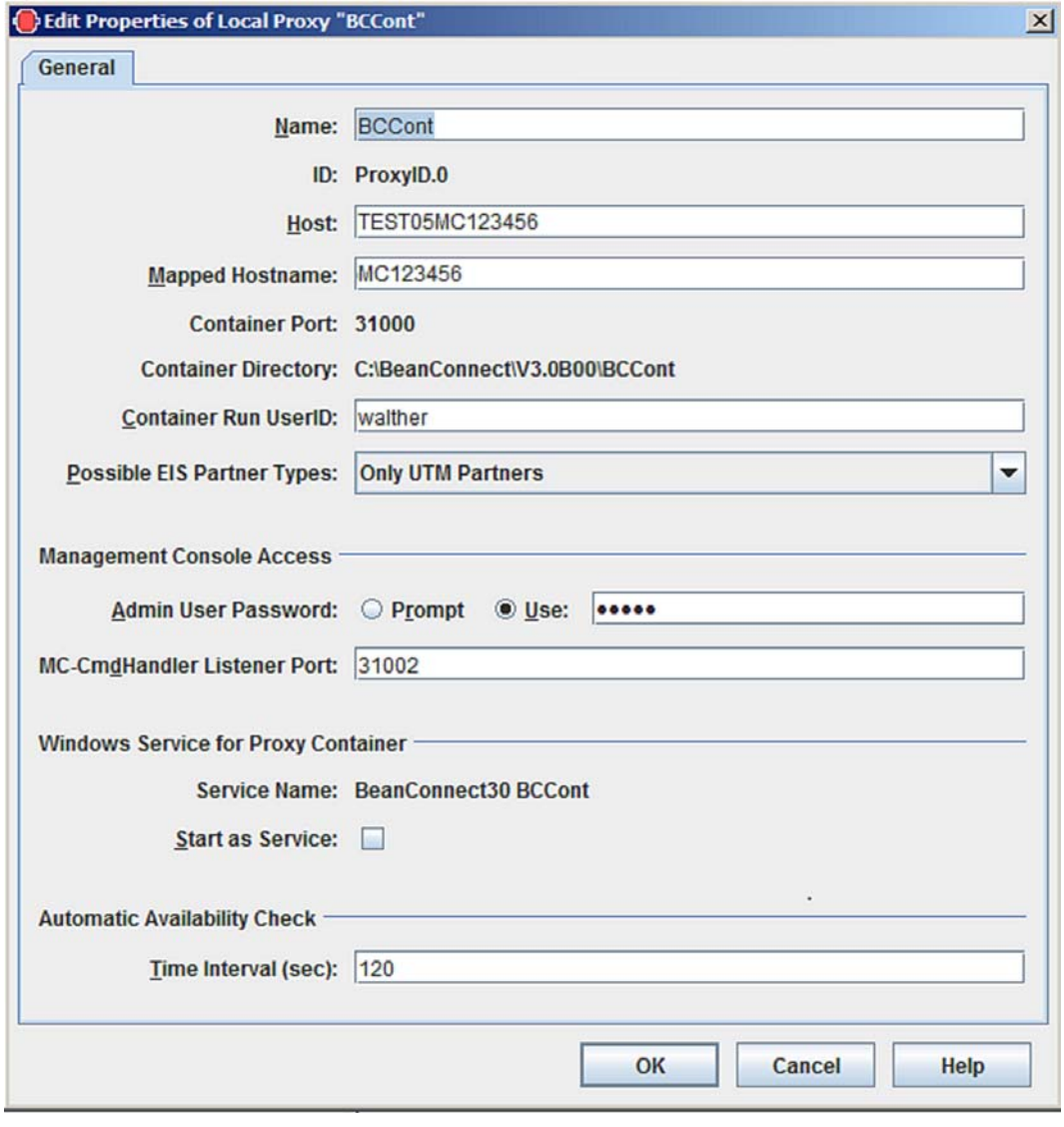


Bild 18: Allgemeine Proxy-Daten

Name

Name des Proxys. Dieser Name wird nur intern von der Management Console zur Unterscheidung der verwalteten Proxys verwendet.

ID

Die Management Console weist jedem Proxy eine **ID** in der Form `ProxyID.<number>` zu. Diese ID wird nach dem erstmaligen Hinzufügen eines Proxys im Dialogfeld angezeigt und kann nicht geändert werden. Die Management Console verwendet diese ID als Namensbestandteil für erstellte Konfigurationsdateien.

Host

Name (max. 63 Zeichen) oder IP-Adresse des Rechners, auf dem der Proxy installiert ist.

Mapped Hostname

Mapped Name des Rechners, auf dem sich der Proxy befindet (max. 8 Zeichen).

Container Port / Container Directory

Diese Felder zeigen die Portnummer und das Home-Verzeichnis des Proxys an. Die Portnummer wird bei der Installation festgelegt.

Container Run UserID

Benutzerkennung, unter der der Proxy Container gestartet wird. Die Management Console vergleicht die hier angegebene Benutzerkennung mit der Benutzerkennung, unter der der MC-CmdHandler läuft, der zur Administration des Proxy Containers verwendet wird. Nur wenn beide Benutzerkennungen identisch sind, ist der Proxy Container aus Sicht der Management Console mit dem verwendeten MC-CmdHandler administrierbar.

Bei lokalen Proxys beachten Sie bitte, dass in bestimmten Situationen, z.B. wenn der MC-CmdHandler des Proxy Containers nicht gestartet ist und der Listener Port dieses MC-CmdHandlers dem Proxy nicht zugewiesen wurde, der "interne" MC-CmdHandler verwendet wird, der im Prozess der Management Console läuft und ohne Kommunikation funktioniert. In diesem Fall muss die hier angegebene Benutzerkennung mit der Benutzerkennung übereinstimmen, unter der die Management Console selbst gestartet wurde.

Possible EIS Partner Types

Gibt an, für welchen Typ von EIS Partnern der Proxy konfiguriert wird. Wenn Sie **Only UTM Partners** auswählen, kann der Proxy nur mit Partnern vom Typ openUTM kommunizieren. Wenn Sie **Only CICS Partners** auswählen, kann der Proxy nur mit Partnern vom Typ CICS kommunizieren. Bei der Option **UTM and CICS Partners** ist die Kommunikation mit beiden Partnertypen möglich. Für die Kommunikation mit CICS-Partnern wird die zusätzliche Registerkarte **Proxy Components** angezeigt.

Für die Kommunikation mit den Partnertypen UTM und CICS müssen jeweils separate Nutzungsrechte erworben werden.

Management Console Access

Im Feld **Admin User Password** geben Sie das Administrationspasswort an, das die Management Console verwendet, um auf den Proxy-Container und den MC-CmdHandler des Proxy-Containers zuzugreifen. Dieses Passwort muss mit dem des Proxy-Containers und des zugehörigen MC-CmdHandlers übereinstimmen. Anstatt hier ein Passwort einzugeben, können Sie auch die Option **Prompt** wählen. In diesem Fall wird das Passwort abgefragt, sobald innerhalb einer Management Console-Sitzung zum ersten Mal auf irgendwelche Proxy-Container-Daten zugegriffen wird.
Das Feld **MC-CmdHandler Listener Port** definiert den Listener Port des MC-CmdHandlers, mit dem der Proxy administriert wird. Standard ist der Wert aus **Container Port** + 2.

Windows Service for Proxy Container

Nur wenn der Proxy-Container auf einem Windows-System läuft: Proxys können auch als Windows-Dienst gestartet werden. Wählen Sie hierzu die Option **Start as Service**. Der Name, unter dem der Proxy als Dienst gestartet werden kann, wurde bei der Installation festgelegt und wird im Feld **Service Name** angezeigt. Wenn der Proxy als Windows-Dienst betrieben wird, kann er nicht im Debug-Modus gestartet werden.

Automatic Availability Check

Zuletzt können Sie im Feld **Time Interval (sec)** das Zeitintervall für die automatische Verfügbarkeitsprüfung einstellen. Die Werte sollten nicht zu klein gewählt werden, damit die Management Console (und die Proxys) nicht zu sehr mit den Verfügbarkeitsprüfungen ausgelastet wird. Empfohlen werden Werte ab 180 Sekunden.

Es wird keine automatische Verfügbarkeitsprüfung durchgeführt, wenn Sie in das Feld nichts eintragen oder den Wert 0 eingeben.

6.3.2 Proxy-Komponenten CICS-Partner

Auf der Registerkarte **Proxy Components** im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** werden die Einstellungen für die Proxy-Komponenten openUTM-LU62 Gateway und den Communication Service definiert.

Diese beiden Proxy-Komponenten werden für die Kommunikation mit CICS-Partnern benötigt und müssen immer auf dem selben Rechner ablaufen. Dies muss nicht der Rechner sein, auf dem der Proxy läuft.

Die nachfolgend gezeigte Registerkarte enthält schon die notwendigen Einträge. Beim ersten Einrichten müssen Sie die Daten für das openUTM-LU62-Gateway und den Communication Service über Zwischendialoge eintragen.

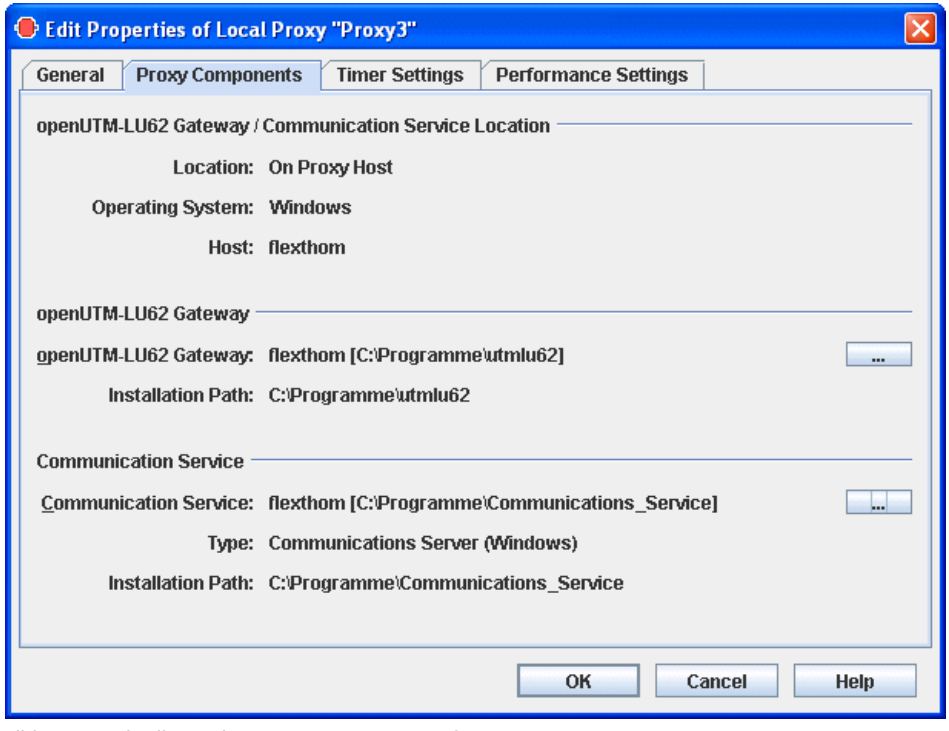


Bild 19: Beschreibung der Proxy-Komponenten für CICS-Partner

openUTM-LU62 Gateway / Communication Service Location

Zeigt den Ort, den Namen und das Betriebssystem des Rechners an, auf dem die Proxy-Komponenten openUTM-LU62-Gateway und Communication Service ablaufen. **Location: On Proxy Host** bedeutet: derselbe Rechner, auf dem der Proxy-Container läuft. **Location: On Separate Host** bedeutet: anderer Rechner.

openUTM-LU62 Gateway

Gibt den Rechner und das Verzeichnis an, in dem das openUTM-LU62 Gateway installiert ist. Um ein Gateway auszuwählen, klicken Sie auf die Schaltfläche **...** und öffnen Sie damit folgenden Zwischendialog:

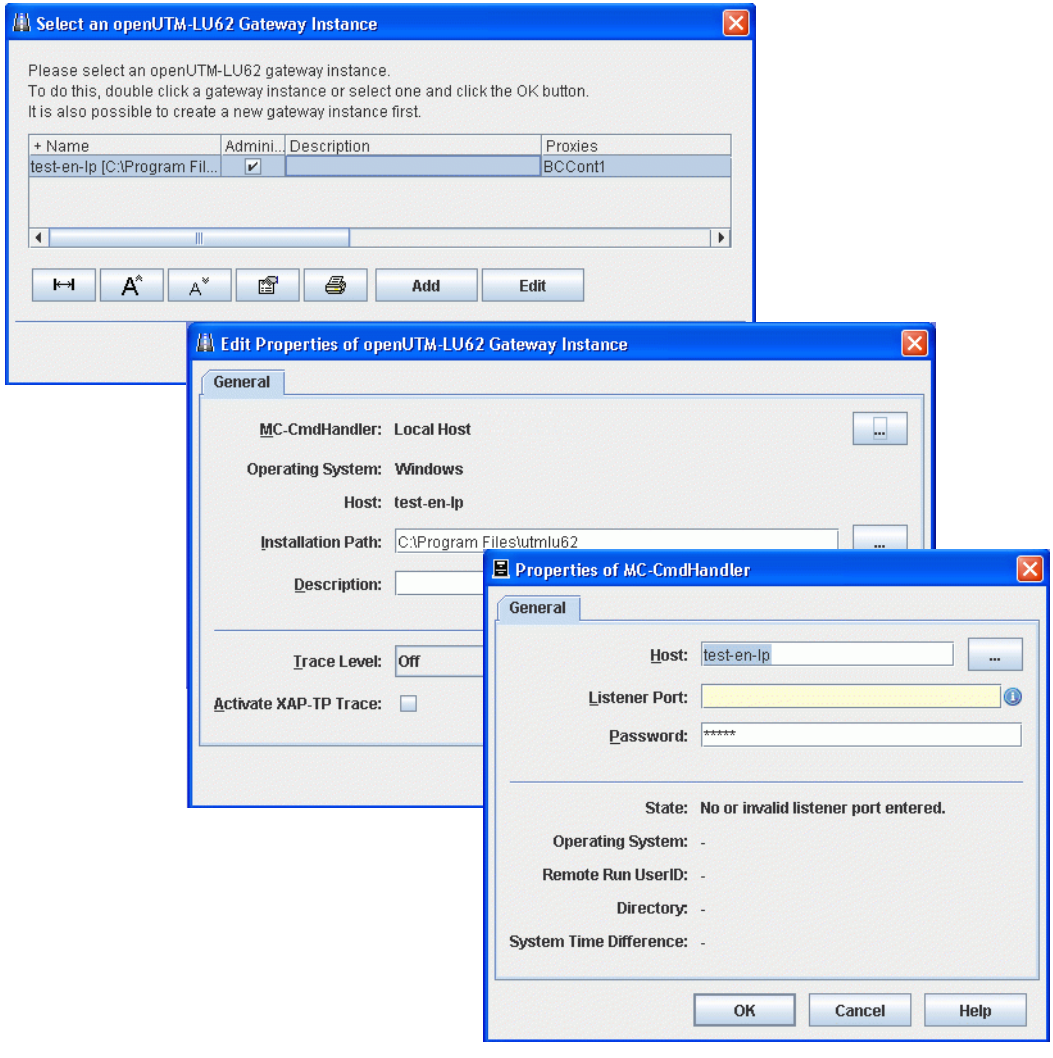


Bild 20: openUTM-LU62 Gateway konfigurieren

In **Select an openUTM-LU62 Gateway Instance** werden die bereits konfigurierten Gateway Instanzen angezeigt. Wenn Sie eine Instanz auswählen und **OK** klicken, dann wird diese Instanz zugeordnet. Mit **Edit** können Sie die Eigenschaften einer ausgewählten Instanz ändern.

Wenn Sie eine neue Instanz konfigurieren möchten, gehen Sie wie folgt vor:

- Klicken Sie auf die Schaltfläche **Add**.
- Öffnen in **Add openUTM-LU62 Gateway Instance** über die Schaltfläche **...** bei **Select MC-CmdHandler** den Dialog **Properties of MC-CmdHandler**.
- Tragen Sie in **Properties of MC-CmdHandler** die Eigenschaften des MC-Handlers ein, mit dem die Komponenten administriert werden, und klicken Sie **OK**.
- Tragen Sie in **Add openUTM-LU62 Gateway Instance** den Installationspfad des openUTM-LU62 Gateways ein und modifizieren Sie ggf. die Trace-Eigenschaften. Klicken Sie **OK**.

Communication Service

Gibt den Rechner, den Typ und das Verzeichnis an, in dem der Communication Service installiert ist. Um einen Communication Service auszuwählen, klicken Sie auf die Schaltfläche **...** und öffnen Sie damit folgenden Zwischendialog. Gehen Sie dabei vor wie beim Eintragen eines openUTM-LU62 Gateways:

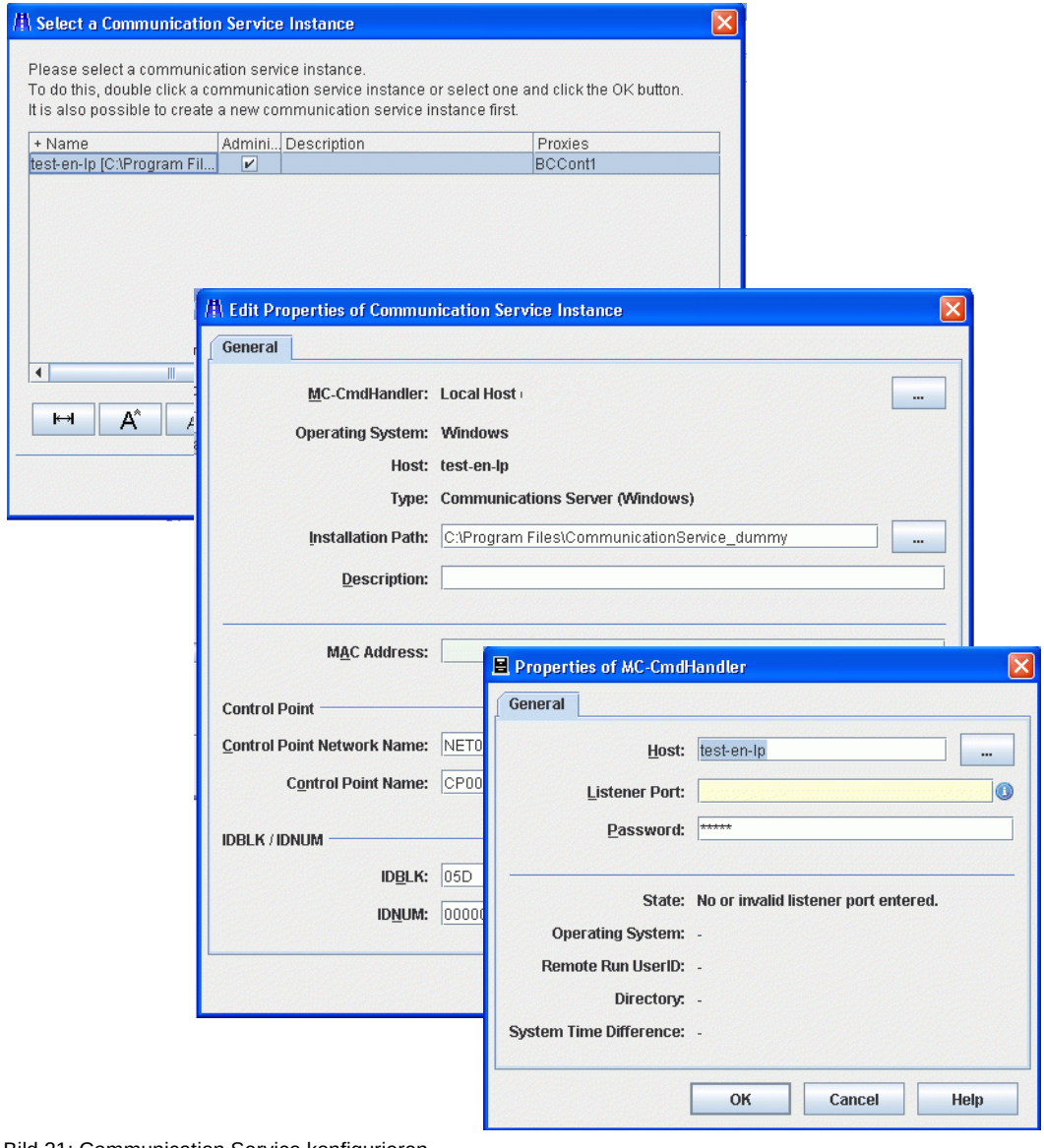


Bild 21: Communication Service konfigurieren

In **Select a Communication Service Instance** werden die bereits konfigurierten Communication Service Instanzen angezeigt. Wenn Sie eine Instanz auswählen und **OK** klicken, wird diese Instanz zugeordnet. Mit **Edit** können Sie die Eigenschaften einer ausgewählten Instanz ändern.

Wenn Sie eine neue Instanz des Communications Service konfigurieren möchten, gehen Sie wie folgt vor:

- Klicken Sie auf die Schaltfläche **Add**.
- Öffnen Sie in **Add Communication Service Instance** über die Schaltfläche **...** bei **Select MC-CmdHandler** den Dialog **Properties of MC-CmdHandler**.
- Tragen Sie in **Properties of MC-CmdHandler** die Eigenschaften des MC-Handlers ein, mit dem die Komponenten administriert werden, und klicken Sie **OK**.
- Tragen Sie in **Add Communication Service Instance** den Installationspfad des Communication Service sowie die Konfigurationsparameter ein (wie unten beschrieben) und klicken Sie **OK**.

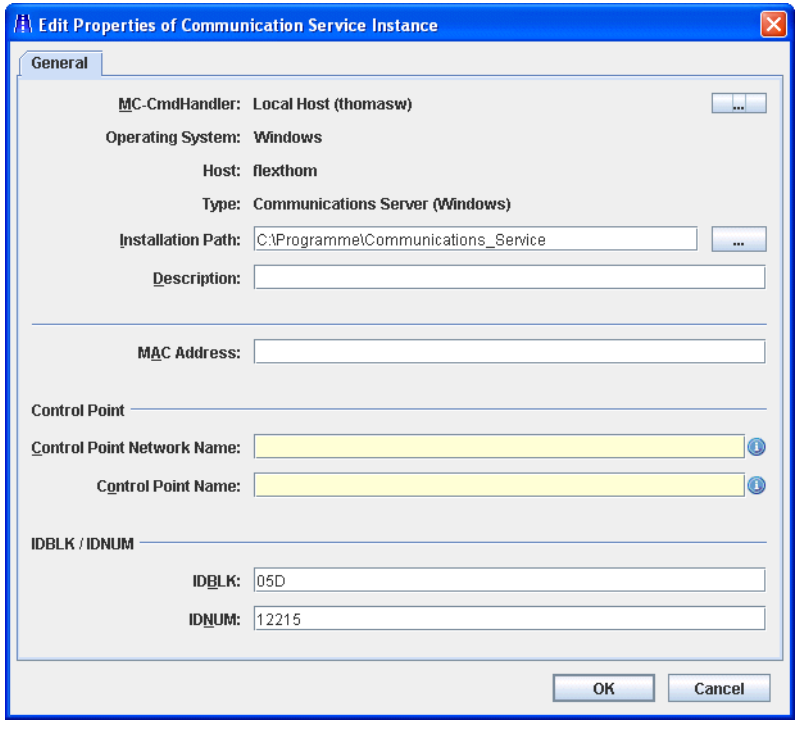


Bild 22: Eigenschaften eines Communication Services konfigurieren

MAC Address
Wenn für mindestens einen EIS Partner **LAN** als DLC-Typ verwendet wird, müssen Sie hier die MAC-Adresse des Rechners eingeben, auf dem der Communication Service läuft. Nähere Informationen hierzu finden Sie im Abschnitt Abschnitt „EIS Partner vom Typ CICS hinzufügen“ auf Seite 224.

Control Point
Der Control Point bezeichnet einen eindeutigen Knoten im SNA-Netzwerk. Über diesen Namen wird die Instanz des Communication Service identifiziert. BeanConnect erzeugt eine Datei mit den notwendigen VTAM-Definitionen, die dann am z/OS™-Mainframe eingetragen werden können (siehe Abschnitt „VTAM auf IBM Mainframe konfigurieren“ auf Seite 270). Der vollständige Control-Point-Name muss im Netzwerk eindeutig sein und besteht aus folgenden zwei Teilen:

- **Control Point Network Name** gibt das Netzwerk an und kann frei vergeben werden. Es empfiehlt sich aber, hierfür den EIS-Netzwerknamen zu verwenden.
- **Control Point Name** bezeichnet den Control Point in diesem Netzwerk. Der Name muss mit der VTAM-Definition am z/OS zusammenpassen.

IDBLK / IDNUM
Diese Werte bilden die sogenannte XID (Node ID) und gehen in die PU-Definition (Physical Unit) für die VTAM-Generierung ein:

- **IDBLK** gibt die Block ID (IDBLK-Wert) für die CICS-Generierung an. Angabe als 3-stellige hexadezimale Zahl, Buchstaben müssen in Großschreibung angegeben werden.
- **IDNUM** gibt die Physical Unit ID (IDNUM-Wert) für die CICS-Generierung an. Angabe als 5-stellige hexadezimale Zahl, Buchstaben müssen in Großschreibung angegeben werden.

Erläuterungen zu SNA-spezifischen Begriffen finden Sie im Glossar.

6.3.3 Administration-Passwort ändern

Das Administrationspasswort können Sie ändern. Dazu expandieren Sie den Proxy-Knoten im Navigationsbaum und wählen Sie im Kontextmenü des Proxys den Menüpunkt **Modify Admin Password**.

Dabei wird auch das Passwort des MC-CmdHandlers geändert, welcher zur Proxy-Administration verwendet wird. Aus diesem Grund sollte jeder Proxy seinen eigenen MC-CmdHandler haben, und nicht ein MC-CmdHandler von mehreren Proxys verwendet werden.

Sie können Ihr Passwort hinterlegen, indem Sie im Eigenschaftsfeld des Proxys für die Eigenschaft **Admin User Password** die Option **Use** wählen. Somit müssen Sie das Passwort bei nachfolgenden Sitzungen nicht mehr eingeben (siehe Abschnitt „Allgemeine Proxy-Daten“ auf Seite 186).

Bei diesen Aktionen muss der MC-Cmd-Handler auf dem Proxy-Rechner gestartet sein. Wenn nicht, starten Sie ihn vorher, siehe Abschnitt „MC-CmdHandler starten“ auf Seite 252.

6.3.4 Konfigurationsoptionen im Expertenmodus

Das Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** zeigt zwei weitere Registerkarten an, wenn Sie die Management Console im Expertenmodus betreiben: **Timer Settings** und **Performance Settings**. Diese Registerkarten stellen zusätzliche Optionen bereit, mit denen sich das Proxy-Verhalten steuern lässt.

Außerdem können Sie im Expertenmodus den **Application Program Interface Mode** und den **Container Application Process Title** ändern.

Den Expertenmodus aktivieren Sie, indem Sie das Menü **Extras – Settings** öffnen und im Eigenschaftsfeld **Management Console Settings** auf der Registerkarte **General** die Option **Expert Mode** wählen.

6.3.4.1 Timer Settings

Die Registerkarte **Timer Settings** enthält Timer, mit denen Sie die Überwachung der Verbindungen innerhalb des Proxys beeinflussen können.

6.3.4.2 Performance Settings

Auf der Registerkarte **Performance Settings** können Sie Einstellungen vornehmen, die die Performance des Proxys beeinflussen.

Der **Proxy Container Mode** bestimmt das Verhalten in Bezug auf Asynchron-Aufträge, wenn der Proxy beendet wird. In der Standardeinstellung werden noch nicht gestartete asynchrone Aufträge gelöscht, wenn der Proxy beendet wird (**Performance Enhanced (Non-durable Asynchronous Processing)**). Dies gilt sowohl für asynchrone Inbound-Aufträge, die noch nicht an den Application Server gesendet wurden bzw. erneut an den Application Server gesendet werden müssen (Redelivery) als auch für asynchrone Outbound-Aufträge, die noch nicht an den EIS Partner gesendet wurden. Dazu gehören insbesondere asynchrone Outbound-Aufträge, deren Start-Zeitpunkt noch nicht erreicht wurde, siehe auch Abschnitt „Lebensdauer von asynchronen Aufträgen“ auf Seite 513.

Wenn Asynchron-Aufträge auch bei Beendigung des Proxy erhalten bleiben sollen, können Sie für die Eigenschaft **Proxy Container Mode** die Einstellung **Durable Asynchronous Processing** wählen.

Im Bereich **Number of Proxy Container Processes** geben Sie die maximale Anzahl von Prozessen ein, die der Proxy gleichzeitig handhaben kann für

- alle Verbindungen (Inbound und Outbound Connection)
- Asynchron-Aufträge

Im Bereich **Number of Parallel Connections** können Sie die maximale Anzahl von parallelen Verbindungen angeben für

- Inbound Connections über das UPIC-Protokoll
- Inbound Connections über das openUTM-Socket-Protokoll und
- Inbound Connections über das RFC1006-Protokoll

Darüber hinaus können Sie die Größe bestimmter Speicherbereiche für den Proxy-Container einstellen.

Weitere Einzelheiten hierzu finden Sie in der Online-Hilfe.

6.3.4.3 Application Program Interface Mode (API Mode)

Das Feld **API Mode** befindet sich auf der Registerkarte **General** und ist auf den Wert **Standard** voreingestellt.

Wenn Sie eine neue Kopplung mit einem EIS Partner konfigurieren möchten, der als API die Programmschnittstelle XATMI verwendet, dann müssen Sie bei **API Mode** den Wert **XATMI** einstellen.

Zusätzlich gibt es noch den Wert **All**. Dieser Modus kann für die Migrationsphase notwendig sein, z.B. wenn von einer Standard-Kopplung auf XATMI-Kopplung umgestellt wird und umgekehrt.

6.3.4.4 Container Application Process Title

Wird nur bei eingeschaltetem Expertenmodus angezeigt.
Enthält den Application Process Title (APT) des Proxy-Containers. Der APT ist Teil der Generierungsinformation eines EIS Partners, damit der EIS Partner eine Verbindung zum Proxy-Container aufbauen kann. Sie können den APT hier auch verändern. Wenn Sie den APT ändern, müssen Sie anschließend den Proxy speichern und die Generierung aktualisieren (Befehl Update Configuration). Dies gilt auch für die EIS Partner des Proxys (Befehl Generate EIS Partner Configuration). Es werden entsprechende Todo-Aktionen erzeugt.

6.4 BeanConnect Proxy Cluster konfigurieren

In einer BeanConnect Cluster Konfiguration können eine oder mehrere Instanzen des BeanConnect Resource Adapters und eine oder mehrere BeanConnect Proxy Instanzen zum Ablauf kommen. Alle Instanzen des Clusters sind identisch konfiguriert. Damit ergeben sich drei unterschiedliche Szenarien:

- n:1-Beziehung: Mehrere Resource Adapter Instanzen im Application Server und eine Proxy Instanz

  Eine n:1-Beziehung ist dann sinnvoll, wenn die Anwendungen in einem Application Server Cluster nur in geringem Maß mit EIS Partnern kommunizieren.

  Auch bei einer n:1-Beziehung müssen Sie einen Proxy Cluster konfigurieren, da sonst ein Multi-Resource Adapter Betrieb vorliegt, der sich völlig anders verhält, siehe Abschnitt „Multi-Resource Adapter Betrieb“ auf Seite 40. Durch den Proxy Cluster ist z.B. sichergestellt wird, dass eine Änderung in allen Instanzen des Resource Adapters durchgeführt wird.

- 1:m-Beziehung: Eine Resource Adapter Instanz und mehrere Proxy Instanzen
  Diese Möglichkeit ist dann sinnvoll, wenn fast ausschließlich Inbound-Kommunikation betrieben wird.

- n:m-Beziehung: Mehrere Resource Adapter Instanzen und mehrere Proxy Instanzen

In der Management Console konfigurieren Sie immer alle Resource Adapter Instanzen, wenn im Application Server mehrere Resource Adapter Instanzen im Cluster ablaufen. Das Konfigurieren des Cluster für den Application Server wird dort durchgeführt.

Gehören mehrere Proxys zum Proxy Cluster, so gibt es einen ausgezeichneten Proxy, den sogenannten **Master Proxy**. Dieser Proxy ist der (erste) Ansprechpartner der Management Console beim Holen der Konfigurationsdaten des Clusters. Ist der momentan festgelegte Master Proxy nicht administrierbar, dann verlangt die Management Console, dass ein anderer, administrierbarer Proxy als Master Proxy festgelegt wird. Die Konfigurationsdaten werden dann von diesem geholt. Bei Änderungen der Konfiguration sorgt die Management Console dafür, dass die Verwaltungsdaten aller Proxys des Proxy Clusters konsistent geändert werden.

6.4.1 Proxy Cluster erzeugen

Wenn Sie einen Proxy Cluster erzeugen möchten, dann müssen Sie zuvor mindestens einen Proxy konfiguriert haben, siehe Abschnitt „BeanConnect Proxy konfigurieren“ auf Seite 185.

Sie erzeugen einen neuen Proxy Cluster, indem Sie unter **BeanConnect Proxies** den gewünschten Proxy markieren und im Kontextmenü den Befehl **Define Proxy Cluster...** ausführen. Der Proxy darf zu diesem Zeitpunkt nicht gestartet sein.

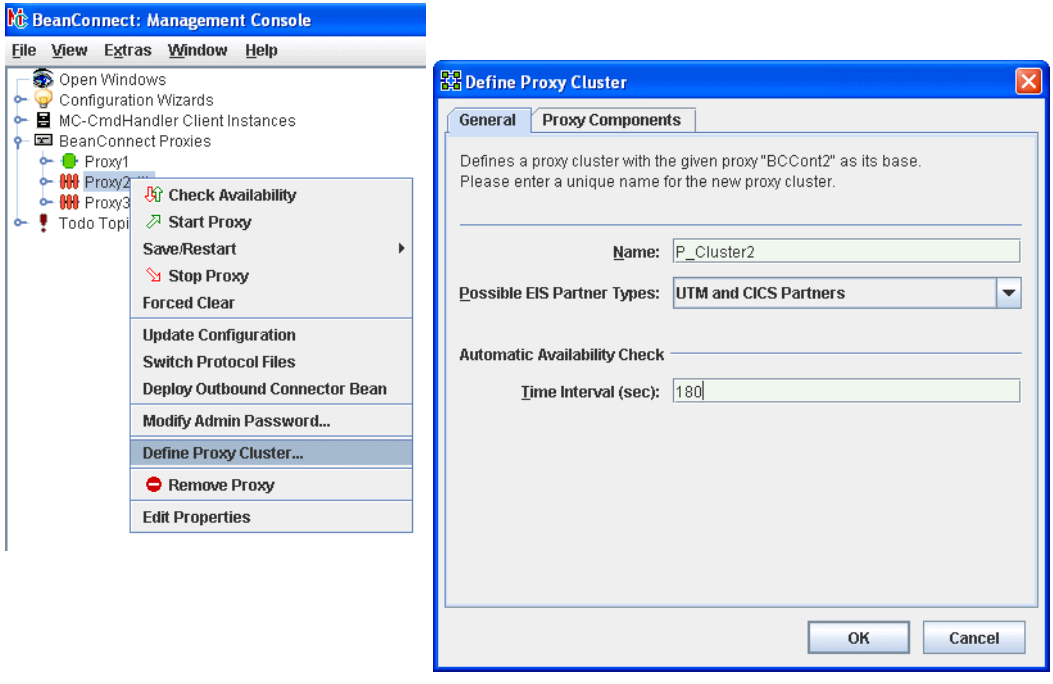


Bild 23: Erzeugen eines neuen Proxy Clusters

Tragen Sie im Dialogfeld **Define Proxy Cluster** bei **Name** den Namen des neuen Proxy Clusters ein. Bei allen anderen Feldern werden die Werte des ausgewählten Proxys eingeblendet. Sie können dieser Werte ändern falls erforderlich.

Wenn Sie den Dialog mit **OK** verlassen, dann wird im Navigationsbaum ein neuer Knoten **BeanConnect Proxy Clusters** erzeugt. Dort wird der Cluster unter dem oben definierten Namen eingetragen, gleichzeitig wird der Proxy aus der Liste der **BeanConnect Proxies** entfernt. Der erste Proxy, der in den Proxy Cluster aufgenommen wird, ist standardmäßig der Master Proxy.

Proxys eines Clusters anzeigen

Sie können sich die Proxys eines Proxy Cluster anzeigen lassen, indem Sie entweder auf den Knoten des Proxy Clusters klicken oder im Kontextmenü den Befehl **Show Cluster Proxies** auswählen. Es wird eine Liste mit allen Proxys angezeigt. Die einzelnen Spalten beschreiben die Eigenschaften der Proxys wie z.B. Name und Adressdaten, Details finden Sie in der Online-Hilfe.

6.4.2 Proxy zum Proxy Cluster hinzufügen

Sie können weitere Proxys zum Proxy Cluster hinzufügen, es sind maximal 32 Proxys pro Cluster möglich. Der betreffende Proxy muss installiert sein und die allgemeinen Proxy-Daten müssen mit der Management Console konfiguriert sein, siehe Abschnitt „Allgemeine Proxy-Daten“ auf Seite 186.

So fügen Sie einen weiteren Proxy zum Cluster hinzu:

- Markieren Sie den gewünschten Proxy und wählen Sie im Kontextmenü den Befehl **Add to Proxy Cluster** aus. Es werden alle konfigurierten Proxy Cluster angezeigt, zu denen der ausgewählte Proxy hinzugefügt werden kann.
- Wählen Sie den gewünschten Cluster per Maus-Klick und bestätigen Sie die Rückfrage mit **Add**. Der Proxy wird zu diesem Cluster hinzugefügt und aus der Liste der Proxys entfernt.

Vorsicht!

Wenn Sie einen Proxy zu einem Proxy Cluster hinzufügen, dann geht die Konfiguration des Proxys verloren, da sie durch die Konfiguration des Proxy Clusters überschrieben wird (der Proxy Cluster wird synchronisiert).

Wenn Sie einen Proxy aus einem n:1- oder n:m-Cluster entfernen, dann ist anschließend keine Inbound-Kommunikation im entfernten Proxy mehr möglich. Dies liegt daran, dass der Proxy im Multi-RA-Betrieb gestartet wird, da er mehr als einen Resource Adapter besitzt, aber keinem Message Endpoint ein gültiger Resource Adapter zugeordnet ist.

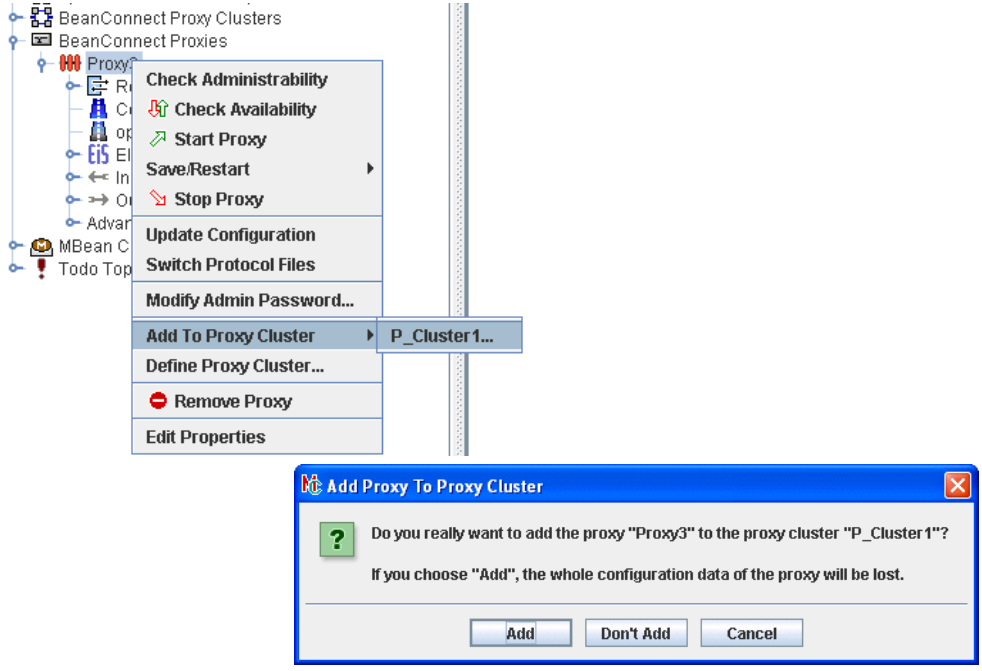


Bild 24: Hinzufügen eines Proxys zu einem Proxy Cluster

6.4.3 Proxy aus Cluster entfernen / Proxy Cluster entfernen

Sie können einen einzelnen Proxy wieder aus dem Cluster entfernen oder den Proxy Cluster selber entfernen.

- Sie entfernen einen Proxy aus dem Proxy Cluster, indem Sie in der Liste der Cluster Proxys den betreffenden Proxy selektieren und im Kontextmenü den Befehl **Remove Proxy From Cluster** auswählen. Nach Bestätigen der Sicherheitsabfrage wird der Proxy aus dem Cluster entfernt und wieder im Bereich **BeanConnect Proxies** aufgelistet.
- Sie entfernen einen Proxy Cluster, indem Sie im Kontextmenü des Proxy Clusters den Befehl **Remove Proxy Cluster** auswählen. Nach dieser Aktion werden alle im Proxy Cluster enthaltenen Proxys wieder im Bereich **BeanConnect Proxies** aufgelistet. Die Proxys werden also nicht deinstalliert, sondern es wird nur der Cluster aufgelöst.

Vorsicht!
Wenn Sie einen Proxy aus einem Proxy Cluster entfernen oder den Proxy Cluster selbst entfernen, dann behält der Proxy die Konfiguration, die er im Proxy Cluster besaß. Es wird also **nicht** wieder der alte Zustand hergestellt, den der Proxy vor dem Hinzufügen zum Proxy Cluster hatte.

6.5 BeanConnect Resource Adapter konfigurieren

Die Konfigurationsdaten des BeanConnect Resource Adapters sind im Deployment Descriptor `ra.xml` des Resource Adapters beschrieben. Dieser Deployment Descriptor ist im BeanConnect-RAR-Archiv enthalten.

Wie Sie die Datei `ra.xml` mit Hilfe eines Texteditors anpassen, ist im Abschnitt Abschnitt „Allgemeine Eigenschaften des Resource Adapters konfigurieren“ auf Seite 95 beschrieben.

Alternativ können Sie die Datei `ra.xml` über die BeanConnect Management Console modifizieren. Dazu müssen Sie den BeanConnect Resource Adapter in der BeanConnect Management Console konfigurieren.

Der Konfigurations-Dialog hängt davon ab, ob der Proxy zu einem Cluster gehört oder nicht.

6.5.1 Resource Adapter hinzufügen (ohne Cluster-Betrieb)

Sie fügen einen Resource Adapter hinzu, indem Sie im Proxy-Teilbaum des Navigationsbaums das Kontextmenü des Knotens **Resource Adapters** öffnen und den Befehl **Add Resource Adapter**... wählen. Alternativ können Sie auch auf den gewünschten Knoten klicken und in der Liste der Resource Adapter den Befehl **Add** ausführen. Der Proxy darf beim Hinzufügen nicht gestartet sein, daher müssen Sie ihn vorher ggf. beenden, z.B. mit dem Befehl **Stop Proxy** im Kontextmenü des Proxys.

Tragen Sie die Parameter des Resource Adapters in das Registerblatt **General** ein:

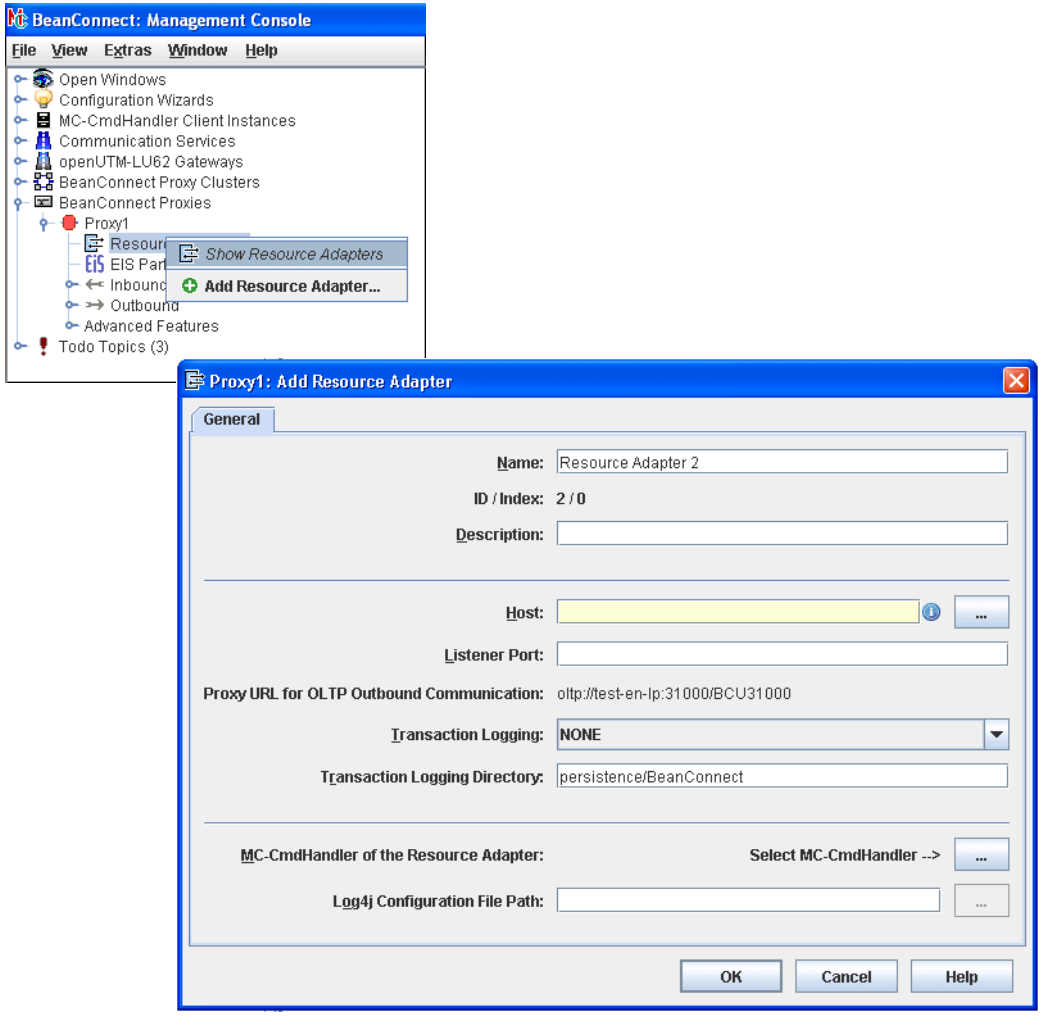


Bild 25: Hinzufügen eines Resource Adapters ohne Cluster-Betrieb

Name

Frei wählbarer Name des Resource Adapters. Dieser Name ist nur innerhalb der Management Console von Bedeutung.

ID / Index

ID ist eine von der Management Console vergebene ID des Resource Adapters. Diese ist eine Zahl zwischen 1 und 256.
Index ist der Index des Resource Adapters, er wird von der Management Console vergeben. Der Index ist nur im Multi-Resource Adapter Betrieb relevant. Diese Eigenschaft entspricht der Property `resourceAdapterIndex` im Deployment Descriptor `ra.xml` des Resource Adapters.

Description

Hier können Sie eine frei wählbare Beschreibung des Resource Adapters eintragen.

Host

Name oder IPv4-Adresse des Rechners, auf dem der Application Server läuft, in dem der Resource Adapter deployt ist.

Listener Port

Listener Portnummer des Resource Adapters für Inbound-Kommunikation. Diese Eigenschaft entspricht der Property `inboundListenerPort` im Deployment Descriptor `ra.xml` des Resource Adapters. Wenn Sie hier 0 eintragen, dann sind für diesen Resource Adapter weder Inbound-Kommunikation noch Verfügbarkeitsprüfung möglich.

Proxy URL of OLTP Outbound Communication

URL, die bei der Outbound-Kommunikation im Application Server verwendet wird. Diese URL wird beim Konfigurieren des Proxy festgelegt und ist nicht änderbar. Diese Eigenschaft entspricht der Property `proxyURL` im Deployment Descriptor `ra.xml` des Resource Adapters.

Transaction Logging

Gibt an, ob bei Outbound-Kommunikation persistente Transaktions-Logs im Resource Adapter geschrieben werden (Auswahl **FILE**) oder nicht (Auswahl **NONE**). Diese Eigenschaft entspricht der Property `transactionLogging` im Deployment Descriptor `ra.xml` des Resource Adapters. Bei konfiguriertem Transaktions-Logging schreibt der Resource Adapter für jede Transaktion eine eigene Log-Datei. Der Dateiname setzt sich zusammen aus dem Prefix `tx.` und einer Nummer.

Transaction Logging Directory

Verzeichnis, unter dem die Transaktions-Logs im Resource Adapter angelegt werden. Diese Eigenschaft entspricht der Property `transactionLogDir` im Deployment Descriptor `ra.xml` des Resource Adapters.

MC-CmdHandler of the Resource Adapter

Hiermit können Sie einen MC-CmdHandler definieren, der auf dem Rechner des Resource Adapters verfügbar ist und über den der Resource Adapter per Management Console administriert werden kann. Dazu gehen Sie wie folgt vor:

- Mit der Schaltfläche **...** bei **Select MC-CmdHandler** erhalten Sie den Folgedialog **Properties of MC-CmdHandler**, in dem Sie die Daten des MC-CmdHandler eingeben (Listener Port, Passwort):

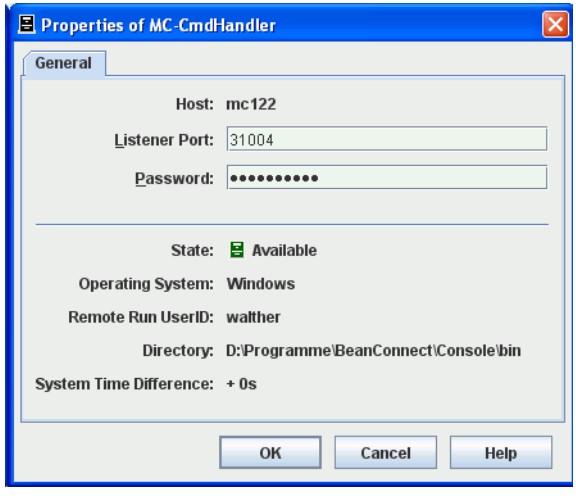


Bild 26: Eigenschaften eines MC-CmdHandlers konfigurieren

Wenn sich das BeanConnect-RAR-Archiv auch auf diesem Rechner befindet, können Sie über diesen MC-CmdHandler die Einträge im Deployment Descriptor `ra.xml` des Resource Adapters editieren oder gemäß der eingegebenen Parameter aktualisieren. Dazu verwenden Sie die Befehle **Edit ra.xml of BeanConnect Resource Adapter RAR** oder **Update ra.xml of BeanConnect Resource Adapter RAR** im Kontextmenü des Resource Adapters, siehe Abschnitt „Konfigurationsdatei des Resource Adapters“ auf Seite 210.
Außerdem lassen sich über diesen MC-CmdHandler die Log4j-Diagnoseeinstellungen ändern.

- In **Log4j Configuration File Path** können Sie den kompletten Pfadnamen der Log4j-Konfigurationsdatei angeben (optional).

  Die Log4j-Konfigurationsdatei liegt immer auf dem Rechner, auf dem der Application Server läuft. Dies muss nicht der Rechner sein, auf dem sich der BeanConnect Resource Adapter befindet, da der Resource Adapter auch erst beim Deployment auf den Rechner des Application Servers hochgeladen werden kann.

  Wenn man die Logging-Konfiguration des Resource Adapters ändern will, muss auch auf dem Rechner, auf dem der Application Server läuft, ein MC-CmdHandler laufen.

6.5.2 Resource Adapter im Cluster-Betrieb hinzufügen

Sie fügen einen Resource Adapter im Cluster-Betrieb hinzu, indem Sie im Cluster-Teilbaum des Navigationsbaums das Kontextmenü des Knotens **Resource Adapters** öffnen und den Befehl **Add Resource Adapter**.. wählen. Alternativ können Sie auch auf den gewünschten Knoten klicken und in der Liste der Resource Adapter den Befehl **Add** ausführen.

Tragen Sie die Parameter des Resource Adapters in die beiden Registerblätter **General** und **Common Properties** ein. Die Parameter in **Common Properties** sind für alle Resource Adapter eines Clusters gleich, eine Änderung bei diesen Properties wirkt sich deshalb auf alle definierten Resource Adapter des Cluster aus, während die Parameter in **General** Resource Adapter spezifisch sind:

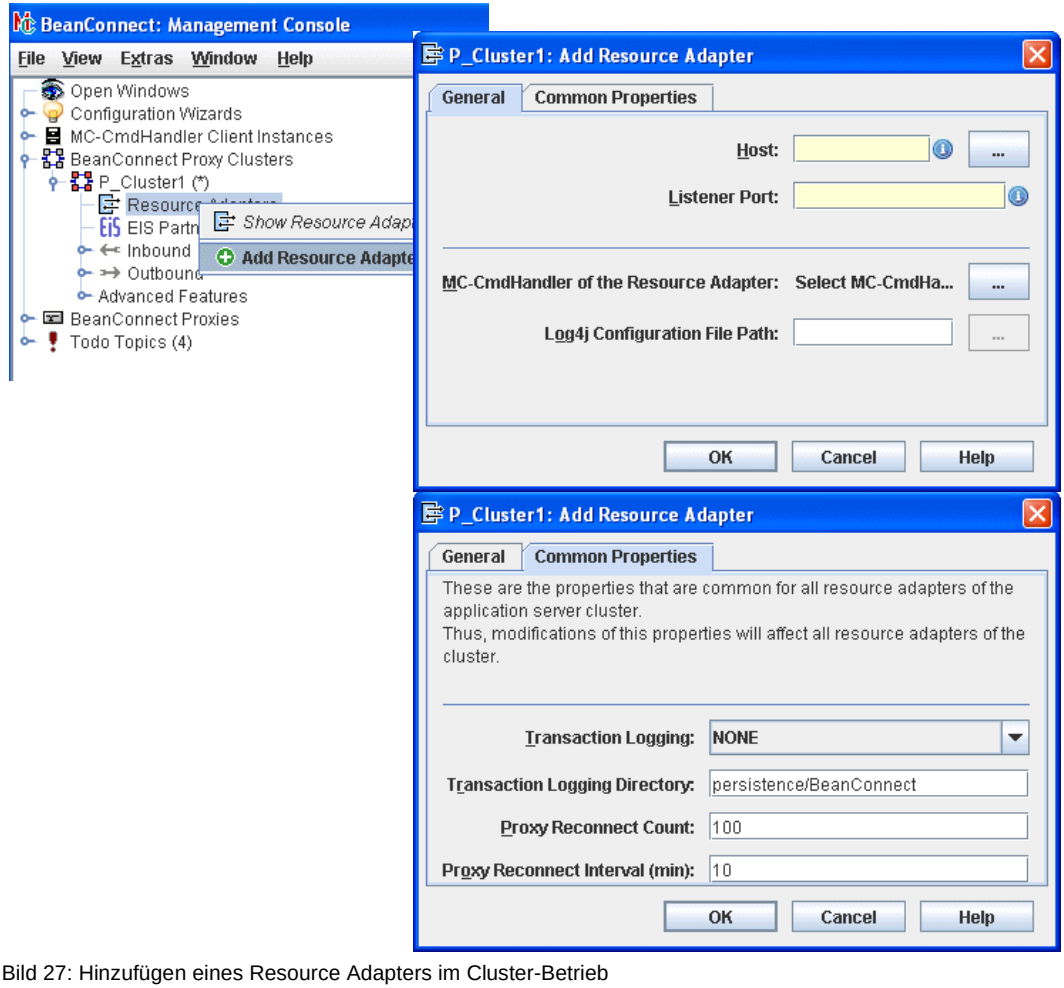


Bild 27: Hinzufügen eines Resource Adapters im Cluster-Betrieb

Host (General)

Name oder IPv4-Adresse des Rechners, auf dem der Application Server läuft, in dem der Resource Adapter deployt ist. Der Host muss dem Wert `<host>` in einem Eintrag `<host:port>` in der Property `resourceAdapterAddresses` im Deployment Descriptor `ra.xml` des Resource Adapters entsprechen.

Listener Port (General)

Listener Portnummer des Resource Adapters für Inbound-Kommunikation. Der in **Host** angegebene Host und der Listener Port müssen einem Eintrag `<host:port>` in der Property `resourceAdapterAddresses` im Deployment Descriptor `ra.xml` des Resource Adapters entsprechen. Ist dort kein Port angegeben, muss Angabe mit dem Wert in der Property `inboundListenerPort` übereinstimmen.

Innerhalb des Clusters erlaubt die Management Console weder den Wert `0` noch eine Leereingabe. Es muss immer ein Port von `1025` bis `65535` angegeben werden, auch dann, wenn kein Inbound geplant ist.

MC-CmdHandler of the Resource Adapter (General)

In diesen Feldern können Sie einen MC-CmdHandler definieren, der auf dem Rechner des Resource Adapters verfügbar ist und über den der Deployment Descriptor `ra.xml` des Resource Adapters per Management Console angepasst werden kann:

- Mit der Schaltfläche **...** bei **Select MC-CmdHandler** erhalten Sie einen Folgedialog, in dem Sie die Daten des MC-CmdHandler eingeben (Rechner, Listener Port, Passwort).

  Wenn sich das BeanConnect-RAR-Archiv auch auf diesem Rechner befindet, können Sie über diesen MC-CmdHandler die Einträge im Deployment Descriptor `ra.xml` des Resource Adapters editieren oder gemäß der eingegebenen Parameter aktualisieren. Dazu verwenden Sie die Befehle **Edit ra.xml of BeanConnect Resource Adapter RAR** oder **Update ra.xml of BeanConnect Resource Adapter RAR** im Kontextmenü des Resource Adapters, siehe Abschnitt „Konfigurationsdatei des Resource Adapters“ auf Seite 210. Außerdem lassen sich über diesen MC-CmdHandler die Log4j-Diagnoseeinstellungen ändern.

- In **Log4j Configuration File Path** können Sie den kompletten Pfadnamen der Log4j Konfigurationsdatei angeben (optional).

  Die Log4j-Konfigurationsdatei liegt immer auf dem Rechner, auf dem der Application Server läuft. Dies muss nicht der Rechner sein, auf dem sich der BeanConnect Resource Adapter befindet, da der Resource Adapter auch erst beim Deployment auf den Rechner des Application Servers hochgeladen werden kann.

  Wenn man die Logging-Konfiguration des Resource Adapters ändern will, muss auch auf dem Rechner, auf dem der Application Server läuft, ein MC-CmdHandler laufen.

Transaction Logging (Common Properties)

Gibt an, ob bei Outbound-Kommunikation persistente Transaktions-Logs im Resource Adapter geschrieben wird (Auswahl **FILE**) oder nicht (Auswahl **NONE**). Diese Eigenschaft entspricht der Property `transactionLogging` im Deployment Descriptor `ra.xml` des Resource Adapters. Bei konfiguriertem Transaktions-Logging schreibt der Resource Adapter für jede Transaktion eine eigene Log-Datei. Der Dateiname setzt sich zusammen aus dem Prefix `tx.` und einer Nummer.

Transaction Logging Directory (Common Properties)

Verzeichnis, unter dem das Logging der Transaktionen im Resource Adapter angelegt wird. Diese Eigenschaft entspricht der Property `transactionLogDir` im Deployment Descriptor `ra.xml` des Resource Adapters.

Proxy Reconnect Count (Common Properties)

Gibt an, nach wieviel Verbindungsanforderungen (`getConnection()`-Aufrufen) eine Neuzuordnung zwischen Resource Adapter Instanz und Proxy-Anwendung initiiert werden soll. Diese Eigenschaft regelt die nutzungsabhängige Neuzuordnung einer Resource Adapter Instanz zu einer Proxy-Anwendung. Sie entspricht der Property `proxyReconnectCount` im Deployment Descriptor `ra.xml` des Resource Adapters.

Der voreingestellte Wert ist 100.

Wenn ein Cluster mit mehr Resource Adapter Instanzen als Proxy Instanzen betrieben wird, dann sollte für diesen Parameter ein größerer Wert oder der Wert 0 eingetragen werden; mit dem Wert 0 wird der Reconnect Counter ausgeschaltet.

Proxy Reconnect Interval (min) (Common Properties)

Gibt an, nach wie vielen Minuten eine Neuzuordnung zwischen Resource Adapter Instanz und Proxy-Anwendung initiiert werden soll. Diese Eigenschaft regelt die zeitabhängige Neuzuordnung einer Resource Adapter Instanz zu einer Proxy-Anwendung, Sie entspricht der Property `proxyReconnectInterval` im Deployment Descriptor `ra.xml` des Resource Adapters.

Der voreingestellte Wert ist 10 Minuten.

Wenn ein Cluster mit mehr Resource Adapter Instanzen als Proxy Instanzen betrieben wird, dann sollte für diesen Parameter ein größerer Wert oder der Wert 0 eingetragen werden; mit dem Wert 0 wird der Reconnect Timer ausgeschaltet.

6.5.3 Konfigurationsdatei des Resource Adapters

Die allgemeinen Konfigurationsdaten des Resource Adapters werden in der Datei `ra.xml` festgelegt, siehe Abschnitt „Allgemeine Eigenschaften des Resource Adapters konfigurieren“ auf Seite 95. `ra.xml` ist im BeanConnect-RAR-Archiv enthalten. Die Management Console kann auf diese Datei zugreifen und die Konfigurations-Properties ändern, wenn eine der beiden folgenden Bedingungen erfüllt sind:

- Das BeanConnect-RAR-Archiv liegt auf dem selben Rechner, auf dem die **Management Console** läuft. Dann muss es entweder auch unter derselben Kennung liegen, unter der die Management Console läuft, oder die Datei-Zugriffsrechte müssen entsprechend gesetzt sein.
- Das BeanConnect-RAR-Archiv liegt auf dem selben Rechner, auf dem ein **MC-CmdHandler** läuft. Dann muss es entweder auch unter derselben Kennung liegen, unter der der MC-CmdHandler läuft, oder die Datei-Zugriffsrechte müssen entsprechend gesetzt sein.

Konfigurationsdatei per Management Console aktualisieren

Sie aktualisieren die Konfigurationsdatei `ra.xml`, indem Sie im Kontextmenü des Resource Adapters den Befehl **Update ra.xml of BeanConnect Resource Adapter RAR** auswählen. In einem Proxy Cluster müssen Sie das Kontextmenü im Knoten **Resource Adapters** auswählen, nicht im Resource Adapter selber.

Damit werden die Werte in `ra.xml` durch die in der Management Console angegebenen Werte überschrieben.

Konfigurationsdatei per Management Console editieren

Sie editieren die Konfigurationsdatei `ra.xml`, indem Sie im Kontextmenü des Resource Adapters den Befehl **Edit ra.xml of BeanConnect Resource Adapter RAR** auswählen. In einem Proxy Cluster müssen Sie das Kontextmenü im Knoten **Resource Adapters** auswählen, nicht im Resource Adapter selber.

Es wird das Dialogfeld **Edit ra.xml of BeanConnect Resource Adapter RAR** geöffnet, in dem Sie alle Properties von `ra.xml` modifizieren können, siehe auch Abschnitt „Allgemeine Eigenschaften in ra.xml festlegen“ auf Seite 96 und Online-Hilfe:

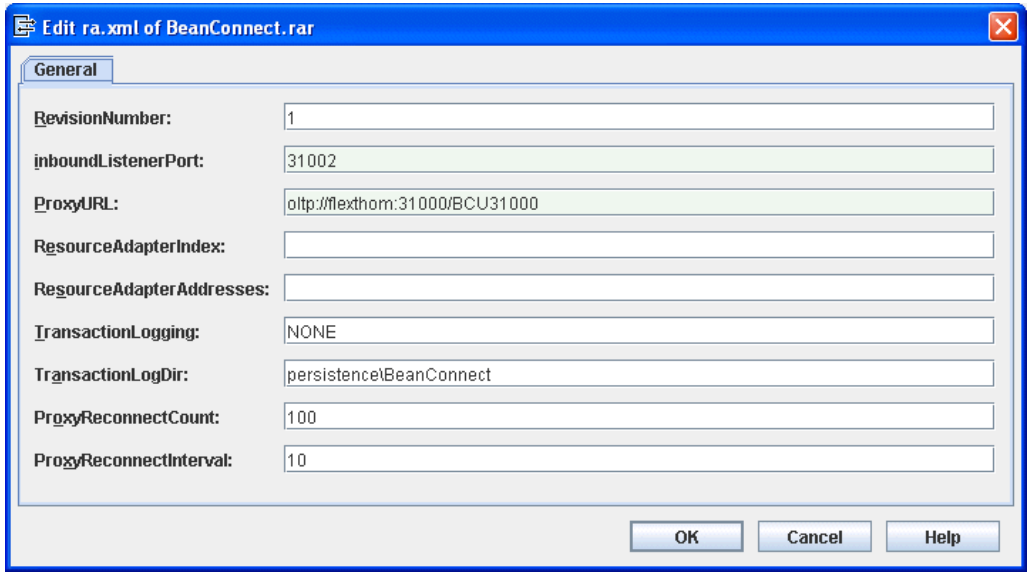


Bild 28: Editieren der Konfigurationsdatei ra.xml

Die geänderten Werte werden nach einem Update oder direktem Editieren erst dann wirksam, nachdem das BeanConnect-RAR-Archiv anschließend deployt wurde, siehe Abschnitt „Deployment des Resource Adapters“ auf Seite 103.

Bitte beachten Sie, dass **vor** der Anwendung des Befehls **Update ra.xml...** die gewünschten Änderungen bei den Properties des/der betroffenen Resource Adapter eingebracht werden müssen.

6.6 EIS Partner konfigurieren

Ein Proxy oder ein Proxy Cluster kann mit mehreren EIS Partnern kommunizieren, d.h. mit mehreren openUTM- und/oder CICS-Anwendungen. Beim Konfigurieren des Proxys/ Proxy Clusters legen Sie fest, mit welchem Typ von EIS Partnern er kommunizieren kann (nur openUTM, nur CICS oder beides).

Damit ein EIS Partner verwaltet werden kann, muss er den Konfigurationsdaten der Management Console hinzugefügt werden. Dabei gibt es nur geringe Unterschiede zwischen Proxy und Proxy Cluster.

Jede Partneranwendung, die der Management Console bekannt gegeben wurde, wird im Proxy-Teilbaum des Navigationsbaums unterhalb des Knotens **EIS Partners** durch ein EIS Partnerobjekt dargestellt.

Sie müssen hier nur EIS Partner konfigurieren, die vom Typ openUTM sind und mit denen über das OSI TP-Protokoll kommuniziert werden soll oder die vom Typ CICS sind.

Klicken Sie auf den Knoten **EIS Partners**, um die Liste der verwalteten EIS Partner des Proxys anzuzeigen.

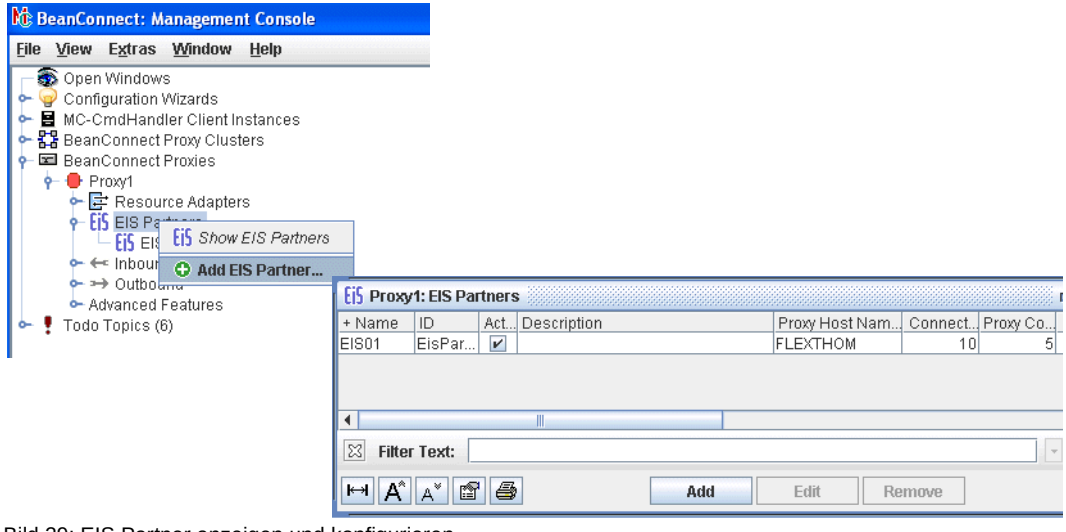


Bild 29: EIS Partner anzeigen und konfigurieren

6.6.1 EIS Partner vom Typ openUTM konfigurieren

Damit Sie einen EIS Partner vom Typ openUTM konfigurieren können, muss der Proxy bzw. Proxy Cluster in der Registerkarte **General** entsprechend konfiguriert sein, siehe Bild 18 auf Seite 186.

6.6.1.1 EIS Partner vom Typ openUTM hinzufügen

So fügen Sie einen neuen EIS Partner hinzu:

1. Klicken Sie unter der Liste der EIS Partner auf die Schaltfläche **Add** oder öffnen Sie das Kontextmenü eines bereits bestehenden EIS Partnerobjekts oder des Knotens **EIS Partners** und wählen Sie den Befehl **Add EIS Partner...**.

   Wenn der Proxy/Proxy Cluster für die beide Partnertypen (UTM und CICS) konfiguriert ist, dann wird der Zwischendialog **Choose EIS Partner Type** eingeschoben. Wählen Sie dort **UTM application** aus.

2. Definieren Sie die Eigenschaften für den EIS Partner. Das Eigenschaftsfeld wird automatisch geöffnet. Es enthält die Registerkarten **General, UTM Partner** und **Availability Check**.

Folgende Eigenschaften werden abgefragt:

Registerkarte General (openUTM-Partner)

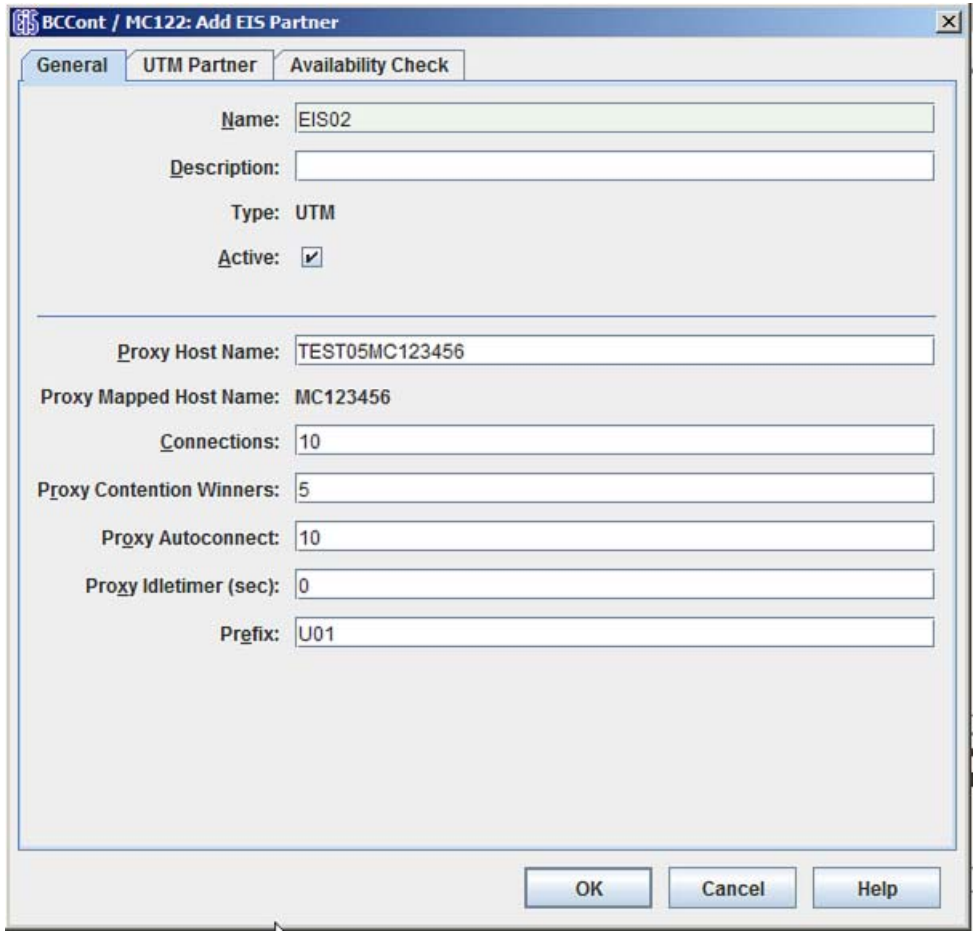


Bild 30: Allgemeine Eigenschaften eines EIS Partners vom Typ openUTM

Name / Description

Name ist der Name des EIS Partners. Der Name muss für den Proxy eindeutig sein. Darüber hinaus können Sie im Feld **Description** eine Beschreibung für den EIS Partner eingeben.
Bei einem existierenden EIS Partner wird die EIS-ID im Feld **ID** unterhalb des Namens angezeigt (nicht änderbar).

Type

Zeigt den Typ des EIS Partners an: hier **UTM**, nicht veränderbar.

Active

Diese Option steuert, ob die Definition eines EIS Partners aktiv ist. Nur aktivierte EIS Partner werden in der Konfiguration des Proxys und der EIS Partner berücksichtigt (Befehl **Update Configuration** im Proxy-Knoten bzw. Proxy Cluster Knoten).

Proxy Host Name

Name des Rechners, auf dem sich der Proxy befindet und wie er in der openUTM-Partneranwendung bekannt ist (max. 63 Zeichen).

Bei einem EIS Partner im Proxy Cluster gibt es für jeden Proxy des Clusters jeweils ein Eingabefeld, das mit dem beim EIS Partner bekannten Rechnernamen des jeweiligen Proxys versorgt werden muss.

Voreinstellung ist der Rechnername, der bei **Edit Properties** vom Proxy unter **General** bei **Host** eingetragen ist. Wurde für den Proxy Host eine IP-Adresse angegeben, muss diese durch den Namen ersetzt werden.

Proxy Mapped Hostname

Mapped Name des Rechners, auf dem sich der Proxy befindet und wie er in der openUTM-Partneranwendung bekannt ist, wenn der reale Rechnername länger als 8 Zeichen ist.

Bei einem EIS Partner im Proxy Cluster gibt es für jeden Proxy des Clusters neben dem Proxy Hostname Feld jeweils ein Feld, das den beim EIS Partner bekannten gemappten Rechnernamen des jeweiligen Proxys enthält.

Eine Änderung ist an dieser Stelle nicht möglich. Der **Proxy Mapped Hostname** kann nur im Eigenschaftsdialog des Proxys geändert werden (Befehl **Edit Properties,** Registerkarte **General**, Feld **Mapped Hostname**).

Connections

Gibt die maximale Anzahl von Verbindungen an, die gleichzeitig zwischen dem Proxy und dem EIS Partner zulässig sind.

Proxy Contention Winners

Gibt die Anzahl von Verbindungen an, für die der Proxy als Contention Winner gelten soll. Die von Ihnen anzugebene Anzahl hängt vom bevorzugten Kommunikationstyp ab (Outbound- oder Inbound-Kommunikation).

Grundsätzlich gilt Folgendes:

- Outbound-Kommunikation: der Proxy sollte der Contention Winner sein.
- Inbound-Kommunikation: der EIS Partner sollte der Contention Winner sein.

Proxy Autoconnect

Definiert die Anzahl der Verbindungen, die beim Start des Proxys aufgebaut werden sollen.

Proxy Idletimer (sec)

Gibt die Zeitspanne in Sekunden an, nach deren Ablauf der BeanConnect Proxy Container die Verbindung zum EIS Partner abbauen soll, falls in dieser Zeitspanne keine Kommunikation über diese Verbindung stattgefunden hat.

Mögliche Werte: 0 (Standardwert) bis 32767.
0 bedeutet, dass der Timer nicht verwendet wird.

Prefix

Das Präfix geht als Bestandteil von Namen ein, die in Konfigurationsanweisungen verwendet werden. Dieses Präfix muss für alle Proxys, die auf demselben Rechner laufen, eindeutig sein. Das Präfix muss aus genau drei Zeichen bestehen (Großbuchstaben oder Ziffern). Das erste Zeichen muss ein Großbuchstabe sein.

Registerkarte UTM Partner

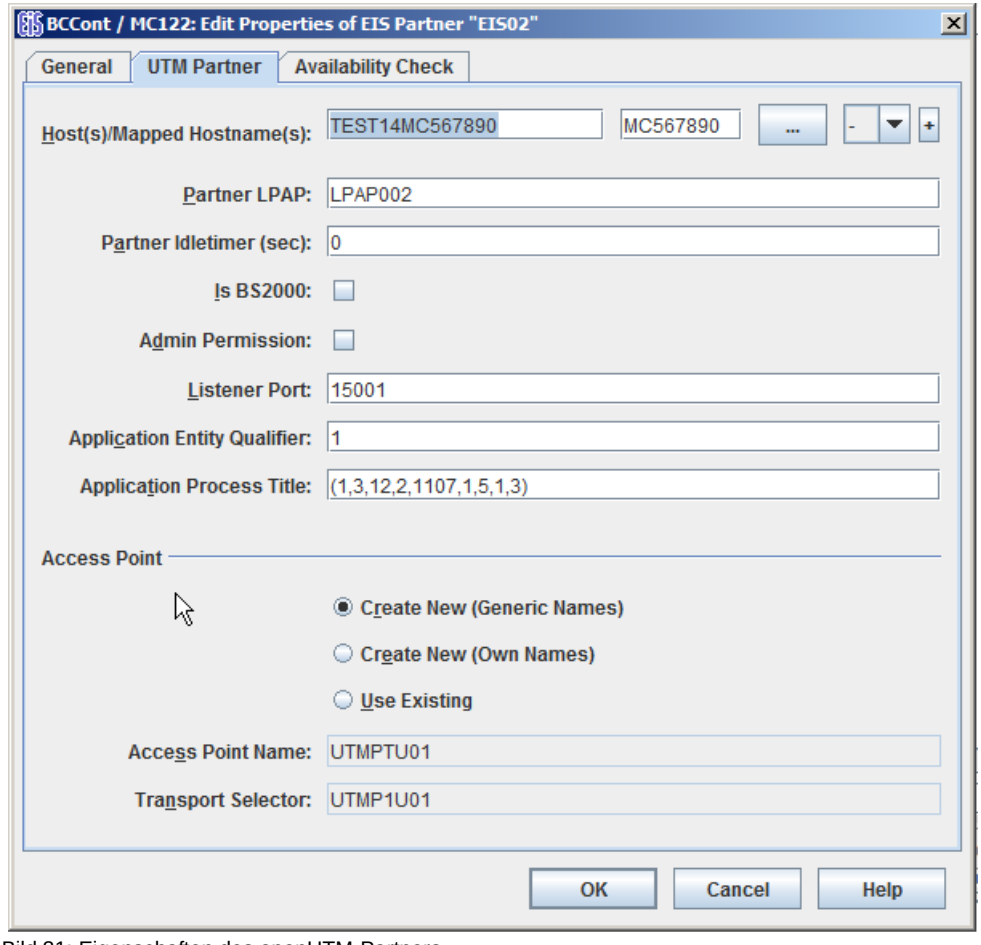


Bild 31: Eigenschaften des openUTM-Partners

Bei aktiviertem Expertenmodus werden zusätzliche Felder angezeigt. Diese werden im Anschluss beschrieben, siehe „UTM Partner - Zusätzliche Felder im Expertenmodus“ auf Seite 221.

Host(s)/Mapped Hostname(s)

Der Name des Rechners, auf dem sich die openUTM-Partneranwendung befindet, 1 bis maximal 63 Zeichen lang, und ggf. der Mapped Name des Rechners (max. 8 Zeichen), falls der Rechnername länger als 8 Zeichen ist.

Gehen Sie wie folgt vor:

1. Im ersten Feld tragen Sie den Rechnernamen ein (1-63 Zeichen).
2. Wenn der Rechnername länger als 8 Zeichen ist, tragen Sie im zweiten Feld den Mapped Hostnamen ein (max. 8 Zeichen).

Wenn die Partneranwendung eine openUTM-Cluster-Anwendung ist, tragen Sie hier die Namen der Rechner und ggf. die Mapped Hostnamen aller Cluster-Knoten ein.

Alternativ dazu können Sie über die Schaltfläche "**...**" die Adresse des Hosts (statt der Namen) angeben.

Wenn der Partner ein openUTM-Cluster ist, geben Sie in der Dropdown-Liste rechts neben der Schaltfläche den Cluster-Knotenindex der Knoten-Anwendung an.

Wenn der Partner eine stand-alone openUTM-Anwendung ist, wählen Sie in der Dropdown-Liste “-“ aus.

Der Knoten-Index des Cluster-Knotens ergibt sich aus der Reihenfolge der CLUSTER-NODE-Anweisungen im KDCDEF-Input der Cluster-Partneranwendung. Der Knoten, der durch die erste CLUSTER-NODE-Anweisung definiert wird, hat den Index 1, der Knoten der zweiten CLUSTER-NODE-Anweisung den Index 2 usw.

Wenn der Partner ein openUTM-Cluster ist, wählen Sie die Schaltfläche **+** (Plus), um einen weiteren Cluster-Knoten hinzufügen. Eine zusätzliche Zeile wird angezeigt, die den gleichen Aufbau hat wie die erste Zeile. Wählen Sie die Schaltfläche **-** (Minus), um einen Cluster-Knoten wieder aus der Konfiguration zu löschen. Sie können maximal 32 Cluster-Knoten angeben. Den letzten verbleibenden Cluster-Knoten können Sie nicht löschen.

Partner LPAP

Gibt den LPAP-Namen an, unter dem die openUTM-Partneranwendung bei einer Inbound-Kommunikation den BeanConnect Proxy und damit den Application Server adressiert, 1 bis maximal 8 Zeichen lang. Die Management Console erzeugt aus diesem Namen eine OSI-LPAP-Anweisung für die Generierung der openUTM-Partneranwendung.

Partner Idletimer (sec)

Gibt die Zeitspanne in Sekunden an, nach deren Ablauf der EIS Partner die Verbindung zum BeanConnect Proxy Container abbauen soll, falls in dieser Zeitspanne keine Kommunikation über diese Verbindung stattgefunden hat.

Es wird empfohlen, als Wert für **Partner Idletimer** einen Wert zu wählen, der kleiner ist als der bei **Proxy Idletimer** angegebene Wert.
0 bedeutet, dass der Timer nicht verwendet wird.
Mögliche Werte: 0 (Standardwert) bis 32767.

Is BS2000

Diese Option legt fest, ob sich die openUTM-Partneranwendung auf einem BS2000-System befindet oder nicht. Ist diese Option aktiviert, dann generiert die Management Console daraufhin KDCDEF- und BCMAP-Anweisungen für ein BS2000-System.

Ist diese Option nicht aktiviert, dann generiert die Management Console die KDCDEF-Anweisungen und die Hostname-Datei für Unix-, Linux- und Windows-Plattformen.

Admin Permission

Diese Option legt fest, ob bei der Outbound-Kommunikation der Proxy und damit implizit auch der Application Server in der openUTM-Partneranwendung Administratorrechte haben soll oder nicht.

Listener Port

Definiert den Port, an dem die openUTM-Partneranwendung auf Aufträge zum Verbindungsaufbau wartet. Zulässige Werte: 102 und 1025 bis 32767

Wenn Sie **Is BS2000** nicht aktiviert haben und bei **Access Point** die Option **Use Existing** wählen, dann müssen Sie hier den Wert eintragen, der in der openUTM-Partneranwendung generiert ist (ACCESS-POINT-Anweisung, Operand LISTENER-PORT).

Application Entity Qualifier

Der **Application Entity Qualifier** ist eine Adress-Komponente für den Access Point der openUTM-Partneranwendung.

Wenn Sie bei **Access Point** die Option **Use Existing** wählen, dann müssen Sie hier den Wert eintragen, der in der openUTM-Partneranwendung generiert ist (ACCESS-POINT-Anweisung, Operand APPICATION-ENTITY-QUALIFIER).

Application Process Title

Der **Application Process Title** ist eine Adress-Komponente für die UTMD-Anweisung der openUTM-Partneranwendung.

Wenn Sie bei **Access Point** die Option **Use Existing** wählen, dann müssen Sie hier den Wert eintragen, der in der openUTM-Partneranwendung generiert ist (UTMD-Anweisung, Operand APPLICATION-PROCESS-TITLE).

Wenn der EIS Partner eine UTM-Cluster-Anwendung ist und der angegebene Application Process Title (APT) aus weniger als 10 Elementen besteht, so ergänzt die Management Console den APT für jede Knoten-Anwendung um den Knoten-Index, um die Eindeutigkeit des APT zu gewährleisten.

Access Point

Definiert die Eigenschaften des Access Points, mit dem die openUTM-Partneranwendung bei Outbound-Kommunikation angesprochen wird. Der Access Point wird in der openUTM-Partneranwendung mit der KDCDEF-Anweisung ACCESS-POINT generiert.

Mit den drei Optionen **Create New (Generic Names)**, **Create New (Own Names)** und **Use Existing** entscheiden Sie, ob die Management Console KDCDEF-Anweisungen für den Access-Point erzeugt oder ob ein in der openUTM-Partneranwendung bereits vorhandener Access-Point verwendet werden soll.

- **Create New (Generic Names)**
  Wenn Sie diese Option auswählen, dann erzeugt die Management Console eine ACCESS-POINT-Anweisung mit generischen Werten. Diese Werte trägt die Management Console in die Felder **Access Point Name**, **Transport Selector** und **Transport Selector Format** ein, sie sind nicht veränderbar.
- **Create New (Own Names)**
  Wenn Sie diese Option auswählen, dann erzeugt die Management Console eine ACCESS-POINT-Anweisung mit eigenen Werten. Diese Werte trägt die Management Console in die Felder **Access Point Name**, **Transport Selector** und **Transport Selector Format** ein. Sie können die Werte verändern.
- **Use Existing**
  Wenn Sie diese Option auswählen, dann erzeugt die Management Console keine ACCESS-POINT-Anweisung. Diese Option ist für den Fall gedacht, dass Sie einen vorhandenen Access-Point in der openUTM-Partneranwendung nutzen. Sie müssen die Werte dieses Access Points in die Felder **Access Point Name**, **Transport Selector** und **Transport Selector Format** eintragen. Außerdem müssen die Werte, die Sie in **Application Entity Qualifier** und **Listener Port** eintragen, in der openUTM-Partneranwendung generiert sein.
- **Access Point Name**
  Name des Access-Points in der openUTM-Partneranwendung.

  Bei **Create New (Generic Names)** steht hier der generierte Name (nicht veränderbar).

  Bei **Create New (Own Names)** tragen Sie hier einen frei wählbaren Namen ein (1 bis maximal 8 Zeichen lang).

  Bei **Use Existing** müssen Sie den Namen eintragen, der in der ACCESS-POINT-Anweisung der openUTM-Partneranwendung generiert ist.
- **Transport Selector**

  Transport Selektor des Access Points in der openUTM-Partneranwendung.

  Bei **Create New (Generic Names)** steht hier der generierte Name (nicht veränderbar).

Bei **Use Existing** müssen Sie den Namen eintragen, der in der ACCESS-POINT-Anweisung der openUTM-Partneranwendung bei T-SEL= generiert ist.

Bei **Create New (Own Names)** tragen Sie hier einen frei wählbaren Namen ein. Dieser muss 1 bis maximal 8 Zeichen lang sein, das erste Zeichen muss ein Großbuchstabe sein, sonst sind Buchstaben, Ziffern sowie #, $ und @ erlaubt.

UTM Partner - Zusätzliche Felder im Expertenmodus

Bei aktiviertem Expertenmodus werden zusätzlich die folgenden Felder angezeigt:

- **Application Program Interface Mode of EIS Partner**

  Dieses Feld ist nur für Partner relevant, die die Schnittstelle XATMI verwenden, siehe Abschnitt „EIS Partner vom Typ XATMI konfigurieren“ auf Seite 234. Bei openUTM-Partneranwendungen, die die KDCS-Schnittstelle verwenden, muss bei **API Mode** immer **Standard** eingestellt sein.

- **Transport Selector Format**
  Format, in dem der Transport Selektor kodiert wird.

  Bei **Create New (Generic Names)** steht hier der generierte Wert (nicht veränderbar).

  Bei **Create New (Own Names)** können Sie wählen zwischen **ASCII**, **EBCDIC** und **TRANSDATA** (Standardwert). Wenn der EIS Partner unter einem BS2000-System läuft, muss **TRANSDATA** als Transport Selector Format verwendet werden.

  Bei **Use Existing** müssen Sie den Wert eintragen, der in der ACCESS-POINT-Anweisung der openUTM-Partneranwendung bei TSEL-FORMAT= generiert ist.

Registerkarte Availability Check (openUTM-Partner)

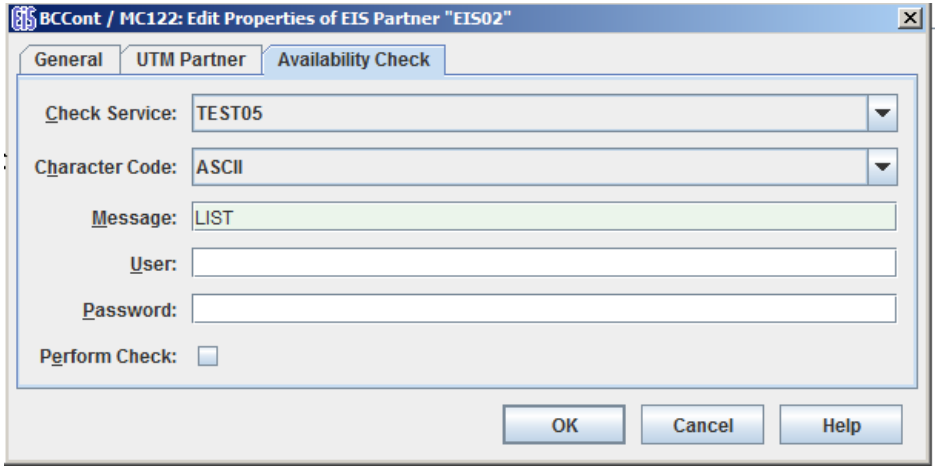


Bild 32: Eigenschaften des EIS Partners für den Availability Check

Mit diesem Dialogfeld können Sie einen Dialog-Service in der Partneranwendung auswählen, der beim Prüfen der Verfügbarkeit (Availability Check) eines EIS Partners aufgerufen wird. Dabei wird die in **Message** definierte Nachricht dem angegebenen Service am EIS Partner übergeben. Sobald die Antwort eintrifft, wird der EIS Partner als verfügbar gekennzeichnet; die Antwort wird in der Management Console ausgegeben.

Sie können dieses Dialogfeld erst dann sinnvoll bearbeiten, wenn Sie für den EIS Partner mindestens einen Outbound Service definiert haben, siehe Abschnitt „Outbound Services konfigurieren“ auf Seite 236.

Check Service

Hier werden alle für diesen Partner definierten Outbound Services angezeigt. Wählen Sie aus der Liste einen geeigneten Service aus. Der Service ist immer vom Typ **Dialog**.

Character Code

Zeichensatz, den der EIS Partner verwendet:
Mögliche Werte: **ASCII** und **EBCDIC**

Message

Nachricht, die beim Prüfen der Verfügbarkeit an den EIS Partner geschickt wird. Die maximale Länge der Nachricht beträgt 80 Zeichen.

User

Name der Benutzerkennung, falls der Service unter einer bestimmten Benutzerkennung aufgerufen werden soll.

Password

Kennwort für die Benutzerkennung, falls diese ein Kennwort erfordert.

Perform Check

Mit dieser Option können Sie die Verfügbarkeitsprüfung für diesen EIS Partner zeitweilig ausschalten, ohne dass die Einstellungen wie z.B. der aufzurufende Service verloren gehen.

Ist diese Option aktiviert, dann wird die Verfügbarkeitsprüfung mit den eingestellten Parametern durchgeführt.

Wenn Sie die Option deaktiveren, dann wird die Verfügbarkeitsprüfung für diesen EIS Partner so lange ausgeschaltet, bis die Option wieder aktiviert wird. Ausgeschaltet wird dabei der Check bei der automatischen Verfügbarkeitsprüfung und der Check, der per Kontextmenü-Befehl des Proxys oder Proxy Clusters angefordert wird. Wird der EIS Partner "per Hand" (über sein Kontextmenü) geprüft, dann wird die Prüfung unabhängig von dieser Einstellung durchgeführt.

6.6.1.2 Konfigurationsdateien für EIS Partner vom Typ openUTM

Die Management Console erstellt Konfigurationsfragmente für die EIS Partner. Wenn Sie die Proxy-Konfiguration das nächste Mal speichern (siehe Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249), generiert die Management Console alle Konfigurationsanweisungen für alle EIS Partner. Voraussetzung dafür ist, dass Sie im Register **General** die Option **Active** aktiviert haben.

Es werden Dateien mit KDCDEF-Anweisungen erstellt, für openUTM-Partneranwendungen auf einem BS2000-System werden zusätzlich Dateien mit BCMAP-Anweisungen erzeugt.

Die Konfigurationsdateien finden Sie im Verzeichnis `<MC_home>/genfiles` unter folgenden Namen:

- Einzelner Proxy:

  `ProxyID.<p-id>.EISPartnerID.<e-id>.UTM.txt` (KDCDEF-Anweisungen)

  `ProxyID.<p-id>.EISPartnerID.<e-id>.BCMAP.txt` (BCMAP-Anweisungen)

  `ProxyID.<p-id>.EISPartnerID.<e-id>.HOSTNAME.txt` (Hostname-Datei für das Mapping langer Rechnernamen, nicht für BS2000-Systeme)

- Proxy Cluster:

  `ClusterID.<c-id>.EISPartnerID.<e-id>.UTM.txt` (KDCDEF-Anweisungen)

  `ClusterID.<c-id>.EISPartnerID.<e-id>.BCMAP.txt` (BCMAP-Anweisungen)

  `ClusterID.<p-id>.EISPartnerID.<e-id>.HOSTNAME.txt` (Hostname-Datei für das Mapping langer Rechnernamen, nicht für BS2000-Systeme)

`<MC_home>` ist das Verzeichnis, unter dem die Management Console installiert ist. `<p-id>`, `<e-id>` und `<c-id>` bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

Die Datei mit BCMAP-Anweisungen wird nur für openUTM-Partner in einem BS2000-System erzeugt (Option **Is BS2000** ist aktiviert).

Weitere Einzelheiten zu den Konfigurationsaufgaben beim EIS Partner finden Sie in Kapitel „Konfiguration im EIS Partner anpassen“ auf Seite 263.

6.6.2 EIS Partner vom Typ CICS konfigurieren

Damit Sie einen EIS Partner vom Typ CICS konfigurieren können, muss der Proxy oder Proxy Cluster in der Registerkarte **General** entsprechend konfiguriert sein, siehe z.B. „Possible EIS Partner Types“ auf Seite 188.

Ein EIS Partner im Proxy Cluster wird im Wesentlichen genau so konfiguriert wie bei einem einzelnen Proxy bis auf einen kleinen Unterschied beim Konfigurieren des Communication Service.

6.6.2.1 EIS Partner vom Typ CICS hinzufügen

So fügen Sie einen neuen EIS Partner hinzu:

1. Klicken Sie unter der Liste der EIS Partner auf die Schaltfläche **Add** oder öffnen Sie das Kontextmenü eines bereits bestehenden EIS Partnerobjekts oder des Knotens **EIS Partners** und wählen Sie den Befehl **Add EIS Partner**.

   Wenn der Proxy/Proxy Cluster für die beide Partnertypen (UTM und CICS) konfiguriert ist, dann wird der Zwischendialog **Choose EIS Partner Type** eingeschoben. Wählen Sie dort **CICS application** aus.

2. Definieren Sie die Eigenschaften für den EIS Partner. Das Eigenschaftsfeld wird automatisch geöffnet. Es enthält die Registerkarten **General, Communication Service**, **CICS Partner** und **Availability Check**.

Folgende Eigenschaften werden abgefragt:

Registerkarte General (CICS-Partner)

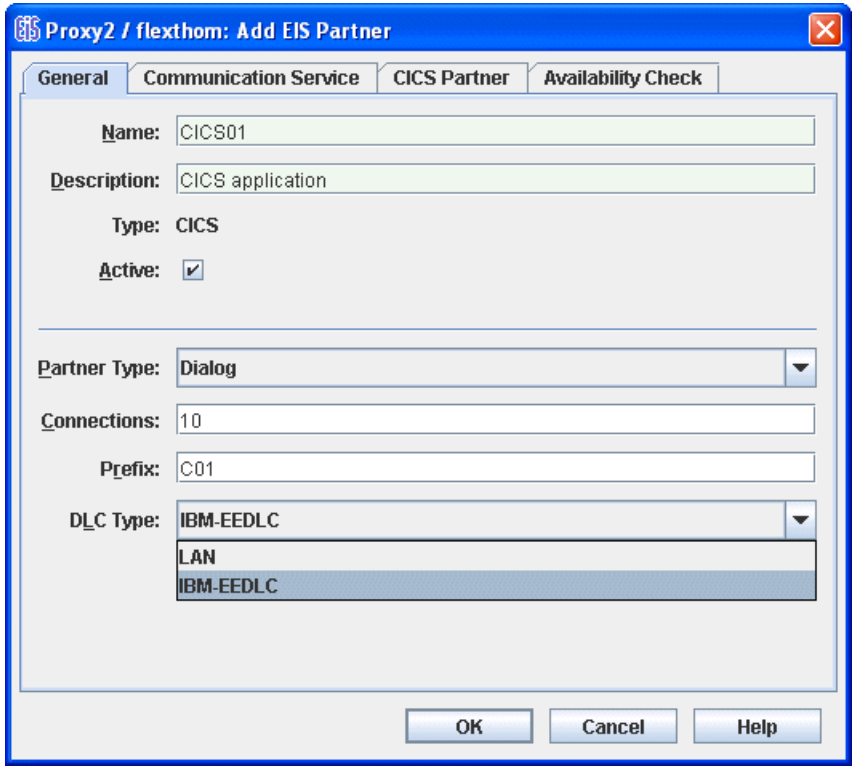


Bild 33: Allgemeine Eigenschaften eines EIS Partners vom Typ CICS

Name / Description

Name des EIS Partners. Der Name muss für den Proxy eindeutig sein. Darüber hinaus können Sie im Feld **Description** eine Beschreibung für den EIS Partner eingeben. Bei einem existierenden EIS Partner wird die EIS-ID im Feld **ID** unterhalb des Namens angezeigt (nicht änderbar).

Type

Zeigt den Typ des EIS Partners an: hier **CICS**, nicht veränderbar.

Active

Diese Option steuert, ob die Definition eines EIS Partners aktiv ist. Nur aktivierte EIS Partner werden in der Konfiguration des Proxys und der EIS Partner berücksichtigt. Nicht aktivierte EIS Partner werden im Navigationsbaum als durchgestrichen gekennzeichnet.

Partner Type

Gibt den Typ der Outbound-Kommunikation mit dem CICS-Partner an:

- **Dialog**: Kommunikation über Dialog-Services.
- **Asynchronous**: Kommunikation über Asynchron-Services.

Ein CICS-Partner kann in BeanConnect entweder nur als Dialog- oder nur als Asynchron-Partner konfiguriert werden. Sollen für einen realen CICS-Partner beide Kommunikations-Typen möglich sein, muss der reale CICS-Partner zweimal mit denselben Adressdaten und unterschiedlichen LU-Namen konfiguriert werden: Einmal als Dialog-Partner und einmal als Asynchron-Partner.

Connections

Gibt die maximale Anzahl von Verbindungen an, die gleichzeitig zwischen dem Proxy und dem EIS Partner zulässig sind.

Prefix

Das Präfix geht als Bestandteil von Namen ein, die in Konfigurationsanweisungen verwendet werden. Dieses Präfix muss für alle Proxys, die auf demselben Rechner laufen, eindeutig sein. Das Präfix muss aus genau drei Zeichen bestehen (Großbuchstaben oder Ziffern), damit die erzeugten Namen eindeutig sind. Das erste Zeichen muss ein Großbuchstabe sein.

DLC Type

Gibt die Kommunikationsart der Verbindung zwischen Proxy und EIS Partner an. Der DLC-Typ kann **LAN** oder **IBM-EEDLC** sein.

Registerkarte Communication Service

Dieses Registerblatt hat bei Proxy Clustern mit mehreren Proxys ein anderes Layout, daher werden beide Varianten dargestellt.

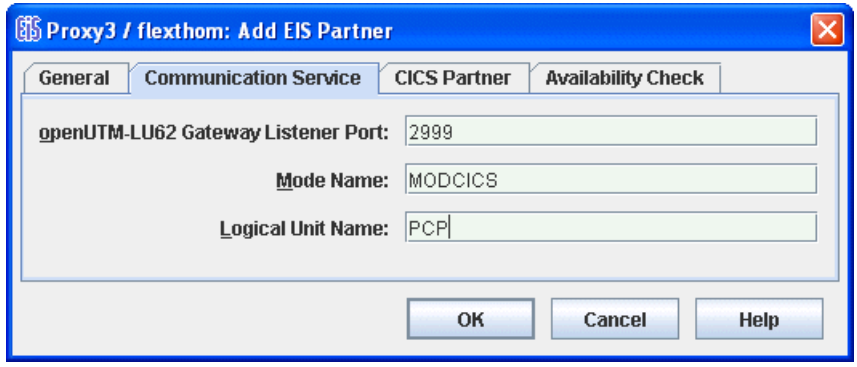


Bild 34: Eigenschaften des Communication Services eines CICS-Partners (einzelner Proxy)

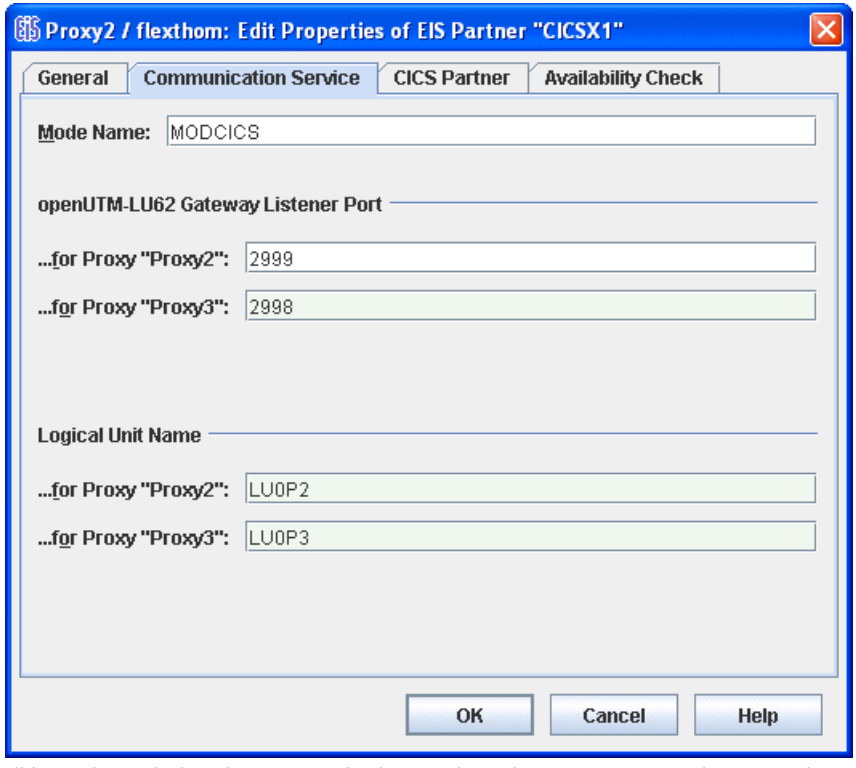


Bild 35: Eigenschaften des Communication Services eines CICS-Partners im Proxy Cluster

Mode Name

Gibt den Namen eines Eintrags in der VTAM MODETAB auf dem z/OS-Mainframe an. Ein solcher Eintrag beschreibt die Eigenschaften von Sessions zwischen Communication Service und VTAM. Es muss der Name eines Eintrags angegeben werden, der für LU6.2-Kommunikation mit CICS geeignet ist.

openUTM-LU62 Gateway Listener Port

Gibt die partnerspezifische Nummer des Port an, an dem das openUTM-LU62 Gateway auf Nachrichten wartet. Diese Portnummer darf in keiner Konfiguration eines Kommunikationsservice mit einem anderen EIS Partner und von keiner anderen Anwendung verwendet werden.

Bei einem Proxy Cluster wird pro Proxy ein Feld ausgegeben. Tragen Sie dort für jeden Proxy die Portnummer ein, an dem das openUTM-LU62-Gateway Nachrichten für den EIS Partner empfängt. Die Portnummern müssen sich unterscheiden und eindeutig einem Paar "EIS Partner <-> Proxy" zugeordnet sein, d.h. für unterschiedliche Paare "EIS Partner <-> Proxy" darf nicht dieselbe Portnummer vergeben werden.

Logical Unit Name

Gibt den eindeutigen Anwendungsnamen des Proxys im SNA-Netzwerk an.

Bei einem Proxy Cluster wird pro Proxy ein Feld ausgegeben, in das Sie den Anwendungsnamen des jeweiligen Proxys eingeben.

Registerkarte CICS Partner

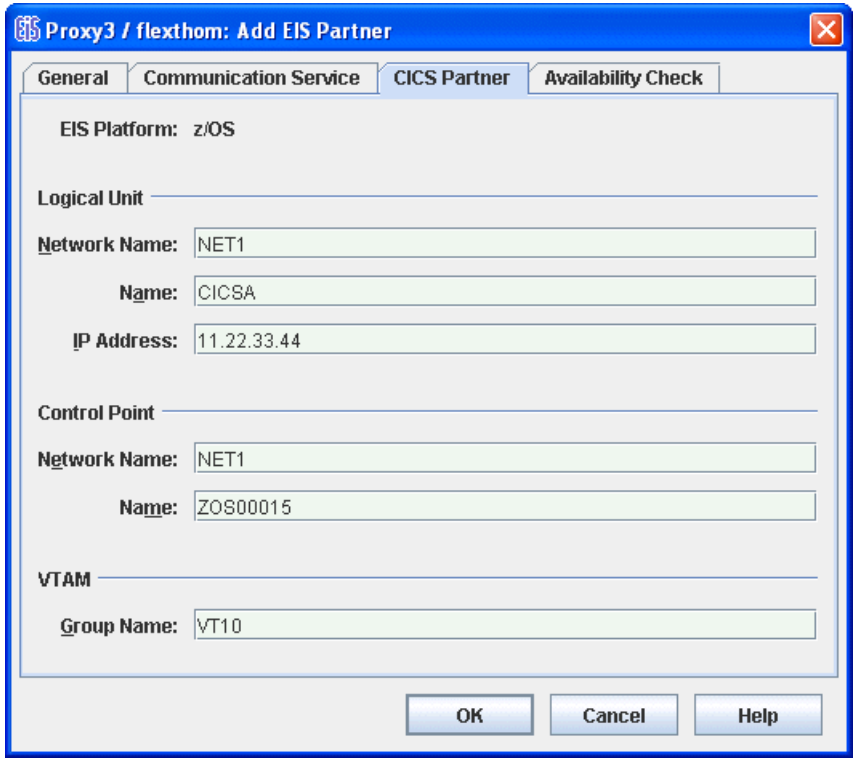


Bild 36: Eigenschaften des CICS-Partners auf einem z/OS-Mainframe

Das Layout dieses Dialogfeldes hängt vom DLC-Typ ab. Manche Felder werden nur bei einer bestimmten Einstellung angezeigt. Im Folgenden werden alle Felder beschrieben.

EIS Platform

Gibt die Plattform an, auf der der CICS-Partner läuft. Hier ist derzeit nur **z/OS** möglich, nicht änderbar.

Logical Unit

Beschreibt die Logical Unit des CICS-Partners.

- **Network Name** ist die NETID in den VTAM-Startoptionen am z/OS-Mainframe.
- **Name** ist der Anwendungsname der CICS-Region wie er am z/OS-Mainframe in der VTAM-Anweisung APPL angegeben wurde.
- Für **IP Address** ist die IP-Adresse des z/OS-Mainframe anzugeben, auf dem CICS läuft.

Control Point

Gibt den Control Point von VTAM am z/OS-Mainframe an.

- **Network Name** ist die NETID in den VTAM-Startoptionen am z/OS-Mainframe.
- **Name** ist die Angabe beim Parameter SSCPNAME in den VTAM-Startoptionen am z/OS-Mainframe.

VTAM

Group Name gibt den VTAM-Gruppennamen an. Dies ist der Name des GROUP-Makros in der VTAM Major Node-Definition für den Enterprise Extender.

MAC Address

MAC-Adresse des z/OS-Mainframe, auf dem der CICS-Partner läuft, falls der DLC-Typ LAN ist. Andernfalls wird dieses Feld nicht angezeigt.

Erläuterungen zu SNA-spezifischen Begriffen finden Sie im Glossar.

Registerkarte Availability Check (CICS-Partner)

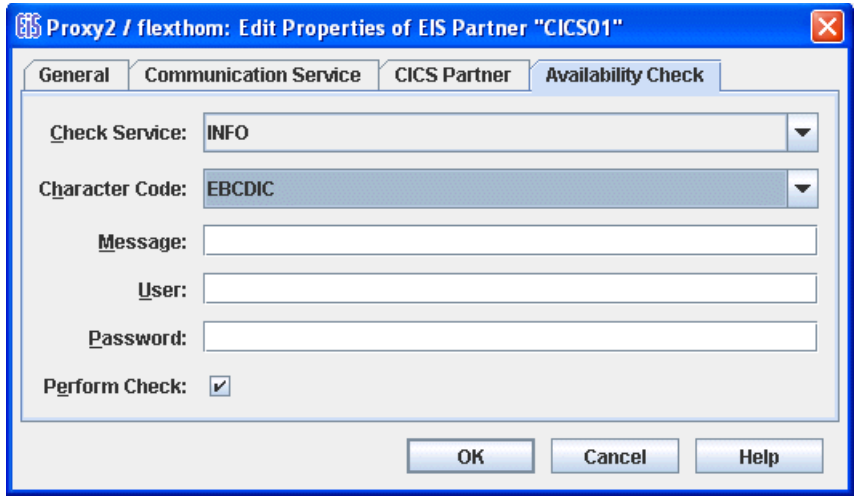


Bild 37: Eigenschaften des EIS Partners vom Typ CICS für den Availability Check

Mit diesem Dialogfeld können Sie einen Dialog-Service in der Partneranwendung auswählen, der beim Prüfen der Verfügbarkeit (Availability Check) eines EIS Partners aufgerufen wird. Dabei wird die in **Message** definierte Nachricht dem angegebenen Service der EIS Partneranwendung übergeben. Sobald die Antwort eintrifft, wird der EIS Partner als verfügbar gekennzeichnet; die Antwort wird in der Management Console ausgegeben.

Sie können dieses Dialogfeld erst dann sinnvoll bearbeiten, wenn Sie für den EIS Partner mindestens einen Outbound Service definiert haben, siehe Abschnitt „Outbound Services konfigurieren“ auf Seite 236.

Check Service

Hier werden alle für diesen Partner definierten Outbound Services angezeigt. Wählen Sie aus der Liste einen geeigneten Service aus. Der Service ist immer vom Typ **Dialog**.

Character Code

Zeichensatz, den der EIS Partner verwendet:
Mögliche Werte: **ASCII** und **EBCDIC**

Message

Nachricht, die beim Prüfen der Verfügbarkeit an den EIS Partner geschickt wird. Die maximale Länge der Nachricht beträgt 80 Zeichen.

User

Name der Benutzerkennung, falls der Service unter einer bestimmten Benutzerkennung aufgerufen werden soll.

Password

Kennwort für die Benutzerkennung, falls diese ein Kennwort erfordert.

Perform Check

Mit dieser Option können Sie die Verfügbarkeitsprüfung für diesen EIS Partner zeitweilig ausschalten, ohne dass die Einstellungen wie z.B. der aufzurufende Service verloren gehen.

Ist diese Option aktiviert, dann wird die Verfügbarkeitsprüfung mit den eingestellten Parametern durchgeführt.

Wenn Sie die Option deaktiveren, dann wird die Verfügbarkeitsprüfung für diesen EIS Partner so lange ausgeschaltet, bis die Option wieder aktiviert wird. Ausgeschaltet wird dabei der Check bei der automatischen Verfügbarkeitsprüfung und der Check, der per Kontextmenü-Befehl des Proxys oder Proxy Clusters angefordert wird. Wird der EIS Partner "per Hand" (über sein Kontextmenü) geprüft, dann wird die Prüfung unabhängig von dieser Einstellung durchgeführt.

6.6.2.2 Konfigurationsdateien für die EIS Partner vom Typ CICS

Die Management Console erstellt Konfigurationsfragmente für den definierten Proxy und die EIS Partner. Wenn Sie die Proxy-Konfiguration das nächste Mal speichern, generiert die Management Console automatisch alle Konfigurationsdateien (siehe Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249). Es werden alle Konfigurationsanweisungen für den Proxy und für alle definierten EIS Partner generiert.

Die Management Console erstellt Konfigurationsanweisungen für folgende Komponenten:

- BeanConnect Proxy-Container
- openUTM-LU62 Gateway des Proxys
- Alle Partneranwendungen des BeanConnect Proxys
- VTAM (für die Verbindung)
- Communication Service, d.h. SNAP-IX bzw. IBM Communications Server (für die Proxy-Seite der Verbindung)

Die Konfigurationsdateien finden Sie im Verzeichnis <MC_home>/genfiles unter folgenden Namen:

- Einzelner Proxy:

  ProxyID.<p-id>.EISPartnerID.<e-id>.CICS.txt

  Diese Datei enthält Generierungsanweisungen für die CICS-Partneranwendung.

  ProxyID.<p-id>.EISPartnerID.<e-id>.VTAM.txt

  Diese Datei enthält Generierungsanweisungen für die Kommunikationskomponenten auf der Seite des EIS Partners (VTAM).

- Proxy Cluster:

  ClusterID.<c-id>.ProxyID.<p-id>.EISPartnerID.<e-id>.CICS.txt

  Diese Datei enthält Generierungsanweisungen für die CICS-Partneranwendung.

  ClusterID.<c-id>.ProxyID.<p-id>.EISPartnerID.<e-id>.VTAM.txt

  Diese Datei enthält Generierungsanweisungen für die Kommunikationskomponenten auf der Seite des EIS Partners (VTAM).

<MC_home> ist das Verzeichnis, unter dem die Management Console installiert ist. <p-id>, <e-id> und <c-id> bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

Weitere Einzelheiten zu den Konfigurationsaufgaben beim EIS Partner finden Sie in Kapitel „Konfiguration im EIS Partner anpassen“ auf Seite 263.

6.6.3 EIS Partner vom Typ XATMI konfigurieren

Wenn Sie diese Funktion nutzen möchten, müssen Sie den Expertenmodus aktivieren.

Wenn Sie einen EIS Partner vom Typ XATMI konfigurieren möchten, dann muss der Proxy mit **API Mode XATMI** oder **ALL** konfiguriert sein, siehe Abschnitt „Application Program Interface Mode (API Mode)“ auf Seite 196.

Beim Typ des EIS Partners müssen Sie beim Konfigurieren des Proxys entweder **Only UTM Partners** oder **UTM and CICS Partners** angeben. Einen EIS Partner vom Typ XATMI konfigurieren Sie immer wie einen Partner vom Typ openUTM.

So fügen Sie einen neuen XATMI-Partner hinzu:

1. Klicken Sie unter der Liste der EIS Partner auf die Schaltfläche **Add** oder öffnen Sie das Kontextmenü eines bereits bestehenden EIS Partnerobjekts oder des Knotens **EIS Partners** und wählen Sie den Befehl **Add EIS Partner**.
2. Definieren Sie die Eigenschaften für den EIS Partner. Das Eigenschaftsfeld wird automatisch geöffnet. Es enthält die Registerkarten **General, UTM Partner** und **Availability Check**.

   Versorgen Sie die Felder wie einem openUTM-Partner, siehe Abschnitt „EIS Partner vom Typ openUTM hinzufügen“ auf Seite 213. Der einzige Unterschied zu einem EIS Partner vom Typ **openUTM** besteht darin, dass bei eingeschaltetem Expertenmodus im Feld **API Mode** der Wert **XATMI** angezeigt wird.

Es werden die gleichen Konfigurationsdateien erzeugt wie bei einem EIS Partner vom Typ **openUTM**, siehe auch Abschnitt „Konfigurationsdateien für EIS Partner vom Typ openUTM“ auf Seite 223.

6.6.4 EIS Partner entfernen

Sie entfernen einen EIS Partner, indem Sie im Kontextmenü des EIS Partners den Befehl **Remove EIS Partner** wählen. Alternativ können Sie einen oder mehrere EIS Partner aus der Liste auswählen und unter der Liste auf die Schaltfläche **Remove** klicken.

6.7 Outbound-Kommunikation konfigurieren

Mit Outbound-Kommunikation ist die Kommunikation vom Application Server zum EIS Partner gemeint. Für eine Outbound-Kommunikation müssen Sie Outbound Services und Outbound Communication Endpoints konfigurieren.

Ein Outbound Service stellt einen Service (z.B. Transaktionscode) innerhalb des EIS Partners dar. Jeder Service eines EIS Partners, der vom Application Server aus aufgerufen werden soll, muss als Outbound Service konfiguriert sein. Er kann entweder implizit verwendet werden, falls er einem Communication Endpoint als Service zugewiesen ist, oder er kann explizit in der EJB über die Methode `setServiceName` gesetzt werden.

Jeder Outbound Communication Endpoint stellt innerhalb eines EIS Partners einen Kommunikationsendpunkt für Outbound-Kommunikation dar und ist deshalb EIS-spezifisch. Einem Outbound Communication Endpoint wird über die Konfigurations-Property `connectionURL` eine Connection Factory zugewiesen, die somit einem bestimmten EIS Partner zugewiesen ist. Ein EIS Partner kann mehrere Outbound Communication Endpoints haben.

Beispiele zur Verwendung der BeanConnect-Programmierschnittstellen bei der Entwicklung von EJBs finden Sie im Abschnitt „Code-Beispiele für Outbound-Kommunikation“ auf Seite 466.

6.7.1 Outbound Services konfigurieren

Outbound Services konfigurieren Sie, indem Sie im Navigationsbaum eines Proxys unterhalb des Knotens **Outbound** auf den Knoten **Services** klicken.

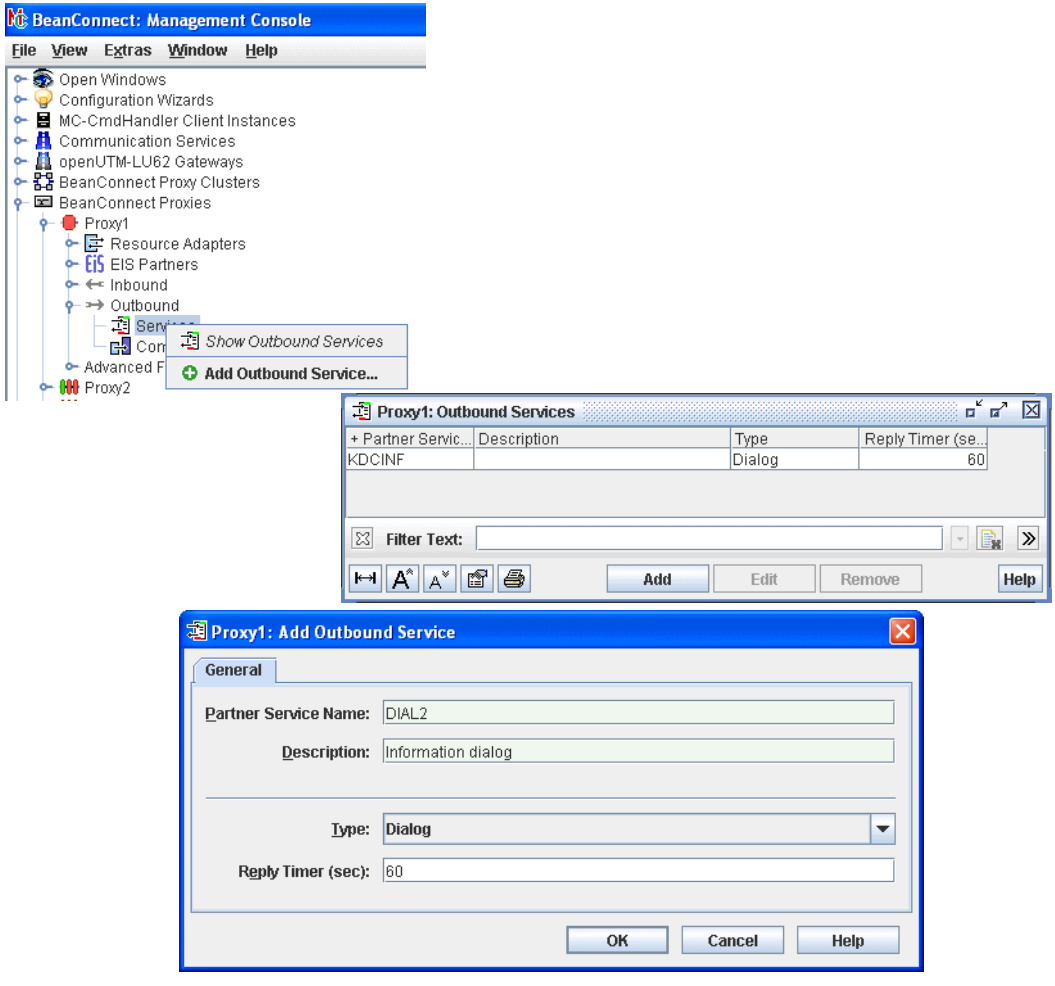


Bild 38: Outbound Services konfigurieren

Sie zeigen die Liste der Outbound Services eines Proxys an, indem Sie unter dem Knoten **Outbound** den Teilbaum öffnen und auf **Services** klicken. Alternativ können Sie das Kontextmenü des Knotens **Services** öffnen und den Befehl **Show Outbound Services** wählen.

Einen neuen Outbound Service fügen Sie hinzu, indem Sie unter der Liste auf die Schaltfläche **Add** klicken. Sie können aber auch aus dem Kontextmenü eines bereits bestehenden Services oder im Knoten **Services** den Befehl **Add Outbound Service**... wählen. Das Eigenschaftsfeld wird geöffnet.

Folgende Eigenschaften müssen für einen Outbound Service festgelegt werden:

Partner Service Name / Description

Gibt den Namen des Services innerhalb des EIS Partners an, der durch diesen Outbound Service vertreten wird. Darüber hinaus können Sie im Feld **Description** eine Beschreibung für den Outbound Service eingeben.

Type

Gibt den Kommunikationstyp des Services an. Der Kommunikationstyp kann **Dialog** oder **Asynchronous** sein.

Reply Timer (sec)

Der Proxy überwacht die Antworten eines Outbound Services. Wenn innerhalb der hier definierten Zeit keine Antwort eingegangen ist, setzt der Proxy die entsprechende Transaktion zurück. Dieser Parameter kann nur für Services vom Typ **Dialog** eingestellt oder geändert werden.

Outbound Service entfernen

Einen Outbound Service entfernen Sie, indem Sie im Kontextmenü des Services den Befehl **Remove Outbound Service** wählen. Alternativ können Sie einen oder mehrere Services aus der Liste auswählen und unter der Liste auf die Schaltfläche **Remove** klicken.

6.7.2 Outbound Communication Endpoints konfigurieren

Outbound Communication Endpoints konfigurieren Sie, indem Sie im Navigationsbaum eines Proxys unter dem Knoten **Outbound** auf den Knoten **Communication Endpoints** klicken.

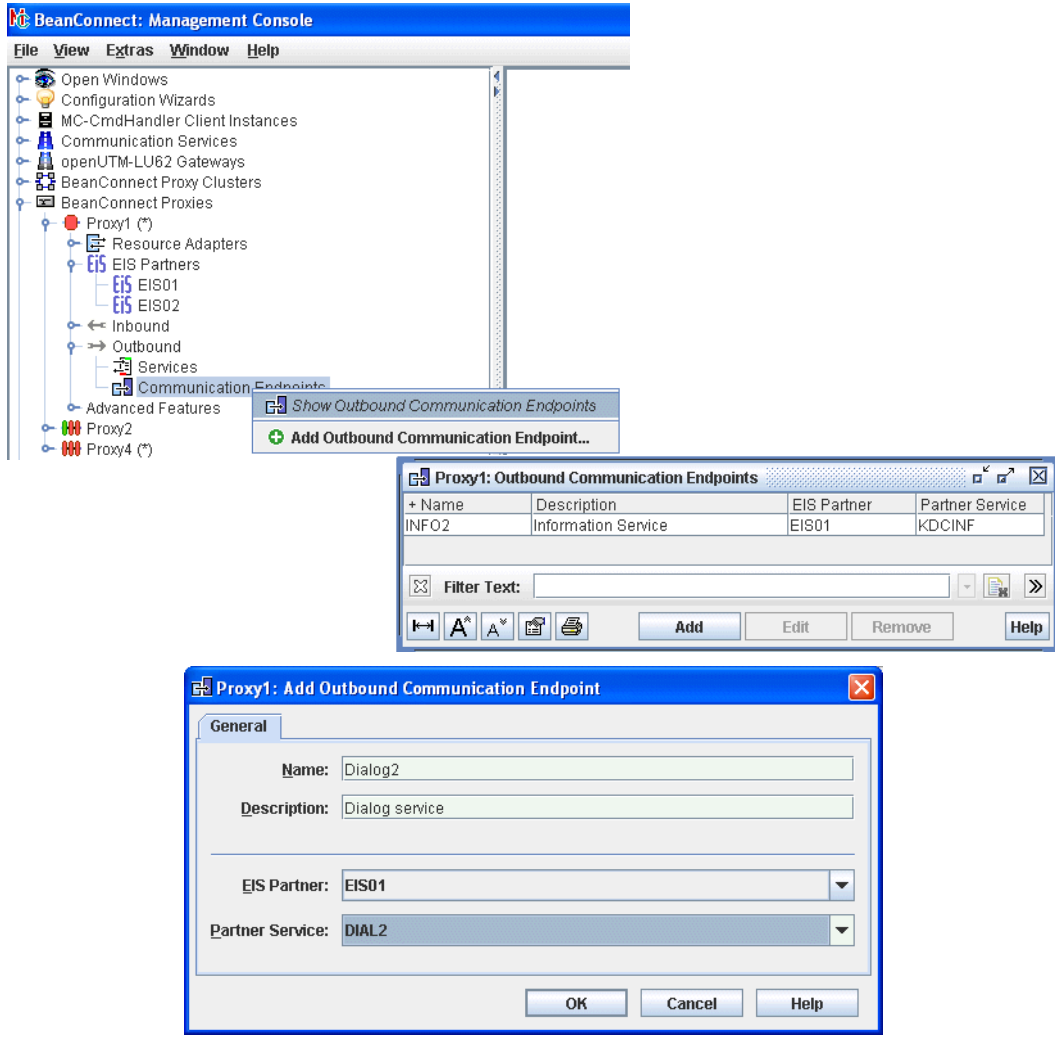


Bild 39: Outbound Communication Endpoints konfigurieren

Sie zeigen die Liste der Outbound Communication Endpoints eines Proxys an, indem Sie auf den Knoten **Communication Endpoints** klicken. Oder Sie öffnen das Kontextmenü des Knotens **Communication Endpoints** und wählen den Befehl **Show Outbound Communication Endpoints**.

Einen neuen Outbound Communication Endpoint fügen Sie hinzu, indem Sie auf die Schaltfläche **Add** klicken, die sich unterhalb der Tabelle befindet. Sie können aber auch aus dem Kontextmenü eines bereits bestehenden Endpoints oder im Knoten **Communication Endpoints** den Befehl **Add Outbound Communication Endpoint...** wählen. Das Eigenschaftsfeld wird geöffnet.

Folgende Eigenschaften müssen für einen Outbound Communication Endpoint festgelegt werden:

Name / Description

Name gibt den symbolischen Namen des Outbound Communication Endpoints an. Zusätzlich können Sie im Feld **Description** eine Beschreibung für den Outbound Communication Endpoint eingeben.

Der Deployer einer Bean verwendet diesen Namen beim Deployment der Bean im Application Server, um den Bezug zu dem Service innerhalb des EIS Partners herzustellen. Weitere Einzelheiten zu diesem Thema finden Sie im Abschnitt „Kommunikation für Outbound über OSI TP / LU6.2 konfigurieren“ auf Seite 109.

EIS Partner

Name des EIS Partners, zu dem der Communication Endpoint gehört. Der EIS Partner muss zuvor eingerichtet worden sein.

Partner Service

Gibt den tatsächlichen Namen des Services innerhalb des EIS Partners an, der durch diesen Outbound Communication Endpoint vertreten wird. Der Service muss bereits als Outbound Service definiert worden sein. Alle definierten Services werden in der Liste angezeigt.

Jeder `<connector-instance>`-Eintrag, der in der Datei `weblogic-ra.xml` enthalten ist, muss auf einen definierten Outbound Communication Endpoint zeigen (siehe Abschnitt „Konfigurations-Properties für OSI TP / LU6.2 definieren“ auf Seite 111).

Einen Outbound Communication Endpoint entfernen Sie, indem Sie im Kontextmenü des Endpoints den Befehl **Remove Outbound Communication Endpoint** wählen. Alternativ können Sie einen oder mehrere Endpoints aus der Liste auswählen und unter der Liste auf die Schaltfläche **Remove** klicken.

6.8 Inbound-Kommunikation konfigurieren

Bei Inbound-Kommunikation sendet ein EIS Partner Nachrichten an eine Message Endpoint Application in einem Java EE Application Server.

Der Name des Message Endpoints, der im Deployment Descriptor der OLTP Message-Driven Bean definiert ist, muss dem Proxy bekanntgegeben werden. Dazu konfigurieren Sie in der Management Console einen Inbound Message Endpoint. Der Name dieses Inbound Message Endpoints muss mit der Property `messageEndpoint` der OLTP Message-Driven Bean übereinstimmen, die in der Datei `ejb-jar.xml` definiert ist.

Beispiele zur Verwendung der BeanConnect-Programmierschnittstellen für die Entwicklung von OLTP Message-Driven Beans finden Sie im Abschnitt „Code-Beispiel für Inbound-Kommunikation“ auf Seite 484.

6.8.1 Inbound Message Endpoints konfigurieren

Auf die Inbound Message Endpoints kann über den Knoten **Message Endpoints** zugegriffen werden, der sich im Proxy-Teilbaum des Navigationsbaums unter dem Knoten **Inbound** befindet.

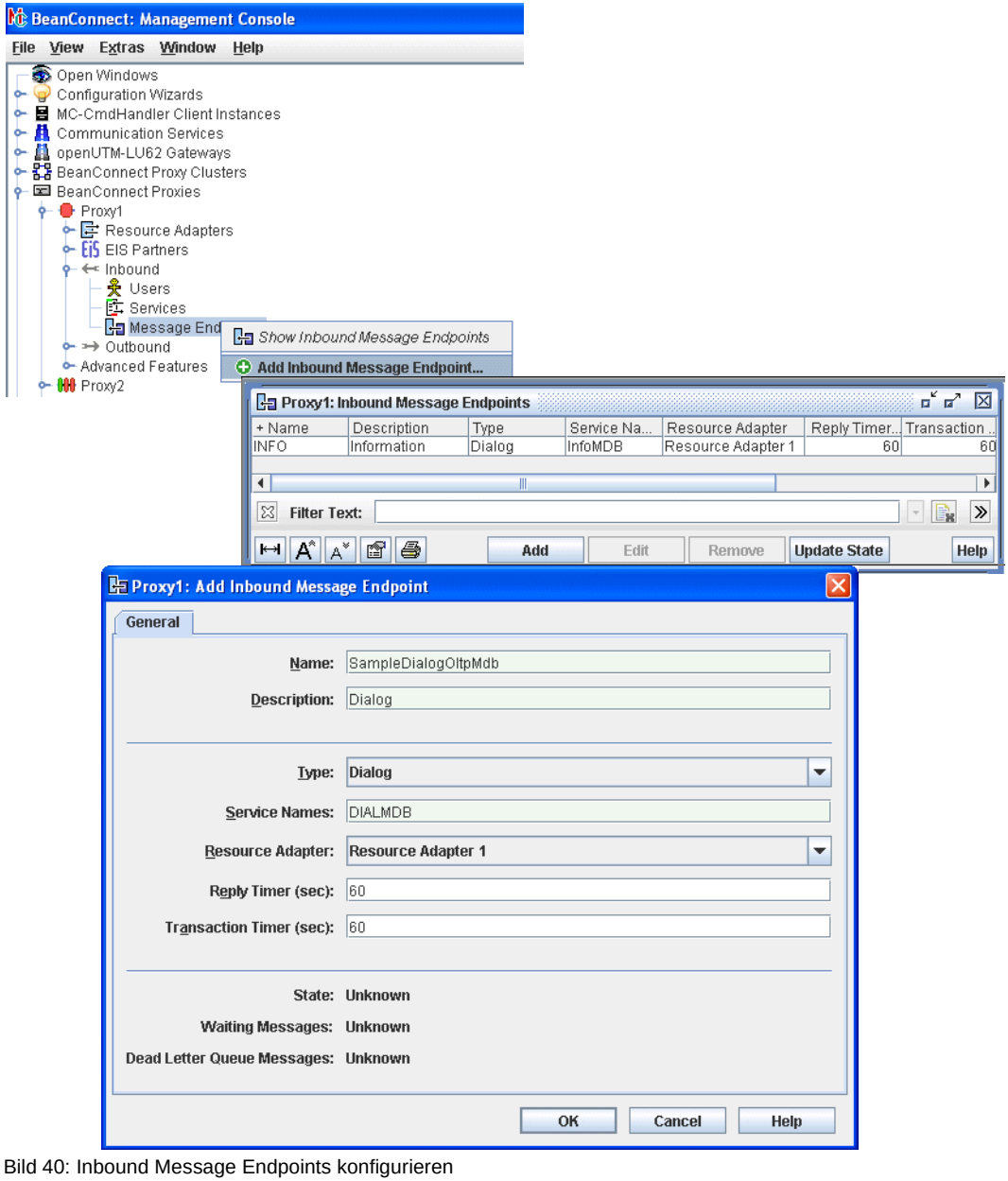


Bild 40: Inbound Message Endpoints konfigurieren

Sie zeigen die Liste der Inbound Message Endpoints eines Proxys an, indem Sie auf den Knoten **Message Endpoints** klicken. Oder Sie öffnen das Kontextmenü des Knotens **Message Endpoints** und wählen den Befehl **Show Inbound Message Endpoints**.

Einen neuen Inbound Message Endpoint fügen Sie hinzu, indem Sie unter der Liste auf die Schaltfläche **Add** klicken. Sie können aber auch aus dem Kontextmenü eines bereits bestehenden Endpoints oder im Knoten **Message Endpoints** den Befehl **Add Inbound Message Endpoint...** wählen. Das Eigenschaftsfeld wird geöffnet.

Folgende Eigenschaften müssen für einen Inbound Message Endpoint festgelegt werden:

Name / Description

Name gibt den symbolischen Namen des Inbound Message Endpoints an. Dieser Name muss mit dem Namen übereinstimmen, der im Deployment Descriptor der OLTP Message-Driven Bean innerhalb des Application Servers verwendet wird (Property `messageEndpoint` in der Datei `ejb-jar.xml`, siehe Abschnitt „Kommunikation für Inbound konfigurieren“ auf Seite 141). Zusätzlich können Sie im Feld **Description** eine Beschreibung für den Inbound Message Endpoint eingeben.

Type

Gibt den Kommunikationstyp der Verbindung an. Abhängig von dem Message-Listener-Interface, das von der OLTP Message-Driven Bean implementiert wird, kann der Kommunikationstyp **Dialog** oder **Asynchronous** sein.

BeanConnect unterstützt folgende Message-Listener-Interfaces (die als `messaging-type` in der Datei `ejb-jar.xml` definiert sind):

- `net.fsc.jca.communication.AsyncOltpMessageListener`
  (asynchrone Kommunikation)
- `net.fsc.jca.communication.OltpMessageListener`
  (dialogbasierte Kommunikation)
- `javax.resource.cci.MessageListener`
  (dialogbasierte Kommunikation)

Service Names

Definiert einen oder mehrere Inbound Services, die dem Inbound Message Endpoint zugewiesen sind; ein Service-Name darf bis zu acht Zeichen lang sein. Wenn Sie mehrere Services angeben, müssen diese durch Komma getrennt sein. Einem Inbound Service muss aber immer genau ein Inbound Message Endpoint zugeordnet sein.

Für die Inbound-Kommunikation mit einem openUTM-Partner über OSI TP muss jeder dieser Namen ausdrücklich beim EIS Partner generiert werden. Dazu wird der Name als Wert des Parameters RTAC in einer LTAC-Anweisung angegeben.

In einem CICS-Programm wird der Service-Name zur Adressierung des Inbound Message Endpoints verwendet.

Sie können die Eigenschaften eines Inbound Services ändern, siehe Abschnitt „Inbound Services konfigurieren“ auf Seite 245.

Resource Adapter

Hier wählen Sie im Multi Resource Adapter Betrieb den Resource Adapter aus, der dem Inbound Message Endpoint zugeordnet ist. Wenn nur ein Resource Adapter definiert ist, dann wird dieser angezeigt (nicht veränderbar). Im Cluster-Betrieb wird dieses Feld nicht ausgegeben.

Reply Timer (sec)

Überwacht die Antwortzeit des Resource Adapters beim Aufruf der OLTP Message-Driven Bean.

Falls nach Ablauf der Zeit noch keine Antwort vom Resource Adapter eingetroffen ist, baut der Proxy Container die Verbindung zum Resource Adapter ab und setzt ggf. die Transaktion zurück.

Der Wert 0 bedeutet keine Überwachung.

Transaction Timer (sec)

Überwacht die Transaktionsdauer im Application Server, wenn die Transaktion an den Application Server propagiert wird. Der Wert 0 bedeutet keine Überwachung.

Eine Transaktion wird an den Application Server propagiert,

- wenn die Methode `onMessage()` der OLTP Message-Driven Bean mit dem Transaktionsattribut `Required` deployt und für **Type** `Asynchron` ausgewählt wurde oder
- wenn die Methode `onMessage()` der OLTP Message-Driven Bean mit dem Transaktionsattribut `Required` deployt, für **Type** `Dialog` ausgewählt wurde und eine Transaktion vom EIS in den Proxy propagiert wurde.

Falls die Transaktion nach Ablauf der Zeit nicht beendet wurde, dann wird die Transaktion zurückgesetzt.

Berücksichtigen Sie bitte die Verarbeitungsdauer beim EIS Partner, z.B. für einen Datenbankzugriff. Stellen Sie den Wert daher nicht zu niedrig ein.

Wenn Sie beide Timer einschalten (beide Werte > 0), dann sollten Sie den **Transaction Timer** mindestens so groß wählen wie den **Reply Timer**.

State

Zeigt den Status des Inbound Message Endpoints an. Der Inbound Message Endpoint kann sich im Zustand **Unknown**, **Available** oder **Not Available** befinden.

- **Unknown** bedeutet, dass der Status noch nicht überprüft wurde.
- **Available** bedeutet, dass ein Inbound Message Endpoint mit diesem Namen im Resource Adapter vorhanden ist. Der Proxy-Container muss verfügbar sein, damit der Status eines Inbound Message Endpoints ermittelt werden kann.
- **Not Available** bedeutet, dass kein Inbound Message Endpoint mit diesem Namen im Resource Adapter vorhanden ist.

Den Status können Sie aktualisieren, indem Sie unter der Liste auf die Schaltfläche **Update State** klicken. Alternativ können Sie auch das Kontextmenü des Endpoints öffnen und den Befehl **Update State** wählen. Es werden zusätzlich die Namen der anderen im Resource Adapter verfügbaren Inbound Message Endpoints angezeigt, die Sie nicht mit der Management Console definiert haben.

Waiting Messages

Dieses Feld wird nur im Expertenmodus angezeigt. Der Wert gibt die Anzahl der Nachrichten an, die an den bei **Service Names** angezeigten Service gerichtet sind und derzeit im Proxy Container warten.

Unknown bedeutet, dass der Status noch nicht überprüft wurde. Werte `> 0` sind nur beim Typ **Asynchron** möglich.

Dead Letter Queue Messages

Dieses Feld wird nur im Expertenmodus angezeigt. Der Wert gibt die Anzahl der Nachrichten an, die ursprünglich an diesen Message Endpoint gerichtet waren und die jetzt in der Dead Letter Queue des Proxy Containers stehen.

Unknown bedeutet, dass der Status noch nicht überprüft wurde. Werte `> 0` sind nur bei Type `Asynchron` möglich.

Wenn der Proxy zu dem Zeitpunkt nicht läuft, zu dem der Inbound Message Endpoint hinzugefügt wird, muss die Konfiguration vor dem nächsten Start des Proxys aktualisiert werden (**Update Configuration**, siehe Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249).

Einen Inbound Message Endpoint entfernen Sie, indem Sie im Kontextmenü des Endpoints den Befehl **Remove Inbound Message Endpoint** wählen. Alternativ können Sie einen oder mehrere Endpoints aus der Liste auswählen und unter der Liste auf die Schaltfläche **Remove** klicken.

6.8.2 Inbound Services konfigurieren

Ein Inbound Service ist ein Service, den ein EIS Partner bei Inbound-Kommunikation adressiert. Sie geben den Namen des Services beim Konfigurieren eines Inbound Message Endpoints an. Sie können bei einem Inbound Message Endpoint auch mehrere Servicenamen angeben, einem Servicenamen muss aber immer genau ein Inbound Message Endpoint zugeordnet sein.

Sie können die Codierungseigenschaften eines Inbound Services ändern. Dazu gehen Sie wie folgt vor:

- Klicken Sie auf den Knoten **Services** im Knoten **Inbound** und klicken Sie auf **Show Inbound Services**. Es wird die Liste aller Inbound Services angezeigt.
- Markieren Sie in der Liste den gewünschten Inbound Service und klicken Sie auf die Schaltfläche **Edit** (Alternative: Doppelklick auf den Service).
- Konfigurieren Sie den Inbound Service im Dialogfeld **Edit Properties of Inbound Service...** .

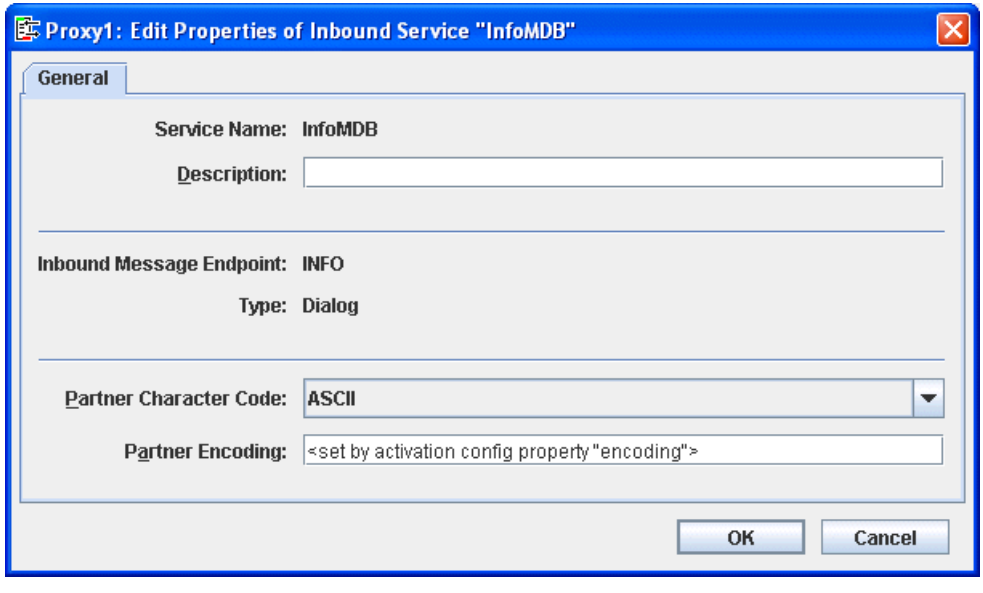


Bild 41: Inbound Service konfigurieren

Es werden folgende Eigenschaften für einen Inbound Service angezeigt oder können festgelegt werden:

Service Name

Name des Inbound Services wie er beim zugeordneten Message Endpoint angegeben wurde, nicht veränderbar. Unter diesem Namen wird der Service vom EIS Partner adressiert.

Description

Hier können Sie eine Beschreibung eingeben.

Inbound Message Endpoint

Name des Communication Message Endpoint, nicht veränderbar.

Type

Typ des Inbound Services (**Dialog** oder **Asynchronous**), nicht veränderbar.

Partner Character Code

Gibt die Art der Zeichensatzcodierung im EIS Partner an, mögliche Auswahl **ASCII** oder **EBCDIC**. Diese Einstellung wird verwendet, um eine korrekt codierte Nachricht an den EIS Partner zu senden, wenn vor Aufruf des Inbound Message Endpoints ein Fehler auftritt.

Partner Encoding

Name einer Code-Tabelle, die für die Konvertierung von Byte-Code (z.B. EBCDIC) in Java Unicode verwendet wird. Wenn Sie hier eine Code-Tabelle angeben, werden folgende Werte im Deployment Descriptor der Message-Driven Bean überschrieben:

- `encoding` wird durch den hier angegebenen Wert ersetzt
- `encodingActive` wird auf `true` gesetzt

Der Wert `<set by activation config property "encoding">` (Voreinstellung) oder eine leere Eingabe bewirken, dass die Einstellung im Deployment Descriptor verwendet wird.

Ist der Proxy mit **API Mode: All** konfiguriert, dann wird Im Expertenmodus zusätzlich die Option **XATMI** angezeigt. Sie müssen diese Option aktivieren, wenn der Service von einem XATMI-Partner verwendet werden soll.

6.8.3 Benutzer für den Zugriff auf Inbound Message Endpoints einrichten

Wenn der EIS Partner für die Inbound-Kommunikation Benutzerkennungen verwendet, müssen Sie diese im Proxy definieren, da der Auftrag sonst abgelehnt wird. Benötigt die Benutzerkennung ein Passwort, müssen Sie dieses ebenfalls definieren.

Wenn Sie die JCA 1.6 Funktionalität Security Inflow nutzen möchten, muss der EIS Partner für die Inbound-Kommunikation Benutzerkennungen verwenden.

Ansonsten ist deren Einrichtung optional.

Sie greifen auf die Benutzereinträge über den Knoten **Users** zu, der sich im Proxy-Teilbaum des Navigationsbaums unter dem Knoten **Inbound** befindet.

Die Liste der Benutzer zeigen Sie an, indem Sie auf den Knoten **Users** klicken. Alternativ können Sie das Kontextmenü des Knotens **Users** öffnen und den Befehl **Show Inbound Users** wählen.

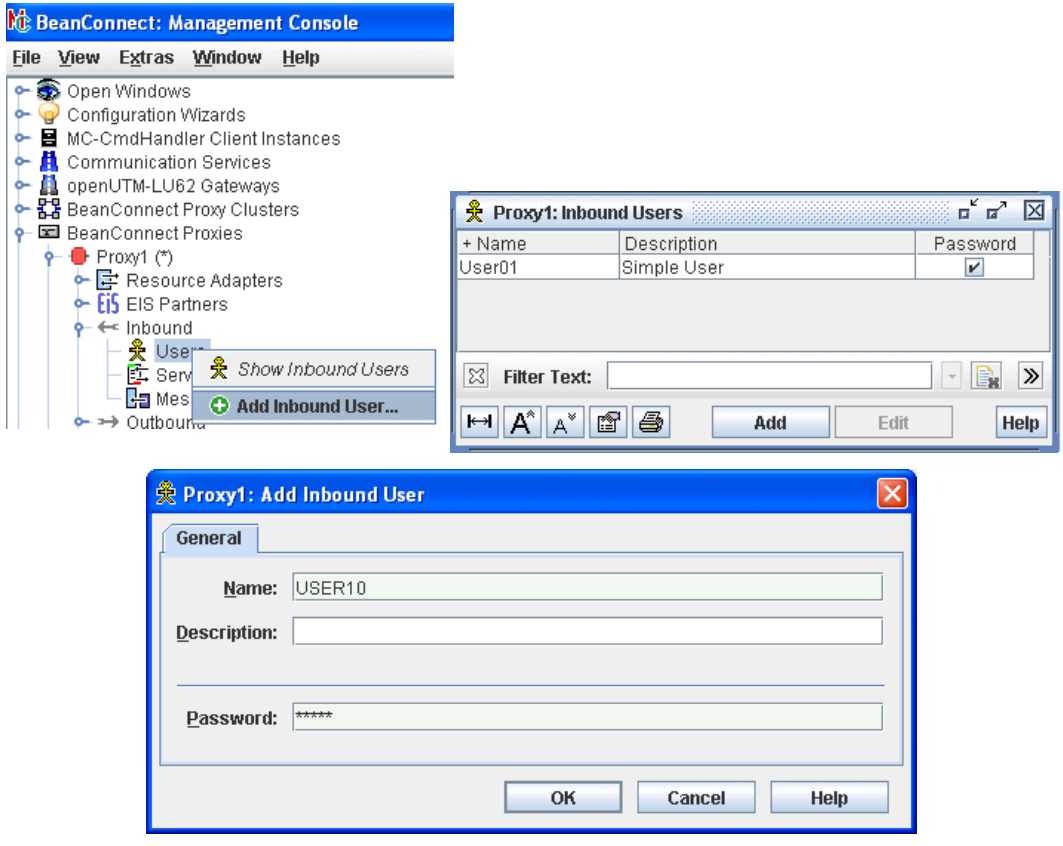


Bild 42: Benutzer für Inbound-Kommunikation konfigurieren

Einen neuen Benutzer fügen Sie hinzu, indem Sie unter der Liste auf die Schaltfläche **Add** klicken. Sie können aber auch aus dem Kontextmenü eines bereits bestehenden Benutzers oder im Knoten **Users** den Befehl **Add Inbound User...** wählen.

Geben Sie im Feld **Name** den gewünschten Namen ein. Darüber hinaus können Sie im Feld **Description** eine Beschreibung sowie im Feld **Password** ein Passwort für den Inbound User eingeben. Die Eingabe von Beschreibung und Passwort ist optional.

Einen Benutzer entfernen Sie aus der Management Console, indem Sie im Kontextmenü des Benutzers den Befehl **Remove Inbound User** wählen. Alternativ können Sie einen oder mehrere Benutzer aus der Liste auswählen und unter der Liste auf die Schaltfläche **Remove** klicken.

6.8.4 Fehlermeldungs-Präfix für Inbound konfigurieren

Treten bei der Inbound-Kommunikation Fehler auf, so wird ggf. eine Fehlermeldung an den betroffenen EIS Partner gesendet. Diese Fehlermeldungen haben standardmäßig das Präfix **BCSYSEX**.

Sie können dieses Präfix wie folgt ein- und ausschalten:

- Wählen Sie im Kontextmenü des Knoten **Inbound** den Befehl **Configure Inbound Error Prefix...** .

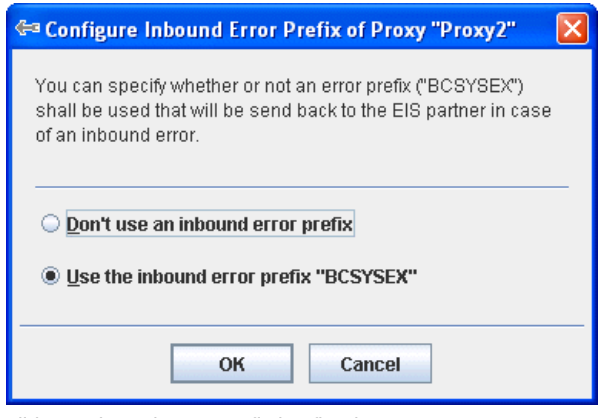


Bild 43: Inbound Error Präfix konfigurieren

- Wählen Sie im Dialogfeld die entsprechende Option aus:

  Don't use an inbound error prefix (Präfix ausschalten)

  Use the inbound error prefix "BCSYSEX" (Präfix einschalten)

Der betroffene Proxy darf dabei nicht laufen. Die Änderung wird beim Speichern des Proxys aktiviert.

6.9 Konfiguration eines BeanConnect Proxys speichern und aktivieren

Nachdem Sie mit der Management Console entweder einen neuen Proxy hinzugefügt oder die Konfiguration eines bereits bestehenden Proxys geändert haben, müssen Sie die Konfiguration speichern und aktivieren, damit sie wirksam werden kann. Verwenden Sie die Todo-Liste, um Informationen zu ausstehenden Aktivitäten zu erhalten (siehe Abschnitt „Todo-Aktionen“ auf Seite 171).

Die folgende Beschreibung gilt auch, wenn Sie den BeanConnect Proxy im Cluster betreiben.

Die erforderlichen Schritte hängen von den Änderungen ab, die Sie vorgenommen haben.

Wurden für das openUTM-LU62 Gateway und den Communication Service Daten wie der LU-Name oder der Control Point Name geändert, muss die Konfiguration gespeichert und für die Komponenten ein Restart durchgeführt werden. Änderungen werden erst danach wirksam.

Damit die Änderungen wirksam werden, muss die Konfiguration gespeichert und für den Proxy-Container ein Restart durchgeführt werden, wenn Sie

- die Einstellungen für die Kommunikation mit dem Resource Adapter geändert haben (siehe Abschnitt „BeanConnect Resource Adapter konfigurieren“ auf Seite 202).
- einen Inbound User, einen Inbound Message Endpoint, einen Outbound Service oder einen Outbound Communication Endpoint erstellt, verändert oder gelöscht haben.

Wählen Sie einen der folgenden Befehle aus dem Kontextmenü des Proxys:

- Wenn der Proxy läuft: **Save/Restart** – **Save & Restart Proxy**
- Wenn der Proxy nicht läuft: **Save/Restart** – **Save** und anschließend den Befehl **Start Proxy**, um den Proxy zu starten.

Die nachfolgenden Schritte sind notwendig und ausreichend, wenn Sie einen EIS Partner hinzugefügt oder gelöscht oder wenn Sie seine Konfiguration geändert haben. Solange der Proxy läuft, kann die neue Konfiguration nicht aktiviert werden. Verfahren Sie deshalb nach Abschluss der Konfigurationsaktivitäten wie folgt:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Save/Restart** – **Save**.

   Die neu definierte Konfiguration ist somit gespeichert und es werden neue Konfigurationsdateien für alle EIS Partner des Proxys generiert.

2. Wenn der Proxy gerade läuft, beenden Sie ihn, indem Sie in seinem Kontextmenü den Befehl **Stop Proxy** aufrufen. Alle Proxy-Komponenten werden beendet.
3. Die neue Konfiguration aktivieren Sie, indem Sie im Proxy-Kontextmenü den Befehl **Update Configuration** aufrufen.

4. Den Proxy starten Sie, indem Sie in seinem Kontextmenü den Befehl **Start Proxy** aufrufen. Der Proxy und die Proxy-Komponenten werden mit der geänderten Konfiguration gestartet.
5. Ggf. müssen die neuen Konfigurationsdateien für den EIS Partner beim EIS Partner eingebracht werden.

6.10 Management Console Command Handler (MC-CmdHandler) konfigurieren

Der Management Console Command Handler (MC-CmdHandler) ist eine stand-alone Java-Anwendung, mit der die Management Console entfernte Proxys, Proxy-Komponenten, Resource Adapter oder die Log4j-Konfiguration verwalten kann.

Zu jedem BeanConnect Proxy wird auch ein MC-CmdHandler installiert. Zusätzlich können Sie den MC-CmdHandler separat auf anderen Rechnern als dem Proxy-Rechner installieren. Dies ist in folgenden Fällen nötig:

- Wenn der Zugriff auf benötigte Dateien wie z.B. den Deployment Descriptor `ra.xml` im BeanConnect-RAR-Archiv oder die Log4j-Konfiguration des Resource Adapters nicht möglich ist, weil sie auf einem anderen Rechner und/oder in anderen Kennungen liegen.
- Wenn der Zugriff auf Komponenten wie openUTM-LU62 oder Communication Services nicht möglich ist, weil sie auf einem anderen Rechner und/oder in anderen Kennungen liegen.

Der MC-CmdHandler ist ein Socket-Listener, der am Listener Port auf Aufträge wartet, die von der Management Console vergeben wurden. Der MC-CmdHandler kann grundlegende Dateitransfer-Aufgaben ausführen, wie z.B. Verzeichnisse überwachen, Informationen zu Dateien bereitstellen, Dateien holen und aktualisieren. Darüber hinaus können auch Skripts auf einem entfernten System ausgeführt werden.

Folgende Bedingungen müssen erfüllt sein, bevor die oben aufgeführten Komponenten auf den entfernten Rechnern verwaltet werden können:

- Der MC-CmdHandler muss auf dem Rechner gestartet werden, auf dem der Proxy bzw. die zu administrierenden Komponenten laufen (Resource Adapter, openUTM-LU62 und Communication Service).
- Die Management Console muss auf den MC-CmdHandler zugreifen können.

6.10.1 Sicherheit und Berechtigungen

Der MC-CmdHandler prüft die Benutzer-Autorisierung. Auf diese Weise wird verhindert, dass nicht-autorisierte Personen von entfernten Rechnern mit dem MC-CmdHandler auf die Dateisysteme zugreifen und sie manipulieren. Zugriffsversuche werden nur dann akzeptiert, wenn das (verschlüsselte) Passwort, das die Anforderung begleitet, mit dem Passwort des MC-CmdHandlers übereinstimmt.

Für den MC-CmdHandler gelten dieselben Berechtigungen wie für die Systembenutzerkennung unter der der MC-CmdHandler gestartet wurde. Dies ist von Bedeutung, wenn der MC-CmdHandler auf Dateien zugreift oder Skripts ausführt. Daher ist es notwendig, den MC-CmdHandler jeweils unter der selben Kennung zu starten wie die zu administrierend(en) Komponent(en).

Hinweise zur Benutzung des MC-CmdHandlers

Soll ein MC-CmdHandler zur Administration eines Proxys verwendet werden, sind folgende Punkte zu beachten:

- Bei der Aufnahme des Proxys in die Verwaltungsdaten der Management Console muss das Passwort des verwendeten MC-CmdHandlers mit dem Administrations-Passwort des aufzunehmenden Proxys übereinstimmen.
- Beim Ändern des Administrations-Passworts für einen Proxy wird das Passwort des MC-CmdHandlers, der für die Administration des betreffenden Proxys verwendet wird, ebenfalls geändert.

  Falls also mehrere Proxys auf einem Rechner installiert sind, sollte aus diesem Grund für jeden Proxy ein eigener MC-CmdHandler verwendet werden.

6.10.2 MC-CmdHandler verwalten

Sie finden die Skripts zum Starten, Prüfen und Beenden des MC-CmdHandlers im Home-Verzeichnis des Proxy-Containers unter `shsc` bzw. im Installationsverzeichniss des MC-CmdHandlers unter `shsc`, wenn der MC-CmdHandler separat installiert wurde.

6.10.2.1 MC-CmdHandler starten

MC-CmdHandler auf Unix-/Linux-Systemen starten

Sie starten den MC-CmdHandler mit folgendem Skript im Home-Verzeichnis des Proxy-Containers bzw. im Installationsverzeichnis des MC-CmdHandlers:

- `shsc/startmccmdhandler.sh`

Soll der MC-CmdHandler beim nächsten Abmelden nicht automatisch beendet werden, müssen Sie ihn entweder als Dienst starten (siehe Abschnitt „MC-CmdHandler als Dienst konfigurieren“ auf Seite 253) oder ihn mit folgendem Kommando starten:

```
nohup shsc/startmccmdhandler.sh &
```

MC-CmdHandler auf Windows-Systemen starten

Sie starten den MC-CmdHandler mit folgendem Skript im Home-Verzeichnis des Proxy-Containers bzw. im Installationsverzeichnis des MC-CmdHandlers:

- `shsc\startmccmdhandler.cmd`

Liegt der MC-CmdHandler im Home-Verzeichnis des Proxy-Containers, können Sie ihn auch über die Programmgruppe starten. Wählen Sie dazu:

Start - Programme - FUJITSU Software BeanConnect V3.0B00 - Proxy <container> - MC-CmdHandler - MC-CmdHandler Startup

6.10.2.2 MC-CmdHandler beenden

MC-CmdHandler auf Unix-/Linux-Systemen beenden

Sie beenden den MC-CmdHandler mit folgendem Skript im Home-Verzeichnis des Proxy-Containers bzw. im Installationsverzeichnis des MC-CmdHandlers:

- `shsc/shutmccmdhandler.sh`

MC-CmdHandler auf Windows-Systemen beenden

Sie beenden den MC-CmdHandler mit folgendem Skript im Home-Verzeichnis des Proxy-Containers bzw. im Installationsverzeichnis des MC-CmdHandlers:

- `shsc\shutmccmdhandler.cmd`

Liegt der MC-CmdHandler im Home-Verzeichnis des Proxy-Containers, können Sie ihn auch über die Programmgruppe beenden. Wählen Sie dazu:

Start - Programme - FUJITSU Software BeanConnect V3.0B00 - Proxy <container> - MC-CmdHandler - MC-CmdHandler Shutdown

6.10.2.3 MC-CmdHandler als Dienst konfigurieren

MC-CmdHandler als Dienst auf Unix-/Linux-Systemen konfigurieren

Wenn der MC-CmdHandler als Dienst gestartet werden soll, dann muss dieser konfiguriert werden. Dazu tragen Sie in die Konfigurationsdatei `/etc/init.d/bcmccmdhandler.dat` für jeden zu startenden Dienst eine Zeile ein. Diese Zeile enthält die Kennung, unter der der Dienst gestartet werden soll, sowie das Verzeichnis, unter der der MC-CmdHandler installiert wurde, z.B.:

- `proxyuser /home2/proxyuser/BCCONT`

Sie starten/beenden den MC-CmdHandler als Dienst, indem Sie folgendes Skript aufrufen:

- `/etc/init.d/bcmccmdhandler.sh start | stop`

  Alternativ können Sie das Skript auch mit den Optionen `restart` oder `reload` aufrufen. `restart` und `reload` sind identisch und beinhalten jeweils `stop` und `start`.

Sie entfernen einen Dienst wieder aus dem Unix-/Linux-System, indem Sie die entsprechende Zeile in obiger Konfigurationsdatei löschen.

Für das Eintragen und Löschen des Dienstes, sowie für den Aufruf des skripts `/etc/init.d/bcmccmdhandler.sh`, benötigen Sie Systemverwalterrechte. Weitere Hinweise zum Löschen eines Dienstes finden Sie in Abschnitt „BeanConnect Tools deinstallieren“ auf Seite 90.

MC-CmdHandler als Dienst auf Windows-Systemen konfigurieren

Wenn der MC-CmdHandler ohne BeanConnect Container installiert wird, dann wird der MC-Cmdhandler schon beim Installieren als Dienst eingetragen unter dem Namen `BeanConnect MCCmdhandler <portnummer>` und dem Autostart-Typ `Manuell`.

Wenn der MC-CmdHandler zusammen mit dem BeanConnect Container installiert wird, dann müssen Sie den MC-CmdHandler anschließend explizit als Dienst eintragen. Dazu steht das Skript `shsc/MCCmdHandler_InstallSrv.cmd` im Home-Verzeichnis des Containers zur Verfügung. Rufen Sie dieses Skript unter Administrationsberechtigung auf:

- `<Proxy_home>/shsc/MCCmdHandler_InstallSrv.cmd`

Dieses Skript trägt Ihnen den Dienst mit dem Autostart-Typ `Manuell` ein.

Will man über den MC-CmdHandler einen Proxy Container administrieren, der auf einem entfernten Rechner installiert ist, so kann es unter Umständen erforderlich sein, den Dienst unter dem entsprechenden Benutzerkonto laufen zu lassen und nicht unter dem Systemkonto (Standardwert bei der Installation des MC-CmdHandlers). Die Benutzerkonto-Einstellung ändern Sie über die Windows-Systemsteuerung (Systemsteuerung/Verwaltung/Dienste).

Mit dem Skript `shsc/MCCmdHandler_UnInstSrv.cmd` (Aufruf unter Administrationsberechtigung) können Sie den Dienst wieder entfernen.

6.11 Management Console als JMX-Client konfigurieren

Die Management Console enthält einen JMX-Client. Damit kann die Management Console aktuelle statistische und administrative Daten des laufenden Resource Adapters lesen und bestimmte Attributwerte auch ändern, siehe Abschnitt „Überwachen des Resource Adapters mit der Management Console“ auf Seite 296.

Die Daten werden mit Hilfe von MBeans bereitgestellt, daher wird ein JMX-Client auch MBean-Client genannt.

Als JMX-Client kann die Management Console auf alle MBeans der betreffenden Application Server Instanz zugreifen. In diesem Kapitel werden jedoch nur die MBeans beschrieben, die den Resource Adapter betreffen.

6.11.1 Definierte MBeans des Resource Adapters

Der BeanConnect Resource Adapter bietet mehrere Typen von MBeans an. Für jede MBean werden folgende Arten von Daten zur Verfügung gestellt:

- Attributes

  Dies sind Werte, z.B. Konfigurationswerte oder Statistikzähler. Von den Attributen der MBeans sind die meisten nur lesbar, einige können über die MBean-Oberfläche auch geändert werden.

- Operationen

  Dies sind Operationen (Methoden), die mit Hilfe von MBeans ausgeführt werden können, wie z.B. das Zurücksetzen von Zählern.

- Notifications

  Dies sind Benachrichtigungen, die bei bestimmten Ereignissen (z.B. das Rücksetzen einer Transaktion) an den MBean-Client geschickt werden. Damit die Benachrichtungen zugeschickt werden, muss der MBean-Client sie explizit abonnieren.

Folgende MBeans stehen zur Verfügung (Beschreibungstext jeweils englisch):

- ResourceAdapter MBean

  MBean Beschreibung:

  Administration interface of the BeanConnect Resource Adapter

  Diese MBean stellt die Konfigurationseinstellung des Resource Adapters dar, wie sie in der Datei `ra.xml` definiert sind. Pro Resource Adapter gibt es eine MBean diesen Typs.

Operationen:

– `checkProxyApplication`
Prüft ob die Proxy-Anwendung vefügbar ist.

– `selectProxyApplication`
Startet eine Neuauswahl der Proxy-Anwendung wenn notwendig.

- ManagedConnectionFactory MBean

  Es gibt drei Ausprägungen des ManagedConnectionFactory MBean:

  – Für nicht-transaktionale OSI TP/LU6.2-Verbindungen das OltpMBean:

    MBean Beschreibung:

    Administration interface of a non-transactional BeanConnect OltpManagedConnectionFactory

  – Für transaktionale OSI TP/LU6.2-Verbindungen das OltpTaMBean:

    MBean Beschreibung:

    Administration interface of a transactional BeanConnect OltpManagedConnectionFactory

  – Für UPIC-Verbindungen das UpicMBean:

    MBean Beschreibung:

    Administration interface of a BeanConnect UpicManagedConnectionFactory

  Diese MBeans stellen die Konfigurationseigenschaften der einzelnen Managed Connection Factories dar (Deployment Daten) und geben Auskunft über die Nutzung der Verbindungen (Statistikdaten). Die drei Ausprägungen dieses MBean unterscheiden sich in den angezeigten Attributen.

  Für jede Managed Connection Factory existiert eine entsprechende MBean.

  Operationen:

  – `cleanupPool`
  ManagedConnections aus dem Pool entfernen

  – `resetStatisticValues`
  Statistik zurücksetzen

- Inbound MBean

  MBean Beschreibung:

  Statistic data of inbound connections

  Diese MBean zeigt die statistischen Daten aller Inbound-Aktivitäten an, die keinem Message Endpoint zugeordnet werden können. Es gibt nur eine solche MBean.

  Operationen:

  – `resetStatisticValues`
    Statistik zurücksetzen

- MessageEndpoint MBean

  MBean Beschreibung:

  Administration interface of a BeanConnect Message Endpoint

  Diese MBeans stellen die Konfigurationseigenschaften der einzelnen Message Endpoints dar (Deployment Daten) und geben Auskunft über die Nutzung der Message Endpoints (Statistikdaten). Für jeden Message Endpoint existiert eine entsprechende MBean.

  Operationen:

  – `resetStatisticValues`
    Statistik zurücksetzen

- Logging MBean

  MBean Beschreibung:

  Administration interface of BeanConnect Logging

  Diese MBean stellt die Log4j-Konfigurationseinstellungen dar. Es werden alle Log4j-Logger und ihr LogLevel angezeigt. Es gibt nur eine solche MBean.

  Operationen:

  – `getLogLevel`
    LogLevel eines Log4j Loggers anzeigen

  – `setLogLevel`
    LogLevel eines Log4j Loggers ändern

Weitere Einzelheiten zu den von den MBeans gelieferten Daten und den Möglichkeiten, die die Administration bietet, finden Sie in Abschnitt „Überwachen des Resource Adapters mit der Management Console“ auf Seite 296.

6.11.2 JMX-Client in der Management Console einrichten

Der JMX-Client benötigt für die Kommunikation mit Oracle WebLogic Server folgende zusätzliche Java Bibliotheken:

entweder `wljmxclient.jar` und `wlclient.jar`

oder `wlfullclient.jar`.

Wenn die Management Console nicht auf dem selben Rechner wie der Application Server läuft, dann müssen Sie diese Bibliotheken von der offiziellen Download-Seite von Oracle herunterladen und bereitstellen.

In jedem Fall müssen diese zusätzlichen Java Bibliotheken bekannt gemacht werden. Dazu muss der Classpath der Management Console erweitert werden. Die Skripts `mc.cmd` (Windows-Systeme) und `mc.sh` (Solaris-, Linux-Systeme) zum Starten der Management Console enthalten unter `Enable WebLogic JMX client` bereits die zugehörigen Anweisungen für die Bibliothek `wlfullclient.jar`. Um die Anweisungen zu nutzen, müssen Sie die Skripts mit der Option `-weblogicjmx` starten. Andernfalls werden die Anweisungen unter `Enable WebLogic JMX client` nicht durchlaufen.

Auf Windows-Systemen finden Sie das Skript `mc.cmd` unter `<MC_home>/bin`, dabei ist `<MC_home>` das Installationsverzeichnis der Management Console.

Auf Solaris-/Linux-Systemen finden Sie das Skript `mc.sh` im Verzeichnis `Console/bin` des installierten BeanConnect. Dieses Skript `mc.sh` müssen Sie für den JMX-Client anpassen. Gestartet wird die Management Console jedoch mit dem Skript `startconsole.sh`, das im Installationsverzeichnis der Management Console liegt.

6.11.2.1 Einrichten eines JMX-Clients

Ein JMX-Client ist normalerweise einem Resource Adapter zugeordnet. Sie können aber auch einen freien "stand-alone" JMX-Client einrichten, siehe Abschnitt „Freie JMX-Clients einrichten“ auf Seite 260.

JMX-Client für Resource Adapter einrichten

Sie richten einen JMX-Client für einen Resource Adapter ein, indem Sie im Kontextmenü des Resource Adapters den Befehl **Define MBean Client** auswählen und im Dialogfeld **MBean Client Properties** die Eigenschaften des MBean-Clients festlegen.

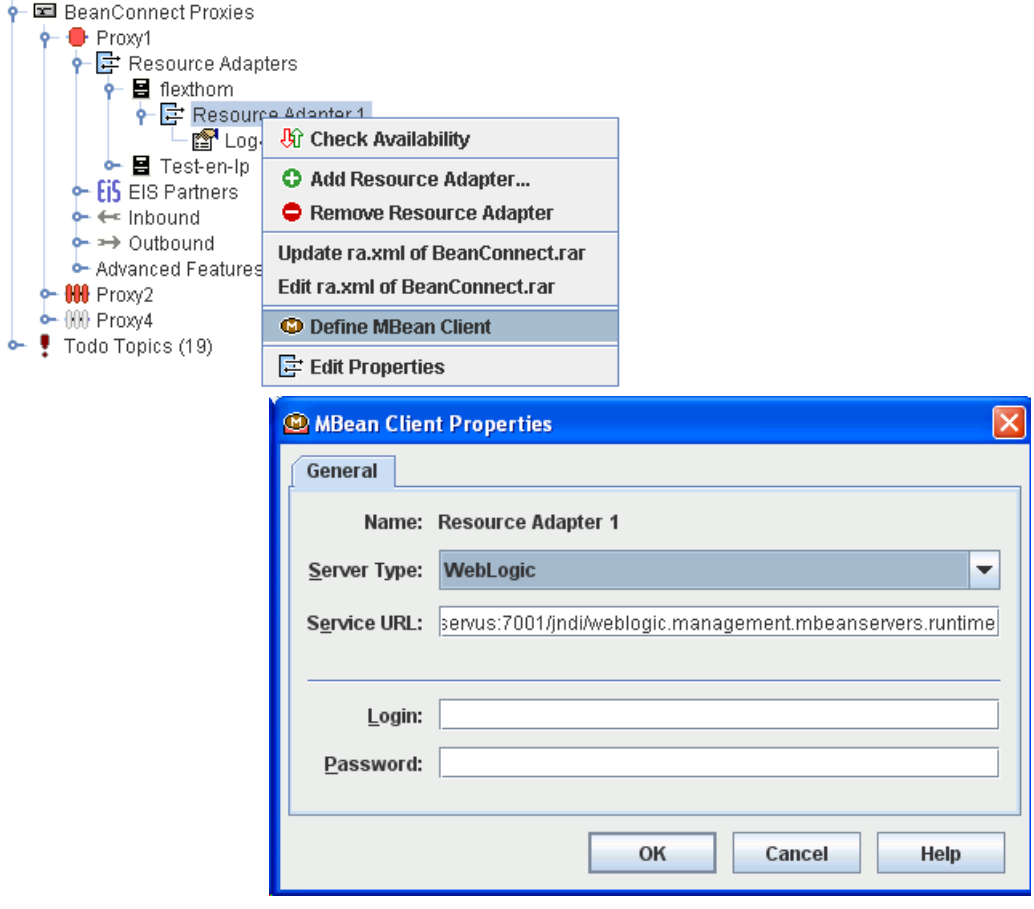


Bild 44: JMX-Client für Resource Adapter einrichten

Name

Name des Resource Adapters, nicht veränderbar.

Server Type

Wählen Sie den Typ des Application Servers aus.

Server URL

gibt die URL an, die zum Aufbau der Verbindung zum JMX-Server verwendet werden soll. Das Format der URL hängt vom Typ des Application Servers ab. Die Management Console schlägt eine Standard-URL vor, die Sie ggf. modifizieren müssen. Die Standard-URL hat folgendes Format:

```
service:jmx:iiop://<server-host>:7001/jndi/
weblogic.management.mbeanservers.runtime
```
(Oracle WebLogic Server)

`<server-host>` ist der Name des JMX-Servers, dahinter steht jeweils die Portnummer des Servers. Die Management Console trägt hier den Namen des Rechners ein, auf dem der Resource Adapter läuft, und schlägt eine Standard-Portnummer für den JMX-Server vor. Sie müssen die Portnummer eventuell anpassen.

Login

Kennung auf dem Application Server, die für das Anmelden benötigt wird. Die Kennung besitzt in der Regel Administrationsberechtigung.

Password

Passwort für die angegebene Kennung.

Der neu eingerichtete MBean-Client wird durch einen eigenen Knoten unterhalb des Resource Adapters symbolisiert.

Freie JMX-Clients einrichten

Ein freier JMX-Client ist ein Client, der keinem Resource Adapter zugeordnet ist. Sie richten einen derartigen Client ein, indem Sie im Menü **File** den Befehl **Add MBean Client** auswählen. Im Dialogfeld **MBean Client Properties** geben Sie die Eigenschaften ein. Dabei gehen Sie ähnlich vor wie bei einem Client mit festem Resource Adapter mit dem Unterschied, dass Sie den Namen des JMX-Client vergeben müssen und dass der Name des JMX-Client-Rechners nicht vorbelegt ist. Die auf diese Weise definierten MBean-Clients werden auf oberster Ebene unter dem Knoten **MBean Clients** aufgelistet. Dieser Knoten existiert nur, wenn Sie mindestens einen freien MBean-Client eingerichtet haben.

6.11.2.2 Verbindung zum JMX-Server auf- und abbauen

Sie müssen die Verbindung zwischen Management Console und JMX-Server explizit auf- und abbauen.

Verbindung zum JMX-Server aufbauen

Wählen Sie im Kontextmenü der MBean den Befehl **Connect to MBean Server**. Wenn die Verbindung aufgebaut werden konnte, dann ändert sich das Symbol des MBean-Knotens und unterhalb des MBean-Knotens werden alle verfügbaren MBean-Domänen angezeigt. Falls die Verbindung nicht zustande kommt, wird eine Fehlermeldung ausgegeben, welche auf die Fehlerursache hinweisen kann.

Verbindung zum JMX-Server abbauen

Sie bauen die Verbindung zum JMX-Server ab, indem Sie im Kontextmenü des MBean-Clients den Befehl **Disconnect From MBean Server** auswählen. Mit dem Abbau der Verbindung werden die Inhalte der MBean-Domänen-Knoten aus dem Navigationsbaum entfernt.

6.11.2.3 Entfernen eines JMX-Clients

Sie entfernen einen JMX-Client, indem Sie im Kontextmenü des MBean-Clients den Befehl **Remove MBean Client** auswählen und dies in der Abfrage bestätigen. Damit entfernt die Management Console den Knoten aus dem Navigationsbaum, schließt alle eventuell geöffneten Fenster und löscht die Konfigurationsdaten aus ihren Verwaltungsdateien.

7 Konfiguration im EIS Partner anpassen

Für eine Kommunikation zwischen dem Application Server und dem EIS Partner müssen neben der Konfiguration von Application Server, Resource Adapter und Proxy zusätzliche Konfigurationsaufgaben durchgeführt werden. Diese betreffen das EIS sowie die Plattform, auf der das EIS läuft.

Dieses Kapitel enthält Informationen zu folgenden Themen:

- Konfiguration im EIS Partner vom Typ openUTM anpassen
- Konfiguration im EIS Partner vom Typ CICS anpassen

Weitere Einzelheiten zur Konfiguration von openUTM-Anwendungen finden Sie in der Dokumentation zu openUTM.

7.1 Konfiguration im EIS Partner vom Typ openUTM anpassen

Ein EIS Partner vom Typ openUTM ist in der Regel eine openUTM-Anwendung, die über Outbound oder Inbound mit dem Application Server kommuniziert. Es ist jedoch auch möglich, dass eine Anwendung aus dem Umfeld von openUTM wie z.B. eine UPIC-Anwendung Inbound auf den Application Server zugreift.

7.1.1 Verbindungen zwischen openUTM und BeanConnect definieren

openUTM-Partner können über folgende Protokolle mit BeanConnect verbunden sein:

- OSI TP
- UPIC
- Socket oder RFC1006

Die folgenden Abschnitte zeigen, welche Parameter dafür im EIS Partner zu konfigurieren sind.

Für die Verbindung zu einer openUTM-Partneranwendung auf einem BS2000-System müssen Sie ggf. noch BCMAP-Einträge erstellen.

7.1.1.1 OSI TP Verbindung zwischen BeanConnect und openUTM definieren

Eine OSI TP Verbindung kann sowohl für Outbound- als auch für Inbound-Kommunikation verwendet werden. Eine OSI TP Verbindung ermöglicht sowohl transaktionale als auch nicht-transaktionale Kommunikation.

Eine OSI TP Verbindung zwischen BeanConnect und openUTM erfordert eine `OSI-LPAP-` und eine `OSI-CON-`Anweisung innerhalb der openUTM Generierung des EIS Partners (KDCDEF).

Die Generierung auf der Seite des Proxys wird durchgeführt, wie im Abschnitt „EIS Partner konfigurieren“ auf Seite 212 beschrieben. Für die EIS Partner-Seite generiert die BeanConnect Management Console diese Anweisungen in einer Textdatei im Verzeichnis `<MC_home>/genfiles`.

Der Name der generierten Input-Datei lautet bei einem einzelnen Proxy:

`ProxyID.<p-id>.EisPartnerID.<e-id>.UTM.txt`

Der Name der generierten Input-Datei lautet bei einem Proxy Cluster:

`ClusterID.<c-id>.EisPartnerID.<e-id>.UTM.txt`

`<p-id>`, `<e-id>` und `<c-id>` bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

BeanConnect überträgt diese Konfigurationsdatei nicht automatisch auf die EIS-Plattform. Die generierte Input-Datei muss mit Hilfe üblicher Dateiübertragungs-Methoden auf den Rechner des EIS Partners übertragen werden. Anschließend kann der openUTM-Administrator die Datei für die Konfiguration verwenden.

7.1.1.2 UPIC-Verbindung für Outbound-Kommunikation zwischen openUTM-Partner und BeanConnect definieren

Aus Sicht des openUTM-Partners handelt es sich bei dem BeanConnect Resource Adapter um einen UPIC-Client.

Eine UPIC-Verbindung von BeanConnect zum openUTM-Partner erfordert eine `TPOOL`-Anweisung innerhalb der KDCDEF-Generierung der openUTM-Partneranwendung. Die Management Console generiert diese Anweisung nicht.

Die Konfiguration ermöglicht nicht-transaktionale UPIC-Outbound-Kommunikation.

Die openUTM-Verbindung wird wie folgt konfiguriert:

```
BCAMAPPL UTMSERV
   ,T-PROT=RFC1006
  [,LISTENER-PORT = 11111]   // nur für Unix-/Linux-/Windows-Systeme erforderlich
  [,TSEL-FORMAT=T|A|E]       // nur für Unix-/Linux-/Windows-Systeme erforderlich

TPOOL NUMBER=99
   ,PTYPE=UPIC-R
   ,LTERM=UPIC#R
   ,BCAMAPPL=UTMSERV
   ,PRONAM=*ANY
   ,Connect-Mode=MULTI
```

Beachten Sie Folgendes:

- `TSEL-FORMAT` muss mit der Konfigurations-Property `connectionURL`, dem Parameter `TSEL` und dem Attribut `rt` übereinstimmen (siehe „connectionURL“ auf Seite 112).
- `UTMSERV` und die Portnummer müssen mit den Werten übereinstimmen, die in den Parametern `remote` und `port` der Konfigurations-Property `connectionURL` oder im Programm angegeben sind.

Beispiel 11 Beispiel einer URL

Die oben konfigurierte openUTM-Partneranwendung kann mit der Konfigurations-Property `connectionURL` wie folgt adressiert werden:

```
<config-property name="connectionURL"
                 value="upic://host:11111/UTMSERV?rt=t"/>
```

(für Unix-/Linux-/Windows-Systeme)

7.1.1.3 Socket-Verbindung zwischen openUTM-Partner und BeanConnect definieren

Eine Socket- oder RFC1006-Verbindung von einem openUTM-Partner zu BeanConnect erfordert ein `PTERM/LTERM` Anweisungspaar innerhalb der KDCDEF-Generierung der openUTM-Partneranwendung, entweder mit `PTYPE=SOCKET` für das openUTM-Socket-Protokoll oder mit `PTYPE=APPLI` für das RFC1006-Protokoll. Die Management Console generiert diese Anweisungen nicht.

Die Konfiguration ermöglicht nicht-transaktionale asynchrone Inbound-Kommunikation.

7.1.1.4 BCMAP-Eintrag definieren für openUTM-Partner auf BS2000-Systemen

Auf dem BS2000-Partnersystem ist in BCAM ein zusätzlicher Konfigurationsschritt erforderlich, wenn für die Kommunikation nicht die Portnummer 102 verwendet wird.

Um den `OSI-CON`-, `PTERM`- oder `TPOOL`-Anweisungen der openUTM-Konfiguration einen anderen Port als den Standard-Port `102` zuzuweisen, müssen Sie einen `BCMAP`-Eintrag definieren.

Bei der Konfiguration eines BS2000-EIS Partners mit der BeanConnect Management Console wird im Verzeichnis `<MC_home>/genfiles` eineTextdatei mit dem passenden BCMAP-Eintrag erzeugt.

Der Name der generierten Input-Datei lautet bei einem einzelnen Proxy:

`ProxyID.<p-id>.EisPartnerID.<e-id>.BCMAP.txt`

Der Name der generierten Input-Datei lautet bei einem Proxy Cluster:

`ClusterID.<c-id>.EisPartnerID.<e-id>.BCMAP.txt`

`<p-id>`, `<e-id>` und `<c-id>` bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

BeanConnect überträgt diese Konfigurations-Datei nicht automatisch auf die EIS-Plattform. Die generierte Input-Datei muss mit Hilfe üblicher Dateiübertragungs-Methoden auf den Rechner des EIS Partners übertragen werden. Anschließend kann der BCAM-Administrator sie zur Durchführung von Konfigurationsaufgaben verwenden.

Weitere Einzelheiten zur Konfiguration von BCMAP-Einträgen finden Sie in der BCAM-Dokumentation.

7.1.1.5 Mapping von langen Rechnernamen bei openUTM-Partnern auf offenen Plattformen

openUTM-Partner auf Unix-, Linux- und Windows-Systemen verwenden das Hostname Mapping, um lange reale Rechnernamen (> 8 Zeichen) auf kurze Namen abzubilden. Das Mapping wird über Einträge in der Hostname-Datei der UTM-Anwendung durchgeführt.

Wenn der Proxy Hostname länger als 8 Zeichen ist, muss bei dem openUTM-Partner in der Hostname-Datei ein Mapping von einem kurzen (Mapped Hostname) auf den realen Rechnernamen des Proxys definiert werden.

Bei der Konfiguration eines Unix-, Linux- oder Windows-Windows-EIS Partners mit der BeanConnect Management Console wird im Verzeichnis `<MC_home>/genfiles` eine Textdatei mit dem passenden Eintrag erzeugt.

Der Name der generierten Hostname-Datei lautet bei einem einzelnen Proxy:

`ProxyID.<p-id>.EisPartnerID.<e-id>.HOSTNAME.txt`

Der Name der generierten Hostname-Datei lautet bei einem Proxy Cluster:

`ClusterID.<c-id>.EisPartnerID.<e-id>.HOSTNAME.txt`

`<p-id>`, `<e-id>` und `<c-id>` bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

BeanConnect überträgt diese Hostname-Datei nicht automatisch auf die EIS-Plattform. Die generierte Datei muss mit Hilfe üblicher Dateiübertragungs-Methoden auf den Rechner des openUTM-Partners übertragen werden. Existiert dort schon eine Hostname-Datei, müssen die Einträge für den Proxy/Proxy-Cluster in diese eingefügt werden. Existiert dort noch keine Hostname-Datei, muss die übertragene Datei in der UTM-Anwendung aktiviert werden. Einzelheiten zur Konfiguration und Aktivierung von Hostname-Dateien in openUTM finden Sie im openUTM-Handbuch „Anwendungen generieren“ unter „Mapped Hostnamen verwenden ...“.

7.1.1.6 Mapping von langen Rechnernamen bei UTM-Partnern auf BS2000-Systemen

Für das Mapping von langen Rechnernamen gibt es im BS2000-System keinen UTM-eigenen Mechanismus. Das Mapping muss mit BS2000-Mitteln, z.B. einem Eintrag in der FQDN-Datei erfolgen. Die Angaben von realem Rechnernamen und Mapped Hostnamen in BeanConnect müssen der Konfiguration im BS2000-System entsprechen.

7.1.2 Verbindungen zwischen anderen EIS Partnern und BeanConnect definieren

Jede UPIC-Anwendung und jede Anwendung, die Transport-Level-Protokolle wie RFC1006 oder das openUTM-Socket-Protokoll verwendet, kann als EIS Partner für nicht-transaktionale Inbound-Kommunikation verwendet werden, wobei eine UPIC-Anwendung nur Dialog-Kommunikation verwenden kann.

Weitere Einzelheiten zur Kommunikation zwischen UPIC-Anwendungen und Anwendungen, die Transport-Level-Protokolle wie RFC1006 oder das openUTM-Socket-Protokoll verwenden und BeanConnect, finden Sie in der Dokumentation zu openUTM.

7.2 Konfiguration im EIS Partner vom Typ CICS anpassen

Ein EIS Partner vom Typ CICS ist eine CICS-Anwendung, die auf einem IBM Mainframe abläuft.

7.2.1 CICS-Partner konfigurieren

Folgende Konfigurationsaufgaben müssen in CICS durchgeführt werden:

- Definition einer Connection, mit der die Verbindungsparameter festgelegt werden. Diese Definition enthält den Namen der Logical Unit (Parameter `NETNAME`) der CICS-Partneranwendung.
- Definition einer Session, mit der die Sitzungsparameter festgelegt werden. Diese Definition enthält den Namen der Connection (Parameter `CONNECTION`) und den Namen des Session Mode (Parameter `MODE`). Über den Mode sind dann die Eigenschaften einer Session festgelegt.

Die BeanConnect Management Console generiert eine Textdatei, die die Definitionen von `CONNECTION` und `SESSION` enthält (siehe Abschnitt „EIS Partner vom Typ CICS konfigurieren“ auf Seite 224).

Der Name der generierten Input-Datei lautet bei einem einzelnen Proxy:

`ProxyID.<p-id>.EisPartnerID.<e-id>.CICS.txt`

Der Name der generierten Input-Datei lautet bei einem Proxy im Cluster:

`ClusterID.<c-id>.ProxyID.<p-id>.EisPartnerID.<e-id>.CICS.txt`

`<p-id>`, `<e-id>` und `<c-id>` bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

BeanConnect überträgt die Datei nicht automatisch auf die EIS-Plattform. Sie muss mit Hilfe üblicher Datei-Transfermethoden auf den Rechner des EIS Partners übertragen werden. Der CICS-Administrator kann sie dann für die Konfiguration verwenden.

7.2.2 VTAM auf IBM Mainframe konfigurieren

Für CICS-Partner, die auf einem IBM Mainframe (z/OS-System) laufen, muss die Komponente VTAM (Virtual Telecommunications Access Method) konfiguriert werden. VTAM ist ein IBM-Produkt, das auf dem z/OS-Mainframe läuft und den Zugriff auf eine auf diesem z/OS-Mainframe vorhandene CICS-Region ermöglicht. Proxy-Anfragen werden zuerst an VTAM und dann an CICS weitergeleitet. Daher sind einige VTAM-Definitionen erforderlich.

VTAM-Definitionen werden unter Verwendung von Makros codiert. Die Management Console generiert während des Konfigurationslaufs eine Input-Datei für die VTAM-Konfiguration (siehe Abschnitt „EIS Partner vom Typ CICS konfigurieren“ auf Seite 224). Diese Input-Datei enthält Definitionen zur Physical Unit (PU) in VTAM.

Der Name der generierten Input-Datei lautet bei einem einzelnen Proxy:

`ProxyID.<p-id>.EisPartnerID.<e-id>.VTAM.txt`

Der Name der generierten Input-Datei lautet bei einem Proxy im Cluster:

`ClusterID.<c-id>.ProxyID.<p-id>.EisPartnerID.<e-id>.VTAM.txt`

`<p-id>`, `<e-id>` und `<c-id>` bezeichnen die von der Management Console vergebenen IDs für Proxy, EIS Partner und Proxy Cluster.

BeanConnect überträgt die VTAM-Konfigurationsdatei nicht automatisch auf die EIS-Plattform. Sie muss mit Hilfe üblicher Datei-Transfermethoden auf den Rechner des EIS Partners übertragen werden. Anschließend kann der VTAM-Administrator die Datei für die Konfiguration verwenden. Bitte beachten Sie, dass der VTAM-Administrator einige VTAM-Definitionen an die Bedürfnisse der Partneranwendung anpassen muss. BeanConnect kann während der Datei-Generierung die Eindeutigkeit der Definitionen nicht gewährleisten, da BeanConnect keinen Zugriff auf die vollständigen VTAM-Definitionen hat. Aus diesem Grund enthält die Konfigurationsdatei einige Fragezeichen, die durch die passenden Werte ersetzt werden müssen.

8 BeanConnect administrieren

Dieses Kapitel beschreibt die Administrationsaufgaben beim Betrieb von BeanConnect.

Alle notwendigen Operationen können Sie mit der BeanConnect Management Console ausführen. Dazu gehören:

- Proxy-Container und Proxy-Komponenten starten und beenden
- Verfügbarkeit des Proxys prüfen

Zum Starten und Beenden des Proxy-Containers sind auch Skripts verfügbar. Diese stehen auf Windows-Systemen in der Programmgruppe des Proxy-Containers zur Verfügung.

Darüber hinaus bietet die Management Console ein Command Line Interface (MC-CLI) zur Automatisierung von Administrationsfunktionen per Skript. Detaillierte Information dazu finden Sie in Kapitel „Command Line Interface der BeanConnect Management Console (MC-CLI)“ auf Seite 309.

Die Verwaltung von Proxys, die auf entfernten Systemen laufen, ist nur mit der Management Console möglich.

Dieses Kapitel enthält Informationen zu folgenden Themen:

8.1 BeanConnect Proxy mit der Management Console administrieren

Die Management Console kann mehrere installierte Proxys administrieren. Diese Proxys können entweder als lokale Proxys auf demselben Rechner wie die Management Console oder als entfernte Proxys auf einem anderen Rechner installiert sein.

Proxys können mit der Management Console administriert werden, wenn eine der folgenden Bedingungen erfüllt ist:

- Aus Sicht der Management Console handelt es sich bei dem Proxy um einen lokalen Proxy, der unter derselben Benutzerkennung läuft wie die Management Console.
- Bei dem Proxy handelt es sich um einen (ggf. entfernten) Proxy, dessen zugehöriger MC-CmdHandler verfügbar ist und über die Management Console erreicht werden kann.

Die Management Console stellt die notwendigen Verwaltungsfunktionen im Kontextmenü der Proxy-Knoten bereit.

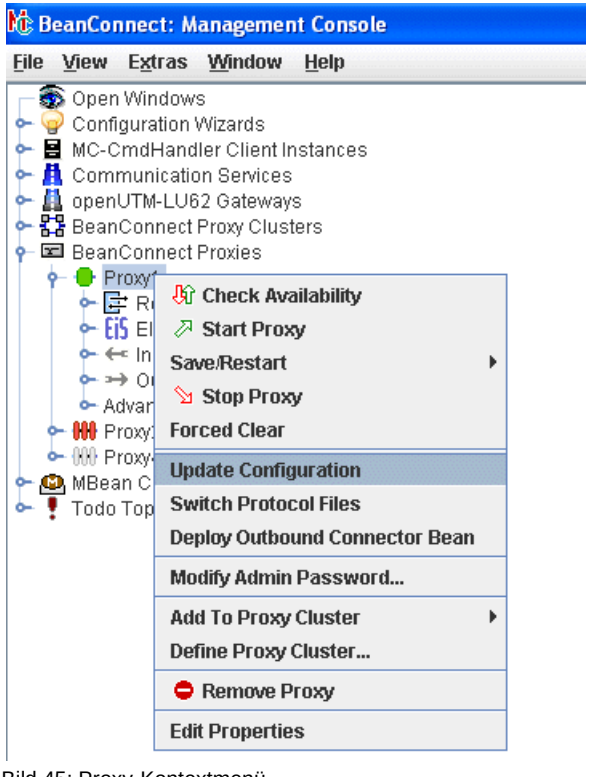


Bild 45: Proxy-Kontextmenü

8.1.1 Starten eines Proxy

Der Proxy-Container bzw. die Proxy-Komponenten müssen konfiguriert sein, um sie erfolgreich starten zu können.

Einen Proxy starten Sie, indem Sie im Navigationsbaum das Kontextmenü des Proxy-Knotens öffnen und den Menüpunkt **Start Proxy** wählen.

Wenn der Proxy nur für Partner vom Typ openUTM konfiguriert ist, dann wird er sofort gestartet.

Wenn der Proxy (auch) für Partner vom Typ CICS konfiguriert ist, dann wird ein Auswahldialogfeld geöffnet, in dem Sie die Komponenten des Proxys wählen, die Sie starten möchten:

- Proxy-Container
- openUTM-LU62 Gateway
- Communication Service

Vor dem Start von openUTM-LU62 und Communication Service müssen zunächst die zugehörigen MC-CmdHandler gestartet werden.

Beim Starten des openUTM-LU62 Gateways (CICS) können Sie zusätzlich die Option **Cold Start** angeben. Bei einem Kaltstart gehen alle Restart-Informationen verloren.

Die Management Console prüft vor dem Starten zuerst, ob der Proxy-Container bzw. die Proxy-Komponente bereits verfügbar ist.

Das openUTM-LU62 Gateway sowie der Communication Service können erst gestartet werden, wenn für den Proxy mindestens ein EIS Partner vom Typ CICS definiert wurde. Nähere Informationen hierzu erhalten Sie im Abschnitt „BeanConnect Proxy konfigurieren“ auf Seite 185.

Für CICS-Partner:
Der Administrator muss Mitglied der vordefinierten SNA-Administratorengruppe `sna` sein, um den Communication Service administrieren zu können. Die Benutzergruppe `sna` ist nach der Installation des Communication Service vorhanden. Bei dem Administrator handelt es sich um den Benutzer, unter dessen Benutzerkennung der MC-CmdHandler gestartet wurde.

Die Management Console gibt entsprechende Meldungen in einem Aktions-Dialogfeld zur Überwachung der Aktionen und Ergebnisse aus. Sie können

- prüfen, welche Aktionen bereits ausgeführt wurden, sowie die Ergebnisse der abgeschlossenen Aktionen überwachen,
- im Fehlerfall detaillierte Informationen zu den Ergebnissen anzeigen lassen,

- gegebenenfalls die gesamte Aktion abbrechen. Dabei werden nur die noch nicht ausgeführten Teilaktionen nicht mehr ausgeführt. Teilaktionen, die schon ausgeführt wurden, werden bei einem Abbruch nicht mehr rückgängig gemacht.

Wenn eine Verfügbarkeitsprüfung durchgeführt wurde, werden die Statussymbole vor dem betreffenden Knoten im Navigationsbaum je nach Ergebnis farbig dargestellt (grün, wenn der Proxy-Container oder bei CICS-Partnern die Proxy-Komponente läuft, oder rot, wenn sie nicht läuft). Bei einem openUTM-Proxy ist das Symbol einteilig und beschreibt den Status des Proxy-Containers. Bei CICS-Proxys besteht das Symbol aus 3 Teilen:

Die Einzelsymbole stehen für folgende Komponenten:

Links	Proxy-Container
In der Mitte	openUTM-LU62 Gateway
Rechts	Communication Service

Eine Verfügbarkeitsprüfung wird automatisch in vordefinierten Zeitintervallen durchgeführt, sie kann aber auch manuell gestartet werden (siehe Abschnitt „Verfügbarkeit von BeanConnect Proxys überprüfen“ auf Seite 287).

Alle Proxy-Komponenten müssen aktiv (grün) sein, um eine Kommunikation zwischen Application Server und EIS Partner zu ermöglichen.

Bei einem CICS-Partner reicht das nicht immer aus, d.h. es kann vorkommen, dass die Kommunikation trotz grüner Symbole nicht korrekt funktioniert. In diesem Fall müssen Sie zusätzlich die aufgebauten Verbindungen und Sessions überprüfen, siehe Abschnitt „Diagnosedaten des openUTM-LU62 Gateways“ auf Seite 560.

Proxy-Container als Windows-Dienst starten

Auf Windows-Systemen können Sie mit der Management Console auch einen Proxy-Container als Windows-Dienst starten:

1. Wählen Sie den Proxy im Navigationsbereich der Management Console aus.
2. Wählen Sie im Kontextmenü den Eintrag **Edit Properties**, und wählen Sie auf der Registerkarte **General** die Option **Start as Service**.
3. Speichern Sie die Konfiguration mit **Save/Restart** - **Save**. Beim nächsten Start wird der Proxy als Windows-Dienst gestartet. Sollte der Proxy schon laufen, wird diese Änderung nicht schon beim angebotenen Restart wirksam, sondern erst nach dem Beenden und neuerlichem Start des Proxy.

Weitere Einzelheiten finden Sie im Abschnitt „Als Windows-Dienst starten“ auf Seite 279.

8.1.2 Restart eines Proxys

Der Restart eines Proxys umfasst eine Kombination aus Beenden (bei Bedarf) und Starten des Proxys bzw. der einzelnen Proxy-Komponenten. Dies ist nach bestimmten Konfigurationsaktivitäten notwendig (siehe Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249).

Sie stoßen einen Restart des Proxys an, indem Sie im Navigationsbaum das Kontextmenü des Proxy-Knotens öffnen und den Menüpunkt **Save/Restart – Restart Proxy** wählen.

Bei einem Proxy, der für CICS-Partner konfiguriert ist, wird ein Dialogfeld geöffnet, in dem Sie die Proxy-Komponenten auswählen können, für die Sie einen Restart durchführen möchten. Auch hier ist ein Aktions-Dialogfeld zur Prüfung der Aktionen und deren Ergebnisse verfügbar.

Der Proxy-Container wird neu geladen, d.h. die einzelnen Prozesse werden nacheinander beendet und erneut gestartet. Dabei bleibt der Proxy-Container als Ganzes verfügbar.

Bei einem Proxy, der für CICS-Partner konfiguriert ist, werden das openUTM-LU62 Gateway und der Communication Service zuerst beendet und anschließend wieder gestartet.

8.1.3 Beenden eines Proxys

Einen Proxy beenden Sie, indem Sie im Navigationsbaum das Kontextmenü des Proxy-Knotens öffnen und den Menüpunkt **Stop Proxy** wählen.

Bei einem Proxy, der für CICS-Partner konfiguriert ist, können Sie hier, genau wie beim Starten des Proxys, einzelne Proxy-Komponenten auswählen. Die Management Console zeigt ein Aktions-Dialogfeld zur Überprüfung der Aktionen und deren Ergebnisse an.

8.1.4 Besonderheiten beim Cluster-Betrieb

Ein Proxy Cluster lässt sich auf entsprechende Weise administrieren wie ein einzelner Proxy. Die Management Console stellt die notwendigen Verwaltungsfunktionen über das Kontextmenü des Proxy Clusters bereit. Über den Befehl **Show Cluster Proxies** können Sie sich alle Proxys im Cluster anzeigen lassen.

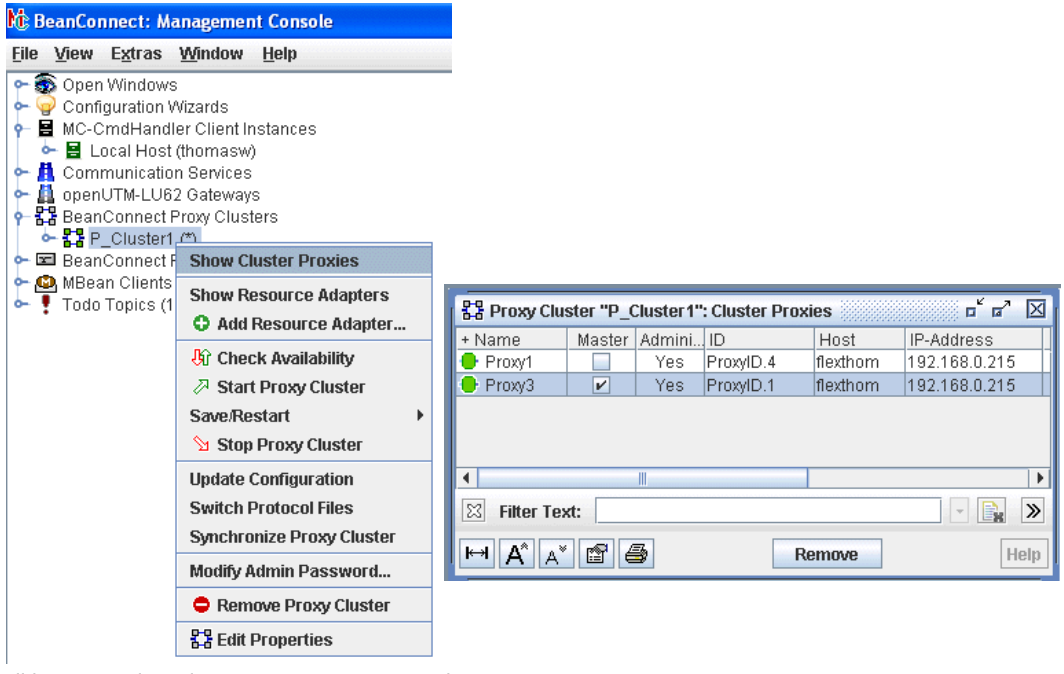


Bild 46: Verwalten eines BeanConnect Proxy Clusters

Das Starten, Beenden und Sichern gilt immer für alle Proxys eines Proxy Clusters.

Wenn Sie einen einzelnen Proxy starten oder beenden möchten, dann markieren Sie den Proxy im Arbeitsbereichsfenster **Cluster Proxies** und wählen den entsprechenden Kontextmenü-Befehl.

Wenn Sie einen Resource Adapter im laufenden Betrieb auf einen anderen Proxy umschalten möchten, dann sollten Sie dazu MBeans verwenden, siehe Abschnitt „Resource Adapter im Cluster auf anderen Proxy umschalten“ auf Seite 306.

Beim Verwalten eines Proxy Clusters gibt es im Vergleich zu einem einzelnen Proxy folgende Unterschiede:

- Wenn mehrere Proxys in einem Cluster vorhanden sind, dann ist immer ein Proxy der Master Proxy. Dieser ist in der Liste **Cluster Proxies** in der Spalte **Master** entsprechend markiert. Dieser Proxy ist der (erste) Ansprechpartner der Management Console beim Holen der Konfigurationsdaten des Clusters. Gibt es Änderungen, sorgt die Management Console dafür, dass diese in allen Proxys nachgezogen werden.
- Ist einer der Proxys zeitweilig nicht administrierbar, dann kann es vorkommen, dass die Daten der Proxys nicht konsistent sind. Für diesen Fall können Sie die Proxys nachträglich mit dem Befehl **Synchronize Proxy Cluster** im Kontextmenü des Proxy Clusters synchronisieren. Dabei übernehmen die anderen Proxys die Daten des Master Proxys.

8.2 BeanConnect Proxy-Container auf Befehlsebene administrieren

BeanConnect stellt einige Skripts und Programme bereit, mit denen Sie Proxy-Container auf Befehlsebene oder auf Windows-Systemen über die Programmgruppe des Proxy-Containers administrieren können.

8.2.1 Proxy-Container starten

Sie können einen lokalen Proxy-Container über ein Skript starten.

Auf Windows-Systemen können Sie zusätzlich die Programmgruppe des Proxy-Containers verwenden oder den Proxy-Container als Windows-Dienst starten.

8.2.1.1 Mit Skript starten

Mit folgenden Skripts können Sie einen lokalen Proxy-Container starten.

Im Home-Verzeichnis des Proxy-Containers steht zum Starten eines lokalen Proxys im Unterverzeichnis `shsc` die Prozedur `startcontainer` zur Verfügung.

Gehen Sie wie folgt vor:

1. Öffnen Sie eine Shell oder ein Eingabefenster für DOS-Befehle.
2. Wechseln Sie in das Home-Verzeichnis des Proxy-Containers.
3. Rufen Sie das Skript wie folgt auf:

   `shsc/startcontainer.sh` (auf Solaris-/Linux-Systemen) oder

   `shsc\startcontainer.cmd` (auf Windows-Systemen)

8.2.1.2 Über die Programmgruppe des Proxy-Containers auf Windows-Systemen starten

Einen lokalen Proxy-Container können Sie auf einem Windows-System auch über die Programmgruppe des Proxy-Containers starten:

1. Öffnen Sie die Programmgruppe
   FUJITSU Software BeanConnect V3.0B00 - Proxy <proxy_cont_name>
2. Wählen Sie den Befehl **Proxy Container Startup - <proxy_cont_name>**.

8.2.1.3 Als Windows-Dienst starten

Wenn BeanConnect auf Windows-Systemen installiert ist, wird der Proxy-Container als Dienst unter dem Namen `BeanConnect30B <proxy_cont_name>` eingerichtet. Der Starttyp für den Dienst lautet standardmäßig `Manuell`. Wenn Sie sicherstellen möchten, dass der Proxy-Container immer verfügbar ist, wählen Sie als Starttyp die Option `Automatisch`.

So starten Sie einen lokalen Proxy-Container als Dienst:

1. Öffnen Sie die Programmgruppe **Start** - **Einstellungen** - **Systemsteuerung** und wählen Sie den Eintrag **Verwaltung** - **Dienste**.
2. Öffnen Sie das Kontextmenü des Dienstes **BeanConnect30B <proxy_cont_name>** und wählen Sie den Befehl **Starten**.

Einen Proxy-Container können Sie auch mit der Management Console als Dienst starten (siehe Abschnitt „Starten eines Proxy“ auf Seite 273).

Wenn der Proxy-Container als Dienst ausgeführt wird, wird seine erste Ausgabe an stdout in die Datei `utmp.out` und seine erste Ausgabe an stderr in die Datei `utmp.err` geschrieben. Bei einem Umschalten der Protokoll-Dateien heißen die Dateien dann: `utmp.err.<Zeitstempel>` bzw. `utmp.out.<Zeitstempel>`. Die Protokoll-Dateien des letzten Anwendungslaufs werden automatisch im Verzeichnis `out-err` gesichert.

Wenn Sie den Proxy-Container als Dienst starten, wird kein DOS-Fenster für die Ausgabe an stdout oder für Fehlermeldungen geöffnet. Sie können jedoch ein Fenster eigens für diesen Zweck öffnen. Der Output wird während des Betriebs ausgegeben und automatisch aktualisiert (siehe Kapitel „Logging, Diagnose und Fehlerbehebung“ auf Seite 517).

8.2.1.4 Nach einem fehlerhaften Abbruch eines Proxy-Container-Laufs starten

Wenn der Proxy-Container nicht gestartet werden kann, z.B. weil der vorherige Lauf fehlerhaft beendet wurde, gehen Sie folgendermaßen vor:

- Auf einem Solaris-/Linux-System wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript `shsc/remove.sh` auf.
- Auf einem Windows-System wählen Sie **Forced Clear** aus der Programmgruppe **FUJITSU Software BeanConnect V3.0B00 - Proxy <proxy_cont_name>**
  oder
  wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript `shsc\remove.cmd` auf.
- Wenn Sie die Management Console im Expertenmodus betreiben (auf Solaris-, Linux- oder Windows-Systemen), dann wählen Sie **Forced Clear** im Kontextmenü des Proxy.

8.2.2 Restart des Proxy-Containers

Ein Restart des Proxy-Containers kann nach bestimmten Konfigurationsaktivitäten erforderlich sein (siehe Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249).

8.2.2.1 Restart mit Skript

Sie können den Restart für einen lokalen Proxy-Container mit einem Skript anstoßen.

Im Unterverzeichnis `shsc` des Home-Verzeichnisses des Proxy-Containers steht für den Restart eines lokalen Proxy-Containers die Prozedur `change` zur Verfügung.

So gehen Sie vor:

1. Öffnen Sie eine Shell oder ein Eingabefenster für DOS-Befehle.
2. Wechseln Sie in das Home-Verzeichnis des Proxy-Containers.
3. Rufen Sie das Skript wie folgt auf:

   `shsc/change.sh` (auf Solaris-/Linux-Systemen) oder

   `shsc\change.cmd` (auf Windows-Systemen)

8.2.2.2 Über die Programmgruppe des Proxy-Containers auf Windows-Systemen neu starten

Den Restart für einen lokalen Proxy-Container können Sie auf einem Windows-System auch über die Programmgruppe des Proxy-Containers anstoßen:

- Öffnen Sie die Programmgruppe
  FUJITSU Software BeanConnect V3.0B00 - Proxy <proxy_cont_name> und wählen Sie den Befehl **Proxy Container Restart**.

8.2.3 Proxy-Container beenden

Einen lokalen Proxy-Container können Sie mit einem Skript oder über die Programmgruppe des Proxy-Containers auf einem Windows-System beenden.

8.2.3.1 Mit einem lokalen Skript beenden

Sie können einen lokalen Proxy-Container mit einem Skript beenden.

Im Unterverzeichnis `shsc` des Home-Verzeichnisses des Proxy-Containers steht für das Stoppen eines lokalen Proxy-Containers die Prozedur `shutcontainer` zur Verfügung.

So gehen Sie vor:

1. Öffnen Sie eine Shell oder ein Eingabefenster für DOS-Befehle.
2. Wechseln Sie in das Home-Verzeichnis des Proxy-Containers
3. Rufen Sie das Skript wie folgt auf:

   `shsc/shutcontainer.sh` (auf Solaris-/Linux-Systemen) oder

   `shsc\shutcontainer.cmd` (auf Windows-Systemen)

8.2.3.2 Über die Programmgruppe des Proxy-Containers auf Windows-Systemen beenden

Einen lokalen Proxy-Container können Sie auf einem Windows-System auch über die Programmgruppe des Proxy-Containers beenden:

- Wählen Sie den Befehl **Proxy Container Shutdown** aus der Programmgruppe **FUJITSU Software BeanConnect V3.0B00 - Proxy <proxy_cont_name>**.

8.2.3.3 Als Windows-Dienst beenden

Wenn ein Proxy-Container als Dienst auf einem Windows-System gestartet wurde, können Sie ihn im Dialogfeld **Dienste** beenden.

1. Öffnen Sie die Programmgruppe **Start** - **Einstellungen** - **Systemsteuerung** und wählen Sie den Eintrag **Verwaltung** - **Dienste**.
2. Öffnen Sie das Kontextmenü des Dienstes **BeanConnect30B <proxy_cont_name>** und wählen Sie den Befehl **Beenden**.

Ein Proxy-Container, der mit dem Starttyp `Automatisch` gestartet wurde, kann nur auf diese Weise beendet werden.

8.3 MC-CmdHandler auf Windows-Systemen als Dienst starten

Wenn der MC-CmdHandler als Dienst auf Windows-Systemen gestartet werden soll, dann muss er zuerst als Dienst konfiguriert werden, siehe „MC-CmdHandler auf Windows-Systemen starten“ auf Seite 253. Danach besitzt der Dienst den Autostart-Typ `Manuell`.

Ändern Sie den Autostart-Typ per Systemsteuerung auf `Automatisch`. Damit wird der Dienst beim Start des Windows-Systems automatisch gestartet.

8.4 openUTM-LU62 Gateway administrieren

Das openUTM-LU62 Gateway kann nicht nur mit der Management Console, sondern auch direkt mit Skripts administriert werden (siehe Abschnitt „BeanConnect Proxy mit der Management Console administrieren“ auf Seite 272). Auf Solaris- oder Linux-Systemen können Sie alle nachfolgend angegebenen Administrationsbefehle aus einer Shell aufrufen.

Nach der Installation des openUTM-LU62 Gateways auf Solaris- oder Linux-Systemen sind alle Benutzer berechtigt, das Gateway zu administrieren. Wenn Sie aus Sicherheitsgründen die Administrationsrechte auf bestimmte Benutzer beschränken möchten, gehen Sie folgendermaßen vor:

Im Home-Verzeichnis des openUTM-LU62 Gateways (Voreinstellung: `/opt/lib/utmlu62`) befindet sich die Datei `u62_users`. In dieser Datei können Sie eine Liste der administrationsberechtigten Benutzer festlegen. Die Benutzernamen müssen durch Leerzeichen, Tabulator, Komma oder Zeilenumbruch getrennt sein.

Wenn die in der Datei `u62_users` definierte Liste Benutzernamen enthält, wird der Administrationszugriff allen Benutzern verweigert, die nicht in der Liste enthalten sind (Ausnahme: `root`, der immer über Administrationsrechte verfügt).

Auf Windows-Systemen ist es empfehlenswert, ein Eingabefenster für DOS-Befehle zu öffnen:

Start - Programme - openUTM-LU62 - Kommando-Eingabe

Von diesem Eingabefenster für DOS-Befehle aus können Sie alle nachfolgend beschriebenen Befehle direkt aufrufen (ohne das Präfix `<LU62_home>/`).

8.4.1 openUTM-LU62-Gateway starten

Der folgende Befehl startet openUTM-LU62 als Hintergrundprozess, wobei `<LU62_home>` das Installationsverzeichnis des openUTM-LU62-Gateway bezeichnet:

```
<LU62_home>/u62_start
```

8.4.2 openUTM-LU62-Gateway beenden

Der folgende Befehl beendet das openUTM-LU62 Gateway, wobei <LU62_home> das Installationsverzeichnis des openUTM-LU62-Gateway bezeichnet:

```
<LU62_home>/u62_adm -e
```

Auf Windows-Systemen können Sie stattdessen den folgenden Eintrag aus der Programmgruppe wählen:

Start - Programme - openUTM-LU62 - Beenden von openUTM-LU62

8.4.3 Statusinformationen des openUTM-LU62-Gateway anzeigen

Der folgende Befehl zeigt aktuelle Statusinformationen zum openUTM-LU62 Gateway an:

```
<LU62_home>/u62_sta
```

Auf Windows-Systemen können Sie stattdessen den folgenden Eintrag aus der Programmgruppe wählen:

Start - Programme - openUTM-LU62 - Status-Anzeige

8.5 Communication Service administrieren

Dieser Abschnitt beschreibt, wie Sie den SNA-Dämon starten und wie Sie den Communication Service (SNAP-IX auf Solaris-Systemen oder IBM Communications Server (auf Linux- und Windows-Systemen) auf Befehlsebene starten.

Weitere Einzelheiten über die Verwaltung von SNAP-IX oder des IBM Communications Servers finden Sie in der Dokumentation des Herstellers.

8.5.1 SNA-Dämon starten und beenden (Linux- und Solaris-Systeme)

Damit der Communication Service administriert werden kann, muss der SNA-Dämon laufen. Andernfalls wird eine Fehlermeldung ausgegeben.

SNA-Dämon starten

1. Wechseln Sie in das Verzeichnis `/opt/sna/bin` (Solaris-Systeme) oder `/opt/ibm/sna/bin` (Linux-Systeme) auf dem System, auf dem der Communication Service installiert wurde.
2. Geben Sie den Befehl `./sna start` ein.

SNA-Dämon beenden

1. Wechseln Sie in das Verzeichnis `/opt/sna/bin` (Solaris-Systeme) oder `/opt/ibm/sna/bin` (Linux-Systeme) auf dem System, auf dem der Communication Service installiert wurde.
2. Geben Sie den Befehl `./sna stop` ein.

8.5.2 Communication Service auf Befehlsebene starten und beenden (Linux- und Solaris-Systeme)

Auf Linux- und Solaris-Systemen generiert die BeanConnect Management Console für einen BeanConnect Proxy nach jeder Änderung der Konfiguration des Communication Service die Skripts `cs-start-all.sh` und `cs-stop.sh`. Diese Skripts werden beim Sichern erzeugt und ermöglichen es, die Proxy-Komponente auf Befehlsebene zu starten oder zu beenden.

Wenn sich alle Proxy-Komponenten auf demselben Rechner befinden, finden Sie die Skripts im Verzeichnis `<Proxy_home>/shsc`, dabei ist `<Proxy_home>` das Home-Verzeichnis des Proxy-Containers.

Wenn openUTM-LU62 Gateway und Communication Service dagegen auf einem separaten Rechner ablaufen, dann wird das Skript beim MC-CmdHandler bereitgestellt, den die Management Console zur Verwaltung dieser Proxy-Komponenten verwendet. Das Skript steht dann im Verzeichnis `<MC-CmdHandler_Home>/shsc`, dabei ist `<MC-CmdHandler_Home>` das Home-Verzeichnis des MC-CmdHandlers.

Communication Service starten

Gehen Sie zum Starten des Communication Services auf Befehlsebene wie folgt vor:

1. Öffnen Sie eine Shell auf dem Rechner, auf dem der Communication Service installiert ist.
2. Wechseln Sie in das Home-Verzeichnis des Proxy-Containers (Installation auf Proxy-Rechner) bzw. in das Home-Verzeichnis des MC-CmdHandlers (Installation auf separatem Rechner).
3. Rufen Sie das Skript auf:

   ```
   shsc/cs-start-all.sh
   ```

Communication Service beenden

Gehen Sie zum Beenden des Communication Services auf Befehlsebene wie folgt vor:

1. Öffnen Sie eine Shell auf dem Rechner, auf dem der Communication Service installiert ist.
2. Wechseln Sie in das Home-Verzeichnis des Proxy-Containers (Installation auf Proxy-Rechner) bzw. in das Home-Verzeichnis des MC-CmdHandlers (Installation auf separatem Rechner).
3. Rufen Sie das Skript auf:

   ```
   shsc/cs-stop.sh
   ```

8.6 Verfügbarkeit von BeanConnect Proxys überprüfen

Unter der Verfügbarkeit von BeanConnect Proxys versteht man die Verfügbarkeit des Proxy Containers (samt der für CICS benötigten Komponenten) und seiner Kommunikationspartner (EIS Partner, Resource Adapter, MC-CmdHandler). Die Verfügbarkeitsprüfung lässt sich als Gesamtprüfung oder einzeln als komponenten- und partnerspezifische Prüfung durchführen.

Im Einzelnen haben Sie folgende Möglichkeiten:

- Verfügbarkeit eines Proxys überprüfen
- Verfügbarkeit eines BeanConnect Resource Adapters überprüfen
- Verfügbarkeit eines openUTM-LU62 Gateways und eines Communication Services überprüfen
- Verfügbarkeit eines MC-CmdHandlers überprüfen
- Verfügbarkeit eines EIS Partners überprüfen

Sie können auch die Verfügbarkeit eines Proxy Clusters überprüfen. Details siehe „Besonderheiten bei einem Proxy Cluster“ auf Seite 290.

8.6.1 Verfügbarkeit eines Proxys überprüfen

Die Verfügbarkeit eines Proxys (samt seiner Kommunikationspartner) können Sie in der Management Console prüfen, indem Sie im Navigationsbaum das Kontextmenü eines Proxy-Knotens öffnen und den Befehl **Check Availability** wählen.

Zusätzlich können Sie die Verfügbarkeit automatisch in vordefinierten Zeitintervallen prüfen lassen, indem Sie im Eigenschaftsfeld des Proxys den Parameter **Automatic Availability Check** einstellen.

Die Management Console überprüft die Verfügbarkeit von:

- Proxy Container
- allen Resource Adaptern, die dem Proxy zugewiesen sind
- allen MC-CmdHandlern
- openUTM-LU62 Gateway und Communication Service, falls der Proxy für CICS-Partner konfiguriert ist
- allen EIS Partnern

Damit die Verfügbarkeit des Resource Adapters und der EIS Partner geprüft werden kann, muss der Proxy-Container laufen, für CICS-Partner auch das openUTM-LU62-Gateway und der Communication Service.

Die Management Console zeigt ein Aktions-Dialogfeld zur Überprüfung der Aktionen und deren Ergebnisse an.

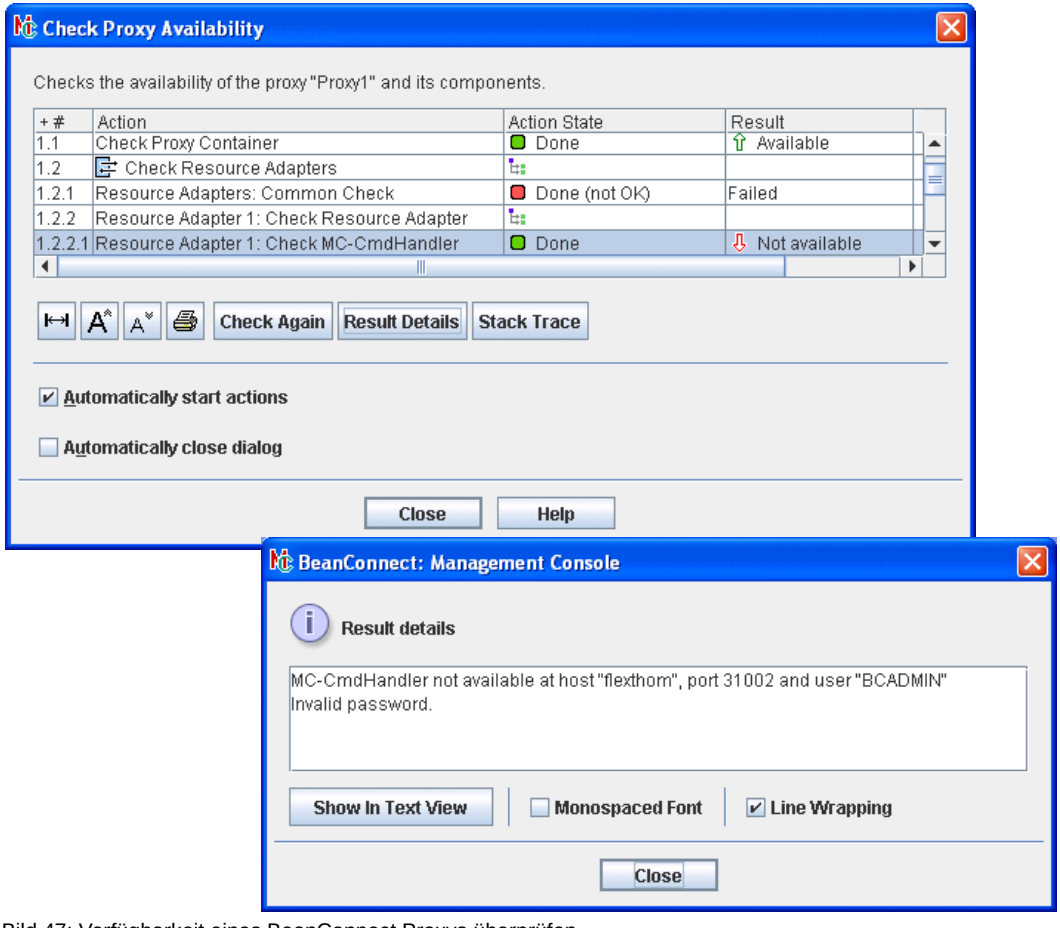


Bild 47: Verfügbarkeit eines BeanConnect Proxys überprüfen

Falls eine der Komponenten nicht verfügbar ist, wählen Sie den Eintrag aus und klicken Sie auf **Result Details**, um detaillierte Prüfungsergebnisse zu erhalten. Diese Daten können für eine Diagnose hilfreich sein.

Beim Konfigurieren des Proxys können Sie ein Zeitintervall für die regelmäßige Verfügbarkeitsprüfung festlegen.

Besonderheiten bei einem Proxy Cluster

- Wenn Sie den Befehl **Check Availability** im Kontextmenü eines Proxy Clusters wählen, werden die Komponenten aller Proxys innerhalb des Clusters geprüft.
- Um die Verfügbarkeit eines einzelnen Proxys im Cluster zu überprüfen, selektieren Sie den Proxy im Arbeitsbereichsfenster **Cluster Proxies** und wählen den Befehl **Check Availability** im Kontextmenü.

8.6.2 Verfügbarkeit eines BeanConnect Resource Adapters überprüfen

Die Verfügbarkeit eines einzelnen Resource Adapters können Sie in der Management Console prüfen, indem Sie im Navigationsbaum das Kontextmenü eines Resource Adapters öffnen und den Befehl **Check Availability** wählen. Der zugehörige Proxy-Container muss dazu laufen.

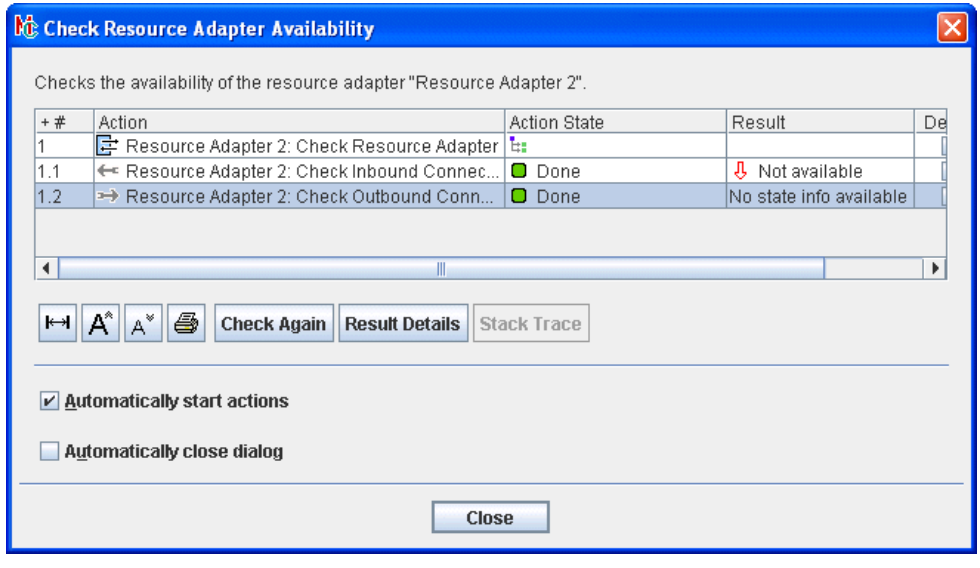


Bild 48: Verfügbarkeit eines Resource Adapters überprüfen

Markieren Sie eine Zeile und klicken Sie auf **Result Details**, um detaillierte Prüfungsergebnisse zu erhalten.

Die detaillierten Prüfungsergebnisse enthalten für jeden konfigurierten Resource Adapter einen Ergebnisstring in folgender Form:

-/+[host:hostname,port:portnummer]

Das Vorzeichen (+/-) zeigt die Verfügbarkeit an:
+ bedeutet verfügbar
- bedeutet nicht verfügbar

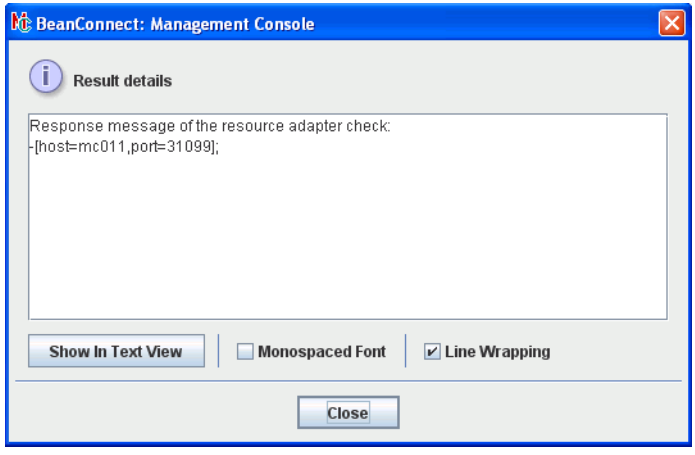


Bild 49: Verfügbarkeit eines Resource Adapters überprüfen - Ergebnisdetails

8.6.3 Verfügbarkeit eines openUTM-LU62 Gateways und eines Communication Services überprüfen

Die Verfügbarkeit der Komponenten openUTM-LU62 Gateway und Communication Service können Sie in der Management Console prüfen, indem Sie auf der obersten Ebene auf den Knoten **openUTM-LU62 Gateway** bzw. **Communication Services** klicken und im Kontextmenü den Befehl **Check Availability** wählen. Für die Überprüfung muss der zugehörige MC-CmdHandler laufen. Dies gilt auch, wenn die Komponente auf demselben Rechner läuft wie der Proxy.

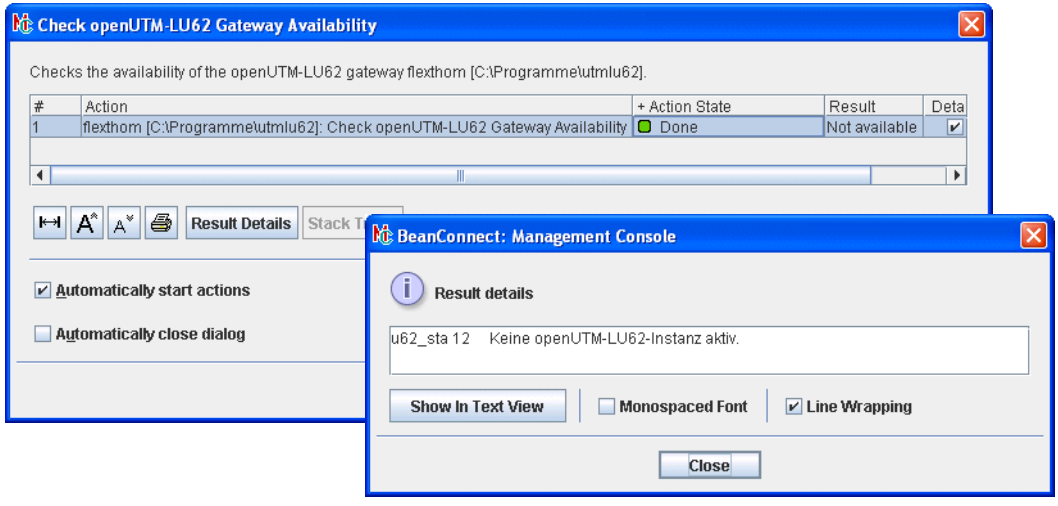


Bild 50: Verfügbarkeit eines openUTM-LU62 Gateways überprüfen

Mit Klicken auf **Result Details** erhalten Sie wie im Bild detaillierte Prüfungsergebnisse.

8.6.4 Verfügbarkeit eines MC-CmdHandlers überprüfen

8.6.4.1 Verfügbarkeit des MC-CmdHandlers mit der Management Console prüfen

Die Verfügbarkeit einer Client Instanz des MC-CmdHandlers können Sie in der Management Console prüfen. Sie öffnen dazu das Arbeitsbereichsfenster **MC-CmdHandler Client Instances**, indem Sie auf der obersten Ebene auf den zugehörigen Knoten klicken und im Kontextmenü den Befehl **Show MC-CmdHandler Client Instances** wählen.

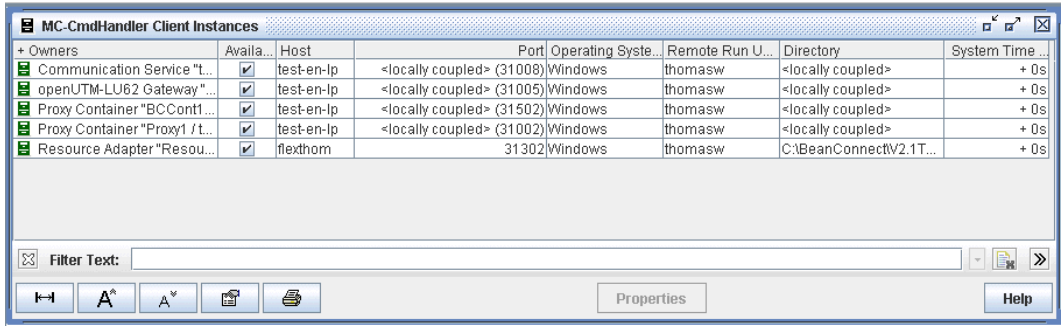


Bild 51: Verfügbarkeit eineMC-CmdHandler Instanz

Die Spalte **Availability** in dieser Tabelle gibt an, ob der MC-CmdHandler verfügbar ist oder nicht.

Sie können die Verfügbarkeit eines entfernten MC-CmdHandlers überprüfen, indem Sie die betreffende Zeile markieren und im Kontextmenü den Befehl **Check Availability** wählen.

"Interne" MC-CmdHandler werden in der Liste als **<locally coupled>** gekennzeichnet. Ihre Verfügbarkeit kann nicht geprüft werden, da sie implizit verfügbar sind.

8.6.4.2 Verfügbarkeit des MC-CmdHandlers auf Befehlsebene prüfen

Solaris- und Linux-Systeme

Mit folgendem Skript im Home-Verzeichnis des Proxy-Containers prüfen Sie, ob der MC-CmdHandler läuft:

- `shsc/checkmccmdhandler.sh`

Bei einem stand-alone MC-CmdHandler befinden sich die Skripts im Verzeichnis `shsc` unterhalb seines Installationsverzeichnisses.

Windows-Systeme

Prüfen Sie, ob der MC-CmdHandler läuft, indem Sie im Menü **Start** aus der Programmgruppe **FUJITSU Software BeanConnect V3.0B00 - Proxy <container> - MC-CmdHandler** den Befehl **MC-CmdHandler Check** wählen.

Die Verfügbarkeit des MC-CmdHandlers können Sie auch mit folgendem Skript im Home-Verzeichnis des Proxy-Containers prüfen:

- `shsc\checkmccmdhandler.cmd`

8.6.5 Verfügbarkeit eines EIS Partners überprüfen

Die Verfügbarkeit eines EIS Partner kann nur geprüft werden, wenn beim Konfigurieren des EIS Partners ein Service im EIS angeben wurde. Dieser Service wird beim Prüfen der Verfügbarkeit aufgerufen und die Ausgabenachricht dieses Services von der Management Console ausgegeben, wenn Sie den Befehl **Result Details** wählen.

Sie prüfen die Verfügbarkeit eines EIS Partners in der Management Console, indem Sie im Navigationsbaum das Kontextmenü des EIS Partners öffnen und den Befehl **Check Availability** wählen. Der zugehörige Proxy-Container muss dazu laufen, für CICS-Partner auch das openUTM-LU62-Gateway und der Communication Service.

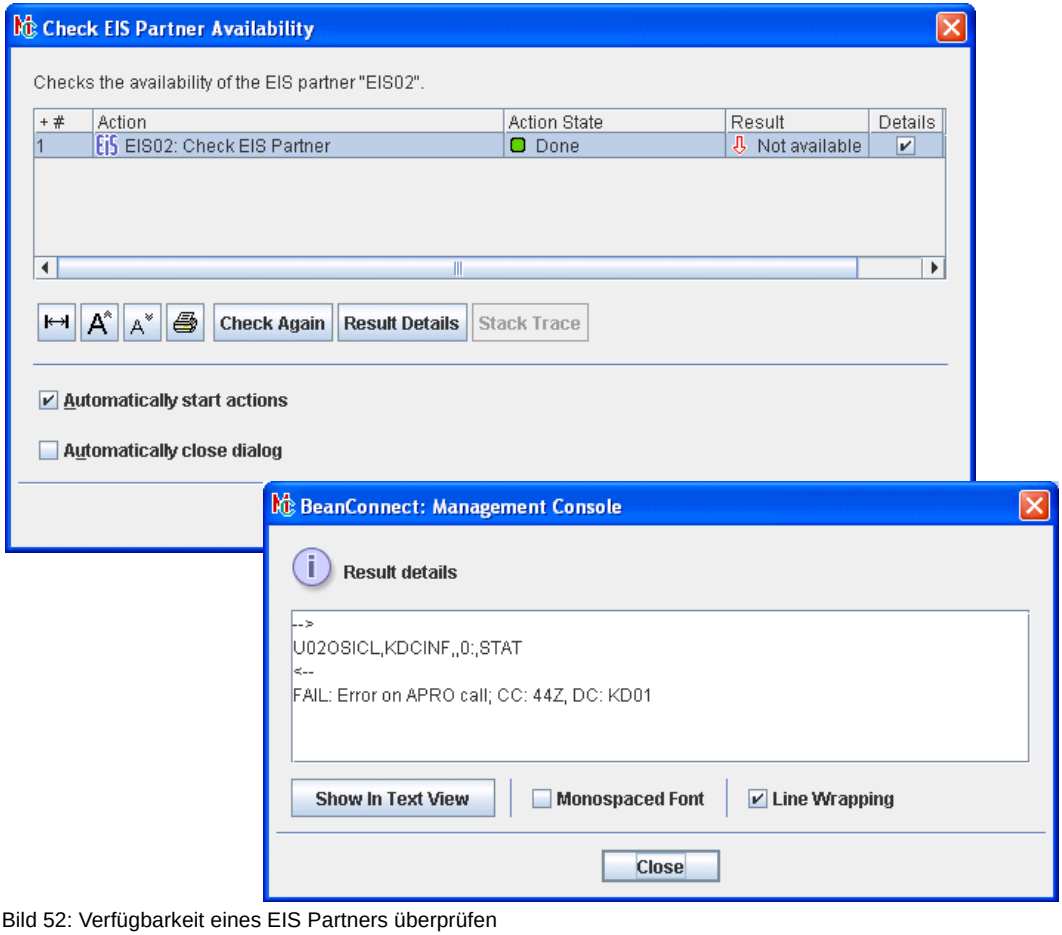


Bild 52: Verfügbarkeit eines EIS Partners überprüfen

Markieren Sie eine Zeile und klicken Sie auf **Result Details**, um die Ausgabenachricht vom EIS zu erhalten. Im Fehlerfall finden Sie wie im Beispiel detaillierte Diagnoseinformationen.

8.7 Überwachen des Resource Adapters mit der Management Console

Der Resource Adapter kann über MBean-Clients überwacht werden. Dazu muss ein MBean-Client in der Management Console konfiguriert sein, siehe Abschnitt „Management Console als JMX-Client konfigurieren“ auf Seite 255.

Der MBean-Client ist einem Resource Adapter fest zugeordnet und wird im Baum des Resource Adapters angezeigt.

Zusätzlich können Sie einen stand-alone MBean-Client definieren, der keinem Resource Adapter zugeordnet ist.

Weitere Informationen dazu entnehmen Sie dem Abschnitt „Freie JMX-Clients einrichten“ auf Seite 260.

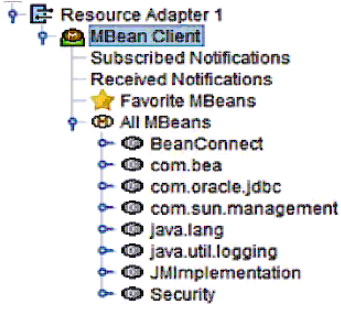


Bild 53: Navigationsbaum einer MBean

Unter dem Knoten **MBean Client** befinden sich die folgenden Knoten:

Subscribed Notifications

Ermöglicht den Zugriff auf alle abonnierten Notifications. Das Abonnieren müssen Sie explizit durchführen.

Received Notifications

Ermöglicht den Zugriff auf alle empfangenen Notifications. Wenn Notifications empfangen wurden, dann wird der Knoten fett dargestellt, dahinter steht in Klammern die Anzahl der empfangenen Notifications.

Statistics Collectors

Ermöglicht den Zugriff auf die Statistik-Kollektoren. Dieser Knoten wird erst dann angezeigt, nachdem Statistik-Kollektoren eingerichtet wurden, siehe Abschnitt „Statistikwerte sammeln und anzeigen“ auf Seite 301.

Favorite MBeans

Zeigt die MBeans an, die in die Favoritenliste aufgenommen wurden. Die Favoritenliste dient dazu, die Übersicht zu erleichtern, indem z.B. häufig verwendete MBeans nochmals dort dargestellt werden. Favoriten werden über das jeweilige Kontextmenü mit dem Befehl **Add to Favorites** definiert.

All Beans

Ermöglicht den Zugriff auf alle MBeans, auch auf diejenigen MBeans, die unter **Favorite MBeans** stehen.

8.7.1 Verbindung zum MBean Server aufbauen

Die meisten Aktionen können nur durchgeführt werden, wenn eine Verbindung zum MBean Server besteht. Falls noch keine Verbindung besteht, wählen Sie im Kontextmenü des MBean-Clients den Befehl **Connect To MBean Server**.

8.7.2 MBean Object Names anzeigen

Durch Aufklappen der Knoten **All MBeans** oder **Favorite MBeans** werden alle definierten MBean-Domänen bzw. alle im Favoriten enthaltenen Domänen angezeigt. Durch weiteres Aufklappen der einzelnen MBean-Domänen erhalten Sie die Object Names der MBeans. Alternativ dazu können Sie jeweils im Kontextmenü den Befehl **Show MBeans** auswählen.

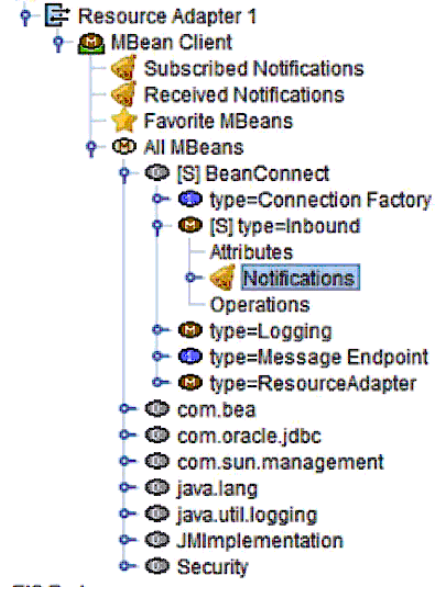


Bild 54: MBean - Teilbaum der Objektnamen

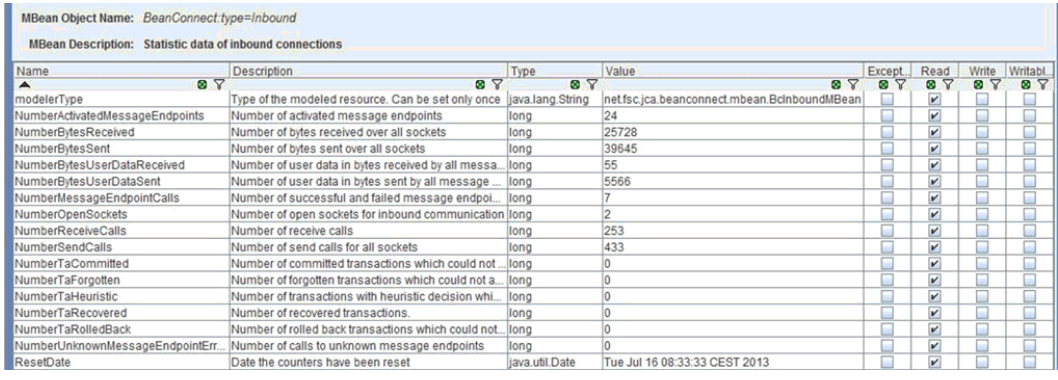
Für jede MBean wird eine Liste angezeigt mit den Elementen **Attributes**, **Notifications** und ggf. **Operations**, falls für die MBean Operationen möglich sind.

8.7.3 MBean Attribute anzeigen und ändern

Eine MBean besitzt eine in der Regel umfangreiche Liste von Attributen. Sie können die aktuellen Werte dieser Attribute über die Management Console ausgeben lassen. Einige dieser Attribute lassen sich auch ändern.

8.7.3.1 MBean Attribute anzeigen

Wenn Sie die Attribute einer MBean ansehen möchten, klappen Sie einen MBean-Knoten auf und klicken Sie auf **Attributes**. Alternativ dazu können Sie auch den Befehl **Show MBean Attributes** im Kontextmenü auswählen.



MBean Object Name: *BeanConnect:type=Inbound*

MBean Description: **Statistic data of inbound connections**

Name	Description	Type	Value	Except...	Read	Write	Writabl...
modelerType	Type of the modeled resource. Can be set only once	java.lang.String	net.fsc.jca.beanconnect.mbean.BcInboundMBean	☐	☑	☐	☐
NumberActivatedMessageEndpoints	Number of activated message endpoints	long	24	☐	☑	☐	☐
NumberBytesReceived	Number of bytes received over all sockets	long	25728	☐	☑	☐	☐
NumberBytesSent	Number of bytes sent over all sockets	long	39645	☐	☑	☐	☐
NumberBytesUserDataReceived	Number of user data in bytes received by all messa...	long	55	☐	☑	☐	☐
NumberBytesUserDataSent	Number of user data in bytes sent by all message ...	long	5566	☐	☑	☐	☐
NumberMessageEndpointCalls	Number of successful and failed message endpoi...	long	7	☐	☑	☐	☐
NumberOpenSockets	Number of open sockets for inbound communication	long	2	☐	☑	☐	☐
NumberReceiveCalls	Number of receive calls	long	253	☐	☑	☐	☐
NumberSendCalls	Number of send calls for all sockets	long	433	☐	☑	☐	☐
NumberTaCommitted	Number of committed transactions which could not ...	long	0	☐	☑	☐	☐
NumberTaForgotten	Number of forgotten transactions which could not a...	long	0	☐	☑	☐	☐
NumberTaHeuristic	Number of transactions with heuristic decision whi...	long	0	☐	☑	☐	☐
NumberTaRecovered	Number of recovered transactions.	long	0	☐	☑	☐	☐
NumberTaRolledBack	Number of rolled back transactions which could not...	long	0	☐	☑	☐	☐
NumberUnknownMessageEndpointErr...	Number of calls to unknown message endpoints	long	0	☐	☑	☐	☐
ResetDate	Date the counters have been reset	java.util.Date	Tue Jul 16 08:33:33 CEST 2013	☐	☑	☐	☐

Bild 55: MBean - Attributtabelle

Wenn Sie in der Tabelle ein Attribut per Maus-Klick auswählen, dann werden die Details im unteren Teils des Fensters angezeigt. Diese Detailansicht ist für umfangreiche Attribute gedacht, die nicht vollständig in der Tabelle angezeigt werden können.

Die Spalten der Tabelle haben folgende Bedeutung:

Name / Description

Name und nähere Beschreibung des Attributes.

Type

Typ des Attributwertes, z.B. String, Integer, Boolean.

Value

Wert des Attributes.

Exception

Gibt an, ob der JMX Server beim Ermitteln des Attributwertes eine Exception geliefert hat. Wenn dies der Fall ist, dann können Sie die Exception über die Detailansicht des Attributes anzeigen.

Read

Gibt an, ob der Attributwert beim JMX Server abgefragt werden kann. Das ist für die meisten Attribute der Fall.

Write

Gibt an, ob der Attributwert geändert werden kann.

Writable

Gibt an, ob der Attributwert über die Management Console geändert werden kann.

Über die Management Console können nur Attribute mit "einfachen" Wertetypen geändert werden (String, Integer, Boolean). Attribute mit komplexen (z.B. zusammengesetzten) Werten können in der Regel nicht über die Management Console geändert werden. Diese Attribute sind dann als **Write**, jedoch nicht als **Writable** gekennzeichnet.

8.7.3.2 MBean Attributwerte ändern

Sie können alle Attribute über die Management Console ändern, die in der Tabelle als **Writable** gekennzeichnet sind. Dazu gehen Sie so vor:

- Wählen Sie im Kontextmenü des Attributes den Befehl **Set MBean Attribute Value** aus. Alternativ dazu können Sie das Attribut markieren und auf die Schaltfläche **Set Value** klicken.
- Ändern Sie den Wert über den nachfolgenden Dialog.

Die Management Console gibt nach der Aktion eine Meldung aus. Diese enthält entweder eine Bestätigung der Änderung oder eine Fehlermeldung, falls die Änderung nicht durchgeführt werden konnte.

8.7.4 Statistikwerte sammeln und anzeigen

Sie können über die Management Console Statistiken erstellen, indem Sie Statistik-Kollektoren einrichten. Diese Statistik-Kollektoren fragen die Werte von MBean Attributen in regelmäßigen Abständen ab.

8.7.4.1 Statistik-Kollektoren einrichten, anzeigen und ändern

Für jedes Attribut einer MBean können Sie einen Statistik-Kollektor einrichten. Die Management Console erzeugt daraufhin unterhalb des MBean-Clients einen Knoten mit dem Namen **Statistics Collectors**. Unter diesem Knoten werden alle Statistik-Kollektoren dieses MBean-Clients angezeigt.

Einen Statistik-Kollektor richten Sie wie folgt ein:

- Lassen Sie sich die Attribute der betreffenden MBean anzeigen, siehe Abschnitt „MBean Attribute anzeigen“ auf Seite 299.
- Markieren Sie das Attribut in der Tabelle und wählen Sie im Kontextmenü des Attributes den Befehl **Collect Attribute Values** aus.

Durch Klicken auf den Knoten **Statistics Collectors** wird eine Tabelle mit allen verfügbaren Statistik-Kollektoren eines MBean-Clients ausgegeben. Alternativ dazu können Sie den Befehl **Show Statistics Collectors** im Kontextmenü dieses Knotens auswählen. Weitere Details zur Bedeutung der Tabellenspalten finden Sie in der Online-Hilfe.

Für jeden Statistik-Kollektor können Sie das Zeitintervall für das Sammeln von Daten ändern. Dazu markieren Sie den betreffenden Statistik-Kollektor in der Tabelle, wählen im Kontextmenü den Befehl **Edit Properties** und ändern das Zeitintervall im nachfolgenden Dialog. Alternativ dazu können Sie Schaltfläche **Edit** unterhalb der Tabelle anklicken.

Sie entfernen einen Statistik-Kollektor aus der Tabelle, indem Sie im Kontextmenü des Statistik-Kollektors den Befehl **Remove MBean Statistics Collector** wählen (Alternative: Schaltfläche **Remove** unterhalb der Tabelle). Mit dem Entfernen eines Statistik-Kollektors werden auch alle Daten gelöscht, die bis zu diesem Zeitpunkt von diesem Kollektor gesammelt wurden.

8.7.4.2 Statistikwerte anzeigen

Sie können sich die Werte eines Statistik-Kollektors ansehen, indem Sie die Tabelle mit den Statistik-Kollektoren anzeigen und im Kontextmenü des gewünschten Kollektors den Befehl **Show Statistic Values** auswählen. Alternativ dazu können Sie den Kollektor markieren und die Schaltfläche **Show Values** unterhalb der Tabelle anklicken.

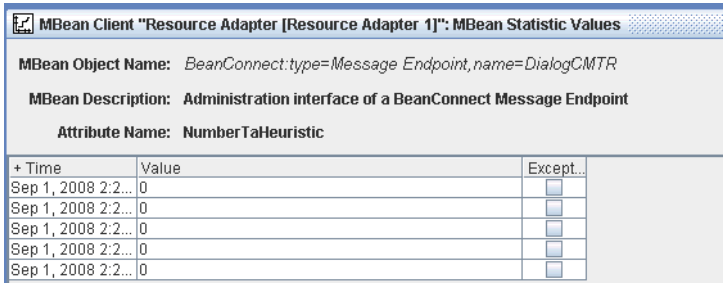


Bild 56: MBean - Statistikwerte

Die Tabelle mit dem gesammelten Statistikwerten besitzt die drei Spalten **Time**, **Value** und **Exception**, siehe Online-Hilfe. In der Spalte **Exception** wird ggf. die Exception ausgegeben, die der JMX-Server beim Ermitteln des Statistikwertes ausgegeben hat. Per Maus-Klick auf einen Statistikwert werden die Details wie z.B. der vollständige Text einer Exception im unteren Bereich des Fenster angezeigt.

8.7.5 MBean Notifications abonnieren und anzeigen

Die MBeans vom Typ **ResourceAdapter**, **ManagedConnectionFactory**, **Inbound** und **MessageEndpoint** bieten Notifications an. Dies sind Nachrichten, die der Resource Adapter bei bestimmten Ereignissen erzeugt und die in der Management Console angezeigt werden können. Damit Notifications angezeigt werden, müssen Sie diese in der Management Console explizit abonnieren (Subscribe Verfahren).

Folgende Tabelle zeigt, welche Notifications Sie abonnieren können:

Typ der MBean	**Name und Bedeutung der Notification**
ResourceAdapter	`BeanConnect.Started` Der Resource Adapter wurde gestartet.
	`BeanConnect.Stopped` Der Resource Adapter wurde beendet.
ManagedConnectionFactory	`BeanConnect.Connection.Error` Für eine Verbindung wurde vom Resource Adapter eine Exception geworfen.
	`BeanConnect.Transaction.Rollback` Die Verbindung ist an einer Transaktion beteiligt, die zurückgesetzt wurde.
	`BeanConnect.Transaction.Heuristic` Für den Zweig der Transaktion, an dem die Verbindung beteiligt ist, wurde eine heuristische Entscheidung getroffen.
Inbound	`BeanConnect.MessageEndpoint.Activiation` Der Application Server hat einen Message Endpoint bei BeanConnect aktiviert.
	`BeanConnect.MessageEndpoint.Deactiviation` Der Application Server hat einen Message Endpoint bei BeanConnect deaktiviert.
	`BeanConnect.MessageEndpoint.Unknown` Es ist eine Nachricht für einen unbekannten Message Endpoint eingetroffen.
	`BeanConnect.Transaction.Rollback` Eine Transaktion wurde bei der Recovery zurückgesetzt.
	`BeanConnect.Transaction.Heuristic` Für eine Transaktion wurde bei der Recovery eine heuristische Entscheidung getroffen.
MessageEndpoint	`BeanConnect.MessageEndpoint.Error` Beim Aufruf des Message Endpoints ist ein Fehler aufgetreten.
	`BeanConnect.MessageEndpoint.Exception` Beim Aufruf des Message Endpoints wurde eine Exception geworfen.

Typ der MBean	Name und Bedeutung der Notification
	`BeanConnect.Transaction.Rollback` Der Message Endpoint ist an einer Transaktion beteiligt, die zurückgesetzt wurde.
	`BeanConnect.Transaction.Heuristic` Für den Zweig der Transaktion, an dem der Message Endpoint beteiligt ist, wurde eine heuristische Entscheidung getroffen.

8.7.5.1 MBean Notifications abonnieren

Sie können die MBean Notifications sowohl alle auf einmal als auch einzeln abonnieren:

- Wenn Sie alle Notifications einer MBean auf einmal abonnieren möchten, dann wählen Sie im Kontextmenü des MBean-Knotens den Befehl **Subscribe MBean Notifications**.
- Wenn Sie eine einzelne Notification abonnieren möchten, expandieren Sie im Teilbaum einer MBean den Knoten **Notifications**. Damit werden alle abonnierbaren Notifications angezeigt. Wählen Sie jetzt im Kontextmenü der gewünschten Notification den Befehl **Subscribe MBean Notifications of this type** aus.

Wenn Sie Notifications wieder abbestellen möchten, dann verwenden Sie den Befehl **Unsubscribe MBean Notification** im Kontextmenü des **Notifications** Knoten einer MBean oder den Befehl **Unsubscribe MBean Notifications of this type** im Kontextmenü eines bestimmten Notification-Typ-Knotens.

Die Einstellungen zu den Notifications bleiben auch nach Beenden der Management Console erhalten. D.h. einmal abonnierte Notifications müssen nicht nach jedem Start der Management Console neu abonniert werden, sondern werden automatisch zugestellt, nachdem die Verbindung zum JMX Server aufgebaut wurde. Dabei werden jedoch keine Notifications zugestellt, die in dem Zeitraum erzeugt wurden, in dem die Management Console nicht an den MBean-Server angemeldet war.

8.7.5.2 MBean Notifications anzeigen

Die Management Console zeigt durch fette Darstellung der entsprechenden Knoten an, dass Notifications empfangen wurden. Hinter dem Knoten steht in Klammern die Anzahl der Notifications. Sie haben folgende Möglichkeiten, die MBean Notifications anzusehen.

- Wenn Sie alle Notifications von allen MBeans des MBean-Clients anzeigen möchten, dann klicken Sie unterhalb des Knotens **MBean Client** auf den Knoten **Received Notifications** oder Sie wählen im Kontextmenü dieses Knotens den Befehl **Show Received Notifications**.

- Wenn Sie alle Notifications einer bestimmten MBean anzeigen möchten, dann klappen Sie den Teilbaum die betreffende MBean auf und klicken auf den Knoten **Notifications**. Alternative: Wählen Sie im Kontextmenü dieses Knotens den Befehl **Show Received Notifications**.
- Wenn Sie alle Notifications eines bestimmten Typs anzeigen möchten, dann klicken Sie auf einen Notification-Typ-Knoten oder wählen **Show Received Notifications** in dessen Kontextmenü aus.

Die Notifications mit ihren Attributen werden in Tabellenform in einem weiteren Fenster aufgelistet. Wenn Sie Detailinformationen für eine Notification haben möchten, dann selektieren Sie die Notification per Maus-Klick oder wählen im Kontextmenü den Befehl **Show MBean Attributes**. Die Details werden im unteren Bereich des Fensters angezeigt. Weitere Informationen zur Bedeutung der Tabellenspalten siehe Online-Hilfe.

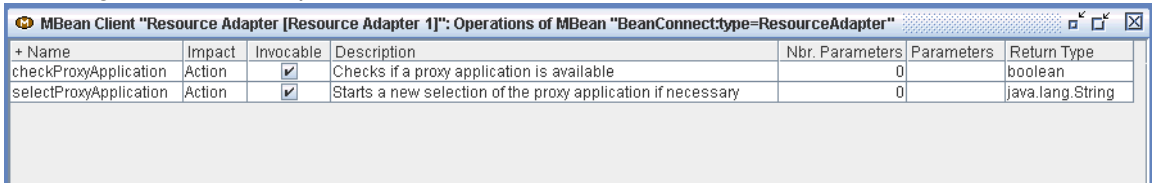
Eine Notification löschen Sie per Befehl **Remove MBean Notification** im Kontextmenü. Alternativ dazu können Sie die Notification markieren und über die Schaltfläche **Remove** unterhalb der Tabelle entfernen.

Empfangene Notifications werden beim Beenden der Management Console nicht gesichert und damit in der nächsten Sitzung auch nicht mehr angezeigt.

8.7.6 MBean Operationen anzeigen und ausführen

Die Management Console bietet die Möglichkeit, Operationen auf MBeans auszuführen. Dies sind bestimmte Aktionen, die im Resource Adapter durchgeführt werden wie z.B. das Rücksetzen von Statistikzählern oder das Prüfen der Verfügbarkeit.

Jede MBean besitzt einen Knoten **Operations**. Durch Klicken auf diesen Knoten oder durch den Befehl **Show MBean Operations** im Kontextmenü wird eine Tabelle ausgegeben, die alle Operationen der betreffenden MBean auflistet.



+ Name	Impact	Invocable	Description	Nbr. Parameters	Parameters	Return Type
checkProxyApplication	Action	☑	Checks if a proxy application is available	0		boolean
selectProxyApplication	Action	☑	Starts a new selection of the proxy application if necessary	0		java.lang.String

Bild 57: MBean Operationen

Wenn Sie eine Operation anklicken, werden die Detailinformationen im Fenster im unteren Bereich angezeigt.

Operationen, die Sie mit der Management Console ausführen können, sind in der Spalte **Invocable** entsprechend gekennzeichnet. Ähnlich wie beim Ändern von Attributwerten kann die Management Console nur Operationen ausführen, die keine oder nur einfache Parametertypen haben. Wenn Sie eine solche Operation ausführen möchten, gehen Sie wie folgt vor:

- Wählen Sie im Kontextmenü der Operation den Befehl **Invoke MBean Operation...**. Alternativ dazu können Sie auch die Operation selektieren und die Schaltfläche **Invoke** unterhalb der Tabelle anklicken.
- Tragen Sie im nachfolgenden Dialog die Aufrufparameter ein (falls nötig). Der Dialog kann aus mehreren Seiten bestehen. Erst wenn Sie mit **OK** bestätigen, wird die Operation durchgeführt.

Die Management Console gibt nach dem Ausführen eine Meldung aus. Diese enthält entweder das Resultat der Operation oder eine Fehlermeldung, falls die Operation nicht durchgeführt werden konnte.

Resource Adapter im Cluster auf anderen Proxy umschalten

Sie können die MBean Operation `selectProxyApplication` dazu benutzen, im laufenden Cluster-Betrieb einen Proxy zu beenden und gleichzeitig den/die Resource Adapter auf einen anderen Proxy umzuschalten ("sanftes" Umschalten). Dies hat den Vorteil, dass der Betrieb dadurch möglichst wenig beeinträchtigt wird. Wenn Sie dagegen einen Proxy eines Proxy Clusters im laufenden Betrieb direkt über Kontextmenü beenden, werden die von diesem Proxy verwalteten Verbindungen sofort abgebaut, was bewirkt, dass z.B offene Transaktionen unterbrochen werden und ggf. zurückgesetzt werden müssen.

Sie können dies durch ein "sanftes" Umschalten vermeiden. Gehen Sie wie folgt vor:

- Beenden Sie den Proxy über das openUTM-Tool `kdcshut` mit Angabe einer Wartezeit.

  Dazu öffnen Sie eine Shell bzw. das Eingabefenster für DOS-Befehle und geben folgendes Kommando ein (auf Windows mit "\"):

  ```
  <openUTM-Server_home>/ex/kdcshut <Proxy_home> time
  ```

  Dabei ist `<openUTM-Server_home>` das Installationsverzeichnis von openUTM, `<Proxy_home>` der komplette Pfadname des Proxys und `time` die Wartezeit in Minuten (Empfehlung: mindestens 10 Minuten). Dieser Aufruf bewirkt, dass einerseits keine neuen Verbindungen mehr angenommen werden und andererseits der Proxy nicht sofort heruntergefahren wird.

- Rufen Sie sofort danach in der Management Console das Resource Adapter MBean auf einer Resource Adapter Instanz auf, die mit dem zu beendenden Proxy zuzammenarbeitet. Rufen Sie in diesem MBean die Operation `selectProxyApplication` auf, um der Resource Adapter Instanz einen anderen Proxy zuzuordnen; die Auswahl des neuen Proxys trifft BeanConnect selbständig nach internen Algorithmen.

  Wenn der Application Server als Cluster konfiguriert ist, dann führen Sie diese Operation für jede Resource Adapter Instanz des Application Server Clusters durch, die dem zu beendenden Proxy zugeordnet ist. Welchem Proxy eine Resource Adapter Instanz zugeordnet ist, erkennen Sie an dem Attribut `CurrentProxyUrl` des Resource Adapter MBean.

- Nach dem Umschalten des Resource Adapters auf den neuen Proxy können Sie den bisherigen Proxy mit einem weiteren `kdcshut`-Aufruf und kurzer Wartezeit (z.B. 5 Minuten) beenden:

  ```
  <openUTM-Server_home>/ex/kdcshut <Proxy_home> 5 G
  ```

  Die Parameter `5` und `G` bewirken, dass der Proxy beendet wird, wenn alle Verbindungen abgebaut sind, spätestens aber, wenn die Wartezeit von 5 Minuten abgelaufen ist.

9 Command Line Interface der BeanConnect Management Console (MC-CLI)

BeanConnect bietet mit dem **M**anagement **C**onsole **C**ommand **L**ine **I**nterface (im Folgenden kurz mit MC-CLI bezeichnet) ein Paket von Jython-Funktionen an, mit dem Sie Funktionen der BeanConnect Management Console aus einem Jython-Skript heraus starten können. Der Einsatz des MC-CLI ist z. B. immer dann sinnvoll, wenn große Datenmengen konfiguriert oder regelmäßig wiederkehrende Administrationsaufgaben ausgeführt werden müssen (siehe auch Abschnitt „Anwendungsszenarien (Beispiele)“ auf Seite 429).

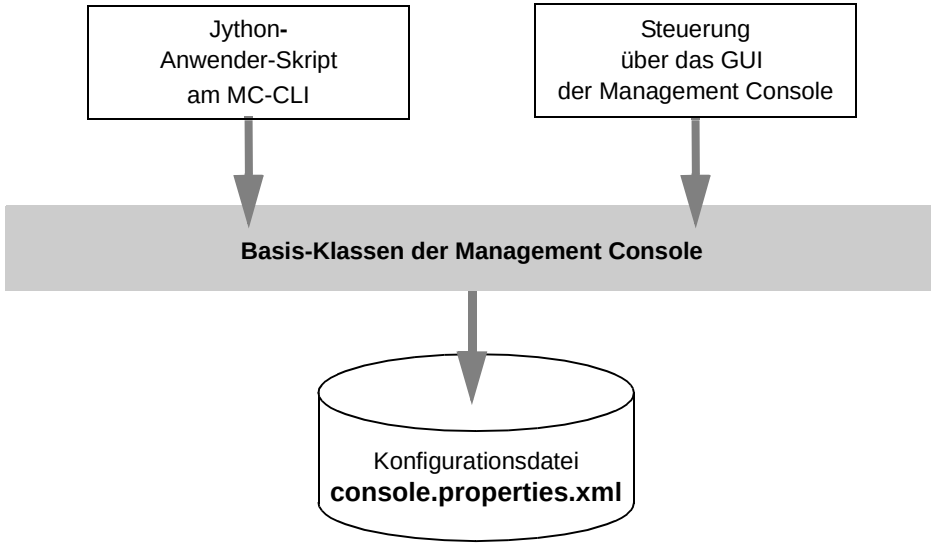


Bild 58: Command Line Interface (CLI) und die grafische Oberfläche (GUI) der Management Console

Sie können dieselbe Konfiguration nacheinander sowohl über die grafische Oberfläche der BeanConnect Management Console als auch über MC-CLI-Skripts bearbeiten.
Die Konfiguration wird in beiden Fällen durch die identische Konfigurationsdatei `console.properties.xml` repräsentiert. Die gleichzeitige Konfiguration und Administration von mehreren Management Console-Sitzungen aus (sowohl über die grafische Oberfläche als auch das MC-CLI) ist nicht möglich.

9.1 Übersicht über das MC-CLI

Alle Proxy Objekte und Proxy Cluster Objekte, d.h. Proxys, Proxy Cluster, Resource Adapter, EIS Partner, Inbound User, Inbound Services, Outbound Services, Inbound Message Endpoints und Outbound Communication Endpoints sowie die Proxy-Komponenten Communication Service und openUTM-LU62 Gateway, können über CLI-Anwenderskripts konfiguriert und administriert werden.

Sie können die oben aufgeführten Objekte erstellen (mit Ausnahme von Proxys), die Eigenschaften der Objekte lesen und ändern sowie Objekte aus der Konfiguration entfernen. Sie können die Eigenschaften der Todo-Topics lesen und Todo-Topics löschen.

Außerdem können Sie Administrationsfunktionen für die Objekte ausführen. Sie können z. B. Verfügbarkeit und Administrierbarkeit prüfen oder Proxys sowie Proxy Cluster starten und beenden.

Module und Funktionen des MC-CLI

Die Schnittstelle MC-CLI besteht aus mehreren Modulen, die jeweils eine Gruppe von Funktionen enthalten:

- Das Modul `BcAdminMain` enthält Funktionen zum Starten und Beenden einer Sitzung der Management Console am MC-CLI.

  Über den Parameter `console_home` wird auf die zentrale Datei `console.properties.xml` zugegriffen, die die Konfigurationsdaten enthält.

- Das Modul `BcAdminAction` enthält Funktionen, die das Ergebnis eines Administrationsaufrufs analysieren und Informationen zum Ergebnis und über alle Teilaktionen zurückgeben.

- Ein Modul für jeden Objekttyp, der über das MC-CLI administriert und konfiguriert werden kann. Jedes dieser Module enthält die Funktionen zur Administration und Konfiguration des jeweiligen Objekttyps. Die zur Verfügung stehenden Module sind in der folgenden Tabelle aufgelistet.

`BcAdminProxy`	Funktionen zur Konfiguration und Administration eines Proxys
`BcAdminProxyCluster`	Funktionen zur Konfiguration und Administration eines ProxyClusters
`BcAdminCommService`	Funktionen zur Konfiguration und Administration eines Communication Services (Proxykomponente zur Kommunikation mit CICS-Partnern)
`BcAdminLu62Gateway`	Funktionen zur Konfiguration und Administration eines openUTM-LU62 Gateways (Proxykomponente zur Kommunikation mit CICS-Partnern)

BcAdminRa	Funktionen zur Konfiguration und Administration eines Resource Adapters
BcAdminEisPartner	Funktionen zur Konfiguration und Administration eines EIS Partners
BcAdminInboundService	Funktionen zur Konfiguration eines Inbound Services
BcAdminInboundMsgEndpoint	Funktionen zur Konfiguration eines Inbound Message Endpoints
BcAdminInboundUser	Funktionen zur Konfiguration eines Inbound Users
BcAdminOutboundService	Funktionen zur Konfiguration eines Outbound Services
BcAdminOutboundCommEndpoint	Funktionen zur Konfiguration eines Outbound Communication Endpoints
BcAdminTodo	Funktionen zur Information und Behandlung eines Todo Topics

Die Module enthalten, je nach Objekttyp verschiedene Funktionen. Eine Liste aller in diesen Modulen enthaltenen Funktionen finden Sie in der folgenden Tabelle.

Welche Funktionen für den jeweiligen Objekttyp angeboten werden und welche Objekteigenschaften gelesen bzw. geändert werden können, ist im Abschnitt „Funktionen“ auf Seite 323 beschrieben.

create()	Neues Objekt zur Konfiguration hinzufügen
getObject()	Vorhandenes Objekt der Konfiguration lesen
authenticate()	Aufrufer authentifizieren
remove()	Objekt aus der Konfiguration entfernen
getProperties()	Eigenschaften des Objekts lesen
modifyProperties()	Eigenschaften des Objekts ändern
getList()	Liste von Objekten eines bestimmten Typs lesen, die mit diesem Objekt verknüpft sind
perform()	Administrative Aktionen starten, z.B. checkAvailability, start, stop
addProxy()	Proxy zum Proxy Cluster hinzufügen
removeProxy()	Proxy aus Proxy Cluster entfernen
getMasterProxy()	Master-Proxy eines Proxy Clusters lesen
setMasterProxy()	Master-Proxy eines Proxy Clusters wechseln
getAssignment()	Zuordnung einer Proxy-Komponente zu einem Proxy lesen
setAssignment()	Zuordnung einer Proxy-Komponente zu einem Proxy ändern
getProxies()	Alle Proxies lesen, denen diese Proxy-Komponente zugeordnet ist

`getGatewayPorts()`	Lu62Gateway Listener Ports eines Proxys bzw. aller Proxys im Cluster lesen
`setGatewayPorts()`	Lu62Gateway Listener Ports eines Proxys bzw. mehrerer Proxys im Cluster ändern
`getLuNames()`	Logical Unit Name eines Proxys bzw. aller Proxys im Cluster lesen
`setLuNames()`	Logical Unit Name eines Proxys bzw. mehrerer Proxys im Cluster ändern

Die genaue Bedeutung der Funktionen und Objekteigenschaften sind in der Online-Hilfe der Management Console beschrieben. Wie Sie die Online-Hilfe öffnen ist im Abschnitt „Online-Hilfe der Management Console starten“ auf Seite 159 beschrieben.

Die Bezeichnungen der Objekteigenschaften sind weitgehend identisch mit denen an der grafischen Oberfläche, zumindest aber eindeutig zuzuordnen.

9.2 MC-CLI-Anwenderskripts erstellen und aufrufen

Nach der Installation der BeanConnect Management Console steht im Unterverzeichnis `lib` des BeanConnect-Installationsverzeichnisses die jar-Datei `BeanConnectMcCli.jar`, die die Jython-Module des MC-CLI enthält. Diese jar-Datei muss beim Aufruf der MC-CLI-Funktionen im `CLASSPATH` enthalten sein.

Zusätzlich muss das Produkt Jython auf Ihrem Rechner installiert sein und das Installationsverzeichnis muss in `PATH/JYTHONPATH` angegeben sein.

Nähere Informationen zum Produkt Jython finden Sie in der BeanConnect-Freigabemitteilung.

9.2.1 Voraussetzungen für den Aufruf eines MC-CLI-Anwenderskripts

Um die Funktionen der am MC-CLI angebotenen Jython-Module nutzen zu können, müssen beim Start eines MC-CLI-Anwender-Skripts folgende Voraussetzungen erfüllt sein.

Die Skripts `startBcAdmin.cmd` (Windows) und `startBcAdmin.sh` (Linux-, Solaris-Systeme) im Unterverzeichnis `cli-sample` des Installationsverzeichnisses der Management Console (siehe Abschnitt „Anwendungsszenarien (Beispiele)“ auf Seite 429) stehen als Beispiel zur Verfügung.

- `CLASSPATH` muss die BeanConnect-jar-Dateien des `lib`-Verzeichnisses enthalten. Der `CLASSPATH` wird durch das Skript `javaenv.cmd` (Windows-Systeme) bzw. `javaenv.sh` (Linux-, Solaris-Systeme) gesetzt. Rufen Sie dieses Skript auf mit:

  `<MC_home>\bin\javaenv.cmd` (Windows) bzw.

  `<MC_home>/bin/javaenv.sh` (Unix-/Linux-Systeme)

  `<console-home>` bezeichnet das Installationsverzeichnis der Management Console.

- `JYTHONPATH` muss das Jython-Installationsverzeichnis enthalten:

  `set JYTHONPATH=<jython_home>` (Windows) bzw.
  `JYTHONPATH=<jython_home>` (Unix-/Linux-Systeme)

- PATH muss um das Jython-Installationsverzeichnis erweitert werden:

  `set PATH=<jython_home>;%PATH%` (Windows) bzw.
  `PATH=<jython_home>:$PATH` (Linux-, Solaris-Systeme)

- Im `Jython`-Kommando, das das CLI-Anwender-Skript startet, müssen Sie folgende VM-Argumente angeben:
  - `-DBEANCONNECTPATH=<beanConnect_lib>`
    `<beanConnect_lib>` ist das Verzeichnis, das die jar-Dateien von BeanConnect enthält.
  - `-DBEANCONNECT_JDK_HOME=<jdk_home>`
    `<jdk_home>` ist das JDK Installationsverzeichnis
  - `-DBEANCONNECT_USERCONS=<MC_home>`
    `<MC_home>` ist das Installationsverzeichnis der Management Console
  - `-Dlog4jCfgFile=<log4j_properties_file>`
    `<log4j_properties_file>` ist die Datei mit den log4j-Eigenschaften. Das ist in der Regel die Datei `log4j.properties.xml` im Unterverzeichnis `config` des Management Console-Installationsverzeichnisses.

9.2.2 Vorbereitung der Konfiguration

Mit den Funktionen des MC-CLI können Sie keine Proxys in die Konfiguration aufnehmen. Neu installierte lokale Proxys erkennt die Management Console beim Start und nimmt sie automatisch in die Konfiguration auf. Sie stehen somit am CLI zur Verfügung.

Entfernte Proxys müssen zuerst über die grafische Oberfläche zur Konfiguration hinzugefügt werden, damit sie über das MC-CLI konfiguriert und administriert werden können.

9.2.3 Aufbau des Anwenderskripts

Um die Funktionen der am MC-CLI angebotenen Jython-Module nutzen zu können, müssen Sie in Ihrem MC-CLI-Skript folgende Vorgaben beachten:

- Jedes MC-CLI-Skript, das Java-Klassen anspricht, muss die entsprechenden `import`-Anweisungen für die Java-Klassen des MC-CLI enthalten (siehe Abschnitt „Java-Klassen“ auf Seite 318):

```
import com.fujitsu.ts.jca.tools.mc.cli.BcDef     as BcDef
import com.fujitsu.ts.jca.tools.mc.cli.BcParameterException
                                                 as BcParameterException
import com.fujitsu.ts.jca.tools.mc.cli.BcObjectException
                                                 as BcObjectException
import com.fujitsu.ts.jca.tools.mc.cli.BcToolException
                                                  as BcToolException
```

- Für jedes MC-CLI Modul, dessen Funktionen im Skript aufgerufen werden, müssen Sie eine `import`-Anweisung `import <mc-cli-modul>` absetzen (z.B. `import BcAdminMain`).
- Zu Beginn des Skripts müssen Sie die Funktion `BcAdminMain.init()` aufrufen, die die Sitzung der Management Console startet und die Konfigurationsdatei liest. Danach können Sie andere MC-CLI-Funktionen aufrufen.
- Sind ein Proxy oder Proxy Cluster mit einem Administrations-Passwort geschützt, müssen Sie vor Aufruf einer Funktion, die den Proxy oder Proxy Cluster als Parameter verwendet, eine erfolgreiche Authentifizierung für dieses Objekt durchführen (siehe Funktion „authenticate() – Beim Proxy Cluster authentifizieren“ auf Seite 405).
- Um Änderungen wirksam zu machen, müssen die Änderungen explizit gesichert werden und in die Konfiguration der jeweiligen Komponente eingearbeitet werden. Das erfolgt mit der Funktion `perform()` und dem Parameter `action` (= "`save`", "`update-config`", "`update-ra-xml`") oder bei `BcAdminMain.close()` mit dem Parameter `save_all=True`
- Am Ende des Skripts muss die Funktion `BcAdminMain.close()` aufgerufen werden, die die Sitzung der Management Console schließt. Erst danach können wieder andere Sitzungen der Management Console mit derselben Konfigurationsdatei gestartet werden.

  Wird eine Sitzung der Management Console nicht ordnungsgemäß beendet, wird ggf. die Synchronisierungsdatei `ConsoleInUse.txt` im Installationsverzeichnis der Management Console nicht gelöscht und jeder weitere Start einer Management Console Sitzung wird abgebrochen. In diesem Fall muss diese Datei von Hand gelöscht werden.

9.2.4 Aufrufparameter angeben

Die Aufrufparameter der MC-CLI-Funktionen sind in der Regel Stellungsparameter. Einige Parameter, die optional sind, sind als Schlüsselwortparameter definiert (im Folgenden mit (kw) gekennzeichnet). Wenn beim Aufruf einer Funktion das Schlüsselwort nicht mit angegeben wird, ist die Stellung des Parameters maßgeblich.

Für die Übergabe der Parameter beim Aufruf der MC-CLI-Funktionen müssen Sie abhängig vom Parametertyp (Objekte, Eigenschaften der Objekte) zusätzlich folgende Regeln beachten:

Objekte

Am MC-CLI gelten für die Angabe der Objekte folgende Regeln:

- Bei den Funktionen `create()` und `getObject()` muss ggf. das übergeordnete `BcObject` vom Typ `BcObjectType.PROXY` oder `BcObjectType.PROXY_CLUSTER` angegeben werden.
- Die Funktionen `create()` und `getObject()` liefern als Ergebnis ein erzeugtes bzw. gelesenes Objekt `BcObject` vom Typ `BcObjectType` zurück.

  Mit diesem Objekt können dann alle weiteren Funktionen für dieses Objekt aufgerufen werden (Parameter `bc_object`).
- Die Funktion `getList()` gibt ein (Jython-) Dictionary zurück mit
  `key`

  Name des Objekts (Eigenschaft **name**)
  `value`

  Objekt vom Typ `BcObject`

  Mit jedem dieser Objekte können dann alle weiteren Funktionen für dieses Objekt aufgerufen werden (Parameter bc_object).

Eigenschaften (Properties)

Am MC-CLI gelten für die Eigenschaften (Parameter `props` bei `create()` oder `modifyProperties()`, Rückgabe bei `getProperties()`) folgende Regeln:

- Alle Eigenschaften werden in Jython-Dictionarys übergeben.
- Die Eigenschaftsnamen bestehen immer aus Kleinbuchstaben, Ziffern, Bindestrich ('-') und Punkt ('.').

- Wenn für ein Objekt an der grafischen Oberfläche der Management Console mehrere Registerkarten mit Eigenschaften existieren, wird den Namen der Eigenschaften die Bezeichnung der jeweiligen Registerkarte als Präfix vorangestellt, z.B. "`utm.`", "`timer.`". Ausnahme: Die Eigenschaften der Registerkarte **General**. Sie erhalten keinen Präfix.
- Eigenschaftsnamen für Zeitwerte enthalten als Postfix die Einheit ("`.sec`" bzw. "`.min`").
- Die Werte der Eigenschaften werden immer als Zeichenfolgen angegeben, auch bei Integer-Werten (z.B. bei Zeitwerten oder Portnummern)
- Kann eine Eigenschaft nur bestimmte Werte annehmen, sind sie in der Java-Klasse `BcDef` definiert. Genaueres ist in den einzelnen Funktionen beschrieben.
- Wird einem Objekt (Proxy, Resource Adapter, …) bei der Konfiguration über die grafische Oberfläche ein MC-CmdHandler zugordnet, werden am CLI nur die entsprechenden Eigenschaften `admin-port` und ggf. `admin-pw` angegeben.

9.3 Java-Klassen

Die Schnittstelle MC-CLI verwendet für die Aufrufparameter und -rückgaben seiner Funktionen die Java-Klassen `BcDef`, `BcObjectType`, `BcObject`. Für die Fehlerbehandlung verwendet MC-CLI Exceptions der Klassen `BcObjectException`, `BcParameterException` und `BcToolException`. Diese Java-Klassen sind im Package `com.fujitsu.ts.jca.tools.mc.cli` enthalten.

In diesem Abschnitt erhalten Sie einen kurzen Überblick über diese Klassen. Vollständig beschrieben sind die Java-Klassen in JavaDoc, die im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses abgelegt sind.

9.3.1 Klasse: BcDef

Die Klasse `BcDef` beschreibt Werte von Objekteigenschaften, Parametern und Rückgaben (action), die nur ganz bestimmte Werte annehmen können. Damit müssen die Werte nicht über vom Anwender definierte Zeichenfolgen gesetzt bzw. überprüft werden.

9.3.2 Klasse: BcObjectType

Der Typ eines Objekts der Klasse `BcObject` ist über die Klasse `BcObjectType` beschrieben. Der Wert kann über die Methode `toString()` gelesen werden. Dieser Wert ist auch Eingabewert für den Parameter `list_type` bei den Funktionen `getList()`.

Folgende Werte sind verfügbar:

Objekte	**toString()**
`BcObjectType.ACTION`	`action`
`BcObjectType.COMMUNICATION_SERVICE`	`communication-service`
`BcObjectType.EIS_PARTNER`	`eis-partner`
`BcObjectType.INBOUND_MSG_ENDPOINT`	`inbound-msg-endpoint`
`BcObjectType.INBOUND_SERVICE`	`inbound-service`
`BcObjectType.INBOUND_USER`	`inbound-user`
`BcObjectType.LU62GATEWAY`	`lu62gateway`
`BcObjectType.OUTBOUND_COMM_ENDPOINT`	`outbound-comm-endpoint`

Objekte	toString()
BcObjectType.OUTBOUND_SERVICE	outbound-service
BcObjectType.PROXY	proxy
BcObjectType.PROXY_CLUSTER	proxy-cluster
BcObjectType.RESOURCE_ADAPTER	resource-adapter
BcObjectType.TODO	todo

9.3.3 Klasse BcObject

Die Klasse `BcObject` repräsentiert Objekte der Management Console am MC-CLI.

Die Funktionen `create()` und `getObject()` liefern als Ergebnis die erzeugten bzw. gelesenen Objekte vom Typ `BcObject` zurück, bei allen weiteren Funktionen kann das anzusprechende Objekt vom Typ `BcObject` als Parameter angegeben werden.

Die Funktion `getList()` gibt ein (Jython-) Dictionary zurück, das als Wert wieder Objekte vom Typ `BcObject` enthält.

Folgende Methoden werden angeboten:

9.3.3.1 getName()

Funktion: `getName()`

Liest den Namen des Objekts

Parameter: keine

Rückgabe: (String)

Name des Objekts. Entspricht der Eigenschaft `name` des Objekts.

9.3.3.2 getObjectType()

Funktion: `getObjectType()`

Liest den Typ des Objekts

Parameter: keine

Rückgabe: (`BcObjectType`)

Typ des Objekts, siehe Abschnitt „Klasse: BcObjectType“ auf Seite 318.

9.3.4 Exceptions

Die Funktionen der MC-CLI geben in der Regel keinen Returncode zurück, der angibt, ob die Funktion erfolgreich ausgeführt worden ist. Wenn eine Funktion nicht ausgeführt werden kann, wird eine Exception aus dem Package `com.fujitsu.ts.jca.tools.mc.cli` geworfen. Diese können ggf. im Skript gefangen und geeignet behandelt werden.

Folgende Exceptions werden geworfen:

9.3.4.1 Klasse: BcObjectException

Bedeutung:

Ein für diese Funktion notwendiges Objekt kann nicht gefunden werden, oder das angegebene Objekt kann nicht verwendet werden.

mögliche Ursachen:

- Die Sitzung der Management Console ist nicht gestartet oder wurde schon geschlossen.
- Bei allen Funktionen, in denen der Parameter `proxy_object` angegeben wurde:
  - Das angegebene Proxy Objekt ist ungültig.
    Grund: falscher Objekt-Typ (BcObjectType) oder die Sitzung, unter der das Objekt erzeugt wurde, ist beendet.

    Oder:
  - Für das Proxy Objekt ist eine Authentifizierung notwendig, die noch nicht oder fehlerhaft durchgeführt wurde.
- Bei allen Funktionen, in denen der Parameter `object_name/bc_object` angegeben werden muss:
  Die Konfiguration enthält kein Objekt des gewünschten Typs mit dem angegebenen Namen bzw. es wurde ein ungültiges Objekt angegeben.

Meldungstext:

`"object not given"` bzw.

`"no object <type> <object_name> given"`

und andere.

Behebung:

Je nach Meldungstext sollte überprüft werden,

- ob eine Sitzung der Management Console gestartet ist,
- ob das Objekt zur aktuellen Sitzung gehört,
- ob für das zugehörige Proxy Objekt eine Authentifizierung durchgeführt wurde,
- ob das Objekt vom richtigen Typ ist (BcObject mit richtigem `BcObjectType`) oder
- ob das angegebene Objekt in der Konfiguration enthalten ist.

9.3.4.2 Klasse: BcParameterException

Bedeutung:

Bei Angabe von Funktionsparametern oder Objekt-Eigenschaften ist ein Fehler aufgetreten.

mögliche Ursachen:

- Die Angabe dieses Parameters oder dieser Eigenschaft ist für diese Funktion und diesen Objekttyp nicht erlaubt. D.h. diese Eigenschaft existiert nicht für diesen Objekttyp oder ist nicht änderbar.
- Der angegebene Wert ist für diesen Parameter oder die angegebene Eigenschaft nicht erlaubt (nicht numerisch, bzw. nicht im erlaubten Wertebereich), oder er ist in Kombination mit anderen Eigenschaften nicht erlaubt.

Meldungstext:

`"invalid / unchangeable property given"` bzw.

`"invalid value <value> for property <key> given"` bzw.

`"invalid parameter <key> given"`

und andere.

Behebung:

Je nach Meldungstext prüfen Sie bitte, ob die angegebene Eigenschaft für diesen Objekttyp existiert bzw. änderbar ist oder ob der angegebene Wert für diesen Parameter erlaubt ist.

9.3.4.3 Klasse: BcToolException

Bedeutung:

In einer der Basis-Klassen der Management Console trat ein Fehler auf.

mögliche Ursachen:

- Dateizugriffsfehler.
- Klassenzugriffsfehler.
- je nach Exceptiontyp weitere Fehler

Meldungstext:

Unterschiedlich je nach Exceptiontyp

Behebung:

Je nach Meldungstext prüfen Sie bitte, ob ein Zugriffsfehler auf eine Datei oder auf Java-Klassen oder ein anderer vom Anwender behebbarer Fehler vorliegt. Andernfalls wenden Sie sich bitte an den Service/Diagnosedienst.

9.4 Funktionen

9.4.1 Allgemeines

Die Schnittstelle MC-CLI besteht aus mehreren Modulen, die jeweils die Funktionen für einen Objekttyp enthalten, der über das MC-CLI konfiguriert und administriert werden kann, siehe auch Abschnitt „Übersicht über das MC-CLI“ auf Seite 310.

In diesem Abschnitt wird jedes Modul mit seinen Funktionen beschrieben. Die Beschreibung erfolgt in alphabetischer Reihenfolge.

Die genaue Bedeutung der Funktionen und Eigenschaften sind in der Online-Hilfe der Management Console beschrieben. Wie Sie die Online-Hilfe öffnen, ist im Abschnitt „Online-Hilfe der Management Console starten“ auf Seite 159 beschrieben.

9.4.1.1 Parameter

Die für die jeweilige Funktion angegebenen Parameter sind in der aufgeführten Reihenfolge anzugeben (Stellungsparameter). Die Schlüsselwortparameter sind je nach Funktion optional. Wann sie angegeben werden müssen, ist der Beschreibung zu entnehmen. Sie sind mit (kw) (= key word parameter) gekennzeichnet.

In den folgenden Funktionsbeschreibungen wird für jeden Parameter angegeben, welcher Datentyp erwartet wird (z.B. **Parameter:** `object_name` (String)).

In einigen Funktionen wird der Jython-Datentyp "Dictionary" eingesetzt. In den Parameterbeschreibungen wird der Begriff wie folgt verwendet: "Dictionary mit den key-value-Paaren..."

9.4.1.2 Eigenschaften

In den folgenden Abschnitten gibt es zu allen Modulen, die Funktionen für einen bestimmten Objekttyp enthalten, einen Unterabschnitt „Eigenschaften eines <Objekt>s“. Die darin aufgeführte Tabelle enthält alle Eigenschaften des Objekts vom genannten Objekt-Typ. Die Bedeutung der Spalten ist wie folgt:

- Die Spalte "Schlüsselwort (key) am MC-CLI" enthält die Namen, über die die Eigenschaften am MC- CLI identifiziert werden,
- die Spalte "Feldname an der grafischen Oberfläche" enthält die Eigenschaftsnamen, die an der grafischen Oberfläche und in der Online-HIlfe der Management Console verwendet werden.

- Die Spalte "Funkt." hat folgende Bedeutung:

c	(create)	Eigenschaft kann bei `create()` angegeben werden
cd	(create/default)	Eigenschaft kann bei `create()` angegeben werden. Ist sie nicht angegeben, wird ein Standard-Wert gesetzt.
cm	(create/mandatory)	Eigenschaft muss bei `create()` angegeben werden. Ist sie nicht angegeben, wird eine `BcParameterException` geworfen.
g	(get)	Eigenschaft wird bei `getProperties()` zurückgegeben
m	(modify)	Eigenschaft kann bei `modifyProperties()` geändert werden
m(s)	(modify/synchronize)	(nur bei ResourceAdapter im Cluster) Eigenschaft kann bei `modifyProperties()` geändert werden; beim Sichern werden alle Resource Adpater im Cluster synchronisiert.

- Die Spalte "Eigenschaftswert" enthält die erlaubten Werte einer Eigenschaft. Dabei bedeutet:

(String)	Zeichenfolge
(String numerisch)	Zeichenfolge, die nur aus Zahlen besteht
`BcDef.<prop>_xxx`	Werte, die in der Java-Klasse `BcDef` definiert sind und mit „`<prop>_`“ beginnen, (z.B. `BcDef.BOOL_xxx` für `BcDef.BOOL_TRUE` und `BcDef.BOOL_FALSE`)

Für die Angabe eines Eigenschaftswertes ist `None` nicht erlaubt. Die Angabe eines leeren Strings löscht den bisherigen Wert einer Eigenschaft.

9.4.1.3 Meldungen

Die Management Console erzeugt beim Ablauf vieler Aktionen Meldungen, die an der grafischen Oberfläche in einem eigenen Protokollfenster ausgegeben werden. Diese Protokollmeldungen werden am MC-CLI mit dem Präfix `MC-CLI:ProtocolMessage` (asynchron) auf `stdout` ausgegeben.

9.4.1.4 Rückgaben

Die unter **Rückgabe** beschriebenen Rückgaben werden nur bei erfolgreicher Bearbeitung der Funktion zurückgegeben.

Die Funktionen des MC-CLI geben keinen Returncode zurück, der angibt, ob die Funktion erfolgreich ausgeführt worden ist. Wenn eine Funktion nicht ausgeführt werden kann, wird eine Exception aus dem Package `com.fujitsu.ts.jca.tools.mc.cli` geworfen (siehe Abschnitt „Exceptions“ auf Seite 320). Diese können im Skript gefangen und geeignet behandelt werden.

Die mit der Funktion `perform()` gestarteten Aktionen entsprechen in der Regel den Aktionen, die an der grafischen Oberfläche einen Aktionsdialog öffnen. Der Aktionsdialog gibt eine Tabelle mit Beschreibung, Status und Ergebnisse aller Teilaktionen aus. Am MC-CLI wird als Rückgabe der Funktion `perform()` in den meisten Fällen ein Objekt der Klasse `BcObject` mit Objekttyp `BcObjectType.ACTION` zurückgegeben. Aus diesem Objekt können über die Funktionen des Moduls `BcAdminAction` diese Informationen extrahiert werden.

9.4.2 BcAdminAction

Der Modul `BcAdminAction` enthält Funktionen, die das Ergebnis eines `perform()`-Aufrufs, ein `BcObject` vom Typ `BcObjectType.ACTION`, analysieren und bestimmte Informationen zurückgeben.

`BcAdminAction` enthält die Funktionen:

- getCheckResults() – Ergebnisse von check-Aktionen anzeigen
- getResults() – Ergebnisse aller Teilaktionen einer Aktion anzeigen
- isFinishedSuccessfully() – Erfolg/Misserfolg einer Aktion anzeigen

9.4.2.1 getCheckResults() – Ergebnisse von check-Aktionen anzeigen

Funktion: `BcAdminAction.getCheckResults()`

Gibt ein Dictionary zurück, das das Ergebnis aller Check-Proxy-Container-Teilaktionen der angegebenen Art enthält.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.ACTION`)

Das Ergebnis eines `perform()`-Aufrufs.

`action_type`

Art des Checks, mögliche Angaben sind:
`BcDef.ACTION_CHECK_ADM`
`BcDef.ACTION_CHECK_AVAIL`

`object_type`

Objekttyp, für den die Check-Aktionen gelesen werden sollen; mögliche Angaben sind:
`BcObjectType.COMMUNICATION_SERVICE.toString()`
`BcObjectType.LU62GATEWAY.toString()`
`BcObjectType.PROXY.toString() (Default)`
`BcObjectType.RESOURCE_ADAPTER.toString()`

Rückgabe: Dictionary mit den key-value-Paaren Proxy Name und Check-Ergebnis (value=`result`) aller Check-Proxy-Container-Teilaktionen.

Beispiele für die Rückgabe:

```
{"BCCnt1": "Available", "BcCnt2": "Not available"}
{"BCCnt1": "Administrable", "BcCnt2": "Not administrable"}
```

Exceptions: `BcObjectException, BcToolException`

Anmerkung: Diese Funktion gibt eine Auswahl der Ergebnisse der Funktion `getResults()` zurück, nämlich die Information über die Administrierbarkeit (Angabe `BcDef.ACTION_CHECK_ADM)` bzw. den Status (Angabe `BcDef.ACTION_CHECK_AVAIL`) der Komponente des angegebenen Typs. Die Rückgabeinformationen enthalten key-value-Paare mit:

– key: Name der Komponente (object_name)
– value: Ergebnis der Teilaktion (result)

wobei `object_name` und `result` die in `getResult()` beschriebenen Eigenschaften der Teilaktion sind. Wenn die angegebene Aktion der Rückgabewert eines Check-Aufrufes für einen Proxy war, enthält das Dictionary nur ein Element. War sie der Rückgabewert eines Check-Aufrufes für einen Proxy Cluster, enthält sie ein Element für jeden Proxy des Clusters.

Beispiel:

```
...
import BcAdminAction
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
bcaction=BcAdminProxy.perform(proxy_obj,action="check-avail")
checkProxy=BcAdminAction.getCheckResults(bcaction,\
                                         BcDef.ACTION_CHECK_AVAIL)
...
```

9.4.2.2 getResults() – Ergebnisse aller Teilaktionen einer Aktion anzeigen

Funktion: `BcAdminAction.getResults()`

Gibt eine Liste von Dictionaries zurück, die die Informationen aller Teilaktionen dieser Aktion enthält.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.ACTION`)

Das Ergebnis eines `perform()`-Aufrufs.

Rückgabe: Liste von Dictionarys mit den key-value-Paaren aller Eigenschaften der Teilaktionen. Für jede Teilaktion wird ein Dictionary zurückgegeben, das die Informationen als key-value-Paare enthält. Sie entsprechen im Wesentlichen den Spalten des Aktionsdialogs an der grafischen Oberfläche, siehe Online-Hilfe der Management Console unter Layout der Management Console – Aktions-Dialogfeld – Layout. Es gibt einige zusätzliche Informationen, die die programmatische Verarbeitung erleichtern. Der Wert (value) ist immer vom Typ String, der auch leer sein kann.

Exceptions: `BcObjectException, BcToolException.`

Beispiel:

```
...
import BcAdminAction
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
bcaction=BcAdminProxy.perform(proxy_obj, "save")
iresDicn=BcAdminAction.getResults(bcaction)
...
```

Schlüsselwort (key) am MC-CLI	Spaltenname an der grafischen Oberfläche	Bedeutung
`action`	**Action**	Text der Teilaktion
`action-type`	z.T. in **Action** enthalten	Typ der Teilaktion (`BcDef.ACTION_ xxx` bzw. `BcDef.ACTION_SA_xxx`)
`details`	**Result Details**	ausführliche Informationen zum Ergebnis, wenn vorhanden
`id`	**#**	Nummer der Teilaktion
`object-name`	z.T. in **Action** enthalten	Name des Objekts, das bei dieser Teilaktion betroffen ist
`object-type`	z.T. in **Action** enthalten	Typ des Objekts (`BcObjectType.xxx.toString()`)
`result`	**Result**	Ergebnis der Teilaktion (`BcDef.ACTION_RESULT_xxx`)
`stack`	**StackTrace**	ausführliche Informationen zum Stack, falls eine Exception aufgetreten ist
`state`	**Action State**	Status der Teilaktion (`BcDef.ACTION_STATE_xxx`)

Anmerkung:

- Die wichtigsten Werte von `object-type` und `action-type` sind in `BcObjectType` und `BcDef` definiert. Es ist aber auch möglich, dass Werte zurückgegeben werden, die nicht vordefiniert sind.
- Um aus der Menge der Informationen bestimmte Ergebnisse herauszufiltern, ist es ggf. sinnvoll, eigene Funktionen zu erzeugen, die aus dem von `getResults()` zurückgegebenen Dictionary einzelne Informationen herauslesen und geeignet aufbereiten. Ein Beispiel steht mit `getCheckResults()` zur Verfügung, ein weiteres ist im Beispielskript `sampleAdminProxyCluster.py` mit der Funktion `printResults()` enthalten.
- Die Nummern `id` der Teilaktionen sind hierarchisch aufgebaut, um anzuzeigen, welche (Teil-) Aktionen zu welchen übergeordneten Aktionen gehören.

9.4.2.3 isFinishedSuccessfully() – Erfolg/Misserfolg einer Aktion anzeigen

Funktion: `BcAdminAction.isFinishedSuccessfully()`

Gibt an, ob die angestoßene Aktion erfolgreich ausgeführt werden konnte.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.ACTION`)

Das Ergebnis eines `perform()`-Aufrufs.

Rückgabe: `True`, wenn die Aktion mit allen Teilaktionen erfolgreich ausgeführt werden konnte, `False`, falls mindestens eine Teilaktion nicht erfolgreich ausgeführt werden konnte.

Exceptions: `BcObjectException`

Beispiel:

```
...
import BcAdminAction
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
...
bcaction=BcAdminProxy.perform(proxy_obj, "save")
isSucc=BcAdminAction.isFinishedSuccessfully(bcaction)
...
```

9.4.3 BcAdminCommunicationService

Der Modul `BcAdminCommunicationService` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Communication Service der Management Console angeboten werden.

`BcAdminCommunicationService` enthält die Funktionen:

- create() – Communication Service zur Konfiguration hinzufügen
- getObject() – Communication Service Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Communication Services lesen
- getProxies() – die dem Communication Service zugeordneten Proxys lesen
- modifyProperties() – Eigenschaften eines Communication Services ändern
- perform() – Administrative Aktionen starten
- remove() – Communication Service entfernen

9.4.3.1 create() – Communication Service zur Konfiguration hinzufügen

Funktion: `BcAdminCommunicationService.create()`

Es wird ein Communication Service zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `props (Dictionary)`

Dictionary mit den key-value-Paaren der Eigenschaften, die dem Communication Service zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Communication Service“ auf Seite 336.
Die Eigenschaften `host`, `install-path`, `cp-name`, `cp-network` und bei nicht lokalem Communication Service `admin-port` und `admin-pw` müssen angegeben werden. Alle anderen Eigenschaften sind optional, bzw. werden mit Standardwerten versorgt.

Rückgabe: `(BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE)`

Der neu zur Konfiguration hinzugefügte Communication Service

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
....
import BcAdminCommunicationService

...
creProps= {"host": "bchost01", "install-path":
"/opt/ibm/sna/bin", "cp-name": "bccpn101", "cp-network": "P390",
"admin-port". "31002", "admin-pw": "admin"}
newCS=BcAdminCommunicationService.create(creProps)
...
```

9.4.3.2 getObject() – Communication Service Objekt aus der Konfiguration lesen

Funktion: `BcAdminCommunicationService.getObject()`

Liest den Communication Service mit angegebenen Namen aus der Konfiguration

Parameter: `props (Dictionary)`

Dictionary mit den key-value-Paaren der Eigenschaften `host` und `install-path` des Communication Service, der gelesen werden soll. Optional kann die Eigenschaft `install-path-gw` angegeben werden, um die eindeutige Zuordnung zu einem openUTM-LU62 Gateway zu gewährleisten.

Rückgabe: `(BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE)`

Der gelesene Communication Service oder `None`, wenn kein Communication Service mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung: Die Eigenschaft `install-path-gw` sollte beim Lesen eines Communication Services angegeben werden, da einige Eigenschaften des Communication Services unter diesem Pfad gesichert werden.

Beispiel:

```
...
import BcAdminCommunicationService
...
getProps={"host": "bchost01", "install-path": "/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
...
```

9.4.3.3 getProperties() – Eigenschaften eines Communication Services lesen

Funktion: `BcAdminCommunicationService.getProperties()`

Liest alle Eigenschaften des angegebenen Communication Service und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE`)

Communication Service, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Communication Service. Mögliche Werte für `key` finden Sie in Abschnitt „Eigenschaften eines Communication Service“ auf Seite 336.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminCommunicationService
...
getProps={"host": "bchost01", "install-path": "/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
csProps=BcAdminCommunicationService.getProperties(cs_obj)
...
```

9.4.3.4 getProxies() – die dem Communication Service zugeordneten Proxys lesen

Funktion: `BcAdminCommunicationService.getProxies()`

Liest alle Proxys, denen der angegebene Communication Service zugeordnet ist und gibt ein Dictionary mit Proxy-Namen und Proxy-Objekten zurück (`bc_object` (`BcObject` vom Typ `BcObjectType.PROXY`)).

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE`)

Communication Service, dessen Proxy-Zuordnung gelesen werden soll.

Rückgabe: Dictionary mit den name-object-Paaren aller Proxys, denen der Communication Service zugeordnet ist.

Exceptions: `BcObjectException, BcToolException`

Anmerkung: Die Liste der Proxy-Namen kann auch über die Eigenschaft `proxies` gelesen werden.

Beispiel:

```
...
import BcAdminCommunicationService
...
getProps={"host": "bchost01", "install-path":"/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
proxyDicn=BcAdminCommunicationService.getProxies(cs_obj)
for proxy_name, proxy_obj in proxyDicn.iteritems():
  print "handle proxy " + proxy_name
...
```

9.4.3.5 modifyProperties() – Eigenschaften eines Communication Services ändern

Funktion: `BcAdminCommunicationService.modifyProperties()`

ändert alle Eigenschaften des angegebenen Communication Service, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE`)

Communication Service, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Communication Service“ auf Seite 336.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminCommunicationService
...
getProps={"host": "bchost01", "install-path": "/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
modProps={"desc":"modified"}
BcAdminCommunicationService.modifyProperties(cs_obj, modProps)
...
```

9.4.3.6 perform() – Administrative Aktionen starten

Funktion: `BcAdminCommunicationService.perform()`

startet für den Communication Service die angegebene Aktion.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE`)

Communication Service, für den die Aktion gestartet werden soll.

`action (String)`

Aktion, die für den angegebenen Communication Service gestartet werden soll. Mögliche Werte sind (siehe `BcDef.ACTION_xxx` in der MC-CLI-JavaDoc)

`check-adm`	überprüft die Administrierbarkeit des Communication Service.
`check-avail`	überprüft die Verfügbarkeit des Communication Service.
`save`	sichert die Änderungen, die für diesen Communication Service in der aktuellen Sitzung gemacht worden sind
`start`	startet den Communication Service
`stop`	beendet den Communication Service

Rückgabe: (`BcObject` mit Typ `BcObjectType.ACTION`):
Enthält alle Informationen über die gestartete Aktion und sämtliche Teilaktionen. Um genauere Informationen zu erhalten, kann eine Funktion des Moduls `BcAdminAction` mit diesem Objekt als Parameter aufgerufen werden

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminCommunicationSecvice
...
getProps={"host": "bchost01", "install-path": "/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
actResult=BcAdminCommunicationService.perform(cs_obj, "check-avail")
if BcAdminAction.isFinishedSuccessfully(actResult):
  resultString=BcAdminAction.getResults(actResult)[0]["result"]
  print "communication service is " +resultString
...
```

9.4.3.7 remove() – Communication Service entfernen

Funktion: `BcAdminCommunicationService.remove()`

Entfernt den angegebenen Communication Service aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.COMMUNICATION_SERVICE`).

Communication Service, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminCommunicationService
...
getProps={"host": "bchost01", "install-path": "/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
BcAdminCommunicationService.remove (cs_obj)
...
```

9.4.3.8 Eigenschaften eines Communication Service

Die folgende Tabelle enthält alle Eigenschaften eines Communication Service.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxy-Komponenten verwalten - Communication Services verwalten - Communication Services, Tabellenspalten und Communication Service Instanz, Eigenschaften.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`admin-port`	**MC-Cmdhandler Listener Port**	cm / g / m [a)]	(String numerisch)
`admin-pw`	**MC-Cmdhandler Password**	cm / m [a)]	(String)
`admin-state`	**Administrable**	g	BcDef.ACTION_RESULT_ADMINISTRABLE oder BcDef.ACTION_RESULT_NOT_ADMINISTRABLE [2]
`cp-name`	**Control Point Name**	cm / g / m [c)]	(String)
`cp-network`	**Control Point Network**	cm / g / m [c)]	(String)
`desc`	**Description**	g / m [c)]	(String)

Schlüsselwort (key) am MC-CLI [1]	Feldname an der grafischen Oberfläche [1]	Funkt.[1]	Eigenschaftswert [1]
host	**Host**	cm / g / m a)e)	(String)
idblk	**IDBLK**	cd / g / m c)	(String)
idnum	**IDNUM**	cd / g / m c)	(String)
install-path	**Installation Path**	cm / g / m	(String)
install-path-gw	**Installation Path(Lu62Gateway)**	c / g / m f)	(String)
log.audit-log	**Audit Logging / Verbose Audits**	cd / g / m b)	BcDef.LOGGING_MODE_xxx mit xxx = ON/OFF/VERBOSE [2]
log.except-log	**Activate Exception Logging**	cd / g / m b)	BcDef.LOGGING_MODE_xxx mit xxx = ON/OFF [2]
log.line-trac	**Activate Line Tracing**	cd / g / m b)	BcDef.LOGGING_MODE_xxx mit xxx = ON/OFF [2]
log.verbose-err	**Verbose Errors**	cd / g / m b)	BcDef.LOGGING_MODE_xxx mit xxx = ON/OFF [2]
mac-addr	**MAC Address**	cd / g / m c)d)	(String)
name	**Name**	g	(String)
op-system	**Operating System**	g [c)]	(String)
proxies	**Proxies**	g [g)]	(String)
type	**Type**	g	(String)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse BcDef) im Unterverzeichnis JavaDoc des Management Console-Installationsverzeichnisses

Anmerkungen zu den Indizes a) bis g) in der Tabelle:

a) Falls beim Erzeugen oder beim Modifizieren eine der Eigenschaften `host`, `admin-port` oder `admin-pw` angegeben wird, wird intern derzugehörige MC CmdHandler gesucht bzw. neu erzeugt und dem Communication Service zugeordnet. Der MC CmdHandler muss dabei verfügbar sein, sonst wird der Aufruf abgewiesen. Bei lokalem Host müssen `admin-port` und `admin-pw` nicht angegeben werden.

b) Die Eigenschaften mit Präfix `log.` sind nur bei Communication Services auf Linux oder SNAP-IX auf Solaris les- und änderbar. Da die Plattform zum Zeitpunkt der Erzeugung nicht bekannt ist, werden bei Erzeugung des Objekts die Defaultwerte gesetzt.

c) Falls der MC-CmdHandler nicht verfügbar ist, wird bei `getProperties()` für diese Eigenschaften der Wert `BcDef.VALUE_UNKNOWN (="<.>")` ausgegeben.

d) Die Eigenschaft `mac-addr` wird nur benötigt, wenn Sie mindestens einen EIS-Partner mit dem DLC-Typ LAN verwenden. Wird er nicht angegeben, wird der Defaultwert (leere Zeichenfolge) angegeben.

e) Die Eigenschaft `host` ist nur änderbar, wenn es sich um den gleichen Host handelt. (d.h. änderbar, wird aber i.d.R. abgewiesen).

f) Die Eigenschaft `install-path-gw` sollte beim Erzeugen eines Communication Services angegeben werden, da einige Eigenschaften des Communication Services unter diesem Pfad gesichert werden. Wird sie nicht angegeben, wird auf dem Host des Communication Services ein schon konfiguriertes Lu62Gateway gesucht. Wird keines gefunden, wird der Aufruf mit `BcParameterException` abgewiesen. Wird die Eigenschaft `install-path-gw` beim Lesen eines Objekts nicht angegeben, können ggf. Eigenschaften des Communication Services nicht gelesen werden.

g) Die Proxys (mit Name und Objekt) können auch über die Funktion `getProxies()` gelesen werden. Hier werden die Namen mit Komma getrennt in einer Zeichenfolge übergeben.
Achtung: Falls die Proxynamen ein Komma enthalten, ist die zurückgegebene Information evtl. nichteindeutig interpretierbar.

9.4.4 BcAdminEisPartner

Der Modul `BcAdminEisPartner` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts EIS Partner der Management Console angeboten werden.

`BcAdminEisPartner` enthält die Funktionen:

- create() – EIS Partner zur Konfiguration hinzufügen
- getGatewayPorts() - openUTM-LU62 Gateway Listener Ports des EIS Partner-Objekts lesen
- getLuNames() - Logical Unit Names des EIS Partner-Objekts lesen
- getObject() – EIS Partner-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines EIS Partners lesen
- modifyGatewayPorts() - openUTM-LU62 Gateway Listener Ports des EIS Partner-Objekts ändern
- modifyLuNames() - Logical Unit Names des EIS Partner-Objekts ändern
- modifyProperties() – Eigenschaften eines EIS Partners ändern
- perform() – Administrative Aktionen starten
- remove() – EIS Partner entfernen

9.4.4.1 create() – EIS Partner zur Konfiguration hinzufügen

Funktion: `BcAdminEisPartner.create()`

Es wird ein EIS Partner zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `object_name (String)`

Name des BeanConnect EIS Partners

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der EIS Partner zugeordnet werden soll.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

`props` (kw)

Dictionary mit den key-value-Paaren der Eigenschaften, die dem EIS Partner zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines EIS Partners“ auf Seite 347.
Die Eigenschaften `name`, `type`, `utm.hosts`, `utm.listener-port` und `utm.partner-lpap` müssen angegeben werden. Alle anderen Eigenschaften sind optional, bzw. werden mit Standardwerten versorgt.

Rückgabe: (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

Der neu zur Konfiguration hinzugefügte EIS Partner

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminEisPartner
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
creProps= {"type":BcDef.PTYPE_UTM,"desc": "myEIS",\
     "utm.hosts":"xyz","utm.partner-lpap":"LPAP",\
     "utm.listener-port": "1234"}
newEis=BcAdminEisPartner.create("TestEIS",proxy_obj,creProps)
...
```

9.4.4.2 getGatewayPorts() - openUTM-LU62 Gateway Listener Ports des EIS Partner-Objekts lesen

Funktion: `BcAdminEisPartner.getGatewayPorts()`

Liest den openUTM-LU62 Gateway Listener Port des EIS Partners bzw. die Liste der Ports für alle Proxys im Cluster, wenn der Eis-Partner im Cluster definiert ist.

Parameter: `bcobject` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

EIS Partner, dessen openUTM-LU62 Gateway Listener Ports gelesen werden sollen.

Rückgabe: Dictionary mit einem Element (oder im Proxy Cluster mehreren Elementen), wobei key der Name des Proxys und value der zugehörige openUTM-LU62 Gateway Listener Port ist.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung:
- Falls der Eis-Partner nicht vom Typ `cics` ist, wird ein `BcObjectException` geworfen.
- Die Portnummern können auch über die Eigenschaft `cs.lu62gateway-port` gelesen werden.

Beispiel:

```
...
import BcAdminEisPartner import BcAdminProxy
...
proxy_name="BCProxy"
proxy_obj=BcAdminProxy.getObject(proxy_name)
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
port_dicn=BcAdminEisPartner.getGatewayPorts(eis_obj)
gw_port=port_dicn[proxy_name]
...
```

9.4.4.3 getLuNames() - Logical Unit Names des EIS Partner-Objekts lesen

Funktion: `BcAdminEisPartner.getLuNames()`

Liest den Logical Unit Name des EIS Partners bzw. die Liste der Logical Unit Names für alle Proxys im Cluster, wenn der Eis-Partner im Cluster definiert ist.

Parameter: `bcobject` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

EIS Partner, dessen Logical Unit Name gelesen werden soll.

Rückgabe: Dictionary mit einem Element (oder im Proxy Cluster mehreren Elementen), wobei key der Name des Proxys und value der zugehörige Logical Unit Name ist.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung:
- Falls der Eis-Partner nicht vom Typ `cics` ist, wird ein `BcObjectException` geworfen.
- Die Logical Unit Namen können auch über die Eigenschaft `cs.lu-name` gelesen werden.

Beispiel:

```
...
import BcAdminEisPartner import BcAdminProxy
...
proxy_name="BCProxy"
proxy_obj=BcAdminProxy.getObject(proxy_name)
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
luname_dicn=BcAdminEisPartner.getLuNames(eis_obj)
luname=luname_dicn[proxy_name]
...
```

9.4.4.4 getObject() – EIS Partner-Objekt aus der Konfiguration lesen

Funktion: `BcAdminEisPartner.getObject()`

Liest den EIS Partner mit angegebenen Namen aus der Konfiguration

Parameter: `object_name` (string)

Name des BeanConnect EIS Partners, der gelesen werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der EIS Partner zugeordnet ist.

Rückgabe: `(BcObject` vom Typ `BcObjectType.EIS_PARTNER)`

Der gelesene EIS Partner, oder `None`, wenn kein EIS Partner mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung: Die Angabe eines `proxy_object` vom Typ `BcObjectType.PROXY` ist nur erlaubt, wenn der Proxy nicht in einem Proxy Cluster enthalten ist. In dem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

Beispiel:

```
...
import BcAdminEisPartner
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
...
```

9.4.4.5 getProperties() – Eigenschaften eines EIS Partners lesen

Funktion: `BcAdminEisPartner.getProperties()`

Liest alle Eigenschaften des angegebenen EIS Partners und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ BcObjectType.EIS_PARTNER)

EIS Partner, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des EIS Partners. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines EIS Partners“ auf Seite 347.
Für EIS Partner mit der Eigenschaft `type`="`cics`" werden keine Eigenschaften mit dem Präfix "`utm.`" ausgegeben

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminEisPartner
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
eisProps=BcAdminEisPartner.getProperties(eis_obj)
...
```

9.4.4.6 modifyGatewayPorts() - openUTM-LU62 Gateway Listener Ports des EIS Partner-Objekts ändern

Funktion: `BcAdminEisPartner.modifyGatewayPorts()`

Ändert den openUTM-LU62 Gateway Listener Port des EIS-Partners bzw. eine Liste der Ports für einen oder mehrere Proxys im Cluster, wenn der Eis-Partner im Cluster definiert ist.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

EIS Partner, dessen openUTM-LU62 Gateway Listener Port geändert werden sollen.

`ports`

Dictionary mit einem Element (oder im Proxy Cluster mehreren Elementen), wobei key (String) der Name des Proxys, value (String) der zugehörige openUTM-LU62 Gateway Listener Port ist.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung:
- Falls der Eis-Partner nicht vom Typ `cics` ist, wird ein `BcObjectException` geworfen.
- Die Portnummern können auch über die Eigenschaft `cs.lu62gateway-port` geändert werden.

Beispiel:

```
...
import BcAdminEisPartner import BcAdminProxy
...
proxy_name="BCProxy"
proxy_obj=BcAdminProxy.getObject(proxy_name)
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
port_dicn={proxy_name, "31002"}
BcAdminEisPartner.modifyGatewayPorts(eis_obj, port_dicn)
...
```

9.4.4.7 modifyLuNames() - Logical Unit Names des EIS Partner-Objekts ändern

Funktion: `BcAdminEisPartner.modifyLuNames()`

Ändert den Logical Unit Namen des EIS-Partners bzw. eine Liste der Logical Unit Namen für einen oder mehrere Proxys im Cluster, wenn der Eis-Partner im Cluster definiert ist.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

EIS Partner, dessen Logical Unit Namen geändert werden sollen.

`ports`

Dictionary mit einem oder im Proxy Cluster mehreren Elementen, wobei key (String) der Name des Proxys, value (String) der zugehörige Logical Unit Name ist.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung:
- Falls der Eis-Partner nicht vom Typ `cics` ist, wird ein `BcObjectException` geworfen.
- Die Portnummern können auch über die Eigenschaft `cs.lu-name` geändert werden.

Beispiel:

```
...
import BcAdminEisPartner import BcAdminProxy
...
proxy_name="BCProxy"
proxy_obj=BcAdminProxy.getObject(proxy_name)
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
luname_dicn={proxy_name, "CIC31"}
BcAdminEisPartner.modifyLuNames(eis_obj, luname_dicn)
...
```

9.4.4.8 modifyProperties() – Eigenschaften eines EIS Partners ändern

Funktion: `BcAdminEisPartner.modifyProperties()`

ändert alle Eigenschaften des angegebenen EIS Partners, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

EIS Partner, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines EIS Partners“ auf Seite 347.
Für EIS Partner mit der Eigenschaft `type="cics"` dürfen keine Eigenschaften mit Präfix "`utm.`" angegeben werden.
Für EIS Partner mit der Eigenschaft `type="utm"` dürfen keine Eigenschaften mit Präfix `"cics."` bzw. `"cs."` angegeben werden.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminEisPartner
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
modProps={"desc":"modified"}
BcAdminEisPartner.modifyProperties(eis_obj, modProps)
...
```

9.4.4.9 perform() – Administrative Aktionen starten

Funktion: `BcAdminEisPartner.perform()`

startet für den EIS Partner die angegebene Aktion.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`)

EIS Partner, für den die Aktion gestartet werden soll.

`action (String)`

Aktion, die für den angegebenen EIS Partner gestartet werden soll. Mögliche Werte sind (siehe `BcDef.ACTION_xxx` in der MC-CLI-JavaDoc)

`check-avail`	überprüft die Verfügbarkeit des EIS Partners.
`gen-config`	erzeugt eine Konfigurationsdatei, die in die Konfiguration des EIS Partners eingebracht werden muss.

Rückgabe: Bei `action="gen-config"`:
- `True` (boolean), wenn die Konfigurationsdateien erzeugt wurden.
- `False` (boolean), wenn die Konfigurationsdateien nicht erzeugt werden konnten.

Bei `action="check-avail"`:
- (String)
  Ausgabenachricht des Check-Services, wenn Proxy und EIS Partner gestartet sind und der Service erfolgreich aufgerufen werden konnte,
- (String)
  Meldung von MC-CLI oder dem Proxy oder dem EIS Partner, wenn der Service nicht erfolgreich aufgerufen werden konnte.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:
```
...
import BcAdminEisPartner
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
isSucc=BcAdminEisPartner.perform(eis_obj,action="gen-config")
...
```

9.4.4.10 remove() – EIS Partner entfernen

Funktion: `BcAdminEisPartner.remove()`

Entfernt den angegebenen EIS Partner aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.EIS_PARTNER`).

EIS Partner, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:
```
...
import BcAdminEisPartner
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
eis_obj=BcAdminEisPartner.getObject("testEIS",proxy_obj)
BcAdminEisPartner.remove (eis_obj)
...
```

9.4.4.11 Eigenschaften eines EIS Partners

Die folgende Tabelle enthält alle Eigenschaften eines EIS Partners.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – EIS Partners – EIS Partner bearbeiten mit den Registerkarten General, UTM Partner, Communication Services, CICS-Partner, Availability Check.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`active`	**Active**	cd / g / m	`BcDef.BOOL_xxx`[2]
`avail.char-code`	**Character Code**	cd / g / m	`BcDef.CHAR_CODE_xxx` [2]
`avail.check-service`	**Check Service**	c / g / m	`BcDef.EIS_AVAIL_NO_CHECK_SERVICE` oder (String)
`avail.message`	**Message**	c / g / m	(String)
`avail.password`	**Password**	c / m	(String)
`avail.user`	**User**	c / g / m	(String)
`cics.cp-name` [3]	**Control Point - Name**	cm / g / m	(String)
`cics.cp-network` [3]	**Control Point - Network Name**	cm / g / m	(String)
`cics.dlc-type` [3]	**DLCType**	cd / g / m	BcDef.DLC_TYPE_xxx [2]
`cics.eis-platform` [3]	**EIS Platform**	g	BcDef.PLATFORM_xxx [2]
`cics.lu-ipaddress` [3]	**Logical Unit - IP Address**	cm / g / m	(String)
`cics.lu-name` [3]	**Logical Unit - Name**	cm / g / m	(String)
`cics.lu-network` [3]	**Logical Unit - Network Name**	cm / g / m	(String)
`cics.mac-address` [3]	**MAC Address**	cm [4] / g / m	(String)
`cics.partner-type` [3]	**PartnerType**	cd / g / m	BcDef.TYPE_xxx [2]
`cics.vtam-group-name` [3]	**Vtam - Group Name**	cm / g / m	(String)
`cs.lu62gateway-port` [3] [5]	**Lu62 Gateway Listener Port**	cm / g / m	(String numerisch)

Schlüsselwort (key) am MC-CLI [1]	Feldname an der grafischen Oberfläche [1]	Funkt.[1]	Eigenschaftswert [1]
cs.lu-name [3] [5]	**Logical Unit Name**	cm / g / m	(String)
cs.mode-name [3]	**Mode Name**	cm / g / m	(String)
connections	**Connections**	cd / g / m	(String numerisch)
desc	**Description**	c / g / m	(String)
name	**Name**	cm / g / m	(String)
id	**ID**	g	(String)
prefix	**Prefix**	cd / g / m [6]	(String)
type	**Type**	cmcf / g	BcDef.PTYPE_UTM / BcDef.PTYPE_CICS
utm.access-point[7]	**Access Point**	cd / g / m	BcDef.ACCESS_POINT_xxx[2]
utm.access-point-name[7]	**Access Point Name**	c / g / m	(String)
utm.admin-permission[7]	**Admin Permission**	cd / g / m	BcDef.BOOL_xxx[2]
utm.api-mode[7]	**Application Program Interface Mode of EIS Partner**	cd / g / m	BcDef.EIS_API_MODE_xxx[2]
utm.appl-entity-qualifier[7]	**Application Entity Qualifier**	cd / g / m	(String)
utm.appl-process-title[7]	**Application Process Title**	cd / g / m	(String)
utm.hosts[7]	**Hosts**	cm / g / m	(String)
utm.mapped-hosts[7]	Mapped Hostnames	cd / g / m	(String)
utm.is-bs2000[7]	**Is BS2000**	cd / g / m	BcDef.BOOL_xxx[2]
utm.listener-port[7]	**Listener Port**	cm / g / m	(String numerisch)
utm.partner-idletimer.sec[7]	**Partner Idletimer**	cd / g / m	(String numerisch)
utm.partner-lpap[7]	**Partner LPAP**	cm / g / m	(String)
utm.proxy-auto-conn[7]	**Proxy AutoConnect**	cd / g / m	(String numerisch)
utm.proxy-cont-winners[7]	**Proxy Contention-Winners**	cd / g / m	(String numerisch)

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`utm.proxy-host`[7]	**Proxy Hostname**	cd / g / m	(String)
`utm.proxy-mapped-host`[7]	Proxy Mapped Hostname	g	(String)
`utm.proxy-idletimer.sec`[7]	**Proxy Idletimer**	cd / g / m	(String numerisch)
`utm.transport-selector`[7]	**Transport Selector**	c / g / m	(String)
`utm.transport-selector-format`[7]	**Transport Selector Format**	cd / g / m	`BcDef.TSEL_FORMAT_xxx`[2]

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323.

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse `BcDef`) im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses.

[3] Alle Eigenschaften mit dem Präfix "`cics.`" und "`cs.`" sind nur bei EIS Partnern vom Typ `cics` les- und änderbar.

[4] Die Eigenschaft `cics.mac-address` ist dann Pflichtparameter beim Erzeugen des Objektes, wenn `"cics.dlc-type"=BcDef.DLC_TYPE_LAN ("lan")` angegeben wurde.

[5] Die Eigenschaften können auch durch die Funktionen `getGatewayPorts()/getLuNames()` gelesen und mit den Funktionen `modifyGatewayPorts()/modifyLuNames()` geändert werden.

[6] Wenn die ID Nummer > 99 ist, wird kein Defaultwert für die Eigenschaft `prefix` erzeugt. Die Eigenschaft muss dann beim Erzeugen explizit angegeben werden.

[7] Alle Eigenschaften mit dem Präfix "`utm.`" sind nur bei EIS Partnern vom Typ `utm` les- und änderbar.

Anmerkungen:

- Die Eigenschaften können nicht in beliebiger Kombination geändert werden. Ggf. sind Eigenschaften nicht änderbar, wenn andere Eigenschaften bestimmte Werte haben. So sind z.B.
  - `avail.user`, `avail.password`, `avail.char-code` nicht änderbar, wenn `avail.check-service=BcDef.EIS_AVAIL_NO_CHECK_SERVICE`
  - `utm.access-point-name`, `utm.transport-selector`, `utm.transport-selector-format` nicht änderbar, wenn `utm-access-point=BcDef.ACCESS_POINT_CREATE_GENERIC`.

    Siehe dazu auch die Beschreibung (Online-Hilfe) der Management Console (z.B. für utm.api-mode).
- Die Eigenschaft **Availability Check – Perform Check** wird hier nicht angeboten, da am MC-CLI keine automatischen Checks durchgeführt werden.

- Für `cs.lu62gateway-port` und `cs.lu-name` wird bei einem Proxy Cluster eine Zeichenfolge mit einer Liste von Werten wie folgt angegeben: `<wert1>(<proxy-name1>)`, `<wert2>(<proxy-name2>)`, … Falls ein Proxy-Name die Sonderzeichen '(' oder ')' enthält, kann die Zeichenfolge nicht mehr eindeutig interpretiert werden. Dann sollten die Funktionen `get/setGatewayPorts()` und `get/setLuNames()` verwendet werden.
- Für `utm.proxy-host` wird bei einem Proxy Cluster eine Zeichenfolge mit einer Liste von Werten wie folgt angegeben: `<wert1>(<proxy-name1>)`, `<wert2>(<proxy-name2>)`, … Falls ein Proxy-Name die Sonderzeichen '(' oder ')' enthält, kann die Zeichenfolge nicht mehr eindeutig interpretiert werden.
- Für `utm.proxy-mapped-host` wird bei einem Proxy Cluster eine Zeichenfolge mit einer Liste von Werten wie folgt angegeben:
  `<mapped1>(<proxy-name1>), <mapped2>(<proxy-name2>)`, …
  Falls ein Proxy-Name die Sonderzeichen '(' oder ')' enthält, kann die Zeichenfolge nicht mehr eindeutig interpretiert werden.
- Für `utm.hosts` wird eine Zeichenfolge mit einer Liste von Werten wie folgt angegeben:
  `<utmhost-name1>, <utmhost-name2>`, …
  Für einen EIS Partner, der keine UTM Cluster-Anwendung ist, besteht die Liste nur aus einem Element.
- Für `utm.mapped-hosts` wird eine Zeichenfolge mit einer Liste von Werten wie folgt angegeben: `<mapped1>(<utmhost-name1>), <mapped2>(<utmhost-name2>)`, …
  Für einen EIS Partner, der keine UTM Cluster-Anwendung ist, besteht die Liste nur aus einem Element.

9.4.5 BcAdminInboundMsgEndpoint

Der Modul `BcAdminInboundMsgEndpoint` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Inbound Message der Management Console angeboten werden.

`BcAdminInboundMsgEndpoint` enthält die Funktionen:

- create() – Inbound Message Endpoint zur Konfiguration hinzufügen
- getObject() – Inbound Message Endpoint-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Inbound Message Endpoints lesen
- modifyProperties() – Eigenschaften eines Inbound Message Endpoints ändern
- remove() – Inbound Message Endpoint entfernen

9.4.5.1 create() – Inbound Message Endpoint zur Konfiguration hinzufügen

Funktion: `BcAdminInboundMsgEndpoint.create()`

Es wird ein Inbound Message Endpoint zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `object_name` (String)

Name des BeanConnect Inbound Message Endpoints

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` oder `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Inbound Message Endpoint zugeordnet werden soll.
Die Angabe eines `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die dem Inbound Message Endpoint zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Message Endpoints“ auf Seite 356.

Rückgabe: `(BcObject` vom Typ `BcObjectType.INBOUND_MSG_ENDPOINT)`

Der zur Konfiguration hinzugefügte Inbound Message Endpoint.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminInboundMsgEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
creProps={"desc":"my msg endpoint",\
"dial-type":BcDef.TYPE_DIALOG,\
"resource-adapter":"myRA","service-names":"TESTIS1, TESTIS2"}
me_obj=BcAdminInboundMsgEndpoint.create("myMEP", proxy_obj,\
props=creProps)
...
```

9.4.5.2 getObject() – Inbound Message Endpoint-Objekt aus der Konfiguration lesen

Funktion: `BcAdminInboundMsgEndpoint.getObject()`

Liest den Inbound Message Endpoint mit angegebenen Namen aus der Konfiguration

Parameter: `object_name` (String)

Name des BeanConnect Inbound Message Endpoint, der gelesen werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `ObjectType.PROXY_CLUSTER`)

Der Proxy bzw. Proxy Cluster, dem der Inbound Message Endpoint zugeordnet ist. Die Angabe eines `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

Rückgabe: `(BcObject` vom Typ `BcObjectType.INBOUND_MSG_ENDPOINT)`

Der gelesene Inbound Message Endpoint, oder `None`, wenn kein Inbound Message Endpoint mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundMsgEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
me_obj=BcAdminInboundMsgEndpoint.getObject("myMEP",proxy_obj)
...
```

9.4.5.3 getProperties() – Eigenschaften eines Inbound Message Endpoints lesen

Funktion: `BcAdminInboundMsgEndpoint.getProperties()`

Liest alle Eigenschaften des angegebenen Inbound Message Endpoints und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_SERVICE`)

Der Inbound Message Endpoint, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Inbound Message Endpoints. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Message Endpoints“ auf Seite 356.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminInboundMsgEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
me_obj=BcAdminInboundMsgEndpoint.getObject("myMEP",proxy_obj)
meProps=BcAdminInboundMsgEndpoint.getProperties(me_obj)
...
```

9.4.5.4 modifyProperties() – Eigenschaften eines Inbound Message Endpoints ändern

Funktion: `BcAdminInboundMsgEndpoint.modifyProperties()`

Ändert alle Eigenschaften des angegebenen Inbound Message Endpoints, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_MSG_ENDPOINT`)

Der Inbound Message Endpoint, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Message Endpoints“ auf Seite 356

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundMsgEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
me_obj=BcAdminInboundMsgEndpoint.getObject("myMEP",proxy_obj)
modProps={"desc":"created","service-names":"TESTIS1, TESTIS2"}
BcAdminInboundMsgEndpoint.modifyProperties(me_obj,modProps)
...
```

9.4.5.5 remove() – Inbound Message Endpoint entfernen

Funktion: `BcAdminInboundMsgEndpoint.remove()`

Entfernt den angegebenen Inbound Message Endpoint aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_MSG_ENDPOINT`)

Inbound Message Endpoint, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminInboundMsgEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
me_obj=BcAdminInboundMsgEndpoint.getObject("myMEP",proxy_obj)
BcAdminInboundMsgEndpoint.remove(me_obj)
...
```

9.4.5.6 Eigenschaften eines Inbound Message Endpoints

Die folgende Tabelle enthält alle Eigenschaften eines Inbound Message Endpoints.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – Inbound – Inbound Message Endpoints – Inbound Message Endpoint, Eigenschaften.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`desc`	**Description**	c / g / m	(String)
`dial-type`	**Type** (**dial** / **asyn**)	cd / g / m	`BcDef.TYPE_xxx`[2]
`dlq-msgs`	**Dead Letter Queue Messages**	g	(String numerisch)
`name`	**Name**	cm / g / m	(String)
`reply-timer.sec`	**ReplyTimer**	cd / g / m	(String numerisch)
`resource-adapter`[3]	**Resource Adapter**	cm / g / m	(String)
`service-names`	**Service Names**	cm / g / m	(String)
`state`	**State**	g	`BcDef.STATE_xxx`[2]
`ta-timer.sec`	**Transaction Timer**	cd / g / m	(String numerisch)
`waiting-msgs`	**Waiting Messages**	g	(String numerisch)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse `BcDef`) im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses

[3] Gehört der Inbound Message Endpoint zu einem Proxy Cluster, wird die Eigenschaft bei `getProperties()` nicht ausgegeben und darf bei `modifyProperties()` nicht angegeben werden.

9.4.6 BcAdminInboundService

Der Modul `BcAdminInboundService` enthält alle Funktionen, die zur Konfiguration und Administration eines Inbound Service-Objekts der Management Console angeboten werden. Objekte vom Typ Inbound Service werden implizit über die Konfiguration der Inbound Message Endpoints erzeugt und entfernt. Daher gibt es keine Funktionen `create()` und `remove()` für Inbound Service.

`BcAdminInboundService` enthält die Funktionen:

- getObject() – Inbound Service-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Inbound Services lesen
- modifyProperties() – Eigenschaften eines Inbound Services ändern

9.4.6.1 getObject() – Inbound Service-Objekt aus der Konfiguration lesen

Funktion: `BcAdminInboundService.getObject()`

Liest den Inbound Service mit angegebenen Namen aus der Konfiguration

Parameter: `object_name` (String)

Name des BeanConnect Inbound Service, der gelesen werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Inbound Service zugeordnet ist.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

Rückgabe: `(BcObject` vom Typ `BcObjectType.INBOUND_SERVICE)`

Der gelesene Inbound Service, oder `None`, wenn kein Inbound Service mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
serv_obj=BcAdminInboundService.getObject("SRV",proxy_obj)
...
```

9.4.6.2 getProperties() – Eigenschaften eines Inbound Services lesen

Funktion: `BcAdminInboundService.getProperties()`

Liest alle Eigenschaften des angegebenen Inbound Services und gibt ein Dictionary mit den key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_SERVICE`)

Inbound Service, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Inbound Services. Mögliche Werte für `key` finden Sie in Abschnitt „Eigenschaften eines Inbound Services“ auf Seite 360.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminInboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
is_obj=BcAdminInboundService.getObject("SRV",proxy_obj)
isProps=BcAdminInboundService.getProperties(is_obj)
...
```

9.4.6.3 modifyProperties() – Eigenschaften eines Inbound Services ändern

Funktion: `BcAdminInboundService.modifyProperties()`

Ändert alle Eigenschaften des angegebenen Inbound Services, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.INBOUND_SERVICE`)

Inbound Service, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Services“ auf Seite 360

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
is_obj=BcAdminInboundService.getObject("SRV",proxy_obj)
modProps{"desc":"modified"}
BcAdminInboundService.modifyProperties(is_obj,modProps)
...
```

9.4.6.4 Eigenschaften eines Inbound Services

Die folgende Tabelle enthält alle Eigenschaften eines Inbound Services.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – Inbound – Inbound Services – Inbound Service bearbeiten, General.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`desc`	**Description**	g / m	(String)
`dial-type`	**Type**	g	`BcDef.TYPE_xxx`[2]
`inbound-msg-endpoint`	**Inbound Message Endpoint**	g	(String)
`name`	**Service Name**	g	(String)
`partner-char-code`	**Partner Character Code**	g / m	`BcDef.CHAR_CODE_xxx`[2]
`partner-encoding`	**Partner Encoding**	g / m	(String)
`xatmi`	**XATMI**	g / m	`BcDef.BOOL_xxx`[2]

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse `BcDef`) im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses

9.4.7 BcAdminInboundUser

Der Modul `BcAdminInboundUser` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Inbound User der Management Console angeboten werden.

`BcAdminInboundUser` enthält die Funktionen:

- create() – Inbound User zur Konfiguration hinzufügen
- getObject() – Inbound User-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Inbound Users lesen
- modifyProperties() – Eigenschaften eines Inbound Users ändern
- remove() – Inbound User entfernen

9.4.7.1 create() – Inbound User zur Konfiguration hinzufügen

Funktion: `BcAdminInboundUser.create()`

Es wird ein Inbound User zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `object_name` (String)

Name des BeanConnect Inbound Users

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Inbound User zugeordnet werden soll. Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

`props (kw)`

Dictionary mit den key-value-Paaren der Eigenschaften, die dem Inbound User zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Users“ auf Seite 366.

Rückgabe: (`BcObject` vom Typ `BcObjectType.INBOUND_USER`)

Der zur Konfiguration hinzugefügte Inbound User.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminInboundUser
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
creProps={"desc":"created","password":"ADMIN"}
usr_obj=BcAdminInboundUser.create("USR",proxy_obj,props=creProps)
...
```

9.4.7.2 getObject() – Inbound User-Objekt aus der Konfiguration lesen

Funktion: `BcAdminInboundUser.getObject()`

Liest den angegebenen Inbound User aus der Konfiguration

Parameter: `object_name` (string)

Name des BeanConnect Inbound Users, der gelesen werden soll.

`proxy_object` (`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Inbound User zugeordnet ist.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nur erlaubt, wenn der Proxy nicht in einem Proxy Cluster enthalten ist. In dem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

Rückgabe: `(BcObject` vom Typ `BcObjectType.INBOUND_USER)`

Der gelesene Inbound User, oder `None`, wenn kein Inbound User mit angegebenen Namen existiert

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundUser
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
user_obj =BcAdminInboundUser.getObject("USR",proxy_obj)
...
```

9.4.7.3 getProperties() – Eigenschaften eines Inbound Users lesen

Funktion: `BcAdminInboundUser.getProperties()`

Liest alle Eigenschaften des angegebenen Inbound Users und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_USER`)

Inbound User, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit key-value-Paaren der Eigenschaften des Inbound Users. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Users“ auf Seite 366.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminInboundUser
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
user_obj=BcAdminInboundUser.getObject("USR",proxy_obj)
userProps=BcAdminInboundUser.getProperties(user_obj)
...
```

9.4.7.4 modifyProperties() – Eigenschaften eines Inbound Users ändern

Funktion: `BcAdminInboundUser.modifyProperties()`

ändert alle Eigenschaften des angegebenen Inbound User, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_USER`)

Inbound User, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Inbound Users“ auf Seite 366

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundUser
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
user_obj=BcAdminInboundUser.getObject("USR",proxy_obj)
modProps={"desc":"modified"}
BcAdminInboundUser.modifyProperties(user_obj,modProps)
...
```

9.4.7.5 remove() – Inbound User entfernen

Funktion: `BcAdminInboundUser.remove()`

Entfernt den angegebenen Inbound User aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.INBOUND_USER`)

Inbound User, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminInboundUser
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
user_obj=BcAdminInboundUser.getObject("USR",proxy_obj)
BcAdminInboundUser.remove (user_obj)
...
```

9.4.7.6 Eigenschaften eines Inbound Users

Die folgende Tabelle enthält alle Eigenschaften eines Inbound Users.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – Inbound – Inbound Users – Inbound User bearbeiten, General.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`desc`	**Description**	c / g / m	(String)
`name`	**Name**	cm / g	(String)
`password`	**Password**	c / m	(String)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

9.4.8 BcAdminLu62Gateway

Der Modul `BcAdminLu62Gateway` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts openUTM-LU62 Gateway der Management Console angeboten werden.

`BcAdminLu62Gateway` enthält die Funktionen:

- create() – openUTM-LU62 Gateway zur Konfiguration hinzufügen
- getObject() – openUTM-LU62 Gateway Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines openUTM-LU62 Gateways lesen
- getProxies() – die dem openUTM-LU62 Gateway zugeordneten Proxys lesen
- modifyProperties() – Eigenschaften eines openUTM-LU62 Gateways ändern
- perform() – Administrative Aktionen starten
- remove() – openUTM-LU62 Gateway entfernen

9.4.8.1 create() – openUTM-LU62 Gateway zur Konfiguration hinzufügen

Funktion: `BcAdminLu62Gateway.create()`

Es wird ein openUTM-LU62 Gateway zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `props (Dictionary)`

Dictionary mit den key-value-Paaren der Eigenschaften, die dem openUTM-LU62 Gateway zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines openUTM-LU62 Gateways“ auf Seite 373.
Die Eigenschaften `host`, `install-path` und bei nicht lokalem openUTM-LU62 Gateway `admin-port` und `admin-pw` müssen angegeben werden. Alle anderen Eigenschaften sind optional, bzw. werden mit Standardwerten versorgt.

Rückgabe: `(BcObject` vom Typ `BcObjectType.LU62GATEWAY)`

Der neu zur Konfiguration hinzugefügte openUTM-LU62 Gateway

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
....
import BcAdminLu62Gateway
...
creProps={"host": "bchost01", "install-path": "/opt/lib/utmlu62",
  "trave-level": BcDef.LOGGING_MODE_2,
  "admin-port". "31002", "admin-pw": "admin"}
gw_obj=BcAdminLu62Gateway.create(creProps)
...
```

9.4.8.2 getObject() – openUTM-LU62 Gateway Objekt aus der Konfiguration lesen

Funktion: `BcAdminLu62Gateway.getObject()`

Liest den openUTM-LU62 Gateway mit angegebenen Namen aus der Konfiguration

Parameter: `props (Dictionary)`

Dictionary mit den key-value-Paaren der Eigenschaften `host` und `install-path` des openUTM-LU62 Gateway, der gelesen werden soll. Optional kann die Eigenschaft `install-path-gw` angegeben werden, um die eindeutige Zuordnung zu einem Communication Service zu gewährleisten.

Rückgabe: (`BcObject` vom Typ `BcObjectType.LU62GATEWAY`)

Der gelesene openUTM-LU62 Gateway oder `None`, wenn kein openUTM-LU62 Gateway mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminCommunicationService
...
getProps={"host": "bchost01", "install-path": "/opt/ibm/sna/bin"}
cs_obj=BcAdminCommunicationService.getObject(getProps)
...
```

9.4.8.3 getProperties() – Eigenschaften eines openUTM-LU62 Gateways lesen

Funktion: `BcAdminLu62Gateway.getProperties()`

Liest alle Eigenschaften des angegebenen openUTM-LU62 Gateway und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.LU62GATEWAY`)

openUTM-LU62 Gateway, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des openUTM-LU62 Gateway. Mögliche Werte für `key` finden Sie in Abschnitt „Eigenschaften eines openUTM-LU62 Gateways“ auf Seite 373.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminLu62Gateway
...
getProps={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(getProps)
gwProps=BcAdminLu62Gateway.getProperties(gw_obj)
...
```

9.4.8.4 getProxies() – die dem openUTM-LU62 Gateway zugeordneten Proxys lesen

Funktion: `BcAdminLu62Gateway.getProxies()`

Liest alle Proxys, denen der angegebene openUTM-LU62 Gateway zugeordnet ist und gibt ein Dictionary mit Proxy-Namen und Proxy-Objekten zurück (`bc_object` (`BcObject` vom Typ `BcObjectType.PROXY`)).

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.LU62GATEWAY`)

openUTM-LU62 Gateway, dessen Proxy-Zuordnung gelesen werden soll.

Rückgabe: Dictionary mit den name-object-Paaren aller Proxys, denen der openUTM-LU62 Gateway zugeordnet ist.

Exceptions: `BcObjectException, BcToolException`

Anmerkung: Die Liste der Proxy-Namen kann auch über die Eigenschaft `proxies` gelesen werden.

Beispiel:

```
...
import BcAdminLu62Gateway
...
getProps={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(getProps)
proxyDicn=BcAdminLu62Gateway.getProxies(gw_obj)
for proxy_name, proxy_obj in proxyDicn.iteritems():
  print "handle proxy " + proxy_name
...
```

9.4.8.5 modifyProperties() – Eigenschaften eines openUTM-LU62 Gateways ändern

Funktion: `BcAdminLu62Gateway.modifyProperties()`

ändert alle Eigenschaften des angegebenen openUTM-LU62 Gateways, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.LU62GATEWAY`)

openUTM-LU62 Gateway, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines openUTM-LU62 Gateways“ auf Seite 373.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminLu62Gateway
...
getProps={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(getProps)
modProps={"desc":"modified"}
BcAdminLu62Gateway.modifyProperties(gw_obj, modProps)
...
```

9.4.8.6 perform() – Administrative Aktionen starten

Funktion: `BcAdminLu62Gateway.perform()`

startet für den openUTM-LU62 Gateway die angegebene Aktion.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.LU62GATEWAY`)

openUTM-LU62 Gateway, für den die Aktion gestartet werden soll.

`action (String)`

Aktion, die für den angegebenen openUTM-LU62 Gateway gestartet werden soll. Mögliche Werte sind (siehe `BcDef.ACTION_xxx` in der MC-CLI-JavaDoc)

`check-adm`	überprüft die Administrierbarkeit des openUTM-LU62 Gateways.
`check-avail`	überprüft die Verfügbarkeit des openUTM-LU62 Gateways.
`save`	sichert die Änderungen, die für diesen openUTM-LU62 Gateway in der aktuellen Sitzung gemacht worden sind
`start`	startet den openUTM-LU62 Gateway
`stop`	beendet den openUTM-LU62 Gateway

Rückgabe: (`BcObject` mit Typ `BcObjectType.ACTION`):
Enthält alle Informationen über die gestartete Aktion und sämtliche Teilaktionen. Um genauere Informationen zu erhalten, kann eine Funktion des Moduls `BcAdminAction` mit diesem Objekt als Parameter aufgerufen werden.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminLu62Gateway
...
getProps={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(getProps)
actResult=BcAdminLu62Gateway.perform(gw_obj, "check-avail")
if BcAdminAction.isFinishedSuccessfully(actResult):
  resultString=BcAdminAction.getResults(actResult)[0]["result"]
  print "communication service is " +resultString
...
```

9.4.8.7 remove() – openUTM-LU62 Gateway entfernen

Funktion: `BcAdminLu62Gateway.remove()`

Entfernt den angegebenen openUTM-LU62 Gateway aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.LU62GATEWAY`).

openUTM-LU62 Gateway, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminLu62Gateway
...
getProps={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(getProps)
BcAdminLu62Gateway.remove (gw_obj)
...
```

9.4.8.8 Eigenschaften eines openUTM-LU62 Gateways

Die folgende Tabelle enthält alle Eigenschaften eines openUTM-LU62 Gateways.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxy-Komponenten verwalten - openUTM-LU62 Gateways verwalten - openUTM-LU62 Gateway, Tabellenspalten und openUTM-LU62 Gateway Instanz, Eigenschaften..

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
admin-port	**MC-Cmdhandler Listener Port**	cm / g / m a)	(String numerisch)
admin-pw	**MC-Cmdhandler Password**	cm / m [a)]	(String)
admin-state	**Administrable**	g	BcDef.ACTION_RESULT_ADMINISTRABLE oder BcDef.ACTION_RESULT_NOT_ADMINISTRABLE [2]
desc	**Description**	g / m [b)]	(String)
host	**Host**	cm / g / m a)c)	(String)
install-path	**Installation Path**	cm / g / m	(String)
install-path-cs	**Installation Path (Communication Service)**	c / g / m f)	(String)
name	**Name**	g [d)]	(String)
op-system	**Operating System**	g [b)]	(String)
proxies	**Proxies**	g [e)]	(String)
trave-level	**Trace Level**	cd / g / m b)	BcDef.LOGGING_MODE_xxx mit xxx = OFF/1/2/3 [2]
xaptp-trace	**XAP-TP Trace**	cd / g / m b)	BcDef.LOGGING_MODE_xxx mit xxx = ON/OFF [2]

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse BcDef) im Unterverzeichnis JavaDoc des Management Console-Installationsverzeichnisses

Anmerkungen zu den Indizes a) bis f) in der Tabelle:

a) Falls beim Erzeugen oder beim Modifizieren eine der Eigenschaften `host`, `admin-port` oder `admin-pw` angegeben wird, wird intern der zugehörige MC CmdHandler gesucht bzw. neu erzeugt und dem openUTM-LU62 Gateway zugeordnet. Der MC CmdHandler muss dabei verfügbar sein, sonst wird der Aufruf abgewiesen. Bei lokalem Host müssen `admin-port` und `admin-pw` nicht angegeben werden.

b) Falls der MC-CmdHandler nicht verfügbar ist, wird bei `getProperties()` für diese Eigenschaften der Wert `BcDef.VALUE_UNKNOWN (="<.>")` ausgegeben.

c) Die Eigenschaft `host` ist nur änderbar, wenn es sich um den gleichen Host handelt. (d.h. änderbar, wird aber i.d.R. abgewiesen).

d) Der Name setzt sich (generisch) aus dem Host-Namen und dem Installations-Pfad zusammen:"`<host> [<install-path>]`"

e) Die Proxys (mit Name und Objekt) können auch über die Funktion `getProxies()` gelesen werden. Hier werden die Namen mit Komma getrennt in einer Zeichenfolge übergeben.
Achtung: Falls die Proxynamen ein Komma enthalten, ist die zurückgegebene Information evtl. nicht eindeutig interpretierbar.

f) Die Eigenschaft `install-path-cs` sollte beim Erzeugen eines openUTM-LU62 Gateways angegeben werden, da es ohne die Verknüpfung mit einem Communication Service Fehler bei der Administration des openUTM-LU62 Gateways geben kann. Ist unter dem angegebenen Pfad noch kein Communication Service konfiguriert, wird ein neues Objekt erzeugt, dass z.T. Dummy-Werte enthält, die für die Nutzung des Communication Services angepasst werden müssen.Wird die Eigenschaft nicht angegeben, wird auf dem Host des openUTM-LU62 Gateways ein schon konfigurierter Communication Service gesucht und dem openUTM-LU62 Gateway zugeordnet.

9.4.9 BcAdminMain

Mit den Funktionen des Moduls `BcAdminMain` können Sie eine Sitzung der Management Console für das MC-CLI starten und beenden sowie die Liste aller Proxys, Proxy Cluster und Todo-Topics der aktuellen BeanConnect-Konfiguration ausgeben lassen.

`BcAdminMain` enthält die Funktionen:

- close() – Management Console-Sitzung beenden
- getList() – Liste aller konfigurierten Objekte eines Objekt-Typs ausgeben
- getVersion() – Version der Management Console auslesen
- init() – Sitzung der Management Console für das MC-CLI starten

9.4.9.1 close() – Management Console-Sitzung beenden

Funktion: `BcAdminMain.close()`

Beendet die Sitzung der Management Console und speichert abhängig vom eingegeben Parameter die noch nicht gesicherten Änderungen.

Parameter: `save_all` (boolean)

`True`	Alle Änderungen sollen gesichert werden.
`False`	Alle Änderungen sollen verworfen werden (Standard).

Rückgabe: keine

Anmerkung:
- Der Aufruf `close()` muss am Ende der Funktionsaufrufe an die Management Console erfolgen, um das Arbeiten von anderen Sitzungen der Management Console zuzulassen. (Serialisierungsende).
- Alle Java-Objekte vom Typ `BcObject`, die in der aktuellen MC-CLI-Sitzung erzeugt wurden, werden mit dem Aufruf von `close()` ungültig und können nicht mehr (z.B. als Parameter) verwendet werden. In einer neuen MC-CLI-Sitzung müssen Sie diese Objekte ggf. neu erzeugen.

Beispiel:

```
...
import BcAdminMain
...

BcAdminMain.close(True)
...
```

9.4.9.2 getList() – Liste aller konfigurierten Objekte eines Objekt-Typs ausgeben

Funktion: `BcAdminMain.getlist()`

Liest die Namen aller Objekte eines Typs aus der aktuellen BeanConnect-Konfiguration und liefert ein Dictionary mit Namen und Objekten zurück.

Parameter: `list_type` (String)

Objekt-Typ, von dem eine Liste aller definierten Objekte der aktuellen Konfiguration erstellt werden soll. Für `list_type` können Sie Folgendes angeben (siehe auch Abschnitt „Klasse: BcObjectType“ auf Seite 318)

String	oder Wert einer der folgenden `toString()`-Methoden
„communication-service"	`BcObjectType.COMMUNICATION_SERVICE.toString()`
"lu62gateway"	`BcObjectType.LU62GATEWAY.toString()`
"proxy"	`BcObjectType.PROXY.toString()`
"proxy-cluster"	`BcObjectType.PROXY_CLUSTER.toString()`
"todo"	`BcObjectType.TODO.toString()`

Rückgabe: Dictionary mit allen Objekten des angegebenen Typs und deren Namen als key.

Exceptions: `BcParameterException, BcObjectException, BcToolException.`

Beispiel:

```
...
import BcAdminMain
...
''' get list of all proxys configured '''
bcproxys=BcAdminMain.getList("proxy")
...
```

Die Liste der Proxys (`list_type= "proxy`") enthält nur einzelne (stand-alone) Proxys, d.h. keine Proxys, die zu einem Proxy Cluster gehören. Die Proxys, die in einem Proxy Cluster enthalten sind, können Sie mit der `getList()`-Funktion des Moduls `BcAdminProxyCluster` lesen.

9.4.9.3 getVersion() – Version der Management Console auslesen

Funktion: `BcAdminMain.getVersion()`

Liest die Version der Management Console und liefert sie als String zurück.

Parameter: keine

Rückgabe: String mit der Version der Management Console

Exceptions: `BcObjectException`

Beispiel:

```
...
import BcAdminMain
...
admver=BcAdminMain.getVersion()
...
```

9.4.9.4 init() – Sitzung der Management Console für das MC-CLI starten

Funktion: `BcAdminMain.init()`

Startet eine Sitzung der Management Console für die Administration der BeanConnect-Konfiguration über das MC-CLI. Im Unterverzeichnis `config` des angegebenen Console-Unterverzeichnisses `console_home` von BeanConnect wird die Konfigurationsdatei `console.properties.xml` gelesen. Ist sie noch nicht vorhanden, wird die Datei `console.properties.xml` erzeugt.

Parameter: `console_home` (String)

Installationsverzeichnis der Management Console.

`log_mc_msgs (BcDef.BOOL_TRUE / BcDef.BOOL_FALSE)`

`BcDef.BOOL_TRUE` (Default):
Die Meldungen, die die Management Console asynchron erzeugt, werden auf stdout ausgegeben.
`BcDef.BOOL_FALSE`:
Die Meldungen, die die Management Console asynchron erzeugt, werden nicht ausgegeben.

Rückgabe: Keine

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Anmerkung: Ursache für eine geworfene `BcToolException` kann eine bereits existierende andere Sitzung der Management Console sein (gestartet mit der grafischen Oberfläche oder über das MC-CLI). Beenden Sie diese.
Existiert keine andere Sitzung, müssen Sie ggf. die Serialisierungsdatei `Console-InUse.txt` im Home-Verzeichnis der Management Console löschen.

Beispiel:

```
...
import sys
import BcAdminMain
...
consoleHome=sys.argv[1]
BcAdminMain.init(consoleHome)
...
```

9.4.10 BcAdminOutboundCommEndpoint

Der Modul `BcAdminOutboundCommEndpoint` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Outbound Communication Endpoint der Management Console angeboten werden.

`BcAdminOutboundCommEndpoint` enthält die Funktionen:

- create() – Outbound Communication Endpoint zur Konfiguration hinzufügen
- getObject() – Outbound Communication Endpoint-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Outbound Communication Endpoint lesen
- modifyProperties() – Eigenschaften eines Outbound Communication Endpoints ändern
- remove() – Outbound Communication Endpoint entfernen

9.4.10.1 create() – Outbound Communication Endpoint zur Konfiguration hinzufügen

Funktion: `BcAdminOutboundCommEndpoint.create()`

Es wird ein Outbound Communication Endpoint zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `object_name`

Name des BeanConnect Outbound Communication Endpoint

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Outbound Communication Endpoint zugeordnet werden soll.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

`props` (kw)

Dictionary mit key-value-Paaren der Eigenschaften, die dem Outbound Communication Endpoint zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Outbound Communication Endpoints“ auf Seite 384.

Rückgabe: `(BcObject` vom Typ `BcObjectType.OUTBOUND_COMM_ENDPOINT)`

Der neu zur Konfiguration hinzugefügte Outbound Communication Endpoint.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundCommEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
crPr={"desc":"created","eis-partner":"EP","partner-service":"TOS"}
ce_obj=BcAdminOutboundCommEndpoint.create("CEND",\
                                          proxy_obj,props=crPr)
...
```

9.4.10.2 getObject() – Outbound Communication Endpoint-Objekt aus der Konfiguration lesen

Funktion: `BcAdminOutboundCommEndpoint.getObject()`

Liest den Outbound Communication Endpoint mit angegebenen Namen aus der Konfiguration

Parameter: `object_name` (String)

Name des BeanConnect Outbound Communication Endpoint, der gelesen werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Outbound Communication Endpoint zugeordnet ist.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

Rückgabe: `(BcObject` vom Typ `BcObjectType.OUTBOUND_COMM_ENDPOINT)`

Der gelesene Outbound Communication Endpoint, oder `None`, wenn kein Outbound Communication Endpoint mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundCommEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ce_obj=BcAdminOutboundCommEndpoint.getObject("CEND",proxy_obj)
...
```

9.4.10.3 getProperties() – Eigenschaften eines Outbound Communication Endpoint lesen

Funktion: `BcAdminOutboundCommEndpoint.getProperties()`

Liest alle Eigenschaften des angegebenen Outbound Communication Endpoint und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.OUTBOUND_COMM_ENDPOINT`)

Der Outbound Communication Endpoint., dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit key-value-Paaren der Eigenschaften des Outbound Communication Endpoints. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Outbound Communication Endpoints“ auf Seite 384.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundCommEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ce_obj=BcAdminOutboundCommEndpoint.getObject("CEND",proxy_obj)
ceProps=BcAdminOutboundCommEndpoint.getProperties(ce_obj)
...
```

9.4.10.4 modifyProperties() – Eigenschaften eines Outbound Communication Endpoints ändern

Funktion: `BcAdminOutboundCommEndpoint.modifyProperties()`

Ändert alle Eigenschaften des angegebenen Outbound Communication Endpoints, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.OUTBOUND_COMM_ENDPOINT`)

Outbound Communication Endpoint, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Outbound Communication Endpoints“ auf Seite 384

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundCommEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ce_obj=BcAdminOutboundCommEndpoint.getObject("CEND",proxy_obj)
modPr={"desc":"modified"}
BcAdminOutboundCommEndpoint.modifyProperties(ce_obj,modPr)
...
```

9.4.10.5 remove() – Outbound Communication Endpoint entfernen

Funktion: `BcAdminOutboundCommEndpoint.remove()`

Entfernt den angegebenen Outbound Communication Endpoint aus der Konfiguration.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.OUTBOUND_COMM_ENDPOINT`)

Der Outbound Communication Endpoint, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundCommEndpoint
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ce_obj=BcAdminOutboundCommEndpoint.getObject("CEND",proxy_obj)
BcAdminOutboundCommEndpoint.remove(ce_obj)
...
```

9.4.10.6 Eigenschaften eines Outbound Communication Endpoints

Die folgende Tabelle enthält alle Eigenschaften eines Outbound Communication Endpoints.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – Outbound – Outbound Communication Endpoints – Outbound Communication Endpoints, Eigenschaften.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`desc`	**Description**	c / g / m	(String)
`eis-partner`	**EIS Partner**	cm / g / m	(String)
`name`	**Name**	cm / g / m	(String)
`partner-service`	**Partner Service**	cm / g / m	(String)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

9.4.11 BcAdminOutboundService

Der Modul `BcAdminOutboundService` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Outbound Service der Management Console angeboten werden.

`BcAdminOutboundService` enthält die Funktionen:

- create() – Outbound Service zur Konfiguration hinzugefügen
- getObject() – Outbound Service-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Outbound Services lesen
- modifyProperties() – Eigenschaften eines Outbound Services ändern
- remove() – Outbound Service entfernen

9.4.11.1 create() – Outbound Service zur Konfiguration hinzugefügen

Funktion: `BcAdminOutboundService.create()`

Es wird ein Outbound Service zur Konfiguration hinzugefügt, dessen Eigenschaften Sie in einem Dictionary am MC-CLI übergeben müssen.

Parameter: `object_name`

Name des BeanConnect Outbound Service.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Outbound Service zugeordnet werden soll. Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

`props` (kw)

Dictionary mit den key-value-Paaren der Eigenschaften, die dem Outbound Service zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Outbound Services“ auf Seite 390.

Rückgabe: `(BcObject` vom Typ `BcObjectType.OUTBOUND_SERVICE)`

Der neu zur Konfiguration hinzugefügte Outbound Service.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
crPr={"desc":"created","dial-type":BcDef.TYPE DIALOG,\
                                        "reply-timer.sec": "90"}
os_obj=BcAdminOutboundService.create("OSRV",proxy_obj,props=crPr)
...
```

9.4.11.2 getObject() – Outbound Service-Objekt aus der Konfiguration lesen

Funktion: `BcAdminOutboundService.getObject()`

Liest den angegebenen Outbound Service aus der Konfiguration

Parameter: `object_name`

Name des BeanConnect Outbound Service, der gelesen werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Outbound Service zugeordnet ist.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nicht erlaubt, wenn der Proxy zu einem Proxy Cluster gehört. In diesem Fall muss hier als Parameter das Proxy Cluster Objekt angegeben werden.

Rückgabe: `(BcObject` vom Typ `BcObjectType.OUTBOUND_SERVICE)`

Der gelesene Outbound Service, oder `None`, wenn kein Outbound Service mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
os_obj=BcAdminOutboundService.getObject("OSRV",proxy_obj)
...
```

9.4.11.3 getProperties() – Eigenschaften eines Outbound Services lesen

Funktion: `BcAdminOutboundService.getProperties()`

Liest alle Eigenschaften des angegebenen Outbound Services und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.OUTBOUND_SERVICE`)

Outbound Service, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit key-value-Paaren der Eigenschaften des Outbound Services. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Outbound Services“ auf Seite 390.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
os_obj=BcAdminOutboundService.getObject("OSRV",proxy_obj)
os_props=BcAdminOutboundService.getProperties(os_obj)
...
```

9.4.11.4 modifyProperties() – Eigenschaften eines Outbound Services ändern

Funktion: `BcAdminOutboundService.modifyProperties()`

Ändert alle Eigenschaften des angegebenen Outbound Services, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.OUTBOUND_SERVICE`)

Der Outbound Service, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Outbound Services“ auf Seite 390

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
os_obj=BcAdminOutboundService.getObject("OSRV",proxy_obj)
modProps={"reply-timer.sec": "180"}
BcAdminOutboundService.modifyProperties(os_obj,modProps)
...
```

9.4.11.5 remove() – Outbound Service entfernen

Funktion: `BcAdminOutboundService.remove()`

Entfernt den angegebenen Outbound Service aus der Konfiguration.

Parameter: `bc_object`
(`BcObject` vom Typ `BcObjectType.OUTBOUND_SERVICE`)

Der Outbound Service, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminOutboundService
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
os_obj=BcAdminOutboundService.getObject("OSRV",proxy_obj)
BcAdminOutboundService.remove (os_obj)
...
```

9.4.11.6 Eigenschaften eines Outbound Services

Die folgende Tabelle enthält alle Eigenschaften eines Outbound Services.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – Outbound – Outbound Services – Outbound Service bearbeiten, General.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
`desc`	**Description**	c / g / m	(String)
`dial-type`	**Type**	cd / g	`BcDef.TYPE_xxx`[2]
`name`	**Partner Service Name**	cm / g	(String)
`reply-timer.sec`	**Reply Timer (sec)**	cd / g / m	(String numerisch)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse `BcDef`) im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses

9.4.12 BcAdminProxy

Das Modul `BcAdminProxy` enthält alle Funktionen, die zur Konfiguration und Administration eines Proxys der Management Console angeboten werden.

Das Modul enthält keine Funktion create(), mit der Sie neue Proxys in die Konfiguration aufnehmen können. Neu installierte lokale Proxys erkennt die Management Console beim Start und nimmt sie automatisch in die Konfiguration auf. Entfernte Proxys müssen zuerst über die grafische Oberfläche zur Konfiguration hinzugefügt werden.

`BcAdminProxy` enthält die Funktionen:

- authenticate() – Beim Proxy authentifizieren
- getAssignment() - das dem Proxy zugeordnete openUTM-LU62 Gateway oder Communication Service lesen
- getList() – Alle im Proxy enthaltenen Objekte eines Objekt-Typs auflisten
- getObject() – Proxy Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Proxys lesen
- modifyProperties() – Eigenschaften eines Proxys ändern
- perform() – Administrative Aktionen für einen Proxy starten
- remove() – Proxy aus der Konfiguration entfernen
- setAssignment() - dem Proxy ein openUTM-LU62 Gateway oder einen Communication Service zuordnen

9.4.12.1 authenticate() – Beim Proxy authentifizieren

Funktion: `BcAdminProxy.authenticate()`

Mit der Funktion authentifizieren Sie sich für die Administration und Konfiguration des angegebenen Proxys.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY)`

Proxy Objekt, bei dem Sie sich authentifizieren wollen.

`password (String)`

Administrationspasswort des Proxys

Rückgabe: `True`

wenn die Authentifizierung erfolgreich war

`BcParameterException`

sonst

Exceptions: `BcObjectException, BcParameterException`

Anmerkung:
- Wenn für ein `BcObject` des Typs `BcObjectType.PROXY` keine erfolgreiche Authentifizierung durchgeführt worden ist, kann keine andere Funktion ausgeführt werden, bei der dieses Objekt als Parameter angegeben wird. Das gilt auch für die Funktionen anderer Module, z.B. `BcAdminInboundUser.create()`.
- Wenn Sie das Administrations-Passwort des Proxys an der grafischen Oberfläche speichern, ist die Authentifizierung nicht notwendig (siehe Online-Hilfe der Management Console: BeanConnect Proxys zur Management Console hinzufügen – Proxy hinzufügen – Proxy Eigenschaften, General: Management Console Access, Admin User Password, Use).

Beispiel:

```
...
import BcAdminProxy
...
BcAdminProxy.authenticate(proxy_obj, admin_pw)
...
```

9.4.12.2 getAssignment() - das dem Proxy zugeordnete openUTM-LU62 Gateway oder Communication Service lesen

Funktion: `BcAdminProxy.getAssignment()`

Liest die CICS-Komponente (openUTM-LU62 Gateway bzw. Communication Service), die dem angegebenen Proxy zugeordnet ist, und gibt ein Dictionary mit dem Element (Name, Object) zurück.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY`)

Proxy-Objekt, dessen CICS-Komponente gelesen werden soll.

`comp_type (String)`

Typ der Komponente: `BcObjectType.LU62GATEWAY.toString()` oder `BcObjectType.COMMUNICATION_SERVICE.toString()`.

Rückgabe: Dictionary mit einem Element, das die gelesene Komponente enthält (Name und `BcObject` vom Typ `BcObjectType.LU62GATEWAY` oder `BcObjectType.COMMUNICATION_SERVICE`).

Exceptions: `BcObjectException, BcParameterException`

Beispiel:

```
...
import BcAdminLu62Gateway
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
gw_dicn=BcAdminProxy.getAssignment(proxy_obj, "lu62gateway")
gw_name=gw_dicn.keys()[0]
gw_obj=gw_dicn[gw_name]
...
```

9.4.12.3 getList() – Alle im Proxy enthaltenen Objekte eines Objekt-Typs auflisten

Funktion: `BcAdminProxy.getList()`

Liest alle Objekte eines bestimmten Typs, die dem Proxy zugeordnet sind, und gibt ein Dictionary mit Namen und Objekten dieses Typs zurück.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY)`

Proxy, dessen Objekte des Typs `list_type` aufgelistet werden sollen.

`list_type` (String)

Objekt-Typ, von dem eine Liste aller Objekte der aktuellen Konfiguration erstellt werden soll. Folgende Werte können für `list_type` angegeben werden (siehe auch Abschnitt „Klasse: BcObjectType“ auf Seite 318):

String	oder Wert einer der folgenden `toString()`-Methoden
`"eis-partner"`	`BcObjectType.EIS_PARTNER.toString()`
`"inbound-msg-endpoint"`	`BcObjectType.INBOUND_MSG_ENDPOINT.toString()`
`"inbound-service"`	`BcObjectType.INBOUND_SERVICE.toString()`
`"inbound-user"`	`BcObjectType.INBOUND_USER.toString()`
`"outbound-comm-endpoint"`	`BcObjectType.OUTBOUND_COMM_ENDPOINT.toString()`
`"outbound-service"`	`BcObjectType.OUTBOUND_SERVICE.toString()`
`"resource-adapter"`	`BcObjectType.RESOURCE_ADAPTER.toString()`

Rückgabe: Dictionary mit allen Objekten dieses Typs, die diesem Proxy zugeordnet sind, und deren Namen als key.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Anmerkung: Wenn der angegebene Proxy zu einem Proxy Cluster gehört, können Sie die im Proxy konfigurierten Objekte nicht mit `BcAdminProxy.getList()` lesen. Sie müssen über `BcAdminProxyCluster.getList()` gelesen werden.

Beispiel:

```
...
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("proxy")
bcDicn=BcAdminProxy.getList(proxy_obj,"eis-partner")
...
```

9.4.12.4 getObject() – Proxy Objekt aus der Konfiguration lesen

Funktion: `BcAdminProxy.getObject()`

Liest den angegebenen Proxy aus der Konfiguration

Parameter: `proxy_name (String)`

Name des BeanConnect-Proxys, der gelesen werden soll.

Rückgabe: `(BcObject` vom Typ `BcObjectType.PROXY)`

Der gelesenen Proxy, oder `None`, wenn kein Proxy mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcToolException`

Anmerkung: Um das zurückgegebene Objekt als Eingabeparameter für weitere Funktionen nutzen zu können, müssen Sie zunächst eine Authentifizierung durchführen (siehe Funktion „authenticate() – Beim Proxy authentifizieren“ auf Seite 392).

Beispiel:

```
...
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
...
```

9.4.12.5 getProperties() – Eigenschaften eines Proxys lesen

Funktion: `BcAdminProxy.getProperties()`

Liest alle Eigenschaften des angegebenen Proxys und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY)`

Proxy, dessen Eigenschaften Sie lesen wollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Proxys (siehe Abschnitt „Eigenschaften eines Proxys“ auf Seite 400).

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminMain
import BcAdminProxy
...
bcproxies=BcAdminMain.getList("proxy")
proxy_obj=bcproxies["BCProxy"]
proxyProps=BcAdminProxy.getProperties(proxy_obj)
...
```

9.4.12.6 modifyProperties() – Eigenschaften eines Proxys ändern

Funktion: `BcAdminProxy.modifyProperties()`

ändert alle Eigenschaften des angegebenen Proxys, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY)`

Proxy, dessen Eigenschaften Sie ändern wollen.

`props` (Dictionary)

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Proxys“ auf Seite 400

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
modProps={"timer.shutdown-time.min": "2"}
BcAdminProxy.modifyProperties(proxy_obj,modProps)
...
```

9.4.12.7 perform() – Administrative Aktionen für einen Proxy starten

Funktion: `BcAdminProxy.perform()`

startet für den Proxy die in `action` angegebene Aktion.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY)`

Proxy, der administriert werden soll.

`action (String)`

Aktion, die für den angegebenen Proxy gestartet werden soll. Mögliche Werte sind (siehe `BcDef.ACTION_xxx` in der MC-CLI-JavaDoc)

`check-adm`	überprüft den Administrationsstatus des Proxys.
`check-avail`	überprüft Verfügbarkeit des Proxys. Es wird zunächst der Proxy Container geprüft.Ist er verfügbar und danach die Komponenten und Kommunikationspartner des Proxys geprüft.
`restart`	beendet und startet den Proxy neu (Restart).
`save`	sichert die Änderungen, die für diesen Proxy in der aktuellen Sitzung gemacht worden sind.
`start`	startet den Proxy.

`stop`	beendet den Proxy.
`update-config`	bringt die gesicherten Änderungen in die Konfiguration des Proxys ein.

`params (kw)`

Dictionary mit den key-value-Paaren der Parameter, die der angegebenen Aktion mitgegeben werden sollen.
Falls `action=check-avail`, kann im Dictionary `params` folgendes key-value-Paar angegeben werden:

`key="all-components“`

`value=BcDef.BOOL_TRUE`
wenn für alle Komponenten des Proxys die Verfügbarkeit überprüft werden soll.
`value=BcDef.BOOL_FALSE`
wenn nur für den Proxy Container die Verfügbarkeit überprüft werden soll (Standard).

Falls `action=start`, `stop`, `restart` oder `update-config`, können im Dictionary params folgende key-value-Paare angegeben werden:

key =	value =
`all-components`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE`
`container`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE`
`communication-service`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE`
`lu62gateway`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE / BcDef.COLDSTART` (nur bei `action=start` möglich)

Dabei bedeuten:

`key=all-components`
Gibt an, ob die Aktion für alle Komponenten durchgeführt werden soll.

`key=container`
Gibt an, ob die Aktion für den Container des Proxys durchgeführt werden soll.

`key=communication-service`
Gibt an, ob die Aktion für den Communication Service des Proxys durchgeführt werden soll.

`key=lu62gateway`
Gibt an, ob die Aktion für das openUTM-LU62 Gateway des Proxys durchgeführt werden soll.

`value=BcDef.BOOL_TRUE`
Die angegebene Aktion soll für diese Komponente des Proxys durchgeführt werden.

`value=BcDef.BOOL_FALSE`
Die angegebene Aktion soll für diese Komponente des Proxys nicht durchgeführt werden (Standard).

value=BcDef.COLDSTART
Die angegebene Aktion soll für diese Komponente des Proxys als Kaltstart durchgeführt werden (nur bei action=start und key=lu62gateway zulässig).

Rückgabe: (BcObject mit Typ BcObjectType.ACTION)

Enthält alle Informationen über die gestartete Aktion und sämtliche Teilaktionen. Um genauere Informationen zu erhalten, kann eine Funktion des Moduls BcAdminAction mit diesem Objekt als Parameter aufgerufen werden

Exceptions: BcParameterException, BcObjectException, BcToolException

Anmerkung:
- Wenn das Proxy Objekt zu einem Proxy Cluster gehört, kann die Aktion "save" nicht mit BcAdminProxy.perform() angestossen werden. Das Sichern der geänderten Eigenschaften erfolgt direkt im Aufruf von modifyProperties() bzw. über BcAdminProxyCluster.perform().
- Wenn das Proxy Objekt zu einem Proxy Cluster gehört, kann die Aktion "updateconfig" nicht mit BcAdminProxy.perform() angestossen werden. Sie muss über BcAdminProxyCluster.perform() angestossen werden.
- Wenn bei den Aktionen start/stop/restart in params das key-value-Paar "all-components", BcDef.BOOL_TRUE angegeben wird, werden andere key-value-Paare ignoriert.

Beispiel:

```
...
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("proxy")
bcaction=BcAdminProxy.perform(proxy_obj,"check-avail")
...
```

9.4.12.8 remove() – Proxy aus der Konfiguration entfernen

Funktion: BcAdminProxy.remove()

Entfernt den angegebenen Proxy aus der Konfiguration.

Parameter: bc_object (BcObject vom Typ BcObjectType.PROXY)

Proxy Objekt, das aus der Konfiguration entfernt werden soll.

Rückgabe: Keine

Exceptions: BcObjectException, BcParameterException.

Beispiel:

```
...
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
BcAdminProxy.remove(proxy_obj)
..
```

9.4.12.9 setAssignment() - dem Proxy ein openUTM-LU62 Gateway oder einen Communication Service zuordnen

Funktion: `BcAdminProxy.setAssignment()`

Ordnet dem angegebenen Proxy eine CICS-Komponente zu (openUTM-LU62 Gateway bzw. Communication Service).

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY`)

Proxy-Objekt, dessen CICS-Komponente zugeordnet werden soll.

`comp_type (String)`

Typ der Komponente: `BcObjectType.LU62GATEWAY.toString()` oder `BcObjectType.COMMUNICATION_SERVICE.toString()`.

`comp_obj` (`BcObject` vom Typ `BcObjectType.LU62GATEWAY` oder `BcObjectType.COMMUNICATION_SERVICE`)

openUTM-LU62 Gateway Objekt oder Communication Service Objekt, das dem Proxy zugeordnet werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException`

Beispiel:

```
...
import BcAdminLu62Gateway
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
gw_props={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(gw_props)
gw_dicn=BcAdminProxy.setAssignment(proxy_obj, "lu62gateway", gw_obj)
...
```

9.4.12.10 Eigenschaften eines Proxys

Die folgende Tabelle enthält alle Eigenschaften eines Proxys.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys verwalten – Proxy verwalten über Kontextmenü – Eigenschaften eines Proxys bearbeiten
und unter:
BeanConnect Proxys zur Management Console hinzufügen – Expertenmodus – Proxys hinzufügen, Timer Settings/ Performance Settings.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
admin-port	**MC-CmdHandler Listener Port**	g/ m	(String numerisch)
api-mode	**Application Program Interface Mode/API Mode**	g/ m [c)]	BcDef.API_MODE_xxx [2]
appl-process-title	**Container Application Process Title**	g/ m	(String)
cluster.cluster-name	(Knoten im Navigationsbaum über Proxy)	g [b)]	(String)
cluster.ip-address	**IP-Address**	g [b)]	(String)
cluster.is-administrable	**Administrable**	g [b)]	BcDef.ADM_STATE_xxx [2]
cluster.is-master	**Master**	g [b)]	BcDef.BOOL_xxx [2]
dir	**Container Directory**	g	(String)
host	**Host**	g [2]	(String)
id	**ID**	g	(String)
mapped-host	**Mapped Hostname**	g [f)]	(String)
name	**Name**	g/ m	(String)
partner-type	**Possible EIS Partner Type**	g/ m [c)]	BcDef.PTYPE_xxx [2]
performance. modify-start-par	**Modify Standard Start Parameter**	g/ m [c)]	BcDef.BOOL_xxx [2]

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
performance.nbr-par -inb-rfc1006-conns	**Number of Parallel Connections: InboundRfc1006**	g/ m [c)]	(String numerisch)
performance.nbr-par -inb-sock-conns	**Number of Parallel Connections: InboundSocket**	g/ m [c)]	(String numerisch)
performance.nbr-par -inb-upic-conns	**Number of Parallel Connections: InboundUpic**	g/m [c)]	(String numerisch)
performance.nbr-par -outb-conns	**Number of Parallel Connections: Outbound**	g [c)]	(String numerisch)
performance. proxy-cache-size	**Proxy Container Storage Area Size: Cache**	g/ m [c)]	(String numerisch)
performance.proxy-mode	**Proxy Container Mode F(ast)/S(ecure)**	g/ m [c)]	BcDef.APPLI_MODE _xxx [2]
performance. proxy-nbr-asyn-tasks	**Number of Proxy Container Processes: asynTasks**	g/ m [c)]	(String numerisch)
performance. proxy-nbr-tasks	**Number of Proxy Container Processes: tasks**	g/ m [c)]	(String numerisch)
performance. proxy-pagepool-size	**Proxy Container Storage Area Size: Pagepool**	g/ m [c)]	(String numerisch)
port	**ContainerPort**	g	(String numerisch)
run-user-id	**ContainerRunUserID**	g/ m [e)]	(String)
start-as-win-service	**Start as a service**	g/ m [d)]	BcDef.BOOL_xxx [2]
timer.asyn-conf.sec	**Asynch. Confirmation**	g/ m [c)]	(String numerisch)
timer.eis-partner -reconnect.min	**EisPartnerReconnect**	g/ m [c)]	(String numerisch)
timer.prep-to-commit.sec	**Prepare To Commit**	g/ m [c)]	(String numerisch)
timer.ra-check	**RA Check Interval**	g/ m [c)]	(String numerisch)
timer.ra-connect-time.sec	**RA Connection Time**	g/ m [c)]	(String numerisch)

Schlüsselwort (key) am MC-CLI [1]	Feldname an der grafischen Oberfläche [1]	Funkt.[1]	Eigenschaftswert [1]
timer.shutdown-time.min	**ShutdownTime**	g/ m [c)]	(String numerisch)
timer.ta-comm.sec	**Transaction Communi-**	g/ m [c)]	(String numerisch)
win-service-name	**Windows Service Name**	g [d)]	(String)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse BcDef) im Unterverzeichnis JavaDoc des Management Console-Installationsverzeichnisses

Anmerkungen (allgemein):

- Passwort und Sicherheitslevel können nur über die grafische Oberfläche der Management Console geändert werden (z.B. ist die Eigenschaft **Admin User Password** am MC-CLI nicht lesbar und nicht änderbar). Für einen sicheren Zugang zur Konfiguration ist es sinnvoll, das Passwort nicht dauerhaft zu speichern, sondern bei jedem Zugriff auf ein Proxy Objekt anzugeben. Das erfolgt über die Funktion authenticate().
- Die Eigenschaft **General/Automatic Availability Check – Time Interval (sec)** wird im MC-CLI nicht angeboten, da im MC-CLI keine automatischen Checks durchgeführt werden.

Anmerkungen zu den Indizes in der Tabelle

a) Die Eigenschaft host ist nicht änderbar.

b) Nur wenn das Proxy Objekt zu einem Proxy Cluster gehört, werden die Eigenschaften mit dem Präfix "cluster." ausgegeben.

c) Wenn das Proxy Objekt zu einem Proxy Cluster gehört, können die Eigenschaften mit dem Präfix "timer." und "performance." und die Eigenschaften partner-type, mapped-host und api-mode nicht mit BcAdminProxy.modifyProperties() geändert werden. Sie müssen über BcAdminProxyCluster.modifyProperties()geändert werden.

d) Bei Proxys auf Unix-/Linux-Systemen werden die Eigenschaften start-as-win-service und win-service-name nicht angeboten (nicht lesbar, nicht änderbar).

e) Nach Neuaufnahme eines Proxys bzw. Neuinstallation muss die Eigenschaft run-user-id gesetzt werden. Andernfalls kann die Funktion perform() nicht ausgeführt werden.

f) Wenn das Proxy Objekt zu einem Proxy Cluster gehört, kann die Eigenschaft mapped-host nicht geändert werden.

9.4.13 BcAdminProxyCluster

Der Modul `BcAdminProxyCluster` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Proxy Cluster der Management Console angeboten werden.

`BcAdminProxyCluster` enthält die Funktionen:

- addProxy() – Proxy zum Proxy Cluster hinzufügen
- authenticate() – Beim Proxy Cluster authentifizieren
- create() – Proxy Cluster zur Konfiguration hinzufügen
- getAssignment() - das dem Proxy Cluster zugeordnete openUTM-LU62 Gateway oder Communication Service lesen
- getList() – Alle Objekte eines Typs im Proxy Cluster auflisten
- getMasterProxy() – Master-Proxy eines Proxy Clusters lesen
- getObject() – Proxy Cluster Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Proxy Clusters lesen
- modifyProperties() – Eigenschaften eines Proxy Clusters ändern
- perform() – Administrative Aktionen starten
- remove() – Proxy Cluster entfernen
- removeProxy() – Proxy aus Proxy Cluster entfernen
- setAssignment() - dem Proxy Cluster ein openUTM-LU62 Gateway oder einen Communication Service zuordnen
- setAssignment() - dem Proxy Cluster ein openUTM-LU62 Gateway oder einen Communication Service zuordnen

9.4.13.1 addProxy() – Proxy zum Proxy Cluster hinzufügen

Funktion: `BcAdminProxyCluster.addProxy()`

Dem angegebenen Cluster einen stand-alone Proxy hinzufügen.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, der erweitert werden soll.

`proxy_object` (`BcObject` vom Typ `BcObjectType.PROXY`).

Proxy, der zum Cluster hinzugefügt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
import BcAdminProxy
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
proxy_obj=BcAdminProxy.getObject("BCProxy")
proxyList=BcAdminProxyCluster.addProxy(clstr_obj,proxy_obj)
...
```

9.4.13.2 authenticate() – Beim Proxy Cluster authentifizieren

Funktion: `BcAdminProxyCluster.authenticate()`

Mit der Funktion können Sie sich für die Administration und Konfiguration beim angegebenen Proxy Cluster authentifizieren.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, bei dem Sie sich authentifizieren wollen.

`password` (String)

Administrationspasswort des Proxy Clusters

Rückgabe: `True`, wenn die Authentifizierung erfolgreich war,
`BcParameterException` sonst

Exceptions: `BcParameterException, BcObjectException`

Anmerkung:
- Für den Proxy Cluster ist das Admin User Password des Master Proxys anzugeben.
- Wenn für ein `BcObject` des Typs `BcObjectType.PROXY_CLUSTER` keine erfolgreiche Authentifizierung durchgeführt worden ist, kann keine andere Funktion ausgeführt werden, bei der dieses Objekt als Parameter angegeben wird. Das gilt auch für die Funktionen anderer Module, z.B. BcAdminInboundUser.create().
- Wenn Sie das Administrations-Passwort des Master-Proxys im Proxy Cluster speichern, ist die Authentifizierung nicht notwendig (siehe auch Online-Hilfe der Management Console: BeanConnect Proxys zur Management Console hinzufügen – Proxy hinzufügen – Proxy Eigenschaften, General: Management Console Access, Admin User Password, Use).

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
clstr_aut=BcAdminProxyCluster.authenticate(clstr_obj,"BCpass")
...
```

9.4.13.3 create() – Proxy Cluster zur Konfiguration hinzufügen

Funktion: `BcAdminProxyCluster.create()`

Fügt einen neuen Proxy Cluster zur Konfiguration hinzu, der den angegebenen Proxy als Master-Proxy enthält.

Parameter: `object_name` (String)

Name des neuen BeanConnect Proxy Clusters

`proxy_object` (`BcObject` vom Typ `BcObjectType.PROXY`)

Proxy, der Master Proxy des Proxy Cluster werden soll.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die dem Proxy Cluster zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Proxy Clusters“ auf Seite 416.

Rückgabe: (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`)

Der neu zur Konfiguration hinzugefügte Proxy Cluster.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Anmerkung: Um das zurückgegebene Objekt als Eingabeparameter für weitere Funktionen nutzen zu können, muss zuerst eine Authentifizierung durchgeführt werden (Funktion `authenticate()`).

Beispiel:

```
...
import BcAdminProxyCluster
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
clstr_obj=BcAdminProxyCluster.create("BCCluster",proxy_obj)
...
```

9.4.13.4 getAssignment() - das dem Proxy Cluster zugeordnete openUTM-LU62 Gateway oder Communication Service lesen

Funktion: `BcAdminProxyCluster.getAssignment()`

Liest die CICS-Komponente (openUTM-LU62 Gateway bzw. Communication Service), die dem angegebenen Proxy Cluster zugeordnet ist, und gibt ein Dictionary mit dem Element (Name, Object) zurück.

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`)

Proxy Cluster Objekt, dessen CICS-Komponente gelesen werden soll.

`comp_type (String)`

Typ der Komponente: `BcObjectType.LU62GATEWAY.toString()` oder `BcObjectType.COMMUNICATION_SERVICE.toString()`.

Rückgabe: Dictionary mit einem Element, das die gelesene Komponente enthält (Name und `BcObject` vom Typ `BcObjectType.LU62GATEWAY` oder `BcObjectType.COMMUNICATION_SERVICE`).

Exceptions: `BcObjectException, BcParameterException`

Anmerkung: Es wird die jeweilige Komponente des Master Proxys gelesen. Es ist möglich, dass anderen Proxys in diesem Cluster andere Komponenten zugeordnet sind. Diese können über die Proxy-Funktion `getAssignment()` in `BcAdminProxy` gelesen werden.

Beispiel:

```
...
import BcAdminLu62Gateway
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
gw_dicn=BcAdminProxyCluster.getAssignment(clstr_obj, "lu62gateway")
gw_name=gw_dicn.keys()[0]
gw_obj=gw_dicn[gw_name]
...
```

9.4.13.5 getList() – Alle Objekte eines Typs im Proxy Cluster auflisten

Funktion: `BcAdminProxyCluster.getList()`

Liest alle Objekte eines bestimmten Typs, die dem Proxy Cluster zugeordnet sind, und gibt ein Dictionary mit Namen und Objekten dieses Typs zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, für den die Objekte des angegebene Typs aufgelistet werden sollen.

`list_type (String)`

Objekt-Typ, von dem eine Liste aller Objekte der aktuellen Konfiguration erstellt werden soll. Folgende Werte können Sie für `list_type` angeben (siehe auch Abschnitt „Klasse: BcObjectType“ auf Seite 318):

String	oder Wert einer der folgenden `toString()`-Methoden
`"eis-partner"`	`BcObjectType.EIS_PARTNER.toString()`
`"inbound-msg-endpoint"`	`BcObjectType.INBOUND_MSG_ENDPOINT.toString()`
`"inbound-service"`	`BcObjectType.INBOUND_SERVICE.toString()`
`"inbound-user"`	`BcObjectType.INBOUND_USER.toString()`
`"outbound-comm-endpoint"`	`BcObjectType.OUTBOUND_COMM_ENDPOINT.toString()`
`"outbound-service"`	`BcObjectType.OUTBOUND_SERVICE.toString()`
`"proxy"`	`BcObjectType.PROXY.toString()`
`"resource-adapter"`	`BcObjectType.RESOURCE_ADAPTER.toString()`

Rückgabe: Dictionary mit allen Objekten dieses Typs, die diesem Proxy Cluster zugeordnet sind, und deren Namen als key.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
proxyList=BcAdminProxyCluster.getList(clstr_obj,"proxy")
...
```

9.4.13.6 getMasterProxy() – Master-Proxy eines Proxy Clusters lesen

Funktion: `BcAdminProxyCluster.getMasterProxy()`

Gibt den Master-Proxy des Proxy Clusters zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`)

Proxy Cluster, dessen Master-Proxy gelesen werden soll.

Rückgabe: `(BcObject` vom Typ `BcObjectType.PROXY`).

Master Proxy des Proxy Clusters

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
m_proxy_obj=BcAdminProxyCluster.getMasterProxy(clstr_obj)
...
```

9.4.13.7 getObject() – Proxy Cluster Objekt aus der Konfiguration lesen

Funktion: `BcAdminProxyCluster.getObject()`

Liest den Proxy Cluster mit angegebenen Namen aus der Konfiguration

Parameter: `cluster_name` (String)

Name des BeanConnect Proxy Clusters, der gelesen werden soll.

Rückgabe: (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

der gelesene Proxy Cluster, oder `None`, wenn kein Proxy Cluster mit entsprechendem Namen existiert

Exceptions: `BcObjectException, BcToolException`

Anmerkung: Um das zurückgegebene Objekt als Aufrufparameter für weitere Funktionen nutzen zu können, muss zuerst eine Authentifizierung durchgeführt werden (Funktion `authenticate`).

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
...
```

9.4.13.8 getProperties() – Eigenschaften eines Proxy Clusters lesen

Funktion: `BcAdminProxyCluster.getProperties()`

Liest alle Eigenschaften des angegebenen Proxy Clusters und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Proxy Clusters“ auf Seite 416

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Proxy Clusters.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
clusterProps=BcAdminProxyCluster.getProperties(clstr_obj)
...
```

9.4.13.9 modifyProperties() – Eigenschaften eines Proxy Clusters ändern

Funktion: `BcAdminProxyCluster.modifyProperties()`

Ändert die Eigenschaften des angegebenen Proxy Clusters. Die neuen Werte der zu ändernden Eigenschaften müssen Sie in einem Dictionary an die Funktion übergeben.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Proxy Clusters“ auf Seite 416

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
...
modProps={"timer.shutdown-time.min": "2"}
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
BcAdminProxyCluster.modifyProperties(clstr_obj,modProps)
...
```

9.4.13.10 perform() – Administrative Aktionen starten

Funktion: `BcAdminProxyCluster.perform()`

startet für den Proxy Cluster eine bestimmte Aktion.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, für den die Aktion gestartet werden soll.

`action (String)`

Aktion, die für den angegebenen Proxy gestartet werden soll. Mögliche Werte sind (siehe `BcDef.ACTION_xxx` in der MC-CLI-JavaDoc)

`check-adm`	überprüft den Administrationsstatus des Proxy Clusters.
`check-avail`	überprüft den Lauf-Status des Proxy Clusters.
`restart`	beendet und startet den Proxy Cluster neu.
`save`	sichert die Änderungen, die für diesen Proxy Cluster in der aktuellen Sitzung gemacht worden sind.
`start`	startet den Proxy Cluster.
`stop`	beendet den Proxy Cluster.
`update-config`	bringt die gesicherten Änderungen in die Konfiguration des Proxy Clusters ein.

`params (kw)`

Dictionary mit den key-value-Paaren der Parameter, die der angegebenen Aktion mitgegeben werden sollen.

Falls `action=start`, `stop`, `restart` oder `update-config`, können im Dictionary params folgende key-value-Paare angegeben werden:

key =	value =
`all-components`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE`
`container`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE`
`communication-service`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE`
`force-start`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE` (nur bei `action=restart` möglich)
`lu62gateway`	`BcDef.BOOL_TRUE / BcDef.BOOL_FALSE / BcDef.COLDSTART` (nur bei `action=start` möglich)

Dabei bedeuten:

`key=all-components`
Gibt an, ob die Aktion für alle Komponenten durchgeführt werden soll.

`key=container`
Gibt an, ob die Aktion für den Container des Proxy Clusters durchgeführt werden soll.

`key=communication-service`
Gibt an, ob die Aktion für den Communication Service des Proxy Clusters durchgeführt werden soll.

`key=force-start`
gibt an, ob die Komponenten des Proxy Clusters, die aktuell nicht laufen, auch gestartet werden sollen.

`key=lu62gateway`
Gibt an, ob die Aktion für das openUTM-LU62 Gateway des Proxy Clusters durchgeführt werden soll.

`value=BcDef.BOOL_TRUE`
Die angegebene Aktion soll für diese Komponente des Proxy Clusters durchgeführt werden.

`value=BcDef.BOOL_FALSE`
Die angegebene Aktion soll für diese Komponente des Proxy Clusters nicht durchgeführt werden (Standard).

`value=BcDef.COLDSTART`
Die angegebene Aktion soll für diese Komponente des Proxy Clusters als Kaltstart durchgeführt werden (nur bei `action=start` und `key=lu62gateway` zulässig).

Rückgabe: (`BcObject` mit Typ `BcObjectType.ACTION`).

- Enthält alle Informationen über die gestartete Aktion und sämtliche Teilaktionen. Um genauere Informationen zu erhalten, kann eine Funktion des Moduls `BcAdminAction` mit diesem Objekt als Parameter aufgerufen werden.
- `None`, wenn die Aktion nicht gestartet wurde (z.B. wenn bei `save` nichts zu sichern ist).

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Anmerkung:
- Bei `action=start`, `stop`, `restart` muss ggf. vorher eine Authentifikation für alle Proxys durchgeführt werden, da sonst die Aktion für die Proxys nicht gestartet werden kann.
- Wenn bei den Aktionen `start/stop/restart/update-config` in `params` das key- value-Paar `all-components`, `BcDef.BOOL_TRUE` angegeben wird, werden andere key-value-Paare ignoriert.

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
bcaction=BcAdminProxyCluster.perform(clstr_obj,"stop")
...
```

9.4.13.11 remove() – Proxy Cluster entfernen

Funktion: `BcAdminProxyCluster.remove()`

Entfernt den angegebenen Proxy Cluster aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
BcAdminProxyCluster.remove(clstr_obj)
...
```

9.4.13.12 removeProxy() – Proxy aus Proxy Cluster entfernen

Funktion: `BcAdminProxyCluster.removeProxy()`

Den angegebenen Proxy aus dem Proxy Cluster entfernen und in die Liste der einzelnen (stand-alone) Proxys stellen.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`).

Proxy Cluster, aus dem der Proxy entfernt werden soll.

`proxy_object` (`BcObject` vom Typ `BcObjectType.PROXY`).

Proxy, der aus dem Cluster entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Anmerkung:
- Ist der angegebene Proxy der Master-Proxy und enthält der Proxy Cluster noch andere Proxys, wird der Aufruf der Funktion abgewiesen.
- Beim Entfernen des letzten Proxys aus dem Cluster wird automatisch der Proxy Cluster entfernt.

Beispiel:

```
...
import BcAdminProxyCluster
import BcAdminProxy
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
proxy_dicn=BcAdminProxyCluster.getList(clstr_obj, "proxy")
proxy_obj=proxy_dicn["BCProxy"]
BcAdminProxyCluster.removeProxy(clstr_obj,proxy_obj)
...
```

9.4.13.13 setAssignment() - dem Proxy Cluster ein openUTM-LU62 Gateway oder einen Communication Service zuordnen

Funktion: `BcAdminProxyCluster.setAssignment()`

Ordnet dem angegebenen Proxy Cluster eine CICS-Komponente zu (openUTM-LU62 Gateway bzw. Communication Service).

Parameter: `bc_object (BcObject` vom Typ `BcObjectType.PROXY_CLUSTER)`

Proxy Cluster Objekt, dessen CICS-Komponente zugeordnet werden soll.

`comp_type (String)`

Typ der Komponente: `BcObjectType.LU62GATEWAY.toString()` oder `BcObjectType.COMMUNICATION_SERVICE.toString()`.

`comp_obj (BcObject` vom Typ `BcObjectType.LU62GATEWAY` oder `BcObjectType.COMMUNICATION_SERVICE)`

openUTM-LU62 Gateway Objekt oder Communication Service Objekt, das dem Proxy Cluster zugeordnet werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException`

Anmerkung: Bei Zuordnung einer CICS-Komponente zu einem Proxy Cluster wird die CICS-Komponente implizit dem Master Proxy und allen weiteren Proxys im Cluster zugeordnet, denen keine CICS-Komponente dieses Typs zugeordnet ist. Für alle Proxys im Cluster, denen explizit eine andere CICS-Komponente dieses Typs zugeordnet ist, bleibt diese Zuordnung erhalten.

Beispiel:

```
...
import BcAdminLu62Gateway
import BcAdminProxyCluster
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
gw_props={"host": "bchost01", "install-path": "/opt/lib/utmlu62"}
gw_obj=BcAdminLu62Gateway.getObject(gw_props)
gw_dicn=BcAdminProxy.setAssignment(clstr_obj, "lu62gateway", gw_obj)
...
```

9.4.13.14 setMasterProxy() – Master-Proxy eines Proxy Clusters wechseln

Funktion: `BcAdminProxyCluster.setMasterProxy()`

Setzt einen anderen Proxy als Master-Proxy des Proxy Cluster.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.PROXY_CLUSTER`)

Proxy Cluster, in dem der Proxy neuer Master-Proxy werden soll.

`proxy_object` (`BcObject` vom Typ `BcObjectType.PROXY`).

Proxy, der neuer Master-Proxy des Proxy Clusters werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminProxyCluster
import BcAdminProxy
...
clstr_obj=BcAdminProxyCluster.getObject("BCCluster")
proxy_dicn=BcAdminProxyCluster.getList(clstr_obj, "proxy")
new_mproxy_obj=proxy_dicn["BCProxy"]
BcAdminProxyCluster.setMasterProxy(clstr_obj, new_mproxy_obj)
...
```

9.4.13.15 Eigenschaften eines Proxy Clusters

Die folgende Tabelle enthält alle Eigenschaften eines Proxy Clusters.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxy Cluster – Proxy Cluster konfigurieren – Proxy Cluster, Eigenschaften und unter:
BeanConnect Proxys zur Management Console hinzufügen – Expertenmodus – Proxys hinzufügen, Timer Settings/ Performance Settings.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
api-mode	**Application Interface Mode/API Mode**	cd / g / m	BcDef.API_MODE_xxx[2]
id	**ID**	g	(String)
name	**Name**	cm / g / m	(String)
partner-type	**Possible EIS Partner Types**	cd / g / m	BcDef.PTYPE_xxx[2]
performance. modify-start-par	**Modify Standard Start Parameter**	cd / g / m	BcDef.BOOL_xxx[2]
performance. nbr-par -inb-rfc1006-conns	**Number of Parallel Connections: InboundRfc1006**	cd / g / m	(String numerisch)
performance. nbr-par-inb-sock-conns	**Number of Parallel Connections: InboundSocket**	cd / g / m	(String numerisch)
performance. nbr-par-inb-upic-conns	**Number of Parallel Connections: InboundUpic**	cd / g / m	(String numerisch)
performance. nbr-par-outb-conns	**Number of Parallel Connections: Outbound**	g	(String numerisch)
performance. proxy-cache-size	**Proxy Container Storage Area Size: Cache**	cd / g / m	(String numerisch)
performance. proxy-mode	**Proxy Container Mode F(ast)/S(ecure)**	cd / g / m	BcDef.APPLI_MODE_xxx2

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.**[1]	**Eigenschaftswert** [1]
performance. proxy-nbr-asyn-tasks	**Number of Proxy Container Processes: asynTasks**	cd / g / m	(String numerisch)
performance. proxy-nbr-tasks	**Number of Proxy Container Processes: tasks**	cd / g / m	(String numerisch)
performance. proxy-pagepool-size	**Proxy Container Storage Area Size: Pagepool**	cd / g / m	(String numerisch)
timer.asyn-conf.sec	**Asynch. Confirmation**	cd / g / m	(String numerisch)
timer.eis-partner -reconnect.min	**EisPartnerReconnect**	cd / g / m	(String numerisch)
timer.prep-to -commit.sec	**Prepare To Commit**	cd / g / m	(String numerisch)
timer.ra-check-interval.sec	**Resource Adapter Check Interval**	cd / g / m	(String numerisch)
timer.ra-connect -time.sec	**Resource Adapter Connec-tionTime**	cd / g / m	(String numerisch)
timer.shutdown -time.min	**ShutdownTime**	cd / g / m	(String numerisch)
timer.ta-comm.sec	**Transaction Communication**	cd / g / m	(String numerisch)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse BcDef) im Unterverzeichnis JavaDoc des Management Console-Installationsverzeichnisses

Die Eigenschaft "General/Automatic Availability Check – Time Interval (sec)" wird am MC-CLI nicht angeboten, da am MC-CLI keine automatischen Checks durchgeführt werden.

9.4.14 BcAdminRA

Der Modul `BcAdminRa` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Resource Adapter der Management Console angeboten werden.

`BcAdminRA` enthält die Funktionen:

- create() – Resource Adapter zur Konfiguration hinzufügen
- getObject() – Resource Adapter-Objekt aus der Konfiguration lesen
- getProperties() – Eigenschaften eines Resource Adapters lesen
- modifyProperties() – Eigenschaften eines Resource Adapters ändern
- perform() – Administrative Aktionen starten

9.4.14.1 create() – Resource Adapter zur Konfiguration hinzufügen

Funktion: `BcAdminRA.create()`

Es wird ein Resource Adapter zur Konfiguration hinzugefügt. Die Eigenschaften des Resource Adapters müssen Sie in einem Dictionary am MC-CLI übergeben.

Parameter: `object_name` (String)

Name des BeanConnect Resource Adapters, der zur Konfiguration hinzugefügt werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Resource Adapter zugeordnet werden soll.

`props` (kw)

Dictionary mit den key-value-Paaren der Eigenschaften, die dem Resource Adapter zugeordnet werden sollen. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Resource Adapters“ auf Seite 424. Die Eigenschaft `host` muss angegeben werden.

Rückgabe: `(BcObject` vom Typ `BcObjectType.RESOURCE_ADAPTER)`

Der neu zur Konfiguration hinzugefügte Resource Adapter.

Exceptions: `BcParameterException, BcObjectException, BcToolException`

Anmerkung: Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nur erlaubt, wenn der Proxy nicht zu einem Proxy Cluster gehört. Für Proxys eines Clusters muss hier das Proxy Cluster Objekt als Parameter angegeben werden.

Beispiel:

```
...
import BcAdminRA
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
creProps ={"host":"xyz","port-inbound":"1234","desc": "created",\
                              "ta-logging": BcDef.TA_LOGGING_NONE}
newRA=BcAdminRA.create("test-RA",proxy_obj,creProps)
...
```

9.4.14.2 getObject() – Resource Adapter-Objekt aus der Konfiguration lesen

Funktion: `BcAdminRA.getObject()`

Liest den Resource Adapter mit angegebenen Namen aus der Konfiguration

Parameter: `object_name` (String)

Name des BeanConnect Resource Adapter, der gelesen werden soll.

`proxy_object`
(`BcObject` vom Typ `BcObjectType.PROXY` / `BcObjectType.PROXY_CLUSTER`)

Proxy bzw. Proxy Cluster, dem der Resource Adapter zugeordnet ist.
Die Angabe von `proxy_object` vom Typ `BcObjectType.PROXY` ist nur erlaubt, wenn der Proxy nicht zu einem Proxy Cluster gehört. Für Proxys eines Clusters muss hier das Proxy Cluster Objekt als Parameter angegeben werden.

Rückgabe: (`BcObject` vom Typ `BcObjectType.RESOURCE_ADAPTER`).

Der gelesene Resource Adapter, oder `None`, wenn kein Resource Adapter mit entsprechendem Namen existiert.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminRA
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ra_obj=BcAdminRA.getObject("test-RA".proxy_obj)
...
```

9.4.14.3 getProperties() – Eigenschaften eines Resource Adapters lesen

Funktion: `BcAdminRa.getProperties()`

Liest alle Eigenschaften des angegebenen Resource Adapters und gibt ein Dictionary mit den key-value-Paaren der Eigenschaften zurück. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Resource Adapters“ auf Seite 424

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.RESOURCE_ADAPTER`).

Resource Adapter, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Resource Adapters.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminRA
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ra_obj=BcAdminRA.getObject("test-RA", proxy_obj)
raProps=BcAdminRA.getProperties(ra_obj)
...
```

9.4.14.4 modifyProperties() – Eigenschaften eines Resource Adapters ändern

Funktion: `BcAdminRa.modifyProperties()`

ändert alle Eigenschaften des angegebenen Resource Adapters, die im angegebenen Dictionary enthalten sind.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.RESOURCE_ADAPTER`).

Resource Adapter, dessen Eigenschaften geändert werden sollen.

`props`

Dictionary mit den key-value-Paaren der Eigenschaften, die geändert werden sollen. Mögliche Werte für key siehe Abschnitt „Eigenschaften eines Resource Adapters“ auf Seite 424

Rückgabe: Keine

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminRA
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ra_obj=BcAdminRA.getObject("test-RA", proxy_obj)
modProps={"desc":"modified"}
BcAdminRA.modifyProperties(ra_obj,modProps)
...
```

9.4.14.5 perform() – Administrative Aktionen starten

Funktion: `BcAdminRA.perform()`

startet für den Resource Adapter die angegebene Aktion.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.RESOURCE_ADAPTER`).

Resource Adapter, für den die Aktion gestartet werden soll.

`action (String)`

Aktion, die für den angegebenen Resource Adapter gestartet werden soll. Mögliche Werte sind (siehe `BcDef.ACTION_xxx`)

`check-avail`	überprüft den Lauf-Status des Resource Adapters.
`update-ra-xml`	bringt die gesicherten Änderungen in die Konfiguration des Resource Adapter ein.

`params (kw)` (Dictionary)

Dictionary mit den key-value-Paaren der Parameter, die der angegebenen Aktion mitgegeben werden sollen.
Bei `action="update-ra-xml"`, muss im Dictionary `params` folgendes key-value-Paar angegeben werden:
`key="rar-file-path"`
`value`= Name der Resource Adapter-rar-Datei, in die die Konfigurationsänderungen eingebracht werden sollen.

Rückgabe: Bei `action="update-ra-xml"`:
True (boolean), wenn update angestossen wurde, sonst BcToolException.
Bei `action="check-avail"`:
`bcaction` (`BcObject` mit Typ `BcObjectType.ACTION`)
mit allen Informationen über die gestartete Aktion und sämtliche Teilaktionen. Um genauere Informationen zu erhalten, kann eine Funktion des Moduls `BcAdminAction` mit diesem Objekt als Parameter aufgerufen werden.

Exceptions: `BcObjectException, BcParameterException, BcToolException`

Beispiel:

```
...
import BcAdminRA
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ra_obj=BcAdminRA.getObject("test-RA", proxy_obj)
bcaction=BcAdminRA.perform(ra_obj,"check-avail")
...
```

9.4.14.6 remove() – Resource Adapter entfernen

Funktion: `BcAdminRA.remove()`

Entfernt den angegebenen Resource Adapter aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.RESOURCE_ADAPTER`).

Resource Adapter, der entfernt werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminRA
import BcAdminProxy
...
proxy_obj=BcAdminProxy.getObject("BCProxy")
ra_obj=BcAdminRA.getObject("test-RA", proxy_obj)
BcAdminRA.remove(ra_obj)
...
```

9.4.14.7 Eigenschaften eines Resource Adapters

Die folgende Tabelle enthält alle Eigenschaften eines Resource Adapters.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
BeanConnect Proxys konfigurieren – Resource Adapter – Resource Adapter, Eigenschaften.

Schlüsselwort (key) am MC-CLI[1]	**Feldname an der grafischen Oberfläche**[1]	**Funkt. im einzelnen Proxy** [1]	**Funkt. im Proxy Cluster** [1]	**Eigenschaftswert**[1]
`admin-port`	**McCmdHandlerPort**	c / g / m	c / g / m	(String numerisch)
`admin-pw`	**McCmdHandler Password**	c / m	c / m	(String)
`desc`	**Description**	c / g / m	-	(String)
`host`	**Host**	cm / g / m	cm / g / m	(String)
`id`	**ID**	g	-	(String numerisch)
`index`	**Index**	g	-	(String numerisch)
`log4j-config-file-path`	**Log4J Configuration File Path**	c / g / m	c / g / m	(String)

Schlüsselwort (key) am MC-CLI[1]	**Feldname an der grafischen Oberfläche**[1]	**Funkt. im einzelnen Proxy** [1]	**Funkt. im Proxy Cluster** [1]	**Eigenschaftswert**[1]
`name`	**Name**	cd / g / m	c / g	(String)
`port-inbound`	**Listener Port**	c / g / m	c / g / m	(String numerisch)
`proxy-reconnect-count`	**ProxyReconnectCount**	–	c / g / m(s)	(String numerisch)
`proxy-reconnect-interval.min`	**ProxyReconnect Interval**	–	c / g / m(s)	(String numerisch)
`proxy-url-outbound`	**Proxy URL for OLTP Outbound Communication**	g	-	(String)
`ta-log-dir`	**TransactionLoggingDir**	cd / g / m	c / g / m(s)	(String)
`ta-logging`	**TransactionLogging**	cd / g / m	c / g / m(s)	`BcDef.TA_LOGGING_xxx` [2]

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse `BcDef`) im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses

Werden die Eigenschaften `proxy-reconnect-count`, `proxy-reconnect-interval.min`, `ta-log-dir`, `ta-logging` für einen Resource Adapter geändert, der zu einem Proxy Cluster gehört, dann wird beim Sichern der Änderung automatisch eine Synchronisierung aller Resource Adapter dieses Proxy Clusters durchgeführt.

9.4.15 BcAdminTodo

Der Modul `BcAdminTodo` enthält alle Funktionen, die zur Konfiguration und Administration eines Objekts Todo Topic der Management Console angeboten werden. Da Todo Topics keine Namen haben, über die sie angesprochen werden können, können Todo Topics nur über die Listenfunktion `BcAdminMain.getList(BcObjectType.TODO.toString())` gelesen werden.

`BcAdminTodo` enthält die Funktionen:

- getProperties() – Eigenschaften eines Todo-Topics lesen
- remove() – Todo-Topic löschen

9.4.15.1 getProperties() – Eigenschaften eines Todo-Topics lesen

Funktion: `BcAdminTodo.getProperties()`

Liest alle Eigenschaften des angegebenen Todo-Topics und gibt ein Dictionary mit key-value-Paaren der Eigenschaften zurück.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.TODO`)

Das Todo-Topic, dessen Eigenschaften gelesen werden sollen.

Rückgabe: Dictionary mit den key-value-Paaren aller Eigenschaften des Todo Topics. Mögliche Werte für key finden Sie in Abschnitt „Eigenschaften eines Todo Topics“ auf Seite 428.

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminTodo
...
todoProps=BcAdminTodo.getProperties(elem)
...
```

9.4.15.2 remove() – Todo-Topic löschen

Funktion: `BcAdminTodo.remove()`

Löscht das angegebene Todo-Topic aus der Konfiguration.

Parameter: `bc_object` (`BcObject` vom Typ `BcObjectType.TODO`)

Das Todo-Topic, das gelöscht werden soll.

Rückgabe: Keine

Exceptions: `BcObjectException, BcToolException`

Beispiel:

```
...
import BcAdminTodo
...
BcAdminTodo.remove(elem)
...
```

9.4.15.3 Eigenschaften eines Todo Topics

Die folgende Tabelle enthält alle Eigenschaften eines Todo Topics.

Die Bedeutung und die erlaubten Werte der jeweiligen Eigenschaften finden Sie in der Online-Hilfe der Management Console unter:
Todo-Aktionen (Todo Topics) – Todo Aktion, Eigenschaften.

Schlüsselwort (key) am MC-CLI [1]	**Feldname an der grafischen Oberfläche** [1]	**Funkt.** [1]	**Eigenschaftswert** [1]
`comm-service`	**Communication Service**	g	(String)
`components`	**Component(s)**	g	(String)
`eis-partner`	**EIS Partner**	g	(String)
`internal`	**Internal**	g	`BcDef.BOOL_xxx`[2]
`lu62gateway`	**openUTM-LU62 Gateway**	g	(String)
`proxy`	**Proxy**	g	(String)
`proxy-cluster`	**Proxy Cluster**	g	(String)
`related-file-name`	**Related Configuration File Name**	g	(String)
`text`	**Text**	g	(String)
`time`	**Time**	g	(String)

[1] Bedeutung der Spalten und Abkürzungen siehe Abschnitt „Eigenschaften“ auf Seite 323

[2] Die möglichen Werte für xxx finden Sie in der JavaDoc (Java-Klasse `BcDef`) im Unterverzeichnis `JavaDoc` des Management Console-Installationsverzeichnisses

9.5 Anwendungsszenarien (Beispiele)

Dieser Abschnitt beschreibt das Vorgehen für verschiedene Anwendungsszenarien. Er beschreibt die Konfiguration einer Out- sowie einer Inbound-Kommunikation mit einer openUTM-Anwendung. Zusätzlich finden Sie hier Beispiele für administrative Tätigkeiten, die Sie per Skript automatisieren können, und eine Liste der Jython-Beispiel-Skripts, die zusammen mit BeanConnect ausgeliefet werden.

Voraussetzung für die Konfiguration über das MC-CLI ist die Installation eines lokalen Proxys oder die Aufnahme eines entfernten Proxys über die grafische Oberfläche der Management Console.

9.5.1 Outbound-Kommunikation mit einer openUTM-Anwendung konfigurieren

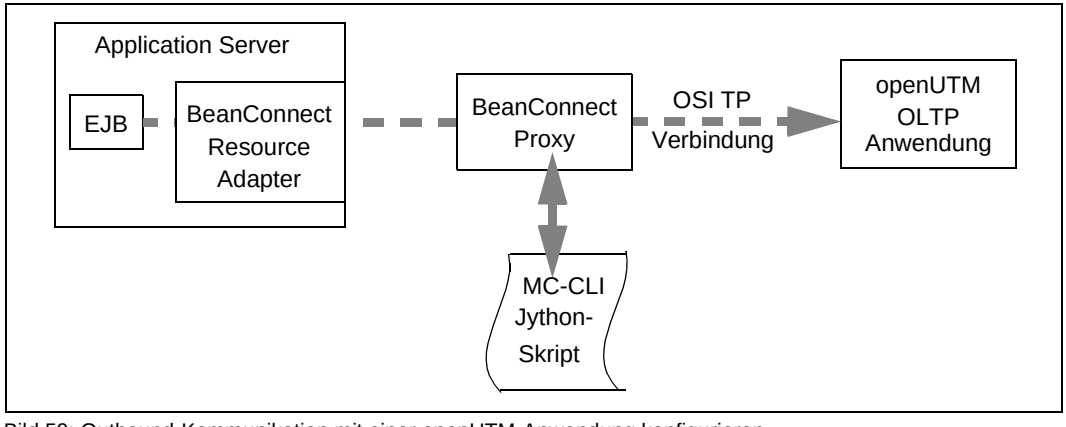


Bild 59: Outbound-Kommunikation mit einer openUTM-Anwendung konfigurieren

Für die Konfiguration einer Outbound-Kommunikation mit einer openUTM-Anwendung müssen Sie in einem Jython-Skript folgende MC-CLI-Funktionen in der angegebenen Reihenfolge aufrufen. Ein Beispiel für die Konfiguration der Outbound-Kommunikation finden Sie in dem mit ausgelieferten Jython-Beispiel-Skript `sampleAdminOutbound.py` im Verzeichnis `cli-sample`.

1. `BcAdminMain.init()`
   Starten Sie eine Sitzung der Management Console, indem Sie die Funktion `BcAdminMain.init(<MC_home>)` aufrufen.

   Die Management Console erkennt einen installierten lokalen Proxy automatisch und öffnet dessen Konfigurationsdatei `console.properties.xml` bzw. legt diese an, wenn sie noch nicht existiert.

2. `BcAdminMain.getlist()` / `BcAdminProxy.getObject()`
   Lesen Sie die Liste aller konfigurierten Proxys mit `BcAdminMain.getList("proxy")` oder lesen Sie ein bestimmtes Proxy Objekt mit `BcAdminProxy.getObject(<proxy_name>)`.

   Die Rückgaben der Aufrufe können Sie bei den Aufrufen unter Punkt 3 bis 8 als Aufrufparameter `<bcproxy>` nutzen.

3. `BcAdminProxy.authenticate()`
   Authentifizieren Sie sich beim Proxy mit dem Aufruf von `BcAdminProxy.authenticate(<bcproxy>, admin_pw)`.

4. `BcAdminProxy.modifyProperties()`
   Konfigurieren Sie den Proxy, indem Sie in Ihrem Skript die Funktion `BcAdminProxy.modifyProperties(<bcproxy>, properties)` mit den gewünschten Eigenschaftswerten aufrufen.

5. `BcAdminEisPartner.create()`
   Definieren Sie ein (oder mehrere) EIS-System(e), indem Sie die Funktion `BcAdminEisPartner.create(<eisPartnerName>,<bcproxy>, properties)` (mehrfach) aufrufen.

6. `BcAdminOutboundService.create()`
   Definieren Sie ein (oder mehrere) Outbound Services, indem Sie die Funktion `BcAdminOutboundService.create(<remoteTacName>,<bcproxy>, properties)` (mehrfach) aufrufen.

7. `BcAdminOutboundCommEndpoint.create()`
   Richten Sie einen (oder mehrere) OutboundCommunicationEndpoint(s) ein, indem Sie die Funktion `BcAdminOutboundCommEndpoint.create (<communicationEndpointName>,<bcproxy>, properties)` (mehrfach) aufrufen.

8. `BcAdminProxy.perform()`
   Sichern und aktualisieren Sie die Proxy Konfiguration mit den Funktionen `BcAdminProxy.perform(<bcproxy>, "save")` und `BcAdminProxy.perform(<bcproxy>, "update-config")`.

9. `BcAdminEisPartner.perform()`
   Sichern Sie die Konfiguration des/der EIS Partner, indem Sie die Funktion `BcAdminEisPartner.perform(<eisPartner>, "gen-config")` aufrufen.

10. `BcAdminMain.close()`
    Beenden Sie die Sitzung der Management Console mit dem Aufruf der Funktion `BcAdminMain.close()`.

9.5.2 Inbound-Kommunikation mit einer openUTM-Anwendung konfigurieren

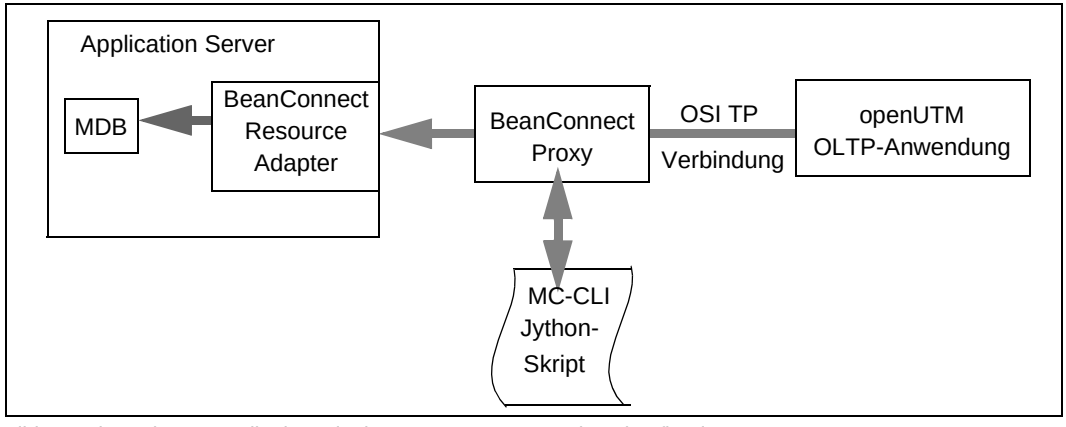


Bild 60: Inbound-Kommunikation mit einer openUTM-Anwendung konfigurieren

Für diese bestehende Proxy Konfiguration aus Abschnitt „Outbound-Kommunikation mit einer openUTM-Anwendung konfigurieren“ auf Seite 429 kann zusätzlich ein Skript zur Konfiguration der Inbound-Kommunikation gestartet werden. Ein Beispiel für die Konfiguration der Inbound-Kommunikation finden Sie in dem mit ausgelieferten Jython-Beispiel-Skript `sampleAdminInbound.py` im Verzeichnis `cli-sample`.

1. `BcAdminMain.init()`
   Starten Sie eine Sitzung der Management Console, indem Sie die Funktion `BcAdminMain.init(<MC_home>)` aufrufen.

   Die Management Console liest die Konfigurationsdatei `console.properties.xml` ein und setzt auf der obigen Konfiguration auf.

2. `BcAdminMain.getlist()` / `BcAdminProxy.getObject()`
   Lesen Sie die Liste aller konfigurierten Proxys mit `BcAdminMain.getList(“proxy“)` oder lesen Sie ein bestimmtes Proxy Objekt mit `BcAdminProxy.getObject(<proxy_name>)`.

3. `BcAdminProxy.authenticate()`
   Authentifizieren Sie sich beim Proxy mit dem Aufruf von `BcAdminProxy.authenticate(<bcproxy>, admin_pw)`.

4. `BcAdminRA.create()`
   Definieren Sie ein (oder mehrere) Resource Adapter, indem Sie die Funktion `BcAdminRA.create(<resourceAdapterName>,<bcproxy>, properties)` (mehrfach) aufrufen.

5. `BcAdminInboundMsgEndpoint.create()`
   Richten Sie einen (oder mehrere) Inbound Message Endpoint(s) ein, indem Sie die Funktion `BcAdminInboundMsgEndpoint.create(<messageEndpointName>, <bcproxy>, properties)` (mehrfach) aufrufen.
6. `BcAdminInboundUser.create()`
   Wenn der EIS Partner Benutzer verwendet, definieren Sie einen (oder mehrere) Inbound User, indem Sie die Funktion `BcAdminInboundUser.create(<userName>, <bcproxy>, properties)` (mehrfach) aufrufen.
7. `BcAdminInboundService.modifyProperties()`
   Legen Sie die Eigenschaften der entsprechenden UTM-Transaktionscodes fest, indem Sie die Funktion `BcAdminInboundService.modifyProperties (<InboundService>, properties)` (mehrfach) aufrufen.
8. `BcAdminProxy.perform()`
   Sichern und aktualisieren Sie die Proxy Konfiguration mit den Funktionen `BcAdminProxy.perform(<bcproxy>, "save")` und `BcAdminProxy.perform(<bcproxy>, "update-config")`.
9. `BcAdminRA.perform()`
   Sichern Sie die Konfiguration des/der Resource Adapter, indem Sie die Funktion `BcAdminRa.perform(<resourceAdapter>,"update-ra-xml", params)` aufrufen.
10. `BcAdminMain.close()`
    Beenden Sie die Sitzung der Management Console mit dem Aufruf der Funktion `BcAdminMain.close()`.

9.5.3 Proxys administrieren

Für die Administration der Proxy Komponenten kann man sich verschiedene Anwendungsszenarien vorstellen, in denen die Administration per Skript komfortabler ist als die über die grafische Oberfläche der Management Console. Beispiele hierfür sind:

- Alle Proxys bzw. alle Proxys, die auf einem bestimmten Host laufen, sollen beendet bzw. neu gestartet werden, z.B. um die Konfiguration zu aktualisieren.
- Für alle Inbound Message Endpoints mit `type=dialog` soll, z.B. wegen Netz-Problemen, der reply-Timer ausgeschaltet werden.
- Nach Beheben der Netz-Probleme soll der reply-Timer wieder auf den Standard-Wert zurückgesetzt werden.

9.5.4 Jython-Beispiel-Skripts

Mit der Management Console werden Jython-Beispiel-Skripts ausgeliefert. Diese Skripts geben die vorhandenen Objekte des entsprechenden Typs mit ihren Eigenschaften aus. Zusätzlich können neue Test-Objekte erzeugt, modifiziert oder gelöscht werden. Die Namen der Testobjekte beginnen mit "test bzw. "TEST". Je nach Konfiguration werden Aktionen (Statusabfrage, Start, Stop) gestartet.

Nach der Installation der Management Console stehen folgende Skripts im Installationsverzeichnis der Management Console im Unterverzeichnis `cli-sample` zur Verfügung:

- `startBcAdmin.cmd` (für Windows-Plattformen) bzw. `startBcAdmin.sh` (für Unix-/Linux-Plattformen) – Start-Skript.

  `startBcAdmin.cmd` bzw. `startBcAdmin.sh` ruft das Skript `sampleMccliStart.py` auf.

- `sampleMccliStart.py` - Start Jython-Skript, von dem aus abhängig von den angegebenen Optionen die gewünschten Beispiel-Skripts aufgerufen werden.
- `sampleAdminCommService.py` - Jython-Skript mit Funktionen des Moduls `BcAdminCommunicationService`.
- `sampleAdminEisPartner.py` – Jython-Skript mit Funktionen des Moduls `BcAdminEisPartner`.
- `sampleAdminInbound.py` – Jython-Skript mit Funktionen der Module `BcAdminInboundUser`, `BcAdminInboundService`, `BcAdminInboundMsgEndpoint`.
- `sampleAdminLu62Gateway.py` - Jython-Skript mit Funktionen des Moduls `BcAdminLu62Gateway`.
- `sampleAdminMain.py` – Jython-Skript mit Funktionen der Module `BcAdminMa`in und `BcAdminTodo`.
- `sampleAdminProxy.py` – Jython-Skript mit Funktionen des Moduls `BcAdminProxy`.
- `sampleAdminProxyCluster.py` – Jython-Skript mit Funktionen des Moduls `BcAdminProxyCluster`.
- `sampleAdminRa.py` – Jython-Skript mit Funktionen des Moduls `BcAdminRA`.
- `sampleAdminOutbound.py` – Jython-Skript mit Funktionen der Module `BcAdminOutboundService`, `BcAdminOutboundCommEndpoint`.
- `sampleInitConfig.template.py` - Template für die Anpassung der Konfigurationsdaten und den gewünschten Umfang der Beispiele.

Einsatz der Jython-Beispiel-Skripts

Gehen Sie in folgendens Schritten vor:

- Setzen Sie im Start-Skript `startBcAdmin.cmd` bzw. `startBcAdmin.sh` die Umgebungsvariable JYTHONPATH.
- Erzeugen Sie eine überschreibbare Kopie von `sampleInitConfig.template.py` mit dem Namen `sampleInitConfig.py`. Diese Datei muss immer im selben Verzeichnis liegen, in dem sich die Skripts befinden (Standard nach Installation: `cli-sample`). Sie wird von allen Jython-Beispiel-Skripts verwendet.
- Passen Sie die Konfigurationsdatei `sampleInitConfig.py` an, siehe Abschnitt „Konfigurationsdatei editieren“.
- Rufen Sie das Start-Skript `StartBcAdmin.cmd/sh` auf, siehe Abschnitt „Start-Skript aufrufen“. Dabei geben Sie über Aufruf-Optionen an, welche Skripts und Funktionen ausgeführt werden sollen.

Sie können jedes Jython-Beispiel-Skript auch als Hauptprogramm starten, müssen dann aber die Parameter, die im Start-Skript versorgt werden, manuell angeben. Daher wird empfohlen, die Skripts über das zentrale Start-Skript mit entsprechenden Optionen zu starten.

Konfigurationsdatei editieren

Die Konfigurationsdatei `sampleInitConfig.py` enthält Angaben zu den Objekten, die von den Beispiel-Skripts gelesen werden sollen. Außerdem werden in dieser Datei Ablaufparameter festgelegt.

Gehen Sie wie folgt vor:

- Ersetzen Sie in `sampleInitConfig.py` die Platzhalter "****" der einzelnen Objektparameter für den Proxy, EIS Partner, Resource Adapter usw. durch die aktuellen Werte. Folgende Objektparameter gelten für alle Skripts:

  console_home
  : Installationsverzeichnis der Management Console. Alternativ kann die Information auch über die Umgebungsvariable BEANCONNECT_USERCONS gelesen werden.

  proxy_name
  : Name des Proxys, für den die Funktionen ausgeführt werden sollen (nicht notwendig bei `sampleAdminMain.py`).

  admin_pw
  : Passwort des Proxys, für den die Funktionen ausgeführt werden sollen (nicht notwendig bei `sampleAdminMain.py`).

Details zu den anderen Parametern findens Sie in der Inline-Beschreibung in `sampleInitConfig.py`.

- Ändern Sie in `sampleInitConfig.py` die voreingestellten Ablaufparameter (Erzeugen, Modifizieren, Sichern, Löschen, Proxy starten/stoppen) falls gewünscht. Es gibt folgende Ablaufparameter:
  - `bCreDelObjs=True/False`
    Test-Objekte erzeugen und entfernen (die Objektnamen beginnen mit "test" bzw. "TEST")
  - `bModObjs=True/False`
    Test-Objekte modifizieren.
  - `bSaveMod=True/False`
    Konfigurationsänderungen speichern.
  - weitere Skript-spezifische Schalter (siehe auch die Kommentare in `sampleInit-Config.py`).

  In der Regel wird bei `bCreDelObjs=bModObjs=bSaveMod=True` in einem ersten Durchlauf der Skripts ein Test-Objekt erzeugt und modifiziert, das in einem zweiten Lauf entfernt wird. Gesteuert wird die Funktionsabfolge in der Regel über den Wert der Eigenschaft `desc` (`"created"` -> `"modified"` -> `remove object`):
  - Existiert ein Test-Objekt nicht, wird es erzeugt mit `desc="created"`.
  - Existiert ein Test-Objekt und ist `desc= created"`, wird es modifiziert (desc="`modified`").
  - Existiert ein Test-Objekt und ist `desc="modified"`, wird es aus der Konfiguration entfernt.

Start-Skript aufrufen

Öffnen Sie die Windows-Eingabeaufforderung bzw. eine Unix- oder Linux-Shell und geben im Verzeichnis `cli-sample` folgendes Kommando ein:

`startBcAdmin.cmd [options]` (Windows-Systeme)

`startBcAdmin.sh [options]` (Unix- und Linux-Systeme)

Mit `options` steuern Sie den Funktionsumfang. Folgende Werte sind möglich:

-help oder -h
: Hilfe-Funktion, gibt nur die möglichen Parameter mit Erklärung aus. Standardwert, wenn keine Option angegeben wurde.

-all
: startet alle Beispiel-Skripts `sampleAdminXxx`.

-cs
: startet `sampleAdminCommService.py`.

-ei	startet `sampleAdminEisPartner.py`.
-gw	startet `sampleAdminLu62Gateway.py`.
-in	startet `sampleAdminInbound.py`.
-main	startet `sampleAdminMain.py`.
-out	startet `sampleAdminOutbound.py`.
-proxy	startet `sampleAdminProxy.py`.
-proxy-cl	startet `sampleAdminProxyCluster.py`.
-ra	startet `sampleAdminRa.py`.
-log	Logging-Funktion, d.h. ausführliche Ausgabe der gelesenen Informationen (Listenelemente, Eigenschaften der gelesenen Objekte). Dies Angabe ist zusätzlich zu allen vorher aufgeführten Optionen möglich.

9.5.5 Erstellen von Jython Skripts aus MC-CLI-Mitschnitten

Das Erstellen von Jython Skripts aus MC-CLI-Mitschnitten ist sinnvoll, wenn eine Abfolge von Aktionen wie z.B. eine Inbound-Konfiguration für mehrere Proxys analog durchgeführt werden soll. Die Vorlage für ein solches Skript erhalten Sie, indem Sie die gewünschten Aktionen für einen Proxy in der Management Console ausführen und anschließend den MC-CLI-Mitschnitt dieses Proxys auf Datei ausgeben. Dieses Skript kann leicht für weitere Proxys angepasst und für deren Konfiguration genutzt werden.

Format der Mitschnitt-Datei

Eine Mitschnitt-Datei wird unter `<MC_home>/cli-rec` abgelegt.

Der Mitschnitt enthält einen Skript-Kopf mit Kommentaren, Import-Anweisungen, dem Startaufruf der Management Console Sitzung und Zuweisung der Proxy-/Cluster-spezifischen Variablen. Darauf folgen die MC-CLI-Aufrufe der mitgeschnittenen Aktionen. Bei Ausgabe auf Datei wird zusätzlich ein "Footer" mit Kommentarzeilen zum möglichen Skript-Ende geschrieben. So ist der Mitschnitt, evtl. mit kleineren Anpassungen, ein ablauffähiges Jython-Skript.

Beispiel

Der Shutdown-Timer für den Proxy Test001 wird über die Management Console auf 3 Minuten gesetzt. Der Mitschnitt wird am Ende der Session auf Datei geschrieben. Die Mitschnitt-Datei unter dem Console-Home-Verzeichnis hat den Namen `2015-06-18.01-45.ProxyID.0.py` und enthält folgende Einträge:

```
''' MC-CLI Recording for                                                    1.
    BeanConnect Management Console BeanConnect V3.0B00 2015-04-21-0607
'''
import BcAdminCommunicationService
import BcAdminEisPartner
import BcAdminInboundMsgEndpoint
import BcAdminInboundService
import BcAdminInboundUser
import BcAdminLu62Gateway
import BcAdminMain
import BcAdminOutboundCommEndpoint
import BcAdminOutboundService
import BcAdminProxy
import BcAdminProxyCluster
import BcAdminRA
import BcAdminTodo
import com.fujitsu.ts.jca.tools.mc.cli.BcObjectException      as BcObjectException
import com.fujitsu.ts.jca.tools.mc.cli.BcParameterException   as BcParameterException
import com.fujitsu.ts.jca.tools.mc.cli.BcToolException        as BcToolException
import com.fujitsu.ts.jca.tools.mc.cli.BcObjectType           as BcObjectType
import com.fujitsu.ts.jca.tools.mc.cli.BcDef                  as BcDef
```

```
''' init BC console for command line interface '''
consoleHome = "C:\\BeanConnect\\V3.0B00\\Console"
BcAdminMain.init(console_home=consoleHome)

''' end of statements for start of MC session '''                    2.

''' record start date/time: 2015-06-18:01-39 '''                     3.
''' MC-CLI record file for Proxy Test001 '''

''' assign proxy object '''
proxy_name = "Test001"
proxy_obj  = BcAdminProxy.getObject(proxy_name)

''' proxy context menu "Edit Properties" '''
''' or button "Edit" in table "BeanConnect Proxies" '''
proxy_dicn = BcAdminProxy.getProperties(proxy_obj)                   4.

''' proxy dialog "Edit Properties" exit with "OK" '''
proxy_dicn_new = {}
proxy_dicn_new["timer.shutdown-time.min"] = "3"                      5.
BcAdminProxy.modifyProperties(proxy_obj,proxy_dicn_new)

''' proxy context menu "Save/Restart" - "Save Proxy" '''
result = BcAdminProxy.perform(proxy_obj,BcDef.ACTION_SAVE)           6.
''' TODO: analyse result '''

''' end of BeanConnect Management Console session '''
''' don't forget to save your changes! '''
''' close BC Management Console '''
BcAdminMain.close(False)
```

Erläuterung:

1. Beginn des Standard-Headers, wird bei jeder Aufzeichnung geschrieben.
2. Ende des Standard-Headers
3. Beginn der Session-spezifischen Aufzeichnung für den Proxy Test001
4. Einlesen der Proxy-Eigenschaften (über Eigenschaftsdialog **Edit Properties**)
5. Änderung des Shutdown-Timers
6. Sichern der Änderungen

10 Interfaces und Programmierung

Dieses Kapitel enthält Informationen zu folgenden Themen:

- BeanConnect-spezifische Interfaces und das Common Client Interface (CCI)
- Outbound-Kommunikation programmieren
- Inbound-Kommunikation programmieren

Bevor eine Kommunikation zwischen einer EJB und einer EIS-Anwendung aufgenommen werden kann, müssen die Konfigurationsdaten von BeanConnect und die Konfiguration der EIS-Anwendung ordnungsgemäß eingerichtet worden sein.

- Bei einer Outbound-Kommunikation ruft eine in einem Java EE Application Server deployte EJB einen Service im EIS Partner auf.
- Bei einer Inbound-Kommunikation sendet ein EIS Partner Nachrichten an eine OLTP Message-Driven Bean, die in einem Java EE Application Server deployt ist.

Die JavaDoc zu BeanConnect, auf die in diesem Kapitel wiederholt verwiesen wird, wird mit der JAR-Datei des Resource Adapters `BC30B00_RA.jar` ausgeliefert und steht nach der Installation des Resource Adapters zur Verfügung.

Sie finden die JavaDoc im Installationsverzeichnis des Resource Adapters:

- Auf Windows-Systemen unter `JavaDoc\api\index.html`
- Auf Unix-/Linux-Systemen unter `JavaDoc/api/index.html`

10.1 BeanConnect-spezifische Interfaces und das Common Client Interface (CCI)

BeanConnect bietet zwei unterschiedliche Interface-Sets an:

- BeanConnect-spezifische Interfaces
- Common Client Interface (CCI)

Empfehlungen: BeanConnect-spezifische Interfaces oder CCI

Das Common Client Interface (CCI) ist in der JCA-Spezifikation von Sun Microsystems™ definiert. Das CCI definiert ein Standard-API und ist primär an den Anforderungen von Tools zur Anwendungsentwicklung und von EAI-Frameworks (Enterprise Application Integration) ausgerichtet. Die Verwendung des CCI ist sinnvoll, wenn Sie bereits mit dem CCI in der JCA-Spezifikation vertraut sind.

In allen anderen Fällen empfiehlt es sich, die BeanConnect-spezifischen Interfaces zu verwenden, da sie die Programmierung wesentlich erleichtern.

Unterschiede zwischen BeanConnect-spezifischen Interfaces und CCI

Bei Outbound-Kommunikation bieten BeanConnect-spezifische Interfaces und das CCI praktisch dieselbe Funktionalität.

Für OLTP-Kommunikation mit openUTM-Partnern und CICS-Partnern haben das BeanConnect-spezifische Interface `EISOltpConnection` und das CCI identische Funktionalität.
Lediglich das BeanConnect-spezifische Interface `EISUpicConnection` bietet für UPIC-Verbindungen zu openUTM-Partnern zusätzliche Funktionen an (siehe „Weitere Funktionalität des Interfaces EISUpicConnection“ auf Seite 445).

Bei Inbound-Kommunikation bieten das BeanConnect-spezifische Interface `OltpMessageListener` und CCI dieselbe Funktionalität für dialogbasierte Kommunikation. Darüber hinaus stellt BeanConnect das Interface `AsyncOltpMessageListener` für asynchrone Kommunikation zur Verfügung.

Interfaces auswählen

So legen Sie ein Interface für Outbound-Kommunikation fest:

- In der Application Server spezifischen Deployment-Descriptor-Datei `weblogic-ra.xml` des Resource Adapters legen Sie das Connection-Factory-Interface mit dem Eintrag `<connection-factory-interface>` fest:
  - Für BeanConnect-spezifische Interfaces:

    `net.fsc.jca.communication.EISOltpConnectionFactory`
    oder
    `net.fsc.jca.communication.EISUpicConnectionFactory` (nur wenn der EIS Partner eine openUTM-Anwendung ist)
  - Für CCI:

    `net.fsc.jca.communication.cci.BCOltpConnectionFactory`
    oder
    `net.fsc.jca.communication.cci.BCUpicConnectionFactory` (nur wenn der EIS Partner eine openUTM-Anwendung ist)
- In der Deployment-Descriptor-Datei `ejb-jar.xml` einer EJB legen Sie das Connection-Factory-Interface mit dem Eintrag `<res-type>` fest:
  - Für BeanConnect-spezifische Interfaces:

    `net.fsc.jca.communication.EISOltpConnectionFactory` oder
    `net.fsc.jca.communication.EISUpicConnectionFactory`
  - Für CCI:

    `net.fsc.jca.communication.cci.BCOltpConnectionFactory` oder
    `net.fsc.jca.communication.cci.BCUpicConnectionFactory`

Für Inbound-Kommunikation legen Sie das gewünschte Interface in der Deployment-Descriptor-Datei `ejb-jar.xml` der Message-Driven Bean fest.

Das Interface legen Sie mit dem Eintrag `<messaging-type>` fest:

- Für BeanConnect-spezifische Interfaces:

  `net.fsc.jca.communication.AsyncOltpMessageListener` oder
  `net.fsc.jca.communication.OltpMessageListener`
- Für CCI:

  `javax.resource.cci.MessageListener`

10.2 Outbound-Kommunikation programmieren

BeanConnect unterstützt für Outbound-Kommunikation mit einer openUTM-Partnernwendung das OSI TP Protokoll, das sowohl für Verteilte Transaktionsverarbeitung als auch für nicht-transaktionale Kommunikation verwendet werden kann, sowie das proprietäre, nicht-transaktionale UPIC-Protokoll für den Zugriff auf openUTM-Anwendungen.

BeanConnect unterstützt für Outbound-Kommunikation mit einer CICS-Partnernwendung das LU6.2-Protokoll, das sowohl für Verteilte Transaktionsverarbeitung als auch für nicht-transaktionale Kommunikation verwendet werden kann.

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- BeanConnect-spezifische Interfaces für Outbound-Kommunikation
- Common Client Interface (CCI) für Outbound-Kommunikation
- Programmierinformationen zu Outbound-Kommunikation
- Programm-Framework für Outbound-Kommunikation
- Outbound-Kommunikation mit XATMI-Partnern

10.2.1 BeanConnect-spezifische Interfaces für Outbound-Kommunikation

Die BeanConnect-spezifischen Interfaces für Outbound-Kommunikation sind im Paket `net.fsc.jca.communication` enthalten.

10.2.1.1 Connection-Factory-Interfaces

Mit einer Connection Factory können Sie eine Verbindung aufbauen. BeanConnect stellt folgende Connection-Factory-Interfaces bereit:

- `EISConnectionFactory`
- `EISOltpConnectionFactory`
- `EISUpicConnectionFactory` (nur wenn der EIS Partner eine openUTM-Anwendung ist)

Die Interfaces `EISOltpConnectionFactory` und `EISUpicConnectionFactory` erweitern das Interface `EISConnectionFactory` ohne jedoch zusätzliche Funktionalität bereitzustellen.

Die Verwendung der Interfaces `EISOltpConnectionFactory` oder `EISUpicConnectionFactory` ist dann sinnvoll, wenn Sie sicherstellen möchten, dass die Kommunikation über das OSI TP Protokoll bzw. über das UPIC-Protokoll durchgeführt wird. Es wird empfohlen, das Interface `EISConnectionFactory` zu verwenden.

Beispiel 12 Überprüfen der Kommunikation auf Verwendung des OSI TP Protokolls (openUTM-Partner)

Ob die Kommunikation mit der EIS-Anwendung mit Hilfe des OSI TP Protokolls verarbeitet wird, überprüfen Sie mit folgender Code-Sequenz:

```
...
net.fsc.jca.communication.EISConnectionFactory cf =
   (EISConnectionFactory)ic.lookup
      ("java:comp/env/<resourceRefName>");
...
if (! (cf instanceof EISOltpConnectionFactory))
   throw new Exception("EISOltpConnectionFactory was
                        expected!");
...
```

Eine Exception wird ausgelöst, wenn nicht das OSI TP Protokoll für die Kommunikation verwendet wird.

10.2.1.2 Connection-Interfaces (Übersicht)

BeanConnect stellt folgende Connection-Interfaces bereit:

- `EISConnection`
- `EISOltpConnection`
- `EISUpicConnection` (nur wenn der EIS Partner eine openUTM-Anwendung ist)

Das Interface `EISOltpConnection` erweitert das Interface `EISConnection` durch die Bereitstellung von zusätzlichen Leistungsmerkmalen wie z.B. asynchrone Kommunikation (weitere Einzelheiten siehe „Weitere Funktionalität des Interface EISOltpConnection“ auf Seite 445).

Das Interface `EISUpicConnection` erweitert das Interface `EISConnection` durch die Bereitstellung von zusätzlichen Leistungsmerkmalen, wie z.B. Merkmale von Datenstrukturen (weitere Einzelheiten siehe „„Weitere Funktionalität des Interfaces EISUpicConnection“ auf Seite 445“).

Es wird empfohlen, die Interfaces `EISOltpConnection` und `EISUpicConnection` nur dann zu verwenden, wenn Sie auch deren zusätzlich angebotenen Funktionen nutzen möchten.

Kommunikationsmethoden des Interfaces EISConnection

Das Interface `EISConnection` wird von einer Reihe von Interfaces erweitert. Jedes von ihnen stellt für Outbound-Kommunikation zwischen der EJB und der EIS-Anwendung Methoden bereit. Für den Datenaustausch sind dies die Methoden `snd()`, `rcv()`, `call()` sowie verschiedene Varianten von `snd()` und `rcv()` wie `rcvRecord()` oder `sndString()`.

Die von diesen Interfaces angebotenen Kommunikationsmethoden unterscheiden sich durch das Datenformat, auf dem die Methoden basieren:

- Für Byte-Array-orientierten Zugriff auf EIS-Anwendungen:

  `EISConnectionByteArray`

- Für Byte-Container-orientierten Zugriff auf EIS-Anwendungen:

  `EISConnectionByteContainer`

  Eine Anwendung kann das Interface `EISConnectionByteContainer` verwenden und das Interface `ByteContainer` implementieren, wenn sie mit einem Service in der Partneranwendung strukturierte Objekte austauschen möchte, die Text und binäre Informationen enthalten. In der Klasse, die das Interface `ByteContainer` implementiert, muss eine Code-Konvertierung für die Textinformation dieses Objekts durchgeführt werden.

  Durch die Bereitstellung eines passenden `ByteContainer`-Objektes wird die Konvertierung von Byte-Streams in Strings während der Ausführung der Methoden `sndRecord()`, `rcvRecord()` oder `call()` durchgeführt. Objekte, die Sie mit Cobol2Java erzeugen, verwenden diese Funktion (siehe Kapitel „Cobol2Java“ auf Seite 615).

- Für OLTP-Message-orientierten Zugriff auf EIS-Anwendungen:

  `EISConnectionOltpMessage`

  Über dieses Interface werden Objekte des Typs `OltpMessage` mit der EIS-Anwendung ausgetauscht. Das Objekt `OltpMessage` dient als Container für den Nachrichteninhalt, der sich aus einem oder mehreren `OltpMessageRecord`-Objekten und/oder aus einem oder mehreren `OltpMessagePart`-Objekten zusammensetzt. Während ein `OltpMessageRecord` beliebig lang sein kann, darf ein `OltpMessagePart` 32767 Bytes nicht übersteigen. `OltpMessagePart`-Objekte werden von einer openUTM-Partneranwendung auf Nachrichtenteile abgebildet.

  `OltpMessageRecord`-Objekte und `OltpMessagePart`-Objekte akzeptieren folgende Datentypen:

  - `byte[]`
  - `String`
  - `ByteContainer`

- Für String-orientierten Zugriff auf EIS-Anwendungen:

  `EISConnectionString`

Für die Code-Konvertierung während des Kommunikationsprozesses steht das Interface `EncodingDef` zur Verfügung.

Weitere Einzelheiten zu diesen Interfaces finden Sie in der JavaDoc zu BeanConnect.

Weitere Funktionalität des Interface EISOltpConnection

Das Interface `EISOltpConnection` erweitert das Interface `EISConnection` um folgende zusätzliche Funktionalität:

- Methoden speziell für die asynchrone Kommunikation (siehe „Asynchrone Kommunikation“ auf Seite 452): `setDelayTime()`, `getDelayTime()`
- Methode `setEndConversation()`.
  Diese Methode legt fest, ob die EIS Partneranwendung die aktuelle Conversation mit dem nächsten Aufruf `send...()` beenden darf oder nicht.

Damit Sie die zusätzliche, vom Interface `EISOltpConnection` bereitgestellte Funktionalität verwenden können, müssen Sie das Objekt `EISConnection`, das Sie mit dem Aufruf `getConnection()` vom Objekt `EISConnectionFactory` erhalten, auf den Typ `EISOltpConnection` casten:

```
con = (EISOltpConnection)cf.getConnection(...);
```

Weitere Funktionalität des Interfaces EISUpicConnection

Das Interface `EISUpicConnection` ist nur relevant, wenn der EIS Partner eine openUTM-Anwendung ist. Es erweitert das Interface `EISConnection` um folgende zusätzliche Funktionalität:

- „Emulation“ von Terminalfunktionen

  openUTM-Conversations, die für Terminals programmiert wurden, können ebenfalls mit Hilfe von BeanConnect angesprochen werden. Die “Emulation” von Terminalfunktionen ist nur mit dem Interface `EISUpicConnection` möglich. Terminalfunktionen wie z.B. Funktionstasten und Cursorpositionen können über dieses Interface genutzt werden.

- Verwendung von Formatnamen (Formatkennzeichen), beim Senden oder Empfangen von Daten

  Beim Datenaustausch zwischen dem Resource Adapter und der openUTM-Partneranwendung können auch Formatnamen (Formatkennzeichen) versendet werden. Zusammen mit den Benutzerdaten kann der Resource Adapter somit Datenstrukturinformationen an den openUTM-Partner senden oder von ihm empfangen. Diese Funktion wird nur unterstützt, wenn der gewünschte Service das KDCS-Interface verwendet.

- Restart-Funktion

  Die openUTM-Partneranwendung führt einen automatischen Service-Restart für Benutzerkennungen durch, die über die Steueranweisung `USER` in Verbindung mit dem Operanden `RESTART=YES` definiert wurden.

- Methode `isInTransaction()`, um den Transaktionsstatus beim EIS Partner abzufragen.

Damit Sie die von dem Interface `EISUpicConnection` zusätzlich bereitgestellte Funktionalität verwenden können, müssen Sie das Objekt `EISConnection` (das Sie mit dem Aufruf `getConnection()` vom Objekt `EISConnectionFactory` erhalten) in den Typ `EISUpicConnection` umwandeln:

```
con = (EISUpicConnection)cf.getConnection(...);
```

10.2.1.3 Kommunikation unter Verwendung von Connection-Interfaces

Die Connection-Interfaces stellen verschiedene Funktionen bereit, die bei der Kommunikation einer EJB mit einer EIS-Anwendung einzeln oder kombiniert verwendet werden können:

- Dialogbasierte Kommunikation (basiert auf den Interfaces `EISConnection`, `EISOltpConnection` und `EISUpicConnection`)
- Asynchrone Kommunikation (basiert auf dem Interface `EISConnection` und auf weiteren Methoden des Interfaces `EISOltpConnection`)
- Transaktionale Kommunikation
- Connection Groups
- Code-Konvertierung

Die folgenden Abschnitte enthalten Informationen zu diesen Themen. Beispiele der am meisten verbreiteten Kommunikations-Szenarien werden in der JavaDoc zu BeanConnect beschrieben.

Dialogbasierte Kommunikation

Die empfohlene Kommunikation zwischen einer EJB und einer EIS-Anwendung verwendet die Methoden des Interfaces `EISConnectionOltpMessage`. Über dieses Interface werden Objekte des Typs `OltpMessage` mit der EIS-Anwendung ausgetauscht. Das Objekt `OltpMessage` dient als Container für den Nachrichteninhalt, der sich aus einem oder mehreren `OltpMessageRecord`-Objekten und/oder aus einem oder mehreren `OltpMessagePart`-Objekten zusammensetzt. Während ein `OltpMessageRecord` beliebig lang sein kann, darf die Länge eines `OltpMessagePart` 32767 Bytes nicht übersteigen.

Die Daten, die gesendet werden sollen, werden zwar zum Zeitpunkt des Aufrufs von `sndOltpMessage()` sofort an BeanConnect übergeben, aber noch nicht zur EIS-Anwendung transferiert. Das tatsächliche Senden einer Anforderung an die EIS-Anwendung geschieht, wenn die erste `rcvOltpMessage()`-Methode für diese Verbindung aufgerufen wird.

Am einfachsten lässt sich ein Dialog-Service in einer EIS-Anwendung mit der Methode `call()` aufrufen. Diese Methode ermöglicht eine RPC-ähnliche Kommunikation mit einer EIS-Anwendung.

Die folgenden Abschnitte geben einen Überblick über die Kommunikation zwischen einer EJB und einem openUTM-/CICS-Programm mit Hilfe von `OltpMessage`-Objekten.

Die unten aufgelisteten Optionen stehen für den Datenaustausch zwischen einer EJB und einem openUTM-/CICS-Programm mit `OltpMessage`-Objekten zur Verfügung:

- Auf OltpMessagePart-Objekten basierender Datenaustausch
- Auf OltpMessageRecord-Objekten basierender Datenaustausch

Weitere Beispiele für verbreitete Kommunikations-Szenarien sowie Code-Beispiele finden Sie in der JavaDoc zu BeanConnect.

Auf OltpMessagePart-Objekten basierender Datenaustausch bei openUTM-Partnern

Mit `MGET` und `MPUT` kann ein openUTM-Programm eine oder mehrere Nachrichtenteile senden oder empfangen. Die Nachrichtenteile dürfen dabei die Größe von 32 KB nicht überschreiten. Hier entsprechen die einzelnen `MessagePart`-Objekte genau den `MGET`- und `MPUT`-Aufrufen im openUTM-Programm.

Der Datenaustausch läuft daher gemäß dem folgenden Schema ab:

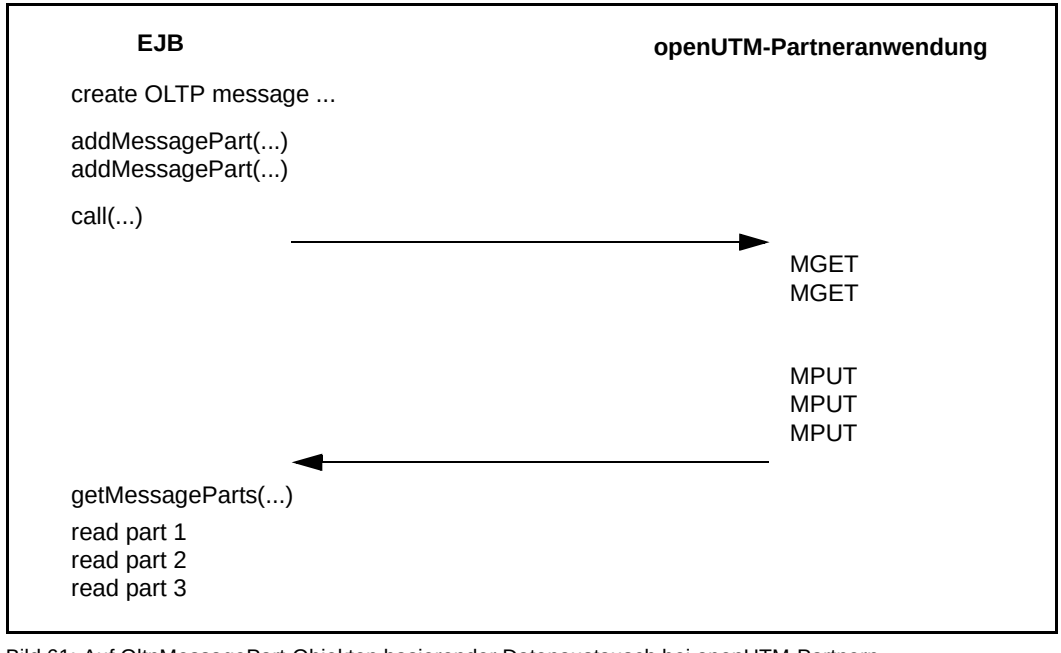


Bild 61: Auf OltpMessagePart-Objekten basierender Datenaustausch bei openUTM-Partnern

Auf OltpMessagePart-Objekten basierender Datenaustausch bei CICS-Partnern

Mit `RECEIVE` und `SEND` kann ein CICS-Programm eine oder mehrere Nachrichtenteile senden oder empfangen. Die Nachrichtenteile dürfen dabei die Größe von 32 KB nicht überschreiten. Hier entsprechen die einzelnen `MessagePart`-Objekte genau den `RECEIVE`- und `SEND`-Aufrufen im CICS-Programm.

Der Datenaustausch läuft daher gemäß dem folgenden Schema ab:

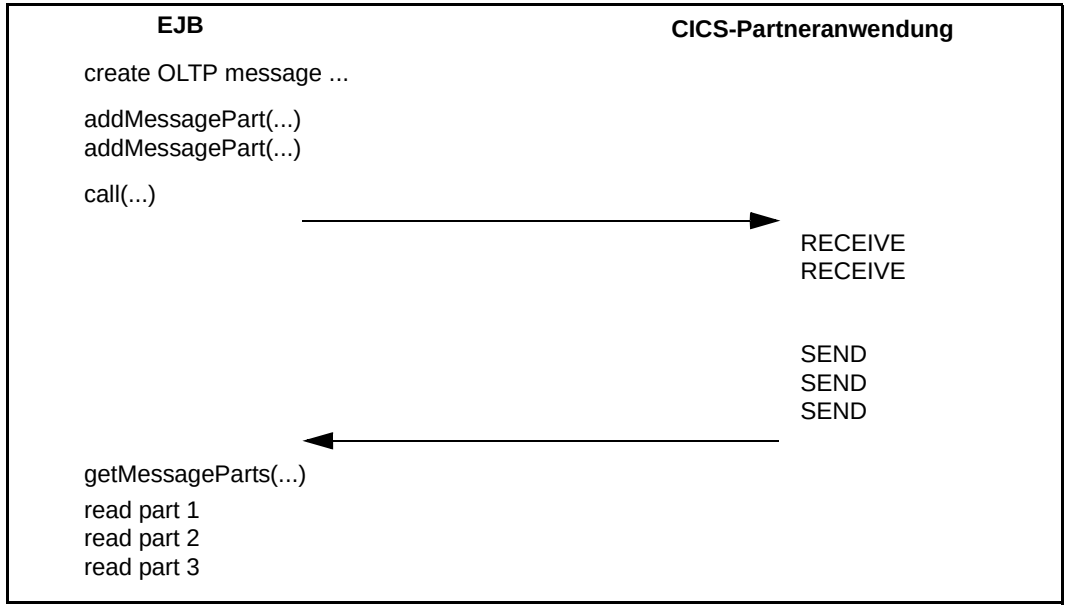


Bild 62: Auf OltpMessagePart-Objekten basierender Datenaustausch bei CICS-Partnern

Auf OltpMessageRecord-Objekten basierender Datenaustausch

Statt mehrerer `MessagePart`-Objekte können Sie innerhalb eines `OltpMessage`-Objekts auch `OltpMessageRecord`-Objekte versenden. Als Java-Programmierer können Sie in diesem Fall Nachrichten senden und empfangen, die größer als 32 KB sind, d.h. Sie müssen sie nicht in 32-KB-Pakete aufteilen.

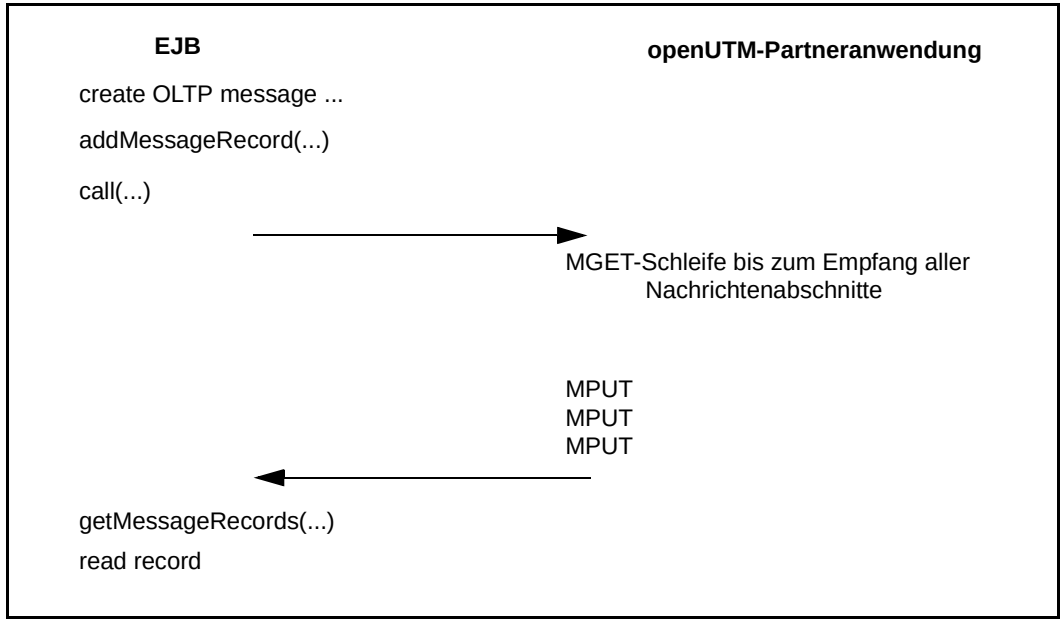


Bild 63: Auf OltpMessageRecord-Objekten (OSI TP Protokoll) basierender Datenaustausch bei openUTM-Partnern

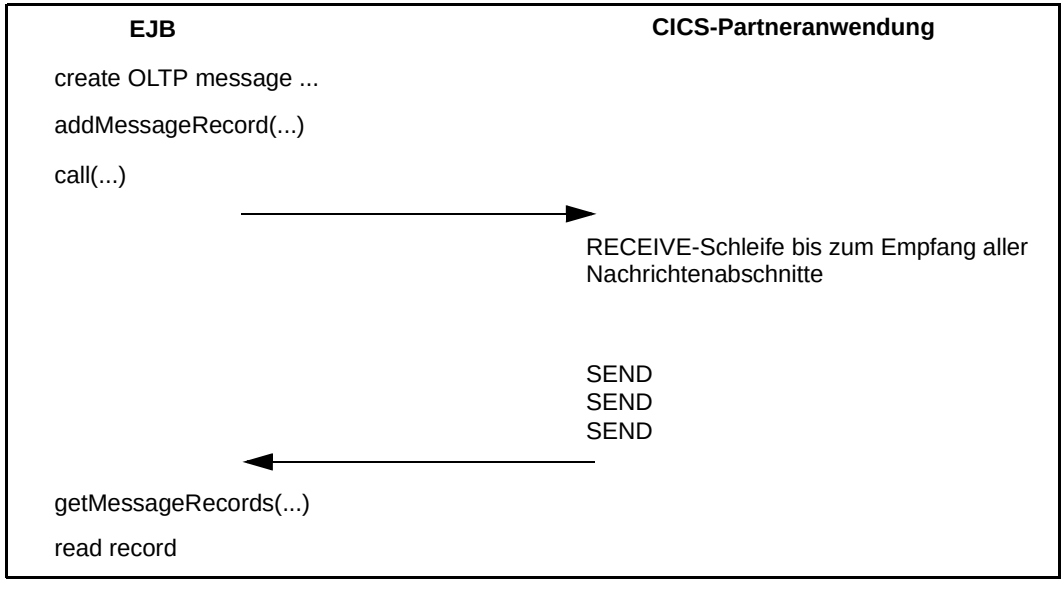


Bild 64: Auf OltpMessageRecord-Objekten basierender Datenaustausch bei CICS-Partnern

Bei der Kommunikation mit einer openUTM-Anwendung über das UPIC-Protokoll können Sie `OltpMessageRecord`-Objekte und `OltpMessagePart`-Objekte verwenden, wenn Sie mit unterschiedlichen Formatnamen arbeiten.

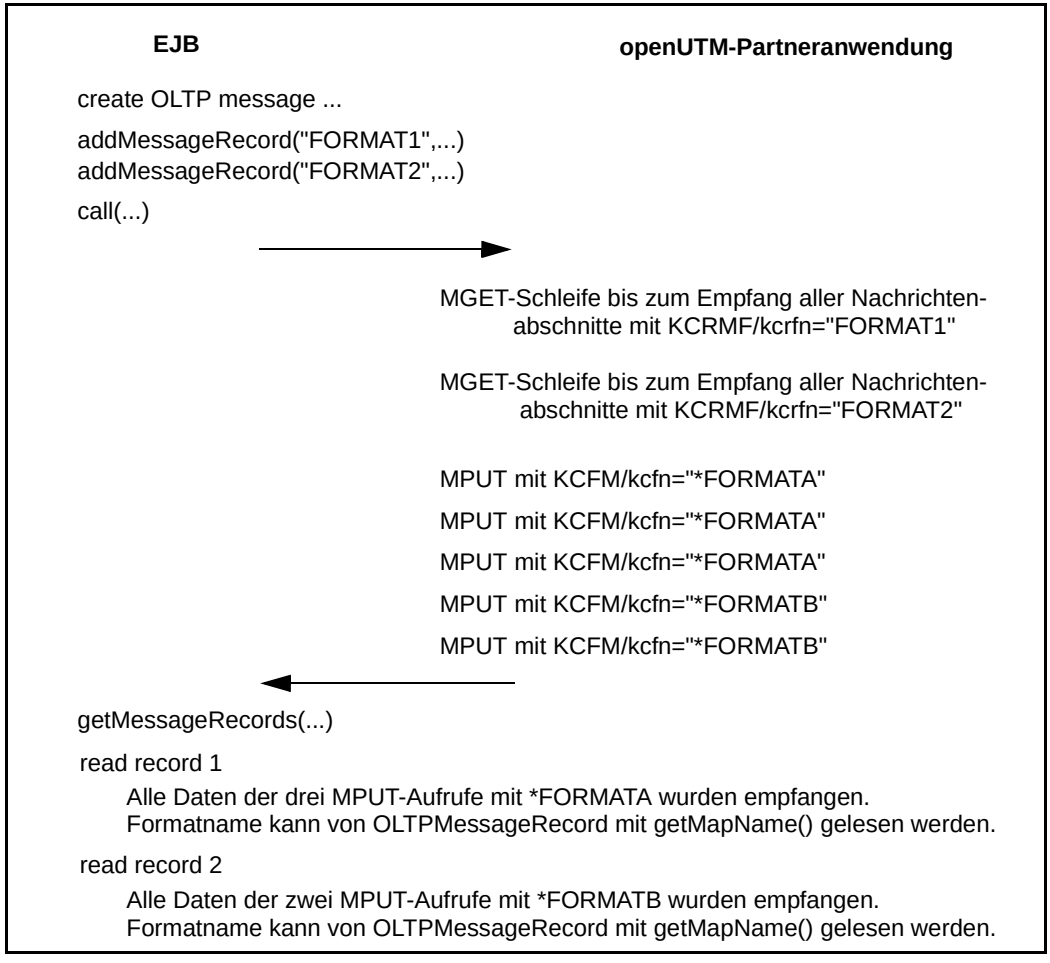


Bild 65: Auf OltpMessageRecord-Objekten (UPIC-Protokoll) basierender Datenaustausch

Asynchrone Kommunikation

Ein Service wird als asynchron bezeichnet, wenn er keine Reply Message sendet.

Das Senden einer Asynchron-Nachricht muss explizit durch die EJB mit den Methoden `snd()` + `flush()`, `sndLast()` oder `sndLastString()` des Interfaces `EISConnection` ausgelöst werden, welche die Nachricht abschließen und den Sendezyklus initiieren.

Eine Asynchron-Nachricht kann entweder sofort oder aber verzögert an den EIS Partner übertragen werden. Für eine verzögertes Senden muss die EJB die Verzögerungszeit vor dem Aufruf der Methode `snd()` festlegen. Die Verzögerungszeit legen Sie mit der Methode `setDelayTime()` fest. Eine verzögerte Nachricht wird für die Dauer der Verzögerungszeit vom BeanConnect Proxy gespeichert. Nach Ablauf dieser Zeit wird die Nachricht an die EIS-Anwendung weitergeleitet.

Eine in einer Transaktion erzeugte asynchrone Nachricht wird nur dann gesendet, wenn die Transaktion erfolgreich beendet wird. Im Falle eines Rücksetzens der Transaktion wird die asynchrone Nachricht nicht gesendet. Eine außerhalb einer Transaktion erzeugte asynchrone Nachricht mit Zeitverzögerung 0 wird sofort gesendet.

Wenn Sie sicherstellen möchten, dass aktuell ein asynchroner Service angesprochen wird, können Sie dies mit der Methode `getPartnerType()` überprüfen. Oder Sie verwenden die Methode `setDelayTime(0)`, die eine Exception auslöst, wenn unerwarteter Weise ein Dialog-Service angesprochen wird. Die Verwendung dieser Methoden ist allerdings mit einem gewissen Performance-Nachteil verbunden.

Beispiele für asynchrone Kommunikationen mit einem EIS Partner finden Sie in der JavaDoc zu BeanConnect.

Transaktionale Kommunikation

Während des Deployments einer Connection Factory muss die Property `transactional` eingestellt werden (siehe „transactional“ auf Seite 116). Eine Connection Factory kann sowohl transaktionale als auch nicht-transaktionale Kommunikation unterstützen, sie kann aber auch auf nicht-transaktionale Kommunikation beschränkt sein:

- Wenn die Connection Factory transaktionale Kommunikation unterstützt, entscheidet die Laufzeitumgebung zu Beginn der Transaktion, ob für eine Conversation dieser Verbindung die Kommunikation transaktional ist oder nicht: Eine Conversation, die gestartet wird während eine Transaktion offen ist, wird immer in die Transaktion eingebunden.

Eine Conversation, die beim Start der Transaktion bereits existierte, aber über die bis zu diesem Zeitpunkt noch keine Kommunikation erfolgte, wird ebenso in die Transaktion eingebunden. In diesen Fällen wird für die Kommunikation ein transaktionales Protokoll, in allen anderen Fällen ein nicht transaktionales Protokoll verwendet.

- Folgendes gilt, falls die Connection Factory keine transaktionale Kommunikation unterstützt: Für Verbindungen, die mit dieser Connection Factory erzeugt wurden, wird immer ein nicht-transaktionales Protokoll verwendet. Dabei spielt es keine Rolle, ob die Kommunikation inner- oder außerhalb der Application Server Transaktion stattfindet. Dies ermöglicht dem Deployer des Resource Adapters, einen EIS-Service ausdrücklich von einer Transaktion auszuschließen, die im Application Server aktiv sein könnte.

Den aktuellen Transaktionsstatus können Sie mit der Methode `isInDistributedTransaction()` des Interfaces `EISOltpConnection` abfragen.

Wenn ein transaktionales Protokoll für die Kommunikation verwendet wird, sind die Transaktionen im Application Server und im EIS Partner Teil einer einzigen, verteilten Transaktion, die als eine Einheit vor- oder zurückgesetzt wird. Bei transaktionaler Kommunikation darf sich eine Conversation nicht über mehr als eine Transaktion erstrecken. Eine Conversation kann jedoch mehr als einen Dialogschritt umfassen. Somit können innerhalb einer Transaktion mehrere `snd()`/`rcv()`-Paare mit der EIS Partner ausgetauscht werden.

Wenn eine EJB innerhalb einer transaktionalen EJB-Methode mehr als eine transaktionale Verbindung verwendet, sollten diese Verbindungen unbedingt in **einer** `EISConnectionGroup` vereinigt werden (siehe „Connection Groups“ auf Seite 453). Auf diese Weise lässt sich eine Ressourcenverknappung vermeiden.

Connection Groups

Das Konzept der Connection Groups ermöglicht einer EJB, gleichzeitig mehrere Nachrichten über mehr als eine Verbindung zu senden. Somit können gleichzeitig mehrere Dialog-Services einer oder mehrerer EIS Partneranwendungen verarbeitet werden, ohne dass es zu einer Zeitverzögerung kommt, wie sie durch die sequentielle Verarbeitung von RPC-ähnlichen Aufrufen verursacht wird.

Mit Hilfe einer `EISConnectionGroup` können Sie eine Verbindung mit einer anderen Verbindung verknüpfen. Die `EISConnectionGroup` erstellen sie mit Hilfe einer `EISConnectionGroupFactory`. Eine `EISConnectionGroupFactory` erhält man mit der Methode `getEISConnectionGroupFactory()` von einer `ConnectionFactory`.

Beispiel:

```
ConnectionFactory cf1 = (ConnectionFactory)ic.lookup(...);
ConnectionFactory cf2 = (ConnectionFactory)ic.lookup(...);
EISConnectionGroupFactory cgf =
                        cf1.getEISConnectionGroupFactory();
```

```
EISConnectionGroup cg = cgf.getConnectionGroup();
con1 = (EISOltpConnection)cf1.getConnection(cg);
con2 = (EISOltpConnection)cf2.getConnection(cg);
```

Ein Code-Beispiel finden Sie im Beispiel 16 auf Seite 468.

Das gleichzeitige Senden der Nachrichten auf allen Verbindungen einer ConnectionGroup wird durch den Aufruf der Methode `execute()` am `EISConnectionGroup`-Objekt ausgelöst. Verbindungen für transaktionale Kommunikation können mit Verbindungen für nicht-transaktionale Kommunikation verknüpft werden. Weitere Einzelheiten hierzu finden Sie in der JavaDoc zu BeanConnect.

Code-Konvertierung

Wann immer `String`- oder `ByteContainer`-Objekte verwendet werden, kann BeanConnect eine Code-Konvertierung durchführen. Die Methoden für das Einrichten der ordnungsgemäßen Umgebung für eine Code-Konvertierung sind in dem Interface `net.fsc.beanta.encoding.EncodingDef` enthalten. Dieses Interface wird durch die `EISConnection`-Interfaces von BeanConnect erweitert. Wenn die Code-Konvertierung zum Deployment-Zeitpunkt noch nicht mit der Konfigurations-Property `encodingActive` aktiviert wurde (siehe „encodingActive“ auf Seite 115), können Sie die Code-Konvertierung über die Methode `setEncodingActive()` einschalten. Die für eine Code-Konvertierung benötigte Code-Tabelle wird während des Deployments beim Definieren der Konfigurations-Property `encoding` zugeordnet (siehe „encoding“ auf Seite 113) oder über die Methode `setEncoding()` des Interfaces `EISConnection`. Weitere Einzelheiten zur Code-Konvertierung von Nachrichten und zur Bereitstellung eigener Konvertierungstabellen finden Sie in Kapitel „Zeichensatz-Konvertierung und Sprachunterstützung“ auf Seite 489) und in der JavaDoc des Packages `net.fsc.beanta.encoding`.

10.2.2 Common Client Interface (CCI) für Outbound-Kommunikation

Die CCI-Interfaces für Outbound-Kommunikation sind in den Packages `javax.resource.cci` und `net.fsc.jca.communication.cci` enthalten. Bei Outbound-Kommunikationen bietet das CCI weitgehend dieselbe Funktionalität wie die BeanConnect-spezifischen Interfaces (ausgenommen ist die von dem Interface `EISUpicConnection` für openUTM-Partner bereitgestellte zusätzliche Funktionalität). Einzelheiten zu dem Programm-Framework, das das CCI-Interface verwendet, finden Sie im Abschnitt „Programm-Framework für Common Client Interface (CCI)“ auf Seite 461.

Connection-Factory-Interfaces

Während des Deployments des Resource Adapters können Sie festlegen, dass Sie Outbound-Kommunikation über CCI verwenden wollen. Dazu geben Sie im Element `<connection-factory-interface>` der Deployment-Descriptor-Datei `weblogic-ra.xml` eines der folgenden Connection-Factory-Interfaces an (siehe „Interfaces auswählen“ auf Seite 441):

- `net.fsc.jca.communication.cci.BCOltpConnectionFactory`
- `net.fsc.jca.communication.cci.BCUpicConnectionFactory`
  (nur wenn der EIS Partner eine openUTM-Anwendung ist)

In Ihrem EJB-Code verwenden Sie eine Connection Factory, um eine Verbindung aufzubauen. Das CCI stellt folgende Connection-Factory-Interfaces bereit:

- `javax.resource.cci.ConnectionFactory`
- `net.fsc.jca.communication.cci.BCOltpConnectionFactory`
- `net.fsc.jca.communication.cci.BCUpicConnectionFactory`
  (nur wenn der EIS Partner eine openUTM-Anwendung ist)

Die Interfaces `BCOltpConnectionFactory` und `BCUpicConnectionFactory` erweitern das Interface `javax.resource.cci.ConnectionFactory`, ohne zusätzliche Funktionalität bereitzustellen.

Die Verwendung der Interfaces `BCOltpConnectionFactory` oder `BCUpicConnectionFactory` ist dann sinnvoll, wenn Sie überprüfen möchten, ob die Kommunikation unter Verwendung des OSI TP Protokolls bzw. des UPIC-Protokolls verarbeitet wird. Beispiel 12 auf Seite 443 gilt analog. Es wird empfohlen, das Interface `javax.resource.cci.ConnectionFactory` zu verwenden.

Connection Interface

Das CCI-Interface für Outbound-Kommunikation heißt `javax.resource.cci.Connection`.

10.2.3 Programmierinformationen zu Outbound-Kommunikation

Dieser Abschnitt enthält Programmierinformationen zur Outbound-Kommunikation zwischen einer EJB und einer EIS Partneranwendung.

10.2.3.1 EIS-Anwendung ansprechen

Der Service einer EIS-Anwendung, der von Ihrer Java EE Anwendung verwendet wird, muss während des Deployments anhand der Konfigurations-Property `connectionURL` konfiguriert werden (siehe Abschnitt „connectionURL“ auf Seite 112). Wenn die Java EE Anwendung mehrere Services des selben EIS Partners in einem EIS verwendet, kann die Methode `setServiceName()` des Connection-Objekts verwendet werden, um explizit einen bestimmten Service zu adressieren:

```
connection.setServiceName(<service_name>);
```

10.2.3.2 Aufrufe von BeanConnect in einer EJB platzieren

Im JNDI-Service des Application Servers müssen Sie einen `lookup()` für eine zuvor konfigurierte Connection Factory ausführen. Eine Connection Factory stellt eine `getConnection()`-Methode bereit, die ein `Connection`-Objekt zurückgibt. Beim Deployment für die Connection Factory wurden die Properties dieser Connection (EIS-Adresse, EIS-Service-Name etc.) vorkonfiguriert. Unabhängig von dem beim Deployment angegebenen Connection-Factory Typ (`EISUpicConnectionFactory` oder `EISOltpConnectionFactory`), erhalten Sie ein `Connection`-Objekt, das das Interface `EISConnection` implementiert. Wenn die Anwendung ein `Connection`-Objekt nicht länger benötigt, muss sie es dem Application Server zum Pooling oder Löschen zurückgeben, indem sie die Methode `close()` für das `Connection`-Objekt aufruft.

Die Anwendung muss sicherstellen, dass sie angeforderte Connections auch in Fehlerfällen wieder mit `close()` frei gibt. Andernfalls kann es zu Folgefehlern im Application Server kommen.

Es wird empfohlen, das JNDI-Interface `lookup()` für eine Connection Factory während der Initialisierung auszuführen. In einer EJB geschieht dies z.B. innerhalb der Methoden `ejbCreate()` oder `setSessionContext()`.

Die Methode `getConnection()` sowie die damit verknüpfte Methode `close()` müssen direkt innerhalb der Business-Methoden aufgerufen werden.

10.2.3.3 Authentisierung (Benutzerkennung und Passwort)

Die Authentisierung findet anhand des Benutzernamens und des Passwortes statt. Dabei wird unterschieden zwischen

- Containergesteuerte Authentisierung
- Anwendungsgesteuerte Authentisierung

Es wird empfohlen, die containergesteuerte Authentisierung zu verwenden.

Containergesteuerte Authentisierung

Bei einer containergesteuerten Authentisierung werden die Zugriffsdaten durch den Container verwaltet. Der EJB-Deployer konfiguriert im EJB-Deployment-Descriptor eine containergesteuerte Authentisierung mit folgendem Eintrag:

```
<res-auth>Container</res-auth>
```

Bei der Verwendung einer containergesteuerten Authentisierung rufen Sie die Methode `getConnection()` ohne Parameter auf.

Details siehe auch Abschnitt „„Containergesteuerte Authentisierung (container-managed)“ auf Seite 118“.

Anwendungsgesteuerte Authentisierung

Bei einer anwendungsgesteuerten Authentisierung werden die Zugriffsdaten im Programmcode der EJB verwaltet. Der EJB-Deployer konfiguriert im EJB-Deployment-Descriptor eine anwendungsgesteuerte Authentisierung mit folgendem Eintrag:

```
<res-auth>Application</res-auth>
```

Im EJB-Quellcode verwenden Sie anstelle des parameterlosen Aufrufs `getConnection()` z.B. folgende Code-Sequenz:

```
javax.naming.InitialContext ic = new InitialContext();

String user = (String)ic.lookup("java:comp/env/User");
String password = (String)ic.lookup("java:comp/env/Password");
net.fsc.jca.communication.PasswordCredential pwc =
  new net.fsc.jca.communication.PasswordCredential
    (user, password);
con= (net.fsc.jca.communication.EISConnection)cf.getConnection(pwc);
```

Benutzerkennung und Passwort in diesem Beispiel werden als Umgebungsvariablen der EJB definiert. Der Deployer kann Umgebungsvariablen erforderlichenfalls anpassen. Auf die Umgebungsvariablen kann mit der Methode `lookup()` zugegriffen werden.

10.2.3.4 Informationen zur Conversation mit der EIS-Anwendung abfragen

Mit der Methode `isInConversation()` des Connection-Objekts können Sie den Status der EIS-Anwendung abfragen.

Beispiel 13 Informationen zur Conversation mit der EIS-Anwendung

So können Sie sicherstellen, dass die Conversation mit der EIS-Anwendung nach Abschluss der Methodenausführung beendet wird:

```
...
String s = con.call("what will be the echo of this");

if (eis.isInConversation())
{
 con.terminate();
 con.close();
 throw new EJBException
   ("EJB Exception: EIS Partner Service not yet terminated ... ");
}
```

10.2.3.5 Programmierhinweise für CICS-Anwendungen

CICS-Transaktionen müssen so entwickelt und codiert sein, dass sie den Prinzipien des Distributed Transaction Programming (DTP) entsprechen. Eine Beschreibung dieser Programmier-Prinzipien finden Sie in der CICS-Dokumentation von IBM, z.B. „CICS Distributed Transaction Programming Guide“.

Die folgenden Einschränkungen und Regeln müssen bei der Outbound-Kommunikation mit BeanConnect beachtet werden:

- Eine CICS-Partneranwendung darf niemals eigenständig `SYNCPOINT` oder `ISSUE PREPARE` verwenden, sondern nur wenn sie von der EJB im Java EE Application Server dazu aufgefordert wird.
- Basic Conversation ist nicht möglich. Basic Conversation wird für CICS mit den Befehlen programmiert, die mit GDS beginnen, wie z.B. `EXEC CICS GDS ALLOCATE`.
- BeanConnect baut LU6.2-Conversations immer mit SYNCLEVEL 0 oder 2 auf, niemals mit SYNCLEVEL 1. Der SYNCLEVEL einer eingehenden Conversation kann beim CICS-API mit `EXTRACT PROCESS` abgefragt werden.
- Der Default-Wert der Kommunikationseigenschaft `endConversation` ist bei CICS-Partnern `false`, bei openUTM-Partnern `true`.

10.2.3.6 Unterstützung von DPL-Programmen (Distributed Program Link) für CICS-Anwendungen

CICS stellt verschiedene Programmierschnittstellen bereit, um ein anderes CICS-Programm aufzurufen oder um von einem anderen CICS-Programm aufgerufen zu werden. Zwei dieser Möglichkeiten sind in diesem Zusammenhang wichtig.

- DTP (Distributed Transaction Processing) ermöglicht einer CICS-Transaktion durch Datenaustausch mit einer CICS-Anwendung zu kommunizieren, die auf einem anderen System läuft. DTP-Programme werden mit Hilfe der Programmierschnittstelle APPC codiert.
- DPL (Distributed Program Link) ermöglicht einem CICS-Programm ein Programm in einem anderen CICS-System aufzurufen und auf die Antwort des aufgerufenen Programms zu warten. Die Daten werden zwischen den Programmen in einem Kommunikationsbereich (`COMMAREA`) ausgetauscht. DPL ähnelt einem RPC-Aufruf.

BeanConnect ermöglicht den Aufruf von Outbound-Transaktionen mit DTP. Allerdings ist es nicht möglich, ein DPL-Programm direkt aufzurufen. Daher wird ein DTP-Programm benötigt, das die Programmverknüpfung umsetzt.

Ein Beispiel für ein solches COBOL-Programm-Fragment mit dem Namen `DPLSERVR.CCP` finden Sie im Verzeichnis `<BC_home>/<proxy_cont_name>/src` (Solaris/Linux) bzw. `<BC_home>\<proxy_cont_name>\src` (Windows). Diese Quelle enthält Hinweise auf die notwendigen Änderungen für die Erzeugung eines neuen Programms.

Das DPL-Programm wird mit dem CICS-Befehl `LINK` aufgerufen, der drei wichtige Parameter hat:

- `PROGRAM`, um den Namen des Programms anzugeben, dem die Kontrolle uneingeschränkt übergegeben werden soll
- `COMMAREA`, um den Kommunikationsbereich anzugeben, der dem verknüpften Programm zur Verfügung steht
- `LENGTH`, gibt die Größe des Kommunikationsbereichs in Bytes an

Die Input-Daten für den verteilten Programmaufruf werden in einer Nachricht empfangen. Die Input-Daten müssen in den Kommunikationsbereich kopiert werden. Wenn das verknüpfte Programm antwortet, müssen die benötigten Output-Daten als Antwort an die EJB gesendet werden.

10.2.4 Programm-Framework für Outbound-Kommunikation

Dieser Abschnitt enthält ein Programm-Framework für die Outbound-Kommunikation zwischen einer EJB und einer EIS-Anwendung. Das Framework enthält die wesentlichen Kommunikationsschritte.

10.2.4.1 Programm-Framework für BeanConnect-spezifische Interfaces

In BeanConnect geben Sie den EIS Partner an, der beim Deployment einer Managed Connection Factory angesprochen werden soll. Sie verwenden die Methode `lookup()`, um nach der Connection Factory zu suchen und um ein Connectivity Objekt durch den Aufruf der Methode `getConnection()` zu erhalten.

Das bereitgestellte Connectivity Objekt implementiert für die Kommunikation mit der EIS-Anwendung das Interface `EISConnection`:

1. Richten Sie den Initial-Kontext ein:

   ```
   javax.naming.InitialContext ic = new InitialContext();
   ```

2. Referenzieren Sie eine Connection Factory:

   ```
   net.fsc.jca.communication.EISConnectionFactory cf =
     (EISConnectionFactory)ic.lookup
       ("java:comp/env/<resource_reference_name>");
   ```

3. Richten Sie die Verbindung ein:

   ```
   net.fsc.jca.communication.EISConnection con = (EISConnection) cf.getCon-
   nection();
   ```

4. Wenn Sie das Interface `EISConnection` verwenden, um den Service-Namen (TAC) bzw. den Namen des EIS-Services festzulegen, gehen Sie wie folgt vor:

   ```
   con.setServiceName(<name_of_the_service>);
   ```

   Beachten Sie jedoch, dass es aus Performance-Gründen empfehlenswert ist, mit den vorkonfigurierten Service-Namen zu arbeiten.

5. Bilden Sie die zu sendende Nachricht:

   ```
   String requestMessage = "...";
   ```

6. Rufen Sie die EIS-Anwendung auf und empfangen Sie die Reply Message:

   ```
   String replyMessage = con.call(requestMessage);
   ```

7. Schließen Sie die Verbindung:

   ```
   con.close();
   ```

Weitere Einzelheiten zur Programmierung der Interfaces `EISConnection` und `EISOltpConnection` finden Sie in der JavaDoc zu dem entsprechenden Interface.

Ein Beispiel finden Sie im Beispiel 14 auf Seite 466.

10.2.4.2 Programm-Framework für Common Client Interface (CCI)

In BeanConnect geben Sie beim Deployment einer Managed Connection Factory den EIS Partner an, der über diese Managed Connection Factory angesprochen werden soll. Sie verwenden die Methode `lookup()`, um nach der Connection Factory zu suchen und um ein Connectivity Objekt durch den Aufruf der Methode `getConnection()` zu erhalten.

Über die CCI-Verbindung, die Sie von der CCI-Connection Factory erhalten, können Sie ein Interaction Objekt anfordern. Das Interaction Objekt implementiert eine `execute()`-Methode, um eine Interaktion zu initiieren. Die Methode `execute()` kennt außer einem `BCCciInteractionSpec`-Objekt auch noch Input- und Output-Datensatz. In diesem `BCCciInteractionSpec`-Objekt definieren Sie ein `interactionVerb` (`SYNC_SEND`, `SYNC_SEND_RECEIVE` oder `SYNC_RECEIVE`) und gleichzeitig den Namen der EIS-Anwendung.

Somit können Sie eine Interaktion durch Bereitstellung passender Daten steuern, anstatt Methoden aufzurufen. Die Standardeinstellung für `interactionVerb` lautet `SYNC_SEND_RECEIVE`.

Weitere Einzelheiten hierzu finden Sie in der JavaDoc zu BeanConnect.

Programm-Framework für dialogbasierte Kommunikation (CCI)

Für dialogbasierte Kommunikation mit der Serveranwendung mittels des CCI-Interfaces in BeanConnect führen Sie folgende Aktionen aus:

1. Richten Sie den Initial-Kontext ein:

   ```
   javax.naming.InitialContext ic = new InitialContext();
   ```

2. Referenzieren Sie eine Connection Factory:

   ```
   javax.resource.cci.ConnectionFactory cf =
     (ConnectionFactory)ic.lookup("java:comp/env/eis/myEIS");
   ```

   Der von `eis/myEIS` referenzierten Connection Factory wird ein Dialog-Service als Standard-Service zugewiesen.

3. Richten Sie die Verbindung ein:

   ```
   javax.resource.cci.Connection con =(Connection)cf.getConnection();
   ```

   Alternativ kann eine EJB sicherheitsbezogene Daten (Benutzerkennung/Passwort) in einem `BCCciConnectionSpec` Objekt an BeanConnect weiterleiten. In diesem Fall geben Sie ein:

```
net.fsc.jca.communication.cci.BCCciConnectionSpec;
cred = new BCCciConnectionSpec("myuser", "mypass");

javax.resource.cci.Connection con =    (Connection)cf.getConnection(cred);
```

4. Erstellen Sie ein `Interaction`-Objekt und ein `InteractionSpec`-Objekt:

   ```
   Interaction ix = (Interaction)con.createInteraction();
   BCCciInteractionSpec is = new
      BCCciInteractionSpec(InteractionSpec.SYNC_SEND_RECEIVE);
   ```

   Während ein `Interaction`-Objekt aus dem `Connection`-Objekt erstellt wird, für das es verwendet werden soll, wird das `InteractionSpec`-Objekt mit einem Konstruktor der Implementierungsklasse erstellt.

   Ein `Interaction`-Objekt ermöglicht einer EJB die Kommunikation mit einer EIS-Anwendung. Ein `InteractionSpec`-Objekt verfügt über Properties, mit denen eine Interaktion mit dieser EIS-Anwendung durchgeführt werden kann. Es wird von einer Interaktion zur Ausführung der angegebenen Funktion in der EIS-Anwendung verwendet.

5. Erstellen Sie ein `BCRecord`-Objekt, das als Container für die Nachricht an das EIS dient, sowie ein weiteres `BCRecord`-Objekt, das als Container für die Reply Message dient. Dies geschieht mittels eines `BCRecordFactory`-Objekts:

   ```
   net.fsc.jca.communication.cci.BCRecordFactory rf =
      (BCRecordFactory)cf.getRecordFactory();
   net.fsc.jca.communication.cci.BCRecord reqrec = (BCRecord)
   rf.createBCRecord("request");
   net.fsc.jca.communication.cci.BCRecord replrec = (BCRecord)
   rf.createBCRecord("reply");
   ```

   Nach dem Erstellen enthält ein `BCRecord`-Objekt ein leeres `OltpMessage`-Objekt, das aus dem `BCRecord`-Objekt abgerufen werden kann. In der Folge kann das `BCRecord`-Objekt mit `OltpMessageRecord`- oder `OltpMessagePart`-Objekten bestückt werden.

6. Versorgen Sie das `OltpMessage`-Objekt der für das EIS bestimmten Nachricht mit Daten (hier: zwei `OltpMessagePart`-Objekte):

   ```
   net.fsc.jca.communication.OltpMessage request =
   reqrec.getOltpMessage();
   request.addMessagePart("request – message part1");
   request.addMessagePart("request – message part2");
   ```

7. Führen Sie die Interaktion aus:

   ```
   ix.execute(is, reqrec, replrec);
   ```

   Das aus diesem Aufruf zurückgegebene `BCRecord`-Objekt enthält wieder ein `OltpMessage`-Objekt, das seinerseits die Antwort enthält, die von der EIS-Anwendung in einem `OltpMessageRecord` oder in einem `OltpMessagePart`-Objekt gesendet wurde.

8. Reply Message empfangen:

```
net.fsc.jca.communication.OltpMessage reply =
replrec.getOltpMessage();
java.util.Iterator<net.fsc.jca.communication.OltpMessagePart> it =
reply.getMessageParts();

net.fsc.jca.communication.OltpMessagePart msgPart;
String msgText="";

while (it.hasNext()) {
   msgPart = (OltpMessagePart) it.next();
   msgText += msgPart.getText();
}
```

9. Schließen Sie die Verbindung:

```
con.close();
```

Weitere Informationen zur Programmierung der CCI-Interfaces finden Sie in der JavaDoc zur CCI.

Ein Code-Beispiel finden Sie im Beispiel 15 auf Seite 467.

Programm-Framework für asynchrone Kommunikation (CCI)

Für asynchrone Kommunikation mit der Serveranwendung mittels des CCI-Interfaces in BeanConnect führen Sie folgende Aktionen aus:

1. Richten Sie den Initial-Kontext ein:

```
javax.naming.Context ic = new InitialContext();
```

2. Referenzieren Sie eine Connection Factory:

```
javax.resource.cci.ConnectionFactory cf =
(ConnectionFactory)ic.lookup("java:comp/env/eis/myAsyncEIS");
```

3. Richten Sie die Verbindung ein:

```
javax.resource.cci.Connection con =(Connection)cf.getConnection();
```

   Alternativ kann eine EJB sicherheitsbezogene Daten (Benutzerkennung/Passwort) in einem `BCCciConnectionSpec` Objekt an BeanConnect weiterleiten. In diesem Fall geben Sie ein:

```
net.fsc.jca.communication.cci.BCCciConnectionSpec cred =
   new BCCciConnectionSpec("myuser", "mypass");

 javax.resource.cci.Connection con =    (Connection)cf.getConnection(cred);
```

4. Erstellen Sie ein `Interaction`-Objekt und ein `InteractionSpec`-Objekt:

   Ein `Interaction`-Objekt ermöglicht einer EJB die Kommunikation mit einer Partneranwendung. Ein `InteractionSpec`-Objekt verfügt über Properties, mit denen eine Interaktion mit einer Partneranwendung durchgeführt werden kann. Es wird von einer Interaktion zur Ausführung der angegebenen Funktion in der EIS-Anwendung verwendet.

   Während ein `Interaction`-Objekt aus dem `connection`-Objekt erstellt wird, für das es verwendet werden soll, wird das `InteractionSpec`-Objekt mit einem Konstruktor der Implementierungsklasse erstellt:

   ```
   Interaction ix = (Interaction)con.createInteraction();
   net.fsc.jca.communication.cci.BCCciInteractionSpec is = new
      BCCciInteractionSpec(InteractionSpec.SYNC_SEND,
                                                  "ASYNTAC");
   ```

   Die Zuweisung des asynchronen Services `ASYNTAC` legt fest, dass asynchrone Kommunikation stattfindet.

   Hier erzielt `SYNC_SEND` dieselbe Wirkung wie ein `flush()`-Aufruf bei der Verwendung der BeanConnect-spezifischen Interfaces.

5. Erstellen Sie ein `BCRecord`-Objekt, das als Container für die Anforderungsnachricht dient. Dies geschieht mit Hilfe einer `recordFactory`:

   ```
   net.fsc.jca.communication.cci.BCRecordFactory rf=
      (BCRecordFactory)cf.getRecordFactory();
   net.fsc.jca.communication.cci.BCRecord reqrec =
   rf.createBCRecord("request");
   ```

   Nach dem Erstellen enthält ein `BCRecord`-Objekt ein leeres `OltpMessage`-Objekt, das aus dem `BCRecord`-Objekt abgerufen werden kann, und dem dann `OltpMessageRecord`-Objekte und/oder `OltpMessagePart`-Objekte zugewiesen werden können.

6. Versorgen Sie das `OltpMessage`-Objekt des Anforderungsdatensatzes mit Daten (hier: zwei `OltpMessagePart`-Objekte):

   ```
   net.fsc.jca.communication.OltpMessage request =
   reqrec.getOltpMessage();
   request.addMessagePart("request – message part1");
   request.addMessagePart("request – message part2");
   ```

7. Führen Sie die Interaktion aus:

   ```
   ix.execute(is, reqrec);
   ```

8. Schließen Sie die Verbindung:

   ```
   con.close();
   ```

10.2.5 Outbound-Kommunikation mit XATMI-Partnern

Wenn Sie die Outbound-Kommunikation mit XATMI-Partnern programmieren, dann sind folgende Besonderheiten zu beachten:

- Transaktionssicherung

  Ob die Kommunikation mit der Partneranwendung mit oder ohne Commit FU erfolgt (TRAN oder NOTRAN bei XATMI), wird durch die Transaktionsumgebung im Anwenderprogramm und die transaktionale Eigenschaft der `ConnectionFactory` bestimmt.

- Typisierte Puffer

  Es werden nur Typisierte Puffer vom Typ X_OCTET unterstützt.

- Nachrichtenlänge und Nachrichtenteile

  Bei Verwendung der OltpMessage-Schnittstelle darf ein Nachrichtenteil einer an einen XATMI-Partner gerichteten Nachricht maximal 32.000 Bytes lang sein. Größere Nachrichtenteile werden mittels einer `OltpMessageException` abgewiesen. Die gleiche Restriktion auf 32.000 Bytes für einen Nachrichtenteil gilt für alle Methoden der EISConnection Interfaces, für die bei anderen Partnern eine Begrenzung auf 32.767 Bytes gilt.

  Nachrichten an XATMI-Partner im Request/Reply Mode dürfen nur einen Nachrichtenteil umfassen und dieser darf maximal 32.000 Bytes lang sein.

  Nachrichten an XATMI-Partner im Conversational Mode dürfen mehr als einen Nachrichtenteil umfassen. Jeder dieser Nachrichtenteile darf bis zu 32.000 Bytes lang sein.

  Nachrichten, die an BeanConnect über Interface-Methoden übergeben werden, an denen die Nachrichtenlänge nicht beschränkt ist, z.B. `sndRecord(String)`, werden von BeanConnect bei der Kommunikation mit XATMI-Partnern im Conversational Mode in Teilnachrichten von maximal 32.000 Bytes Länge fragmentiert. Für XATMI-Partner im Request/Reply-Mode werden Nachrichten mit einer Länge von mehr als 32.000 Bytes mit einer Exception abgewiesen.

- Der Empfang einer FAILURE_RI wird einer Anwendung mittels einer `EISConnectionException` angezeigt.

- Der Default-Wert der Kommunikationseigenschaft `endConversation` ist bei XATMI-Partnern immer `false`.

10.2.6 Code-Beispiele für Outbound-Kommunikation

Dieser Abschnitt erhält die folgenden Code-Beispiele:

- Dialogbasierte Kommunikation unter Verwendung des Interfaces EISConnection.
- Dialogbasierte Kommunikation unter Verwendung des Interfaces CCI.
- Connection Groups unter Verwendung des Interfaces EISConnectionGroup.

Beispiel 14 Dialogbasierte Kommunikation unter Verwendung des Interfaces EISConnection

```
...
public String callService(String request) throws EJBException
{
  net.fsc.jca.communication.EISConnectionFactory cf = null;
  net.fsc.jca.communication.EISConnection con = null;
  String reply = null;

  try
  {
   javax.naming.InitialContext ic =
     new javax.naming.InitialContext();
   cf = (net.fsc.jca.communication.EISConnectionFactory)
     ic.lookup("java:comp/env/eis/myEIS");
   con= cf.getConnection();

   reply = con.call(request);

   con.close();
   con = null;

   return reply;
  }
  catch (Exception e)
  {
   if (con != null)
   {
    con.close();
   }
   throw new EJBException ("EJB Exception: " + e);
  }
}
```

Damit Sie diese Methode verwenden können, muss der Deployment Descriptor der EJB folgende Informationen beinhalten:

```
<resource-ref>
 <res-ref-name>eis/myEIS</res-ref-name>
  <res-type>net.fsc.jca.communication.EISConnectionFactory</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Unshareable</res-sharing-scope>
</resource-ref>
```

Beispiel 15 Dialogbasierte Kommunikation unter Verwendung des Interfaces CCI

```
...
public String sndRcvJavax(String user, String name, String data)
{

  String retValue = "";
  Connection connection = null;

  try {
    Context ic = new InitialContext();
    ConnectionFactory cf =
      (ConnectionFactory)ic.lookup("java:comp/env/eis/myEIS");

    ConnectionSpec cred = new BCCciConnectionSpec (user, "");
    connection = cf.getConnection(cred);
    Interaction ix = connection.createInteraction();

    InteractionSpec is =
      new BCCciInteractionSpec(
                    InteractionSpec.SYNC_SEND_RECEIVE,"HELLO");
    BCRecordFactory recordFactory =
                  (BCRecordFactory)cf.getRecordFactory();
    BCRecord in = recordFactory.createBCRecord("SendRecord");
    BCRecord out =
              recordFactory.createBCRecord("ReceiveRecord");
    OltpMessage inMsg = in.getOltpMessage();
    inMsg.addMessagePart(data);

    out = (BCRecord)ix.execute(is, in);

    OltpMessage outMsg = out.getOltpMessage();
    Iterator it<OltpMessagePart> = outMsg.getMessageParts();
    OltpMessagePart  msgPart;

    while (it.hasNext()) {
      msgPart = it.next();
```

```
      retValue  += msgPart.getText();
    }

  } catch ( Throwable  ex) {
  // Todo: Fehlerbehandlung
  }// tryCatch
  try    {
    if ( connection != null )
        connection.close();  }
  catch (ResourceException e)    {
    retValue += "\n bei connection.close():\n"+getStackInfo(e);
  }
  return retValue;
} // sndRcvJavax
```

Damit Sie diese Methode verwenden können, muss der Deployment Descriptor der EJB folgende Informationen beinhalten:

```
<resource-ref>
 <res-ref-name>eis/myEIS</res-ref-name>
  <res-type>net.fsc.jca.communication.EISConnectionFactory</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Beispiel 16 Connection Groups unter Verwendung des Interfaces EISConnectionGroup

```
...
EISOltpConnection con1 = null;
EISOltpConnection con2 = null;
  try {
     javax.naming.InitialContext ic =
     new javax.naming.InitialContext();
     cf = (net.fsc.jca.communication.EISConnectionFactory)
     ic.lookup("java:comp/env/eis/myEIS");
     EISConnectionGroupFactory cgf =
                        cf.getEISConnectionGroupFactory();
     EISConnectionGroup cg = cgf.getConnectionGroup();
     con1 = (EISOltpConnection)cf.getConnection(cg, new
                           PasswordCredential("upicusea", ""));
     con2 = (EISOltpConnection)cf.getConnection(cg, new
                           PasswordCredential("upicuseb", ""));
     OltpMessage om1 = con1.createMessage();
     om1.addMessagePart("STAT");
     OltpMessage om2 = con2.createMessage();
```

```
        om2.addMessagePart("osi-con,l=kdcall");
        con1.sndOltpMessage(om1);
        con2.sndOltpMessage(om2);
        cg.execute();
        String s;
        om2 = con2.rcvOltpMessage();
        Iterator iter<OltpMessageRecord> = om2.getMessageRecords();
        for (s = ""; iter.hasNext(); )
            { s+= iter.next()).getText(); }
        String result_o = "";
        result_o = result_o + "OltpConnection: KDCINF osi-con,
                   l=kdcall\n" + s + "\n";
        om1 = con1.rcvOltpMessage();
        iter = om1.getMessageRecords();
        for (s = ""; iter.hasNext(); ) {
             s+= ((OltpMessageRecord)iter.next()).getText(); }
        result_o = addResult_o(result_o, "OltpConnection:
                       KDCINF STAT\n" + s + "\n");
        s = cg.getGroupName();
        result_o = result_o + "OltpConnection: KDCINF STAT\n" +s
                   + "\n");
        con1.close();
        con2.close();
        return result_o;
    } catch(EISConnectionException ex) {
    ...
```

10.3 Inbound-Kommunikation programmieren

Bei Inbound-Kommunikation spricht eine EIS-Anwendung eine OLTP Message-Driven Bean an, die im Application Server deployt ist. Die Kommunikation zwischen der EIS-Anwendung und der OLTP Message-Driven Bean setzt voraus, dass die OLTP Message-Driven Bean BeanConnect mit Hilfe der Management Console bekannt gegeben wurde.

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- OLTP Message-Driven Beans
- Inbound-Kommunikation mit openUTM-Partnern
- Inbound-Kommunikation mit CICS-Partnern
- Inbound-Kommunikation mit anderen EIS Partnern
- Inbound-Kommunikation mit XATMI-Partnern
- BeanConnect-spezifische Interfaces für Inbound-Kommunikation
- Common Client Interface (CCI) für Inbound-Kommunikation
- Code-Beispiel für Inbound-Kommunikation

10.3.1 OLTP Message-Driven Beans

OLTP Message-Driven Beans sind JCA-konforme Message-Endpoint-Anwendungen, die von BeanConnect unterstützt werden. Eine EIS-Anwendung kann über BeanConnect OLTP Message-Driven Beans aufrufen, die in einem Application Server deployt sind.

Um mit einer OLTP Message-Driven Bean kommunizieren zu können, sendet eine EIS-Anwendung eine Nachricht an einen Service, der BeanConnect bekannt ist. BeanConnect leitet die Nachricht an eine OLTP Message-Driven Bean weiter, die für den Message-Endpoint-Namen konfiguriert wurde, der mit diesem Service-Namen verknüpft ist. Die Verknüpfung von Service-Name mit Message-Endpoint-Name wird über die Management Console vorgenommen (siehe Abschnitt „Inbound Message Endpoints konfigurieren“ auf Seite 241).

BeanConnect unterstützt zwei Arten von OLTP Message-Driven Beans:

- OLTP Message-Driven Beans für dialogbasierte Kommunikation
- OLTP Message-Driven Beans für asynchrone Kommunikation

OLTP Message-Driven Beans für dialogbasierte Kommunikation

Eine OLTP Message-Driven Bean für dialogbasierte Kommunikation empfängt Nachrichten von einer EIS-Anwendung und sendet Nachrichten zurück. Das entsprechende BeanConnect-spezifische Interface ist `net.fsc.jca.communication.OltpMessageListener`. Das Interface CCI `javax.resource.cci.MessageListener` entspricht ebenso den Anforderungen für dialogbasierte Kommunikation.

Diese Interfaces ermöglichen es einer OLTP Message-Driven Bean, eine Nachricht von einer EIS-Anwendung zu empfangen und eine Antwortnachricht an die EIS-Anwendung zu senden. Diese Nachrichten können einen oder mehrere Nachrichtenteile enthalten.

OLTP Message-Driven Beans für asynchrone Kommunikation

Eine OLTP Message-Driven Bean für asynchrone Kommunikation kann eine Nachricht von einer EIS-Anwendung empfangen, darf aber keine Antwortnachricht senden. Das entsprechende BeanConnect-spezifische Interface ist `net.fsc.jca.communication.AsyncOltpMessageListener`. Die Nachricht der EIS-Anwendung kann einen oder mehrere Nachrichtenteile enthalten.

10.3.2 Inbound-Kommunikation mit openUTM-Partnern

Bei Inbound-Kommunikation mit einem openUTM-Partner unterstützt BeanConnect außer dem Protokoll OSI TP (transaktional oder nicht-transaktional) auch nicht-transaktionale Transport-Level-Protokolle wie RFC1006 oder das openUTM-Socket-Protokoll.

Damit eine OLTP Message-Driven Bean aufgerufen werden kann, muss die Konfiguration der openUTM-Partneranwendung ordnungsgemäß angepasst werden:

Für asynchrone Kommunikation rufen Sie eine OLTP Message-Driven Bean über das OSI TP Protokoll oder ein Transport-Level-Protokoll auf. Für dialogbasierte Kommunikation hingegen rufen Sie eine OLTP Message-Driven Bean über das OSI TP Protokoll auf.

- Verbindungen zu BeanConnect für die Kommunikation über das OSI TP Protokoll können mit der Management Console konfiguriert werden (siehe Kapitel „BeanConnect konfigurieren“ auf Seite 179).
- Verbindungen zu BeanConnect über ein Transport-Level-Protokoll müssen im EIS konfiguriert werden (siehe Kapitel „Konfiguration im EIS Partner anpassen“ auf Seite 263). Die Management Console unterstützt das Konfigurieren dieser Verbindungen nicht.

- Für jede OLTP Message-Driven Bean, die unter Verwendung des OSI TP Protokolls aufgerufen werden soll, muss im EIS ein LTAC konfiguriert werden. Der diesem LTAC zugewiesene RTAC-Name muss mit dem Namen des Inbound Service des Inbound Message Endpoint identisch sein, der der EJB beim Deployment zugewiesen wurde.
- Die Codierung der Benutzernachrichten wird beim Deployment der OLTP Message-Driven Bean festgelegt (`activation-config` properties `encodingActive` und `encoding`, siehe Abschnitt „Konfigurations-Properties für Inbound-Kommunikation in ejb-jar.xml definieren“ auf Seite 142) oder bei der Konfiguration des Inbound Service, siehe Abschnitt „Inbound Services konfigurieren“ auf Seite 245.

Die Handshake-Funktion des OSI TP Protokolls darf in einer Kommunikation mit OLTP Message-Driven Beans nicht verwendet werden.

Die Nachrichten, die eine openUTM-Partneranwendung an eine OLTP Message-Driven Bean sendet, können aus einem Teil oder auch aus mehreren Teilen bestehen. Die OLTP Message-Driven Bean kann jeden Nachrichtenteil einzeln lesen. Ebenso kann die OLTP Message-Driven Bean die Antwortnachricht aus mehreren Nachrichtenteilen zusammensetzen, die von der openUTM-Partneranwendung mit einer Reihe von `MGET NT`-Aufrufen gelesen werden können. Weitere Einzelheiten finden Sie im Abschnitt „Programm-Framework mit den Interfaces AsyncOltpMessageListener und OltpMessageListener“ auf Seite 479.

Wenn Sie eine OLTP Message-Driven Bean unter Verwendung eines Transport-Level-Protokolls aufrufen, muss dem ersten Nachrichtenteil der Service-Name vorangestellt sein, der dem Inbound Message Endpoint zugewiesen wurde, der seinerseits der EJB beim Deployment zugewiesen wurde.

Weitere Informationen dazu finden Sie in der Dokumentation zu openUTM.

10.3.3 Inbound-Kommunikation mit CICS-Partnern

Bei einer Inbound-Kommunikation mit einer CICS-Anwendung unterstützt BeanConnect das Kommunikationsprotokoll LU6.2. Für die Teilnahme an einer Kommunikation mit BeanConnect gelten folgende Einschränkungen und Regeln:

- Basic Conversation wird nicht unterstützt. Basic Conversation für CICS wird mittels der Befehle programmiert, die mit GDS beginnen.
- PIP-Daten können für den Aufruf `CONNECT PROCESS` nicht verwendet werden. Die Daten gehen verloren.
- Es können keine unterschiedlichen Mode-Namen für unterschiedliche Verbindungen mit demselben Partner verwendet werden. Bei CICS/ESA V4.1 wird der Mode-Name in der `SESSION`-Definition angegeben und kann dann beim CICS-API implizit über den Parameter `SYSID` des Kommandos `ALLOCATE` ausgewählt werden.

- Wenn aufgrund von internen Konnektivitäts-Problemen in BeanConnect eine LU6.2-Conversation mit dem Java EE Application Server nicht eröffnet wird, erhält CICS keine detaillierte Rejection Message. Die Rejection Message steht nur in einer der Protokolldateien von BeanConnect.
- BeanConnect unterstützt SYNCLEVEL 0 (nicht-transaktionale Conversation) und SYNCLEVEL 2 (transaktionale Conversation). Der SYNCLEVEL wird in der CICS-API mit dem Parameter SYNCLEVEL für CONNECT PROCESS festgelegt.
- Wenn eine Inbound-Kommunikation SYNCLEVEL 2 verwendet, muss das CICS-Programm beim Proxy das Transaktionsende mit den Kommandos `SEND LAST` und `SYNCPOINT` bzw. `ISSUE PREPARE` anfordern. Der Proxy beendet dann die Transaktion. CICS kann das Transaktionsende entweder gleich beim Senden der Benutzernachricht anfordern oder erst nach dem Empfang der Antwort.
- Es sind nur Einschritt-Dialoge sind zulässig (ein `SEND INVITE` Aufruf im CICS-Programm). Jedoch können Nachricht und Antwort aus mehreren Teilen bestehen. Für jeden Nachrichtenteil muss ein `SEND` und ein `RECEIVE`-Aufruf ausgeführt werden. Der letzte Teil wird durch den Absender mit Hilfe des Aufrufs `SEND INVITE` angezeigt.
- Wenn für SYNCLEVEL der Wert 0 festgelegt sind, beendet die OLTP Message-Driven Bean die Kommunikation. Das bedeutet, dass das CICS eine Nachricht mit `SEND INVITE` versenden und die entsprechende Reply Message mit RECEIVE empfangen kann. Folglich wird der Dialog beendet und `SEND LAST` ist nicht länger zulässig. Das CICS kann jedoch `SEND LAST` anstatt des Paares `SEND INVITE` / `RECEIVE` ausgeben. Hier sendet das CICS die Nachricht an eine OLTP Message-Driven Bean, ohne eine entsprechende Reply Message zu erhalten.

Ein CICS-Programm für Inbound-Kommunikation muss so entwickelt und codiert sein, dass es dem Distributed Transaction Programming (DTP) entspricht. Eine Beschreibung dieses Programmier-Paradigmas finden Sie in der CICS-Dokumentation von IBM, z.B. „CICS Distributed Transaction Programming Guide“.

10.3.4 Inbound-Kommunikation mit anderen EIS Partnern

Bei Inbound-Kommunikation unterstützt BeanConnect folgende EIS Partner, die keine openUTM- oder CICS-Partner sind:

- UPIC-Partner
- Transportsystem-Partner wie RFC1006-Partner oder openUTM-Socket-Partner

Für Kommunikation, die unter Verwendung nicht-transaktionaler Protokolle wie dem UPIC-Protokoll oder Transport-Level-Protokollen (RFC1006 oder openUTM-Socket-Protokoll) erfolgt, gelten folgende Regeln:

- Über UPIC-Partner können nur OLTP Message-Driven Beans für dialogbasierte Kommunikation aufgerufen werden. UPIC-Nachrichtenteile werden auf `OltpMessagePart`-Objekte abgebildet und umgekehrt.
- Wenn Sie eine OLTP Message-Driven Bean unter Verwendung eines Transport-Level-Protokolls aufrufen, muss dem ersten Nachrichtenteil der Name des Inbound Service vorangestellt sein, der dem Inbound Message Endpoint, welcher der EJB beim Deployment zugewiesen wurde, bei der Konfiguration zugeordnet wurde.
- Die Codierung der Benutzernachrichten wird beim Deployment der OLTP Message-Driven Bean festgelegt (`activation-config` properties `encodingActive` und `encoding`, siehe Abschnitt „Konfigurations-Properties für Inbound-Kommunikation in ejb-jar.xml definieren“ auf Seite 142) oder bei der Konfiguration des Inbound Service (siehe Abschnitt „Inbound Services konfigurieren“ auf Seite 245).

10.3.5 Inbound-Kommunikation mit XATMI-Partnern

Bei Inbound-Kommunikation werden auch openUTM-Partner und UPIC-Partner unterstützt, die das API XATMI verwenden. Die Kommunikation mit dem openUTM-Partner erfolgt dabei immer über das OSI TP Protokoll.

Bei der Kommunikation mit XATMI-Partnern gilt:

- Es werden nur Typisierte Puffer vom Typ `X_OCTET` unterstützt.
- Eine OLTP Message-Driven Bean kann mit Hilfe des `OltpMessageContext`-Objekts ermitteln, ob es sich beim aufrufenden EIS Partner um einen XATMI-Partner handelt und - wenn ja - welches Paradigma dieser verwendet (Request/Reply oder Conversational). Details finden Sie in Abschnitt „Absenderkontexte in der OLTP Message-Driven Bean ermitteln“ auf Seite 477.
- Die Länge eines (codierten) Nachrichtenteils darf 32.000 Bytes nicht überschreiten.
- Beim Request/Reply Paradigma darf nur ein Nachrichtenteil gesendet werden.

10.3.6 BeanConnect-spezifische Interfaces für Inbound-Kommunikation

Folgende BeanConnect-spezifischen Interfaces des Packages `net.fsc.jca.communication` werden bei Inbound-Kommunikation unterstützt:

- `AsyncOltpMessageListener` für asynchrone Kommunikation

  Das Interface stellt für den Empfang einer Inbound Message die Methode `onMessage()` bereit, die die Inbound Message als Parameter enthält.

- `OltpMessageListener` für dialogbasierte Kommunikation

  Das Interface stellt für den Empfang einer Inbound Message und das Senden einer Reply Message die Methode `onMessage()` bereit, die die Inbound Message als Parameter enthält und der EIS-Anwendung eine Reply Message zurückgibt.

10.3.6.1 Programmierinformationen zu OLTP Message-Driven Beans

- Eine OLTP Message-Driven Bean muss genau eines der folgenden Interfaces implementieren:
  - `AsyncOltpMessageListener`
  - `OltpMessageListener`
- Ein `OltpMessage`-Objekt kann aus einem oder mehreren `OltpMessagePart`-Objekten und/oder einem oder mehreren `OltpMessageRecord`-Objekten bestehen. Während ein `OltpMessageRecord` beliebig lang sein kann, darf ein `OltpMessagePart` 32767 Bytes nicht übersteigen.

  Aus `OltpMessagePart`- und `OltpMessageRecord`-Objekten können Sie den Nachrichteninhalt als Objekt eines der folgenden Typen abrufen:

  - `byte[]`
  - `String`
  - `ByteContainer`

  Eine OLTP Message-Driven Bean kann das Interface `ByteContainer` implementieren, wenn sie mit einer EIS-Anwendung strukturierte Objekte austauschen möchte, die Text und Binärdaten enthalten. In diesem Fall muss eine Code-Konvertierung für die Textinformationen des strukturieren Objekts durchgeführt werden. Bei Objekten des Typs `String` wird die Code-Konvertierung von BeanConnect durchgeführt. Weitere Einzelheiten hierzu finden Sie in der JavaDoc zu BeanConnect.

- Die Reihenfolge, in der `OltpMessagePart`- und/oder `OltpMessageRecord`-Objekte zu einem `OltpMessage`-Objekt hinzugefügt oder von einem `OltpMessage`-Objekt zurückgegeben werden, entspricht der Reihenfolge, in der die Nahcricht gesendet wird bzw. empfangen wurde.

- Das Interface `OltpMessageContext` dient zwei Zwecken:
  - Es stellt eine Methode zum Generieren einer Reply Message bereit.
  - Es ermöglicht den Abruf von Statusinformationen.

  Eine OLTP Message-Driven Bean kann die Interface-Methode `OltpMessageContext` nur von der Methode `onMessage()` aus aufrufen.
- Wenn Sie die Methoden `addMessagePart()` oder `addMessageRecord()` innerhalb der Methode `onMessage()` aufrufen, müssen Sie dieselbe `OltpMessage` verwenden, mit der auch die Methoden `createMessagePart()` oder `createMessageRecord()` aufgerufen wurden.
- Asynchrone OLTP Message-Driven Beans empfangen asynchrone Nachrichten, die unabhängig von der Verfügbarkeit des Auftraggebers eintreffen. Aus diesem Grund kann weder eine Reply Message noch eine durch eine asynchrones OLTP Message-Driven Bean ausgelöste Exception an den Auftraggeber der asynchronen Nachricht zurückgesendet werden.
- Wenn zwischen der EIS-Anwendung und BeanConnect transaktional über das OSI TP Protokoll kommuniziert wird, läuft die Methode `onMessage(OltpMessage)` einer dialogbasierten OLTP Message-Driven Bean innerhalb der verteilten Transaktion ab, falls dieser Methode das Transaktionsattribut `Required` zugewiesen wurde.
- Asynchrone OLTP Message-Driven Beans können niemals Teil einer zwischen dem EIS und dem Application Server verteilten Transaktion sein.
- Die Methode `onMessage()` einer asynchronen OLTP Message-Driven Bean, die mit dem Transaktionsattribut `Required` deployt wurde, wird innerhalb einer Transaktion aufgerufen, die vom Proxy gestartet wurde (niemals vom EIS). Wird die Transaktion zurückgesetzt, so wird die asynchrone Nachricht ggf. erneut an die OLTP Message-Driven Bean ausgeliefert.

  Die OLTP Message-Driven Bean kann eine solche Situation ermitteln, indem sie den Wert des Redelivery-Zählers eines asynchronen `OltpMessage`-Objekts auswertet. Die Konfigurations-Property `redeliveryThreshold`, die beim Deployment der OLTP Message-Driven Bean festgelegt wird, definiert die Anzahl der zusätzlichen Auslieferungsversuche für eine Nachricht im Falle einer Störung oder eines Fehlers (siehe „redeliveryThreshold“ auf Seite 145).
- Für den Austausch von Nachrichten, die nicht in ASCII codiert sind, stehen über das Interface `OltpMessageContext` die Methoden des Interfaces `EncodingDef` zur Verfügung.

Weitere Einzelheiten hierzu finden Sie in der JavaDoc zu BeanConnect.

10.3.6.2 Absenderkontexte in der OLTP Message-Driven Bean ermitteln

Eine OLTP Message-Driven Bean kann über das Objekt `OltpMessageContext` Informationen über den Absender ermitteln. Dazu gehören z.B. der Anwendungsname und Rechnername des EIS Partners und der Inbound Service, mit dem die OLTP Message-Driven Bean im Proxy aufgerufen wurde. Das Ermitteln des Inbound Service kann z.B. dann von Interesse sein, wenn einem MessageEndpoint mehrere Inbound Services im ProxyContainer zugeordnet wurden.

Das Objekt `OltpMessageContext` stellt folgende Methoden zur Verfügung, um den Absenderkontext abzufragen:

- `String getBCProxyName()`

  Name der Proxyanwendung, feste Länge 8 Zeichen.

- `String getBCProxyHost()`

  Name des Rechners, auf dem der Proxy läuft, feste Länge 8 Zeichen.

- `String getBCProxyInboundService()`

  Name des aufgerufenen Inbound Service im Proxy, feste Länge 8 Zeichen.

- `enum BCCommunicationProtocolType getBCCommunicationProtocol()`

  Kennzeichen für das Kommunikations-Protokoll über das der EIS Partner den Inbound Service aufgerufen hat.

  – Bei dialogorientierter Kommunikation lässt sich der Typ des Kommunikations-Protokolls (bzw. des Client-Protokolls) aus der Aufzählungsklasse `BCCommunicationProtocolType` ermitteln.

  – Bei asynchroner Kommunikation wird der Protokoll-Typ des logischen Anschlusspunkts im Proxy übergeben, siehe auch `getBCProxyLocalPartnerName()`.

  `BCCommunicationProtocolType` liefert die folgenden Werte:

  '2' entspricht Protokoll-Typ OSI TP

  '3' entspricht Protokoll-Typ UPIC

  '5' entspricht Protokoll-Typ RFC1006

  '6' entspricht Protokoll-Typ SOCKET

- `String getBCPartnerTransportSelector()`

  String mit fester Länge von 8 Zeichen. Bei asynchroner Kommunikation werden Leerzeichen übergeben.

  Bei dialogbasierter Kommunikation wird je nach Protokoll-Typ Folgendes übergeben:

  – Protokoll-Typ UPIC, RFC1006 oder SOCKET: Partnername des Clients im Proxy

  - Protokoll-Typ OSI TP und openUTM-Partner in BS2000-Systemen: BCAM-Anwendungsname des fernen Rechners
  - Protokoll-Typ OSI TP und openUTM-Partner auf offenen Plattformen: T-Selektor der Partneranwendung
  - Protokoll-Typ OSI TP und CICS-Partner: TRANSPORT-SELECTOR, der dem CICS-Partner im openUTM-LU62-Gateway zugeordnet ist

- `String getBCPartnerNetworkSelector()`

  String mit fester Länge von 8 Zeichen. Bei asynchroner Kommunikation werden Leerzeichen übergeben.

  Bei dialogbasierter Kommunikation wird je nach Protokoll-Typ Folgendes übergeben:

  - Protokoll-Typ UPIC, RFC1006 oder SOCKET: Prozessorname des Clients
  - Protokoll-Typ OSI TP und openUTM-Partner im BS2000-Systemen: BCAM-Prozessorname des Rechners, auf dem sich die Partneranwendung befindet
  - Protokoll-Typ OSI TP und openUTM-Partner auf offenen Plattformen: Hostname des Partnerrechners
  - Protokoll-Typ OSI TP und CICS-Partner: NETWORK-SELECTOR, der dem CICS-Partner im openUTM-LU62-Gateway zugeordnet ist

- `String getBCProxyTransportSelector()`

  String mit fester Länge von 8 Zeichen. Bei asynchroner Kommunikation werden Leerzeichen übergeben.

  Bei dialogbasierter Kommunikation wird je nach Protokoll-Typ Folgendes übergeben:

  - Protokoll-Typ UPIC, RFC1006 oder SOCKET: Anwendungsname in der Proxy-Anwendung (BCAMAPPL-Name)
  - Protokoll-Typ OSI TP und openUTM-Partner: TRANSPORT-SELECTOR des ACCESS-POINTS in der Proxy-Anwendung
  - Protokoll-Typ OSI TP und CICS-Partner: TRANSPORT-SELECTOR des zugehörigen ACCESS-POINTS im openUTM-LU62-Gateway.

- `String getBCProxyUserId()`

  Benutzerkennung in der Proxyanwendung oder, wenn der Protokoll-Typ OSI TP ist und der EIS Partner keine Benutzerkennung übergeben hat, der Verbindungsname (ASSOCIATION-Name). Feste Länge 8 Zeichen.

- `String getBCProxyLocalPartnerName()`

  Name des logischen Anschlusspunktes in der Proxyanwendung. Für den Protokoll-Typ OSI TP ist dies der OSI-LPAP-Name, für alle anderen Protokoll-Typen der LTERM-Name. Feste Länge 8 Zeichen.

- `String getBCRaMessageEndpointName()`

  Name des aufgerufenen Message Endpoints.

- `boolean isBCPartnerXATMI()`

  `true`, falls der EIS Partner mit dem BeanConnect-Proxy über die XATMI-Schnittstelle kommuniziert, sonst `false`.

- `boolean isBCPartnerXATMIConversational()`

  `true`, falls der EIS Partner mit dem BeanConnect-Proxy über die XATMI-Schnittstelle kommuniziert und das Conversational Communication Paradigma ausgewählt hat, sonst `false` (d.h. Request/Reply Paradigma).

Strings, die mit der festen Länge 8 zurückgegeben werden, werden am Ende ggf. mit Leerzeichen aufgefüllt.

10.3.6.3 Programm-Framework mit den Interfaces AsyncOltpMessageListener und OltpMessageListener

Eine OLTP Message-Driven Bean empfängt die Inbound-Nachricht als `inMsg`-Parameter der `onMessage()`-Methode. Das empfangene Objekt ist ein `OltpMessage`-Objekt. Aus dem `OltpMessage`-Objekt können Sie ein `OltpMessageContext`-Objekt abrufen, das seinerseits Attribute der empfangenen Nachricht enthält und den dialogbasierten OLTP Message-Driven Beans als Factory zur Erstellung einer Reply Message dient:

1. Zugriff auf den Nachrichtenkontext:

   ```
   OltpMessageContext oltpMsgCtx = inMsg.getMessageContext();
   ```

2. Zugriff auf den Nachrichteninhalt:

   Das Objekt `OltpMessage` erlaubt einen Zugriff auf den Nachrichteninhalt, der in Form von `OltpMessageRecord`- oder `OltpMessagePart`-Objekten verarbeitet wird.

   Im Falle eines `OltpMessagePart`-Objekts geben Sie an:

   ```
   String inMsgTxt = "";
   if (inMsg.countMessageParts() > 0) {
     OltpMessagePart inMsgPart;
     Iterator it<OltpMessagePart> = inMsg.getMessageParts();

     for ( ; it.hasNext(); ) {
       inMsgPart = it.next();
       inMsgTxt += inMsgPart.getText();
     }
   }
   ```

Im Falle eines `OltpMessageRecord`-Objekts geben Sie an:

```
if (inMsg.countMessageRecords() > 0) {
  OltpMessageRecord inMsgRec;
  Iterator it<OltpMessageRecord> = inMsg.getMessageRecords();

  for ( ; it.hasNext(); ) {
    inMsgRec = it.next();
    inMsgTxt += inMsgRec.getText();
  }
}
```

3. Erstellen einer Reply Message (nur im Falle einer OLTP Message-Driven Bean für dialogbasierte Kommunikation):

   Eine OLTP Message-Driven Bean für dialogbasierte Kommunikation verwendet das Interface `OltpMessageContext`, um ein `OltpMessage`-Objekt für die Reply Message zu erstellen:

   ```
   OltpMessage outMsg = oltpMsgCtx.createMessage();
   ```

4. Das Objekt `OltpMessage` muss mit Nachrichteninhalt versorgt werden (nur im Falle einer OLTP Message-Driven Bean für dialogbasierte Kommunkation). Verwenden Sie hierzu das Objekt `OltpMessageRecord` und/oder das Objekt `OltpMessagePart`:

   – Sie sollten die Antwortnachricht mit Hilfe von `OltpMessagePart`-Objekten aufbauen, wenn dem Empfänger der Nachricht eine in **Nachrichtenteile** strukturierte Antwort zugestellt werden soll. Ist der Empfänger eine openUTM-Anwendung, dann liest diese jeden mit einem `OltpMessagePart`-Objekt übergebenen Nachrichtenteil mit einem eigenen MGET-Aufruf.

   – Wenn es nicht wichtig ist, dass die Antwortnachricht in Nachrichtenteile strukturiert wird, dann ist die Verwendung von `OltpMessageRecord`-Objekten vorteilhafter.

Das Code-Beispiel für OLTP Message-Driven Beans für dialogbasierte Kommunikation finden Sie Seite 471 und ein Code-Beispiel für OLTP Message-Driven Beans für asynchrone Kommunikation finden Sie Seite 471.

10.3.7 Common Client Interface (CCI) für Inbound-Kommunikation

Das CCI-Interface für die Inbound-Kommunikation ist `javax.resource.cci.MessageListener`. Dieses Interface stellt dieselbe Funktionalität bereit, wie das BeanConnect-spezifische Interface `OltpMessageListener`.

Zusätzlich können die Interfaces `net.fsc.jca.communication.cci.BCRecord` und `net.fsc.jca.communication.OltpMessage` für Inbound-Kommunikation verwendet werden.

10.3.7.1 Programmierinformationen zu OLTP Message-Driven Beans (CCI)

Eine OLTP Message-Driven Bean muss das Interface `javax.resource.cci.MessageListener` implementieren. OLTP Message-Driven Beans (CCI) erfüllen die Vorgaben für dialogbasierte Kommunikation; es gelten die entsprechenden Regeln, die im Abschnitt „Programmierinformationen zu OLTP Message-Driven Beans“ auf Seite 475 beschrieben sind.

10.3.7.2 Program Framework mit dem Interface javax.resource.cci.MessageListener

Eine OLTP Message-Driven Bean empfängt die Inbound Message als `record`-Parameter der `onMessage()`-Methode des Interfaces `MessageListener`. Das empfangene Objekt ist vom Typ `BCRecord` und enthält ein `OltpMessage`-Objekt. Aus dem `OltpMessage`-Objekt können Sie ein `OltpMessageContext`-Objekt abrufen, das seinerseits Attribute der empfangenen Nachricht enthält und auch den dialogbasierten OLTP Message-Driven Beans als Factory zur Erstellung einer Reply Message dient:

1. Extrahieren Sie das Objekt `OltpMessage` aus dem Objekt `BCRecord`:

   ```
   OltpMessage inMsg = ((BCRecord)record).getOltpMessage();
   ```

2. Richten Sie den Nachrichtenkontext ein:

   ```
   String inMsgTxt;
   OltpMessageContext oltpMsgCtx = inMsg.getMessageContext();
   ```

3. Greifen Sie auf den Nachrichteninhalt zu:

   Das Objekt `OltpMessage` erlaubt einen Zugriff auf den Nachrichteninhalt, der in Form von `OltpMessageRecord`- oder `OltpMessagePart`-Objekten verarbeitet wird. Aus diesen Objekten können Sie den Nachrichteninhalt als Objekt eines der folgenden Typen abrufen:

   - `byte[]`
   - `String`
   - `ByteContainer`

Im Falle eines `OltpMessagePart`-Objekts geben Sie an:

```
if (inMsg.countMessageParts() > 0) {
  OltpMessagePart inMsgPart;
  Iterator it<OltpMessagePart> = inMsg.getMessageParts();

  for ( ; it.hasNext(); ) {
    inMsgPart = it.next();
    inMsgTxt = inMsgPart.getText();
  }
}
```

Im Falle eines `OltpMessageRecord`-Objekts geben Sie an:

```
if (inMsg.countMessageRecords() > 0) {
  OltpMessageRecord inMsgRec;
  Iterator it<OltpMessageRecord> = inMsg.getMessageRecords();

  for ( ; it.hasNext(); ) {
    inMsgRec = (OltpMessageRecord) it.next();
    inMsgTxt = inMsgRec.getText();
  }
}
```

4. Erstellen Sie ein `OLTPMessage`-Objekt für die Reply Message:

   Eine OLTP Message-Driven Bean für dialogbasierte Kommunikation verwendet das Interface `OltpMessageContext`, um ein `OltpMessage`-Objekt für die Reply Message zu erstellen:

   ```
   OltpMessage outMsg = oltpMsgCtx.createMessage();
   ```

5. Das Objekt `OltpMessage` muss mit dem Inhalt der Reply Message versorgt werden. Verwenden Sie hierzu das Objekt `OltpMessageRecord` und/oder das Objekt `OltpMessagePart`.

   Im Falle eines `OltpMessagePart`-Objekts geben Sie an:

   ```
   OltpMessagePart outMsgPart = outMsg.createMessagePart();
   outMsgPart.setText("reply");
   outMsg.addMessagePart(outMsgPart);
   ```

   Im Falle eines `OltpMessageRecord`-Objekts geben Sie an:

   ```
   OltpMessageRecord outMsgRec = outMsg.createMessageRecord("");
   outMsgRec.setText("reply");
   outMsg.addMessageRecord(outMsgRec);
   ```

6. Vor der Rückgabe muss die Reply Message in das Objekt `BCRecord` eingesetzt werden, das anschließend von dieser Methode zurückgegeben wird:

   ```
   ((BCRecord)record).setOltpMessage(outMsg);
   ```

Ein Code-Beispiel finden Sie im Beispiel 19 auf Seite 486.

10.3.8 Code-Beispiel für Inbound-Kommunikation

Dieser Abschnitt enthält die folgenden Code-Beispiele:

- OLTP Message-Driven Beans für dialogbasierte Kommunikation
- OLTP Message-Driven Beans für asynchrone Kommunikation
- OLTP Message-Driven Bean (CCI)

Beispiel 17 OLTP Message-Driven Beans für dialogbasierte Kommunikation

```
package net.fsc.jca.BeanConnect.oltpmdb;

import javax.ejb.EJBException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import java.util.Iterator;
import net.fsc.jca.communication.*;

public class SampleDialogOltpMdbBean
     implements MessageDrivenBean, OltpMessageListener {

 public void ejbCreate()
             throws EJBException {
    // @TODO: add code
 }

 public void setMessageDrivenContext(MessageDrivenContext ctx)
             throws EJBException {
    // @TODO: add code
 }

 public void ejbRemove()
             throws EJBException {
    // @TODO: add code
 }

 public OltpMessage onMessage(OltpMessage inMsg) {
    String inMsgTxt;
    OltpMessageContext oltpMsgCtx = inMsg.getMessageContext();

    // read request
    try {
      if (inMsg.countMessageParts() > 0) {
         OltpMessagePart inMsgPart;
         Iterator it<OltpMessagePart> = inMsg.getMessageParts();

         for ( ; it.hasNext(); ) {
```

```
            inMsgPart = it.next();
            inMsgTxt  = inMsgPart.getText();
            // @TODO: process message part
          }

          // @TODO: process request
        }
      }
      catch (Exception ex) {
        // @TODO: handle exception
      }

      // setup reply
      OltpMessage outMsg = oltpMsgCtx.createMessage();
      OltpMessagePart outMsgPart = outMsg.createMessagePart();
      try {
        outMsgPart.setText("Reply from SampleDialogOltpMdbBean");
      }
      catch (OltpMessageException ex) {
        // @TODO: add exception handling
      }
      outMsg.addMessagePart(outMsgPart);

      return (outMsg);
    }
}
```

Beispiel 18 OLTP Message-Driven Beans für asynchrone Kommunikation

```
package net.fsc.jca.BeanConnect.oltpmdb;

import javax.ejb.EJBException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import java.util.Iterator;
import net.fsc.jca.communication.*;

public class SampleAsynOltpMdbBean
     implements MessageDrivenBean,AsyncOltpMessageListener {
  public void ejbCreate()
            throws EJBException {
    // @TODO: add code
  }

  public void setMessageDrivenContext(MessageDrivenContext ctx)
           throws EJBException {
    // @TODO: add code
  }
```

```
  public void ejbRemove()
            throws EJBException {
    // @TODO: add code
  }

  public void onMessage(OltpMessage inMsg) {
    String inMsgTxt;
    OltpMessageContext oltpMsgCtx = inMsg.getMessageContext();

     // read request

     try {
       if (inMsg.countMessageParts() > 0) {
         OltpMessagePart inMsgPart;
         Iterator it<OltpMessagePart> = inMsg.getMessageParts();

         for ( ; it.hasNext(); ) {
           inMsgPart = it.next();
           inMsgTxt  = inMsgPart.getText();
           // @TODO: process message part
         }

         // @TODO: process request
       }
     }
     catch (Exception ex) {
       // @TODO: handle exception
     }

     return;
  }
}
```

Beispiel 19 OLTP Message-Driven Bean (CCI)

```
package net.fsc.jca.BeanConnect.oltpmdb;

import java.util.Iterator;

import javax.ejb.EJBException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import javax.resource.cci.MessageListener;
import javax.resource.cci.Record;

import net.fsc.jca.communication.OltpMessage;
```

```
import net.fsc.jca.communication.OltpMessageContext;
import net.fsc.jca.communication.OltpMessageException;
import net.fsc.jca.communication.OltpMessagePart;
import net.fsc.jca.communication.cci.BCRecord;

public class SampleCciOltpMdbBean
             implements MessageDrivenBean, MessageListener {
  public Record onMessage(Record record) {

    String inMsgTxt;
    OltpMessage inMsg = ((BCRecord)record).getOltpMessage();
    OltpMessageContext oltpMsgCtx = inMsg.getMessageContext();

    // read request
    try {
      if (inMsg.countMessageParts() > 0) {
        OltpMessagePart inMsgPart;
        Iterator it<OltpMessagePart> = inMsg.getMessageParts();

        for ( ; it.hasNext(); ) {
          inMsgPart = it.next();
          inMsgTxt  = inMsgPart.getText();
          // @TODO: process message part
        }

        // @TODO: process request

      }
    }
    catch (Exception ex) {
        // @TODO: handle exception
    }

    // setup reply

    OltpMessage outMsg = oltpMsgCtx.createMessage();
    OltpMessagePart outMsgPart = outMsg.createMessagePart();
    try {
      outMsgPart.setText("Reply from SampleCciOltpMdbBean");
    }
    catch (OltpMessageException ex) {
      // @TODO: add exception handling
    }
    outMsg.addMessagePart(outMsgPart);
```

```
    ((BCRecord)record).setOltpMessage(outMsg);
    return record;
  }

  /**
   * Method ejbCreate() as required by EJB spec.
   */
  public void ejbCreate()
             throws EJBException {
    // @TODO: add code
  }

  /**
   * Method setMessageDrivenContext() as required by interface
   * javax.ejb.MessageDrivenBean.
   * javax.ejb.MessageDrivenBean#setMessageDrivenContext(
   *                                MessageDrivenContext ctx)
   */


  public void setMessageDrivenContext(MessageDrivenContext ctx)
             throws EJBException {
    // @TODO: add code
  }

  /**
  * Method ejbRemove() as required by interface
  * javax.ejb.MessageDrivenBean.
  *
  * @see javax.ejb.MessageDrivenBean#ejbRemove()
  */
  public void ejbRemove()
             throws EJBException {
  // @TODO: add code
  }
}
```

11 Zeichensatz-Konvertierung und Sprachunterstützung

Dieses Kapitel enthält Informationen zu folgenden Themen:

- Der Abschnitt Zeichensatz-Konvertierung beschreibt die Code-Konvertierung zwischen einem Java-Programm, das Unicode verwendet und der Codierung, die das Partnersystem verwendet.
- Der Abschnitt Sprachunterstützung für die Ausgabe von Meldungen beschreibt die Funktion BeanConnect National Language Support (NLS) für die Sprachunterstützung bei der Ausgabe von Meldungen des BeanConnect Resource Adapters, des BeanConnect Proxys sowie der BeanConnect Management Console.

11.1 Zeichensatz-Konvertierung

Wenn BeanConnect druckbare Daten von Partnern auf einem BS2000- oder einem IBM-Mainframe empfängt, muss zuerst der in 1-Byte-Code codierte Datenstrom (z.B. EBCDIC) in den 2-Byte-Unicode konvertiert werden, damit ein Java-Programm die Daten direkt verarbeiten kann. Entsprechend ist die Konvertierung von 2-Byte-Unicode in den 1-Byte-Code erforderlich, wenn ein Java-Programm Daten über BeanConnect an den BS2000- bzw. CICS-Partner sendet.

Für die Konvertierung eines 1-Byte-Codes in den 2-Byte-Unicode und umgekehrt haben Sie folgende Optionen:

- Standard-Konvertierung zwischen EBCDIC-Code und Unicode für EIS Partner vom Typ openUTM
- Standard-Konvertierung zwischen EBCDIC-Code und Unicode für EIS Partner vom Typ CICS
- Andere vordefinierte Code-Tabellen verwenden
- Benutzerdefinierte Zeichensätze verwenden
- Legacy-Code-Tabellen erstellen und verwenden

Weitere Einzelheiten zu den in diesem Kapitel erläuterten Themen finden Sie in der JavaDoc for BeanConnect im Abschnitt zum Package `net.fsc.beanta.encoding`.

Alle Code-Tabellen, die von den IBM-Systemen verwendet werden, finden Sie unter

`http://www-03.ibm.com/systems/i/software/globalization/codepages.html`

11.1.1 Standard-Konvertierung zwischen EBCDIC-Code und Unicode für EIS Partner vom Typ openUTM

In den meisten Fällen müssen Sie sich nicht um die Konvertierung von EBCDIC-Code in Unicode und umgekehrt kümmern, da BeanConnect die Konvertierung automatisch gemäß der Standard-Code-Tabelle `OSD_EBCDIC_DF04_DRV` vornimmt.

Die Code-Konvertierung findet automatisch statt, wenn folgende Bedingungen erfüllt sind:

- Für die Konfigurations-Property `encodingActive` ist der Wert `true` angegeben.
- Es werden Strings für die Kommunikation verwendet.

Wenn das Java-Programm einen nicht konvertierten Datenstrom in 1-Byte-EBCDIC empfangen soll, müssen Byte-Arrays anstelle von Strings verwendet werden.

Code-Tabelle OSD_EBCDIC_DF04_DRV

Die Tabelle zeigt die Zuordnung von 1-Byte-EBCDIC-Code, druckbaren Unicode-Zeichen und 2-Byte-Unicode an, wie sie in der Code-Tabelle `OSD_EBCDIC_DF04_DRV` definiert sind.

In der folgenden Tabelle zeigt

- die x- und die y-Achse den jeweiligen 1-Byte-EBCDIC-Code
- die erste Zeile in einem Feld den 2-Byte-Unicode
  (führende nicht signifikante Bytes werden nicht angezeigt)
- die zweite Zeile in einem Feld das druckbare Unicode-Zeichen

Zuordnung Byte - Zeichen (Unicode)

↓→	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	00	01	02	03	85 …	09	86 †	7F	87 ‡	8D	8E Ž	0B	0C	0D	0E	0F
1_	10	11	12	13	8F	A	8	97 —	18	19	9C œ	9D	1C	1D	1E -	1F
2_	80 €	81	82 ,	83 *f*	84 „	92 ’	17	1B	88 ˆ	89 ‰	8A Š	8B ‹	8C Œ	5	6	7
3_	90	91 ‘	16	93 “	94 ”	95 •	96 –	4	98 ˜	99 ™	9A š	9B ›	14	15	9E ž	1A
4_	20	A0	E2 â	7C |	E0 à	E1 á	E3 ã	E5 å	E7 ç	F1 ñ	60 `	2E .	3C <	28 (	2B +	F6 ö
5_	26 &	E9 é	EA ê	EB ë	E8 è	ED í	EE î	EF ï	EC ì	0	21 !	24 $	2A *	29)	3B ;	AF ¯
6_	2D -	2F /	C2 Â	A6 ¦	C0 À	C1 Á	C3 Ã	C5 Å	C7 Ç	D1 Ñ	5E ^	2C ,	25 %	5F _	3E >	3F ?
7_	F8 ø	C9 É	CA Ê	CB Ë	C8 È	CD Í	CE Î	CF Ï	CC Ì	A8 ¨	3A :	23 #	A7 §	27 '	3D =	22 "
8_	D8 Ø	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	AB «	BB »	F0 ð	FD ý	FE þ	B1 ±
9_	B0 °	6A j	6B k	6C l	6D m	6E n	6F o	70 p	71 q	72 r	AA ª	BA º	E6 æ	B8 ¸	C6 Æ	A4 ¤
A_	B5 µ	7E ~	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	A1 ¡	BF ¿	D0 Ð	DD Ý	DE Þ	AE ®
B_	A2 ¢	A3 £	A5 ¥	B7 ·	A9 ©	40 @	B6 ¶	BC ¼	BD ½	BE ¾	AC ¬	C4 Ä	D6 Ö	DC Ü	B4 ´	D7 ×
C_	7B {	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	AD	F4 ô	5B [	F2 ò	F3 ó	F5 õ
D_	7D }	4A J	4B K	4C L	4D M	4E N	4F O	50 P	51 Q	52 R	B9 ¹	FB û	5D]	F9 ù	FA ú	FF ÿ
E_	5C \	F7 ÷	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	B2 ²	D4 Ô	DB Û	D2 Ò	D3 Ó	D5 Õ
F_	30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	B3 ³	E4 ä	D9 Ù	FC ü	DA Ú	DF ß

Zuordnung Zeichen (Unicode) - Byte

Ersatzzeichen: **6F**

In der folgenden Tabelle zeigt

- die x- und die y-Achse den jeweiligen 2-Byte-Unicode
- das jeweilige Feld den zugehörigen 1-Byte-EBCDIC- Code

↓ →	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
000_	00	01	02	03	37	2D	2E	2F	16	05	15	0B	0C	0D	0E	0F
001_	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
002_	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
003_	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	**6F**
004_	B5	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
005_	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	CC	E0	DC	6A	6D
006_	4A	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
007_	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	43	D0	A1	7
008_	20	21	22	23	24	4	6	8	28	29	2A	2B	2C	9	A	14
009_	30	31	25	33	34	35	36	17	38	39	3A	3B	1A	1B	3E	5F
00A_	41	AA	B0	B1	9F	B2	63	7C	79	B4	9A	8A	BA	CA	AF	5F
00B_	90	8F	EA	FA	BE	A0	B6	B3	9D	DA	9B	8B	B7	B8	B9	AB
00C_	64	65	62	66	BB	67	9E	68	74	71	72	73	78	75	76	77
00D_	AC	69	ED	EE	EB	EF	BC	BF	80	FC	FE	EC	BD	AD	AE	FF
00E_	44	45	42	46	FB	47	9C	48	54	51	52	53	58	55	56	57
00F_	8C	49	CD	CE	CB	CF	4F	E1	70	DD	DE	DB	FD	8D	8E	DF

11.1.2 Standard-Konvertierung zwischen EBCDIC-Code und Unicode für EIS Partner vom Typ CICS

In den meisten Fällen müssen Sie sich nicht um die Konvertierung von 1-Byte-Code in Unicode und umgekehrt kümmern, da BeanConnect die Konvertierung automatisch gemäß der Standard-JDK-Code-Tabelle Cp1047 vornimmt.

Die Code-Konvertierung findet automatisch statt, wenn folgende Bedingungen erfüllt sind:

- Für die Konfigurations-Property `encodingActive` ist der Wert `true` angegeben.
- Es werden Strings für die Kommunikation verwendet.
- Die Connection URL ist vom Typ `cics://`

Wenn das Java-Programm einen nicht konvertierten Datenstrom in 1-Byte empfangen soll, müssen Byte-Arrays anstelle von Strings verwendet werden.

11.1.3 Andere vordefinierte Code-Tabellen verwenden

Zusätzlich zu der Standard-Code-Tabelle `OSD_EBCDIC_DF04_DRV` werden auch die folgenden Code-Tabellen mit dem Produkt BeanConnect mitgeliefert:

- `OSD_EBCDIC_DF03_IRV` (nur openUTM-Partner)
- `OSD_EBCDIC_DF04_1` (nur openUTM-Partner)
- `OSD_EBCDIC_DF04_15` (nur openUTM-Partner)
- von der JVM (Java Virtual Machine) verwendete Code-Tabellen (openUTM- und CICS-Partner)

BeanConnect unterstützt die von der JVM bereitgestellten Code-Tabellen. Die Liste der von der JVM verwendeten Code-Tabelle finden Sie

- für JDK 1.7 unter

  `http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html`

So wählen Sie die benötigte Code-Tabelle während des Deployments aus:

- Geben Sie in der Konfigurations-Property `encoding` den Namen der gewünschten Code-Tabelle an. Eine JVM-Code-Tabelle geben Sie mit `jdk:<jvm-code-table>` an.
- Aktivieren Sie die Konfigurations-Property `encodingActive`, indem Sie die Einstellung `true` wählen.

So wählen Sie die benötigte Code-Tabelle zur Laufzeit aus:

- Aktivieren Sie im Java-Programm die gewünschte Code-Tabelle mit der Methode `setEncoding()` des Interface `EISConnection` oder des Interface `OltpMessageContext`.

Mit der Management Console kann man für einen Inbound Service eine Code-Tabelle angeben. Diese Angabe überschreibt dann den Wert für `encoding` und setzt `encodingActive` auf `true`, siehe Abschnitt „Inbound Services konfigurieren“ auf Seite 245.

Beispiel 20 Vordefinierte Code-Tabelle verwenden

Die Code-Tabelle `OSD_EBCDIC_DF03_IRV` soll verwendet werden:

```
connection.setEncoding(Encoding.getEncoding("OSD_EBCDIC_DF03_IRV"))
```

Die JVM-Code-Tabelle `CP1047` soll verwendet werden:

```
connection.setEncoding(Encoding.getEncoding("jdk:Cp1047"));
connection.setEncodingActive(true);
```

Auf den nachfolgenden Seiten werden weitere vordefinierte Code-Tabellen angezeigt.

In allen folgenden Tabellen **Zuordnung Byte - Zeichen (Unicode)** zeigen

- die x- und die y-Achse den jeweiligen 1-Byte-EBCDIC-Code
- die erste Zeile in einem Feld den 2-Byte-Unicode
  (führende nicht signifikante Bytes werden nicht angezeigt)
- die zweite Zeile in einem Feld das druckbare Unicode-Zeichen

In allen folgenden Tabellen **Zuordnung Zeichen (Unicode) Byte** zeigen

- die x- und die y-Achse den jeweiligen 2-Byte-Unicode
- das jeweilige Feld den zugehörigen 1-Byte-EBCDIC- Code

Code-Tabelle OSD_EBCDIC_DF03_IRV

Die Tabelle zeigt die Zuordnung von 1-Byte-EBCDIC-Code, druckbaren Unicode-Zeichen und 2-Byte-Unicode an, wie sie in der Code-Tabelle `OSD_EBCDIC_DF03_IRV` definiert sind.

Zuordnung Byte - Zeichen (Unicode)

↓ →	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	00	01	02	03	FFFD	09	FFFD	7F	FFFD	FFFD	FFFD	0B	0C	0D	0E	0F
1_	10	11	12	13	FFFD	0A	08	FFFD	18	19	FFFD	FFFD	1C	1D	1E -	1F
2_	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	17	1B	FFFD	FFFD	FFFD	FFFD	FFFD	05	06	07
3_	FFFD	FFFD	16	FFFD	FFFD	FFFD	FFFD	04	FFFD	FFFD	FFFD	FFFD	14	15	FFFD	1A
4_	20	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	60 `	2E .	3C <	28 (	2B +	7C |
5_	26 &	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	21 !	24 $	2A *	29)	3B ;	FFFD
6_	2D -	2F /	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	5E ^	2C ,	25 %	5F _	3E >	3F ?
7_	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	3A :	23 #	40 @	27 '	3D =	22 "
8_	FFFD	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD
9_	FFFD	6A j	6B k	6C l	6D m	6E n	6F o	70 p	71 q	72 r	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD
A_	FFFD	FFFD	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD
B_	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD	5B [	5C \	5D]	FFFD	FFFD
C_	FFFD	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD
D_	FFFD	4A J	4B K	4C L	4D M	4E N	4F O	50 P	51 Q	52 R	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD
E_	FFFD	FFFD	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	FFFD	FFFD	FFFD	FFFD	FFFD	FFFD
F_	30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	FFFD	7B {	FFFD	7D }	FFFD	7E ~

Zuordnung Zeichen (Unicode) - Byte

Ersatzzeichen: **6F**

↓→	**_0**	**_1**	**_2**	**_3**	**_4**	**_5**	**_6**	**_7**	**_8**	**_9**	**_A**	**_B**	**_C**	**_D**	**_E**	**_F**
000_	00	01	02	03	37	2D	2E	2F	16	05	15	0B	0C	0D	0E	0F
001_	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
002_	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
003_	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	**6F**
004_	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
005_	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	BB	BC	BD	6A	6D
006_	4A	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
007_	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	FB	4F	FD	FF	07
008_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
009_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
00A_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
00B_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
00C_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
00D_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
00E_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**
00F_	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**	**6F**

Code-Tabelle OSD_EBCDIC_DF04_1

Die Tabelle zeigt die Zuordnung von 1-Byte-EBCDIC-Code, druckbaren Unicode-Zeichen und 2-Byte-Unicode an, wie sie in der Code-Tabelle `OSD_EBCDIC_DF04_1` definiert sind.

Zuordnung Byte - Zeichen (Unicode)

↓ →	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	00	01	02	03	85 …	09	86 †	7F	87 ‡	8D	8E Ž	0B	0C	0D	0E	0F
1_	10	11	12	13	8F	A	8	97 —	18	19	9C œ	9D	1C	1D	1E -	1F
2_	80 €	81	82 ,	83 *f*	84 „	92 ’	17	1B	88 ˆ	89 ‰	8A Š	8B ‹	8C Œ	05	06	07
3_	90	91 ‘	16	93 “	94 ”	95 •	96 –	4	98 ˜	99 ™	9A š	9B ›	14	15	9E ž	1A
4_	20	A0	E2 â	E4 ä	E0 à	E1 á	E3 ã	E5 å	E7 ç	F1 ñ	60 `	2E .	3C <	28 (	2B +	7C |
5_	26 &	E9 é	EA ê	EB ë	E8 è	ED í	EE î	EF ï	EC ì	DF ß	21 !	24 $	2A *	29)	3B ;	9F Ÿ
6_	2D -	2F /	C2 Â	C4 Ä	C0 À	C1 Á	C3 Ã	C5 Å	C7 Ç	D1 Ñ	5E ^	2C ,	25 %	5F _	3E >	3F ?
7_	F8 ø	C9 É	CA Ê	CB Ë	C8 È	CD Í	CE Î	CF Ï	CC Ì	A8 ¨	3A :	23 #	40 @	27 '	3D =	22 "
8_	D8 Ø	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	AB «	BB »	F0 ð	FD ý	FE þ	B1 ±
9_	B0 °	6A j	6B k	6C l	6D m	6E n	6F o	70 p	71 q	72 r	AA ª	BA º	E6 æ	B8 ¸	C6 Æ	A4 ¤
A_	B5 µ	AF ¯	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	A1 ¡	BF ¿	D0 Ð	DD Ý	DE Þ	AE ®
B_	A2 ¢	A3 £	A5 ¥	B7 ·	A9 ©	A7 §	B6 ¶	BC ¼	BD ½	BE ¾	AC ¬	5B [	5C \	5D]	B4 ´	D7 ×
C_	F9 ù	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	AD	F4 ô	F6 ö	F2 ò	F3 ó	F5 õ
D_	A6 ¦	4A J	4B K	4C L	4D M	4E N	4F O	50 P	51 Q	52 R	B9 ¹	FB û	FC ü	DB Û	FA ú	FF ÿ
E_	D9 Ù	F7 ÷	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	B2 ²	D4 Ô	D6 Ö	D2 Ò	D3 Ó	D5 Õ
F_	30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	B3 ³	7B {	DC Ü	7D }	DA Ú	7E ~

Zuordnung Zeichen (Unicode) - Byte

Ersatzzeichen: **6F**

↓ →	**_0**	**_1**	**_2**	**_3**	**_4**	**_5**	**_6**	**_7**	**_8**	**_9**	**_A**	**_B**	**_C**	**_D**	**_E**	**_F**
000_	00	01	02	03	37	2D	2E	2F	16	05	15	0B	0C	0D	0E	0F
001_	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
002_	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
003_	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	**6F**
004_	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
005_	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	BB	BC	BD	6A	6D
006_	4A	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
007_	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	FB	4F	FD	FF	7
008_	20	21	22	23	24	4	6	8	28	29	2A	2B	2C	9	A	14
009_	30	31	25	33	34	35	36	17	38	39	3A	3B	1A	1B	3E	5F
00A_	41	AA	B0	B1	9F	B2	D0	B5	79	B4	9A	8A	BA	CA	AF	A1
00B_	90	8F	EA	FA	BE	A0	B6	B3	9D	DA	9B	8B	B7	B8	B9	AB
00C_	64	65	62	66	63	67	9E	68	74	71	72	73	78	75	76	77
00D_	AC	69	ED	EE	EB	EF	EC	BF	80	E0	FE	DD	FC	AD	AE	59
00E_	44	45	42	46	43	47	9C	48	54	51	52	53	58	55	56	57
00F_	8C	49	CD	CE	CB	CF	CC	E1	70	C0	DE	DB	DC	8D	8E	DF

Code Table OSD_EBCDIC_DF04_15

Die Tabelle zeigt die Zuordnung von 1-Byte-EBCDIC-Code, druckbaren Unicode-Zeichen und 2-Byte-Unicode an, wie sie in der Code-Tabelle `OSD_EBCDIC_DF04_15` definiert sind.

Zuordnung Byte - Zeichen (Unicode)

↓ →	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	00	01	02	03	85 …	09	86 †	7F	87 ‡	8D	8E Ž	0B	0C	0D	0E	0F
1_	10	11	12	13	8F	A	8	97 —	18	19	9C œ	9D	1C	1D	1E -	1F
2_	80 €	81	82 ,	83 *f*	84 „	92 ’	17	1B	88 ˆ	89 ‰	8A Š	8B ‹	8C Œ	5	6	7
3_	90	91 ‘	16	93 “	94 ”	95 •	96 –	4	98 ˜	99 ™	9A š	9B ›	14	15	9E ž	1A
4_	20	A0	E2 â	E4 ä	E0 à	E1 á	E3 ã	E5 å	E7 ç	F1 ñ	60 `	2E .	3C <	28 (	2B +	7C |
5_	26 &	E9 é	EA ê	EB ë	E8 è	ED í	EE î	EF ï	EC ì	DF ß	21 !	24 $	2A *	29)	3B ;	9F Ÿ
6_	2D -	2F /	C2 Â	C4 Ä	C0 À	C1 Á	C3 Ã	C5 Å	C7 Ç	D1 Ñ	5E ^	2C ,	25 %	5F _	3E >	3F ?
7_	F8 ø	C9 É	CA Ê	CB Ë	C8 È	CD Í	CE Î	CF Ï	CC Ì	161 š	3A :	23 #	40 @	27 '	3D =	22 "
8_	D8 Ø	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	AB «	BB »	F0 ð	FD ý	FE þ	B1 ±
9_	B0 °	6A j	6B k	6C l	6D m	6E n	6F o	70 p	71 q	72 r	AA ª	BA º	E6 æ	17E ž	C6 Æ	20AC €
A_	B5 µ	AF ¯	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	A1 ¡	BF ¿	D0 Ð	DD Ý	DE Þ	AE ®
B_	A2 ¢	A3 £	A5 ¥	B7 ·	A9 ©	A7 §	B6 ¶	152 Œ	153 œ	178 Ÿ	AC ¬	5B [	5C \	5D]	17D Ž	D7 ×
C_	F9 ù	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	AD	F4 ô	F6 ö	F2 ò	F3 ó	F5 õ
D_	160 Š	4A J	4B K	4C L	4D M	4E N	4F O	50 P	51 Q	52 R	B9 ¹	FB û	FC ü	DB Û	FA ú	FF ÿ
E_	D9 Ù	F7 ÷	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	B2 ²	D4 Ô	D6 Ö	D2 Ò	D3 Ó	D5 Õ
F_	30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	B3 ³	7B {	DC Ü	7D }	DA Ú	7E ~

Zuordnung Zeichen (Unicode) - Byte

Ersatzzeichen: **6F**

↓ →	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
000_	00	01	02	03	37	2D	2E	2F	16	05	15	0B	0C	0D	0E	0F
001_	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
002_	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
003_	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	**6F**
004_	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
005_	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	BB	BC	BD	6A	6D
006_	4A	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
007_	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	FB	4F	FD	FF	7
008_	20	21	22	23	24	4	6	8	28	29	2A	2B	2C	9	A	14
009_	30	31	25	33	34	35	36	17	38	39	3A	3B	1A	1B	3E	5F
00A_	41	AA	B0	B1	9F	B2	D0	B5	79	B4	9A	8A	BA	CA	AF	A1
00B_	90	8F	EA	FA	BE	A0	B6	B3	9D	DA	9B	8B	B7	B8	B9	AB
00C_	64	65	62	66	63	67	9E	68	74	71	72	73	78	75	76	77
00D_	AC	69	ED	EE	EB	EF	EC	BF	80	E0	FE	DD	FC	AD	AE	59
00E_	44	45	42	46	43	47	9C	48	54	51	52	53	58	55	56	57
00F_	8C	49	CD	CE	CB	CF	CC	E1	70	C0	DE	DB	DC	8D	8E	DF

Ergebnisse für Eingaben > 0x00FF

Zeichen (in)		Byte (out)
152	Œ	B7
153	œ	B8
160	Š	D0
161	š	79
178	Ÿ	B9
17D	Ž	BE
17E	ž	9D
20AC	€	9F

11.1.4 Benutzerdefinierte Zeichensätze verwenden

Sie können benutzerdefinierte Zeichensätze auf zwei Arten erzeugen:

- Sie erzeugen Sie einen Custom Charset Provider, der in der JVM eingebettet ist.
- Sie erzeugen eine Legacy Code Tabelle.

11.1.4.1 Custom Charset Provider

Diese Methode empfiehlt sich für die Verwendung neuer benutzerdefinierter Zeichensätze.

Sie können eigene Code-Tabellen für die Verwendung in der JVM erstellen und sie aktivieren, indem Sie die im Abschnitt „Andere vordefinierte Code-Tabellen verwenden“ auf Seite 493 beschriebene Prozedur anwenden. Verwenden Sie anstelle des Namens einer vordefinierten JDK-Code-Tabelle den Namen ihrer eigenen Code-Tabelle mit der Syntax `jdk:<my_code_table>`.

Eine detaillierte Beschreibung der Implementierung und des Deployment Ihrer Code-Tabellen in der Java-Laufzeit-Umgebung finden Sie in der Dokumentation zu Java unter

`http://docs.oracle.com/javase/7/docs/api/java/nio/charset/package-summary.html`

sowie in der BeanConnect JavaDoc zum Package `net.fsc.beanta.encoding`.

Ein Java-Beispiel-Code ist im Lieferumfang von BeanConnect im Package `net.fsc.beanconnect.encoding.sample` enthalten.

Damit Sie Ihre eigenen Code-Tabellen verwenden können, müssen Sie die Klassen in das Verzeichnis `<JDK>/jre/lib/ext` stellen. Weitere Hinweise dazu stehen in "Java Extension Mechanism".

11.1.4.2 Legacy-Code-Tabellen erstellen und verwenden

Die hier beschriebene Methode verwenden Sie, wenn Sie für benutzerdefinierte Zeichensätze nicht wie in Abschnitt „Benutzerdefinierte Zeichensätze verwenden“ auf Seite 501 beschrieben die JVM erweitern wollen, sondern die Erweiterung nur für BeanConnect gelten soll.

Wenn die vordefinierten Code-Tabellen Ihren Anforderungen nicht genügen, können Sie eigene Code-Tabellen erstellen. Eine Java-Beispiel-Source für eine benutzerdefinierte Code-Tabelle ist im Lieferumfang enthalten. Sie finden dieses Beispiel in der JavaDoc zu dem Package net.fsc.beanta.encoding in der Klasse Encoding.CustomEncoder.

Wenn Sie Klassen der eigenen Code-Tabellen verwenden, sollten Sie die Klassen zur Datei `BeanConnect.rar` hinzufügen, um Probleme mit den Class Loaders zu vermeiden.

So wählen Sie die benötigte Code-Tabelle während des Deployments aus:

- Geben Sie in der Konfigurations-Property `encoding` mit `custom:<my-code-table>` den Namen der gewünschten Tabelle an.
- Aktivieren Sie die Konfigurations-Property `encodingActive`, indem Sie die Einstellung `true` wählen.

So wählen Sie die gewünschte Code-Tabelle zur Laufzeit aus:

- Aktivieren Sie im Java-Programm die gewünschte Code-Tabelle mit der Methode `setEncoding()` des Interface `EISConnection` oder des Interface `OltpMessageContext` wie folgt:

```
OwnTable myTable = new OwnTable();
connection.setEncoding(new Encoding.CustomEncoder(myTable));
connection.setEncodingActive(true);
```

Aus Kompatibilitätsgründen wird nicht nur die Vorgehensweise, die für die Verwendung benutzerdefinierter Code-Tabellen beschrieben wurde, unterstützt, sondern auch die folgende Vorgehensweise, bei der die Konvertierung mit zwei Java-Programmen implementiert ist:

```
OwnTable_ByteToChar myTableByteToChar = new
OwnTable_ByteToChar();
OwnTable_CharToByte myTableCharToByte = new
OwnTable_CharToByte();
setEncoding(new Encoding.Custom
      (myTableByteToChar,myTableCharToByte));
```

Diese Code-Tabellen unterstützen nur 1-Byte-Codierung. Das bedeutet, Codierungsquelle und -ziel für ein Zeichen kann nur ein Byte sein. Wenn Sie 2-Byte-Code-Tabellen benötigen, empfehlen wir Ihnen die Verwendung eines benutzerdefinierten Zeichensatzes, der alle Möglichkeiten der Code-Konvertierung bietet.

11.2 Sprachunterstützung für die Ausgabe von Meldungen

BeanConnect erlaubt die Internationalisierung und Lokalisierung von Meldungen des BeanConnect Resource Adapters, des BeanConnect Proxys und der BeanConnect Management Console. Internationalisierung und Lokalisierung bedeutet, dass Meldungen, die eine Komponente an die Benutzer weiterleitet oder in Protokolldateien schreibt, korrekt für die entsprechende Umgebung (Land, Sprache) ausgegeben wird.

Daher bietet Java Klassen an, die die Internationalisierung unterstützen (wie z.B. die Klasse `Locale`). In BeanConnect gibt es zwei Kriterien, um festzustellen, in welcher Sprache Meldungen ausgegeben werden sollen:

- Sprache (z.B. de)
- Land (z.B. DE)

Die Sprachkennung ist durch den internationalen Standard ISO-639 definiert. Die möglichen Länderkennungen sind in ISO-3166 festgelegt.

Bei BeanConnect lauten die Standardeinstellungen für Sprache *en* und *US* für die Länderkennung. Wenn der Benutzer nicht ausdrücklich eine andere Einstellung festlegt, werden die Standardwerte von der aktuellen JVM (`Default-Locale`) übernommen.

BeanConnect bietet zwei Möglichkeiten, diese Standardeinstellungen zu ändern:

- Stellen Sie beim Starten eines Java-Programms die Spracheinstellung ein, indem Sie folgende zwei Systemeigenschaften definieren:
  - `net.fsc.tpbasics.i18n.defaultCountry` und
  - `net.fsc.tpbasics.i18n.defaultLanguage`
- Speichern Sie die gewünschten Einstellungen in der Datei `beanconnect_i18n.properties`, die im Klassenpfad der betreffenden JVM stehen muss.

BeanConnect versucht zuerst, die Einstellungen anhand der Systemeigenschaften zu ermitteln. Wenn keine Systemeigenschaften definiert wurden, sucht das System nach der Property-Datei, um die Eigenschaften von dort zu übernehmen. Standardmäßig sollte die Spracheinstellung in der Datei `beanconnect_i18n.properties` eingestellt werden.

Standardkonfiguration

Bei der Installation des BeanConnect Resource Adapters, des BeanConnect Proxys oder der BeanConnect Management Console wird die JAR-Datei (`BeanConnectI18N.jar`) deployt. Diese Datei muss in den Klassenpfad der JVM eingebunden werden. Standardmäßig hat die Datei folgenden Inhalt:

- `beanconnect_i18n.properties` für die Spracheinstellungen
- die Meldungsdateien für die einzelnen Komponenten (Standardsprache: en, US)
- mehrere Java-Klassen (Msg…class), die unverändert in der JAR-Datei bleiben müssen.

Um den Zugriff auf die Meldungsdateien zu ermöglichen, müssen deren Namen einem bestimmten Muster folgen.

Verschiedene Meldungsdateien mit dem Namen `<component>.properties` stehen zur Verfügung. Diese Dateien werden als Ersatz verwendet, wenn Sie nicht die richtigen Werte für Sprache und Land eingegeben haben.

Die aktuell verwendeten Meldungsdateien sind:

Resource Adapter	`basics.properties`
	`stub.properties`
	`proxy.properties`
	`ui.properties`
	`oltpmsg.properties`
	`jconnect.properties`
	`encoding.properties`

Proxy	`proxy.properties` `oltpmsg.properties`
Management Console	`mc.properties` `tpbasics.properties`

Darüber hinaus kann eine Datei `<component>_en_US.properties` in der JAR-Datei enthalten sein. Dabei handelt es sich normalerweise um eine Kopie der Datei `<component>.properties`, da `en_US` der Standardwert für BeanConnect-Meldungsdateien ist.

Beispiele für Dateinamen:

- `proxy.properties`
- `proxy_de.properties`
- `proxy_en_US.properties`
- `proxy_fr_FR.properties`

Eine neue Sprache einführen

Sie können andere Sprachen unterstützen, indem Sie der JAR-Datei `BeanConnectI18N.jar` Meldungsdateien hinzufügen. Diese Meldungsdateien sind Textdateien, die mit einem Texteditor (wie z.B. Notepad, vi) bearbeitet werden können.

Das nachfolgende Beispiel zeigt die Schritte, welche ausgeführt werden müssen, um deutsche Meldungen zu erhalten.

1. Extrahieren Sie mit dem Befehl `jar` oder dem Programm WinZip die Standardmeldungsdatei (z.B. `proxy.properties`) aus der JAR-Datei.
2. Übersetzen Sie die Meldungen ins Deutsche.
3. Speichern Sie die neue Meldungsdatei unter dem Namen `proxy_de_DE.properties` und fügen Sie sie der JAR-Datei unter demselben Pfad (`net.fsc.tpbasic.i18n.r`) wie die Standarddatei hinzu.
4. Extrahieren Sie die Datei `beanconnect_i18n.properties`.
5. Geben Sie `de` für die Sprache und `DE` für das Land ein:

   ```
   net.fsc.tpbasics.i18n.defaultLanguage=de

   net.fsc.tpbasics.i18n.defaultCountry=DE
   ```

6. Schreiben Sie die Datei `beanconnect_i18n.properties` in die JAR-Datei zurück.

Führen Sie die Schritte 1 bis 3 für alle anderen Meldungsdateien durch (`stub.properties`, `ui.properties`, etc.).

Beachten Sie beim Wechsel in eine andere Sprache, dass die Datei `BeanConnectI18N.jar` für den BeanConnect Resource Adapter in der Datei `BeanConnect.rar` bearbeitet werden muss, bevor der Connector deployt wird.

Für den BeanConnect Proxy und die BeanConnect Management Console muss die Datei `BeanConnectI18N.jar` im BeanConnect-Home-Verzeichnis `<BC_home>/lib` bearbeitet werden.

12 Hoch-Verfügbarkeit und Skalierbarkeit

Der BeanConnect Proxy-Container wird mit Standard-Konfigurationswerten installiert. Für Hochlast-Betrieb kann es notwendig werden, diese Konfiguration zu ändern. Wenn die in der Konfiguration festgelegten Grenzwerte erreicht sind, wird dies in den meisten Fällen mit einer Meldung des Proxy-Containers protokolliert.

Das Kapitel enthält Informationen über:

- Shared Memory im Proxy-Container
- Anzahl der Prozesse im Proxy-Container
- Pagepool Area und Cache im Proxy-Container
- Anzahl der parallelen Verbindungen zum EIS Partner
- Asynchrone Verarbeitung im Proxy-Container
- OSI-SCRATCH-AREA im Proxy-Container
- Anzahl der Semaphore im Proxy-Container

12.1 Shared Memory im Proxy-Container

Alle Nachrichten, die der Proxy-Container sendet oder erhält, werden vor der Verarbeitung durch einen der Prozesse in einem Shared Memory gespeichert. Bei Kommunikations-Szenarien mit großen Datenmengen kann das Shared Memory, in dem die Nachrichten gepuffert werden, zu klein werden.

Dieses Problem beheben Sie über die Umgebungsvariablen `UTM_IPC_LETTER` und `UTM_IPC_EXPT_LETTER`, die in der Start-Prozedur des Proxy-Containers gesetzt werden:

- Solaris- und Linux-Systeme:
  `<Proxy_home>/shsc/startcontainer.sh`
- Windows-Systeme:
  `<Proxy_home>\shsc\startcontainer.cmd`

12.1.1 Shared Memory anpassen

Die notwendige Änderung ist abhängig vom Insert in der Meldung U189:

- **U189** &OBJ1 **(** &PTRM, &PRNM **): IPC Engpass LETT EXTP FULL** oder
  U189 &OBJ1 **(** &PTRM, &PRNM **): IPC Engpass LETT MAX ILETT** oder
  U189 &OBJ1 **(** &PTRM, &PRNM **): IPC Engpass LETT MAX OLETT**

  Die Größe einer Nachricht übersteigt in diesem Fall die Maximalgröße des Datenbereichs für eine Verbindung. Die Maximalgröße hat nach der Installation standardmäßig den Wert 32 (angegeben in 4KB-Einheiten).

  Als Korrektur müssen Sie den Wert der Umgebungsvariablen UTM_IPC_EXTP_LETTER auf einen größeren Wert setzen.

- **U189** &OBJ1 **(** &PTRM, &PRNM **): IPC Engpass LETT IPC FULL**

  Die Größe aller Nachrichten übersteigt in diesem Fall die Maximal-Größe des Datenbereichs für alle Verbindungen. Die Maximalgröße hat nach der Installation standardmäßig den Wert 1600 (angegeben in 4KB-Einheiten).

  Als Korrektur müssen Sie den Wert der Umgebungsvariablen UTM_IPC_LETTER auf einen größeren Wert setzen.

In den meisten Fällen ist die folgende „Daumenregel“ ausreichend:

- UTM_IPC_EXTP_LETTER = Maximalgröße einer Outbound-/Inbound-Nachricht in 4KB-Einheiten
- UTM_IPC_LETTER = <nConn> * UTM_IPC_EXTP_LETTER, aber nicht weniger als 1600 (6,4 MB).

  <nConn> ist dabei die maximale Anzahl von Verbindungen zwischen dem Proxy-Container und seinen Partneranwendungen (EIS Partner, Resource Adapter, Management Console).

12.2 Anzahl der Prozesse im Proxy-Container

Die Anzahl der vom Proxy-Container benötigten Prozesse ist maximal so hoch wie die Summe der für Outbound- und für Inbound-Kommunikation benötigten Prozesse.

Outbound-Kommunikation

Bei Outbound-Kommunikation entspricht die Anzahl der von einer Bean benötigten Prozesse der Anzahl paralleler Verbindungen der Bean, wenn die Verbindungen nicht gruppiert werden. Werden die Verbindungen gruppiert, so ist die Anzahl der benötigten Prozesse so groß wie die Anzahl der zu einer Zeit verwendeten Connection Groups.

Außerdem ist die Anzahl der Prozesse insgesamt davon abhängig, wie viele Beans zu einer Zeit Outbound-Kommunikation verwenden, und ob diese Kommunikation transaktional oder nicht-transaktional ist. Bei transaktionaler Kommunikation ist ein Proxy-Prozess bis zum Transaktionsende exklusiv einer Bean zugeordnet, bei nicht-transaktionaler Kommunikation nur für die Dauer einer Conversation.

Inbound-Kommunikation

Bei Inbound-Kommunikation ist die Anzahl der benötigten Prozesse mindestens so hoch wie die maximale Anzahl paralleler Inbound-Nachrichten. Dies können maximal so viele sein, wie parallele Verbindungen zu allen EIS Partnern existieren.

12.2.1 Prozess-Auslastung anzeigen

Die Auslastung der Container-Anwendung können Sie mit Hilfe der Management Console anzeigen:

1. Öffnen Sie den Teilbaum des Proxys im Navigationsbereich.
2. Wählen Sie unter **Advanced Features** den Eintrag **Properties / Statistics**.

Sie finden den Wert für die Prozess-Auslastung im Arbeitsbereichsfeld **Properties / Statistics** unter **Workload**. Bei einem Wert von über 80% bei **Workload (maximum)** sollten Sie die Anzahl der Prozesse erhöhen (siehe folgender Abschnitt).

12.2.2 Prozess-Anzahl ändern

Standardmäßig werden drei Prozesse gestartet. Sie ändern die Anzahl der Prozesse mit Hilfe der Management Console:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Edit Properties**.
2. Wählen Sie im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** die Registerkarte **Performance Settings**.
3. Ändern Sie unter **Number of Proxy Container Processes** den Wert für **Total**.

Die maximale Anzahl von Prozessen ist intern auf 240 begrenzt. Sie sollten außerdem bedenken, dass eine hohe Prozess-Anzahl auch den System-Overhead für die Verwaltung dieser Prozesse erhöht, sowie die Größe der Recovery-Datei beeinflusst.

Pro Container Prozess wird außerdem zusätzlicher virtueller Speicher benötigt. Beim Ablauf auf Solaris-Systemen im 32 Bitmode benötigt z.B. jeder zusätzliche `utmwork`-Prozess ca. 30-40 MByte zusätzlichen Speicher, der genaue Bedarf ist abhängig von der Konfiguration.

Vorsicht!
Ist der physikalische Speicher (RAM) zu knapp bemessen, werden vom System Speicherbereiche auf Festplatte ausgelagert (Paging), was den Ablauf dramatisch verlangsamen kann.

Sollen mehr als 50 Prozesse gestartet werden, müssen Sie zusätzlich die Anweisungen `MAX TASKS`, `MAX TASKS-IN-PGWT` und/oder `MAX ASYNTASKS` in der Datei `input.system.txt` manuell verändern.

Sie finden die Datei `input.system.txt`

- auf Solaris- und Linux-Systemen im Verzeichnis:
  `<Proxy_home>/`def
- auf Windows-Systemen im Verzeichnis:
  `<Proxy_home>\`def

Passen Sie die Anweisungen an, wie im Folgenden beschrieben:

- `MAX TASKS=<anzahl>`

  Um die Anzahl der Prozesse zu erhöhen, erhöhen Sie den Wert für `<anzahl>` in dieser Anweisung. `<anzahl>` gibt dabei die maximal mögliche Anzahl von Prozessen an.

- `MAX TASKS-IN-PGWT=<anzahl>`

  `<anzahl>` gibt dabei die maximale Anzahl der Nachrichten für Outbound-Kommunikation an, die parallel abgearbeitet werden können.

- `MAX ASYNTASKS=<anzahl>`

  `<anzahl>` gibt dabei die maximale Anzahl der asynchronen Nachrichten für Inbound-Kommunikation an, die parallel abgearbeitet werden können.

Anschließend müssen Sie, um diese Änderungen zu aktivieren, eine Update-Konfiguration und einen Restart für den Proxy durchführen (siehe Abschnitt „Konfiguration eines BeanConnect Proxys speichern und aktivieren“ auf Seite 249):

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Update Configuration**.
2. Wählen Sie im Kontextmenü des Proxys den Befehl **Start Proxy**.

12.3 Pagepool Area und Cache im Proxy-Container

Die benötigte Größe der Speicherbereiche für den Proxy-Container wird bestimmt durch die Größe und Anzahl der Outbound- und Inbound-Nachrichten, die parallel verarbeitet werden. Asynchrone Outbound- und Inbound-Nachrichten belegen so lange Speicher im Proxy-Container, bis sie gesendet bzw. vollständig verarbeitet sind.

Der Cache des Proxy-Containers ist ein Shared Memory. Die Größe des Cache beeinflusst daher den Bedarf an Hauptspeicher im Rechner, auf dem der Proxy-Container läuft.

- Ist der **Proxy Container Mode** auf **Performance Enhanced (Non-durable Asynchronous Processing)** gesetzt (Standard-Einstellung) und ist der Hauptspeicher ausreichend groß, sollten Sie den Cache so groß wie den Pagepool dimensionieren, um Schreib-/Lese-Zugriffe wegen Cache-Engpässen zu vermeiden.
- Ist der **Proxy Container Mode** auf **Durable Asynchronous Processing** gesetzt, kann der Cache dagegen nur Lese-Zugriffe einsparen, da die Daten immer im Pagepool gesichert werden.

Sie finden die Einstellung für den **Proxy Container Mode** wie folgt:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Edit Properties**.
2. Wählen Sie im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** die Registerkarte **Performance Settings**.

Sie müssen den Expertenmodus aktivieren, um die Registerkarte **Performance Settings** anzuzeigen. Detaillierte Informationen finden Sie in der Online-Hilfe der Management Console.

Größe der Speicherbereiche ändern

Nach der Installation ist sowohl die Größe der Pagepool Area als auch die des Cache auf 20 MB eingestellt. Die Größe dieser Speicherbereiche ändern Sie mit Hilfe der Management Console:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Edit Properties**.
2. Wählen Sie im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** die Registerkarte **Performance Settings**.
3. Ändern Sie unter **Proxy Container Storage Area Sizes** die Werte für **Pagepool (MB)** und **Cache (MB)**.

12.4 Anzahl der parallelen Verbindungen zum EIS Partner

Parallele Verbindungen zu einem EIS Partner über das OSI TP oder LU6.2-Protokoll

Die Anzahl der benötigten parallelen Verbindungen zwischen Proxy-Container und EIS Partner wird durch die Anzahl der Nachrichten bestimmt, die maximal gleichzeitig zwischen den Anwendungen im Application Server und den EIS Partnern ausgetauscht werden sollen.

Sie ändern die Anzahl der Verbindungen zu einem EIS Partner mit Hilfe der Management Console:

1. Öffnen Sie den Teilbaum des Proxys im Navigationsbereich.
2. Öffnen Sie den Teilbaum unter dem Knoten **EIS Partners**.
3. Wählen Sie im Kontextmenü des betreffenden EIS Partners den Befehl **Edit Properties**.
4. Passen Sie auf der Registerkarte **General** den Wert im Feld **Connections** an.

Die Management Console gibt das Dialogfeld **Number Of Connections Modified** aus und schlägt dort vor, die Werte der anderen Verbindungsparameter ebenfalls zu ändern. Mit **Accept** können Sie diesen Vorschlag akzeptieren.

Wenn Sie den Vorschlag mit **Cancel** ablehnen, dann beachten Sie bitte, dass Sie selbst sinnvolle Werte für die anderen Verbindungsparameter setzen müssen:
Wenn über die Verbindungen zu einem EIS Partner mehr Inbound- als Outbound-Kommunikation erfolgt, sollten weniger als die Hälfte der Verbindungen als Contention Winner definiert werden. Die Anzahl muss mit der Anzahl der Contention Winner abgestimmt werden, die Sie in der EIS-Konfiguration definiert haben. Die Summe beider Definitionen muss gleich der Gesamtzahl der Verbindungen sein.

Anzahl der parallelen Verbindungen für Inbound-Kommunikation über das UPIC-, RFC1006 oder openUTM-Socket-Protokoll

Sie ändern die Anzahl der UPIC-, Socket- und RFC1006-Verbindungen für die Inbound-Kommunikation mit Hilfe der Management Console:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Edit Properties**.
2. Wählen Sie im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** die Registerkarte **Performance Settings**.
3. Passen Sie unter **Number of Parallel Connections** den Wert für **Inbound UPIC**, **Inbound Socket** bzw. **Inbound RFC1006** an.

Diese Werte beziehen sich auch auf die Anzahl paralleler Verbindungen zu allen EIS-Systemen, die über diese Protokolle kommunizieren. Der Standard-Wert für alle drei Partner-Typen ist 10 parallele Verbindungen.

Die Management Console benötigt ihrerseits ebenfalls eine UPIC-Verbindung für die Kommunikation mit dem Proxy.

12.5 Asynchrone Verarbeitung im Proxy-Container

Bei den Einstellungen für die asynchrone Verarbeitung im Proxy-Container ist sowohl die gewünschte Lebensdauer der Aufträge zu berücksichtigen als auch, ob mit Outbound- oder Inbound-Kommunikation gearbeitet wird.

12.5.1 Lebensdauer von asynchronen Aufträgen

Asynchrone Outbound-Aufträge, die noch nicht an das EIS gesendet wurden, werden standardmäßig beim Beenden des Proxys gelöscht. Dies betrifft

- zeitgesteuerte Aufträge, deren Ablaufzeitpunkt noch nicht erreicht ist.
- Aufträge, die noch nicht gesendet werden konnten, weil keine Verbindung zum EIS Partner besteht.

Asynchrone Inbound-Aufträge, die noch nicht gestartet sind, werden ebenfalls beim Beenden des Proxys gelöscht. Dies betrifft

- Aufträge, die noch nicht an den Application Server gesendet wurden.
- Aufträge, die erneut an den Application Server gesendet werden müssen (Redelivery).

Haben Sie bezüglich der Lebensdauer von asynchronen Aufträgen höhere Anforderungen, stellen Sie mit Hilfe der Management Console den **Proxy Container Mode** auf **Durable Asynchronous Processing**. In diesem Fall sollten Sie ggf. auch die Pagepool-Größe erhöhen (siehe Abschnitt „Größe der Speicherbereiche ändern“ auf Seite 512).

Sie ändern die Einstellung des **Proxy Container Mode** mit Hilfe der Management Console:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Edit Properties**.
2. Wählen Sie im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** die Registerkarte **Performance Settings**.
3. Wählen Sie unter **Proxy Container Mode** die Option **Durable Asynchronous Processing**.

Sie müssen den Expertenmodus aktivieren, um die Registerkarte **Performance Settings** anzuzeigen. Detaillierte Informationen finden Sie in der Online-Hilfe der Management Console.

12.5.2 Inbound-Kommunikation

Die Anzahl der Prozesse, die parallel für die Abarbeitung einer asynchronen Inbound-Nachricht aktiv sein dürfen, ist abhängig vom Anwendungs-Szenario. Sie darf aber höchstens so groß sein wie die Anzahl der Prozesse minus Eins. Standardmäßig wird der Proxy so gestartet, dass zu einem Zeitpunkt nur maximal zwei Prozesse für die Ausführung asynchroner Inbound-Nachrichten aktiv sein dürfen.

Sie ändern die Anzahl der Prozesse mit Hilfe der Management Console:

1. Wählen Sie im Kontextmenü des Proxys den Befehl **Edit Properties**.
2. Wählen Sie im Eigenschaftsfeld **Edit Properties of Local/Remote Proxy** die Registerkarte **Performance Settings**.
3. Ändern Sie unter **Number of Proxy Container Processes** den Wert für **For Asynchronous Jobs**.

Sie müssen den Expertenmodus aktivieren, um die Registerkarte **Performance Settings** anzuzeigen. Detaillierte Informationen finden Sie in der Online-Hilfe der Management Console.

12.6 OSI-SCRATCH-AREA im Proxy-Container

Die Größe des dynamischen Speicherbereichs für OSI TP Verbindungen ist in der Datei `input.system.txt` mit folgender Anweisung festgelegt:

```
MAX OSI-SCRATCH-AREA=<wert>
```

`<wert>` gibt dabei die Größe des Speicherbereichs in KB an.
Der Standardwert beträgt 1024 (1MB), der Maximalwert ist 32767 KB.

Sie finden die Datei `input.system.txt`

- auf Solaris- und Linux-Systemen im Verzeichnis:
  `<Proxy_home>/`def
- auf Windows-Systemen im Verzeichnis:
  `<Proxy_home>\`def

Speicherbereich anpassen

Wenn der dynamische Speicherbereich im laufenden Betrieb zu klein ist (z.B. wegen einer großen Anzahl von parallelen Verbindungen), wird der Anwendungslauf des Proxy-Containers mit der Meldung `K060` abgebrochen. Als Insert der Meldung wird ein Abbruchgrund angegeben, dessen Beschreibung auf einen Speicherengpass hindeutet, z.B. `SACT28`. Die Abbruchgründe sind im Kapitel „Logging, Diagnose und Fehlerbehebung“ auf Seite 517 beschrieben.

Als Maßnahme sollten Sie den Wert der Anweisung `MAX OSI-SCRATCH-AREA=<wert>` erhöhen. Sinnvoll ist als erster Schritt eine Verdoppelung des Wertes. Anschließend müssen Sie die Proxy-Konfiguration aktualisieren (z.B. mit Hilfe der Management Console: Befehl **Update Configuration** im Kontextmenü des Proxy-Containers).

12.7 Anzahl der Semaphore im Proxy-Container

Der Proxy-Container benutzt einen Bereich von Semaphoren für globale Operationen. Bei einer großen Anzahl an Container-Prozessen sollten Sie den Maximalwert für diesen Bereich erhöhen. Bei einem Engpass wird die Meldung `U189` mit dem Insert `SEMA USED` in die Container-Protokolldatei ausgegeben (siehe Seite 595).

Anzahl erhöhen

Die Anzahl der Semaphore ist in der Datei `input.system.txt` mit folgender Anweisung festgelegt:

```
MAX SEMARRAY=(<schluessel>, <anzahl>)
```

Sie finden die Datei

- auf Solaris- und Linux-Systemen im Verzeichnis:
  `<Proxy_home>/`def
- auf Windows-Systemen im Verzeichnis:
  `<Proxy_home>\`def

Um die Anzahl der Semaphore zu erhöhen, erhöhen Sie den Wert für `<anzahl>` in der Anweisung `MAX SEMARRAY`. Für die Berechnung von `<anzahl>` gilt folgende Formel:

```
<anzahl>=nProc/10 + 2
```

`<nProc>` gibt dabei die Anzahl der Container-Prozesse an.

13 Logging, Diagnose und Fehlerbehebung

Dieses Kapitel enthält Informationen zu folgenden Themen:

- Logging mit Log4j
- Logging mit JDK-Logging
- Logging mit Log4j konfigurieren
- Logwriter für Connection Factory
- Diagnose des BeanConnect Resource Adapters
- Diagnose des BeanConnect Proxy-Containers
- Diagnose der BeanConnect Management Console
- Diagnose der BeanConnect Tools
- Diagnose des openUTM-LU62 Gateways
- Diagnose von SNAP-IX auf Solaris-Systemen
- Diagnose des IBM Communications Server auf Linux-Systemen
- Diagnose des IBM Communications Server auf Windows-Systemen
- Diagnosedaten sammeln
- Fehlermeldungen des BeanConnect Proxy-Containers
- Fehlermeldungen des openUTM-LU62 Gateways
- Fehlercodes

Da der BeanConnect Proxy auf openUTM basiert, benötigen Sie zur Diagnose außerdem das openUTM-Handbuch „Meldungen, Test und Diagnose in Unix- und Windows-Systemen“.

13.1 Logging mit Log4j

BeanConnect benutzt das Softwareprodukt Log4j für die Logging-Funktionalität. Log4j ist Bestandteil des Jakarta-Projekts der Apache Software Foundation. Log4j bietet Schnittstellen für Logging-Daten (Laufzeit-Daten, Trace-Records etc.) sowie zum Konfigurieren der Protokoll-Ausgabe.

Log4j verwendet in XML geschriebene Konfigurationsdateien. Die Namen dieser Dateien dürfen nicht verändert werden. Sämtliche BeanConnect-Komponenten, die Log4j verwenden, werden mit vorkonfigurierten Konfigurationsdateien ausgeliefert. Die Namen der Ausgabedateien sind in den Konfigurationsdateien vordefiniert. Die unterschiedlichen BeanConnect-Komponenten verwenden unterschiedliche Namen für ihre Ausgabedateien.

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Grundlagen von Log4j
- Logger
- Appender
- Funktionsweise des Rolling File Appenders

Weitergehende Informationen zum Thema Log4j finden Sie unter
`http://logging.apache.org/log4j/2.x/manual/index.html`

13.1.1 Grundlagen von Log4j

Log4j verwendet zwei Hauptkomponenten: den „Logger“ als Meldungsquelle und den „Appender“, der das Meldungsziel definiert. Die Meldungen, die an einen Logger übergeben werden, werden von allen Appendern ausgegeben, die diesem Logger zugewiesen sind.

13.1.1.1 Logger

Ein Logger ist eine Meldungsquelle. Ein Programm, das Logging-Daten schreibt, erhält von Log4j sogenannte Logger-Objekte und gibt seine Meldungen über diese Objekte aus.

Namensbereich

Der Namensbereich der Logger ist hierarchisch strukturiert. Die Namenskonvention ist dieselbe wie für Java-Packages, d.h. die einzelnen Ebenen der Hierarchie sind mit Punkten im Namen voneinander getrennt. Innerhalb dieser Hierarchie erben die Logger ihre Eigenschaften von ihren Parents, es sei denn es sind explizit eigene Eigenschaften für sie definiert. Die Wurzel der Hierarchie bildet der „Root-Logger“, der keinen eigenen Namen hat und immer vorhanden ist.

Beispiel 21 Logger-Namensbereich

Der Logger mit dem Namen `BeanConnect` ist der (direkte) Parent-Logger des Loggers mit dem Namen `BeanConnect.info` und ist ebenfalls der Parent-Logger des Loggers mit dem Namen `BeanConnect.Datasources.OLTP`.

Level

Der Level ist eine Eigenschaft, die sowohl einem Logger als auch einer Meldung zugewiesen werden kann. Wenn der Logger aufgerufen wird, identifiziert das Logging-Programm den Level der Meldung. Abhängig davon, welcher Level dem betreffenden Logger zugewiesen wurde, entscheidet Log4j, ob die übergebene Meldung protokolliert wird oder nicht. Es werden nur diejenigen Meldungen protokolliert, bei denen der Meldungs-Level größer oder gleich dem Logger-Level ist. Mit dem Level `OFF` wird der Logger deaktiviert.

Log4j unterstützt folgende Level (absteigend):

Level	Bedeutung
`FATAL`	Schwerwiegender Fehler, höchster Level
`ERROR`	Fehler
`WARN`	Warnung
`INFO`	Informationen
`DEBUG`	Debug-Ausgabe
`TRACE`	Trace-Ausgabe, niedrigster Level
`OFF`	Dieser Level kann nur Loggern zugewiesen werden. Sämtliche Meldungen, die über diesen Logger ausgegeben werden, werden unterdrückt. Der Logger ist ausgeschaltet.

Beispiel 22 Logging-Level

Wenn ein Logging-Ereignis mit dem Level `DEBUG` an einen Logger übergeben wird, dem der Level `INFO` zugewiesen ist, wird die Meldung unterdrückt. Meldungen der Level `INFO`, `WARN`, `ERROR` und `FATAL` werden jedoch ausgegeben.

Wenn Sie den Logger-Level `ERROR` zuweisen, werden nur Meldungen mit dem Level `ERROR` und `FATAL` ausgegeben.

13.1.1.2 Appender

Die Meldungsziele werden durch Appender definiert. Log4j bietet eine Reihe unterschiedlicher vordefinierter Appender. Dazu gehören:

- File Appender

  Die Meldungen werden in eine Datei geschrieben.

- Console Appender

  Die Meldungen werden nach `System.out` oder `System.err` geschrieben.

- Socket Appender

  Die Meldungen werden an einen Socket geschrieben und können dadurch auch über Rechner-Grenzen hinweg an einen Log4j-Socket-Reader geschickt werden, der die Meldungen weiter verarbeiten kann (siehe Abschnitt „BeanConnect Management Console als Log4j-Socket-Reader konfigurieren“ auf Seite 527 für den Resource Adapter und für den Proxy).

- Rolling File Appender

  Die Meldungen werden in eine Datei geschrieben. Erreicht die Dateigröße den definierten Grenzwert, wird die Datei geschlossen und die Meldungen werden in eine neue Trace-Datei geschrieben.

Die Logging-Ereignisse, die an einen Logger übertragen werden, werden über die Appender ausgegeben, die dem Logger zugewiesen sind.

Beispiel 23 Ausgabe der Logging-Ereignisse über den Appender

Gibt es einen Logger mit dem Namen `Trace`, dem die Appender `Console` (Console Appender) und `File` (File Appender) zugewiesen sind, wird eine von diesem Logger ausgegebene Meldung sowohl über den File Appender als auch den Console Appender ausgegeben. Sie erscheint in der Datei und auf der Console.

13.1.1.3 Funktionsweise des Rolling File Appenders

Der Rolling File Appender ermöglicht das Logging in eine Datei mit mehreren Backup-Dateien. Sie können den maximalen Platz auf der Platte konfigurieren, der für die Trace-Dateien zur Verfügung stehen soll. Der Rolling File Appender läuft sowohl in einer Single-Tasking-Umgebung (wie im Resource Adapter) als auch in einer Multi-Tasking-Umgebung (wie im Proxy-Container).

Logging-Dateien

Der Rolling File Appender erstellt nur für die interne Nutzung die Datei `<File>.gen` und (in einer Multi-Tasking-Konfiguration) die Datei `<File>.lck`. Diese benötigt er im Betrieb. `<File>` ist der Name der aktuellen Logging-Datei, z.B. `BeanConnect.logging.txt`. Diese Datei wird z.B. in der Konfigurationsdatei für den Appender `BeanConnectShortLoggingFile` angegeben.

Der Rolling File Appender schreibt immer in die Logging-Datei `<File>`. Beim Umschalten wird

- die Datei `<File>` in eine Backup-Datei kopiert.
- die Datei `<File>` neu geschrieben.

Beim Umschalten werden die ältesten vorhandenen Backup-Dateien gelöscht, wenn:

- nach dem Kopieren der Datei mehr Backup-Dateien vorhanden sind als durch `<MaxNbrBackupFiles>` definiert sind, und
- `<MaxNbrBackupFiles>` größer als 0 ist.

Angaben in spitzen Klammern (<>) wie `<File>` oder `<MaxNbrBackupFiles>` sind Eigenschaften des Appenders. Weitere Informationen finden Sie in Abschnitt „Vordefinierte Logging-Konfiguration eines Resource Adapters“ auf Seite 537 bzw. Abschnitt „Vordefinierte Logging-Konfiguration eines Proxys“ auf Seite 544.

Beispiel 24 Logging-Dateien

In diesem Beispiel ist `<File>=<Filename>.txt und <MaxNbrBackupFiles>=3`.

Situation vor dem Umschalten:

<Filename>.txt	In Benutzung
<Filename>.13.txt	Älteste Backup-Datei
<Filename>.14.txt	
<Filename>.15.txt	Neueste Backup-Datei

Beim Umschalten wird `<Filename>.txt` in `<Filename>.16.txt` kopiert, `<Filename>.13.txt` wird gelöscht und `<Filename>.txt` wieder beschrieben.

Situation nach dem Umschalten:

<Filename>.txt	In Benutzung
<Filename>.14.txt	Älteste Backup-Datei
<Filename>.15.txt	
<Filename>.16.txt	Neueste Backup-Datei

13.2 Logging mit JDK-Logging

In den BeanConnect Produkten wird für das Logging standardmäßig das Paket Apache Log4j verwendet. Da der BeanConnect Resource Adapter in einem Application Server abläuft, kann es vorkommen, dass in diesem Umfeld kein Log4j benutzt werden soll/darf.

Um dieses Problem zu lösen, wird im Resource Adapter durchgängig das BeanConnect Logging Framework verwendet. Dieses Logging Framework basiert auf dem Paket Apache Jakarta Commons Logging mit BeanConnect-spezifischen Erweiterungen.

Das BeanConnect Logging Framework unterstützt standardmäßig die beiden folgenden Logging-Pakete:

- Apache Log4j
- Java Logging API

Das Logging mit Apache Log4j ist die Standardeinstellung. Sie wird verwendet, wenn Sie an dem ausgelieferten BeanConnect-RAR-Archiv des Resource Adapters nichts verändern.

Wenn Sie das Java Logging verwenden möchten, dann gehen Sie wie folgt vor:

- Löschen Sie die Datei `BeanConnectLog4j.jar` aus dem BeanConnect-RAR-Archiv des Resource Adapters.

Die Funktionsweise des JDK-Logging ist ähnlich wie bei Log4j, siehe Abschnitt „Grundlagen von Log4j“ auf Seite 518.

Sie müssen die BeanConnect-spezifischen Logger manuell in eine Logging-Konfigurationsdatei von JDK eintragen. Eine Beschreibung des JDK-Loggings finden Sie in der Dokumentation von Java unter
`http://docs.oracle.com/javase/7/docs/technotes/guides/logging/overview.html`

13.3 Logging mit Log4j konfigurieren

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Logging für BeanConnect Resource Adapter und Proxy konfigurieren
- Log4j-Konfigurationsdatei mit der BeanConnect Management Console bearbeiten
- BeanConnect Management Console als Log4j-Socket-Reader konfigurieren
- Logging-Ereignisse auf der BeanConnect Management Console anzeigen
- Logging-Datei von Log4j in der BeanConnect Management Console anzeigen

13.3.1 Logging für BeanConnect Resource Adapter und Proxy konfigurieren

Für den **BeanConnect Resource Adapter** ist die Log4j-Konfigurationsdatei `BeanConnect.log4j.properties.xml` vorkonfiguriert. Diese XML-Datei ist nach der Installation des Resource Adapters (siehe Kapitel „BeanConnect installieren“ auf Seite 49) im Unterverzeichnis `config` abgelegt.

Um die Datei `BeanConnect.log4j.properties.xml` für den Resource Adapter mit Hilfe der Management Console zu bearbeiten, muss auf dem Rechner, auf dem der Application Server läuft, ein MC-CmdHandler verfügbar sein.

Für den **BeanConnect Proxy** ist die Log4j-Konfigurationsdatei `log4j.properties.xml` vorkonfiguriert. Nach der Installation des Proxys (siehe Kapitel „BeanConnect installieren“ auf Seite 49) ist diese XML-Datei im Unterverzeichnis `config` des Proxys abgelegt.

Die Änderungen an der Logging-Konfiguration treten erst in Kraft, wenn Sie den Proxy starten bzw. speichern und neu laden (Befehl **Save/Restart** - **Save & Restart** im Kontextmenü des Proxys).

Sie können die Konfigurationseinträge in beiden Konfigurationsdateien mit einem Texteditor bearbeiten. Alternativ können Sie die Dateien auch, wie nachfolgend beschrieben, über die Management Console ändern. In der Management Console können Sie Logger hinzufügen und die vorhandenen Logger und Appender konfigurieren und löschen.

13.3.1.1 Logger konfigurieren

Sie passen die Konfiguration eines Loggers mit Hilfe der Management Console an:

1. Öffnen Sie die Log4j-Konfigurationsdatei wie folgt:

   BeanConnect Resource Adapter:

   ► Klicken Sie im Navigationsbaum des Resource Adapters auf den Knoten **Log4j Configuration**.

   BeanConnect Proxy:

   ► Starten Sie, falls erforderlich, den MC-CmdHandler für den entfernten Proxy.

   ► Klicken Sie im Navigationsbaum des Proxys auf die Knoten **Advanced Features** - **Diagnosis** - **Configuration** - **Proxy Container Log4j**.

   Die Management Console zeigt die Konfiguration im Arbeitsbereichsfenster **Logging Configuration** an.

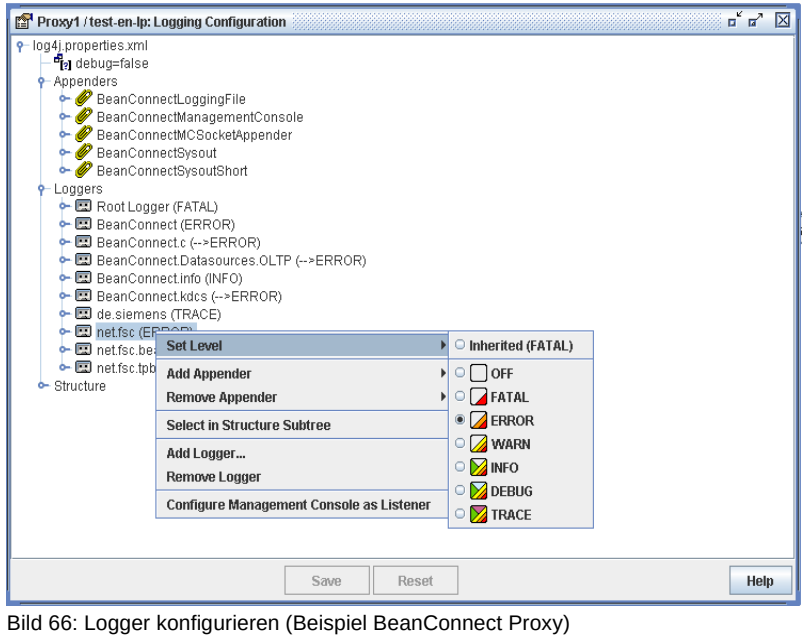


Bild 66: Logger konfigurieren (Beispiel BeanConnect Proxy)

Die Logger werden in einer Baumstruktur angezeigt. Alle vorhandenen Logger sind unter dem Knoten **Loggers** aufgelistet. Die Bezeichnung eines Loggers besteht aus seinem Namen sowie dem zugewiesenen Level in Klammern. Hat der Logger den Level von seinem Parent geerbt, ist dies durch einen Pfeil gekennzeichnet (-->).

2. Um die Konfiguration zu ändern, klicken Sie mit der rechten Maustaste auf einen Logger oder einen seiner untergeordneten Knoten und wählen Sie den entsprechenden Befehl aus dem Kontextmenü.

Detaillierte Informationen zu den Arbeitsbereichsfenstern und den Kommandos der Kontextmenüs finden Sie in der Online-Hilfe der Management Console.

13.3.1.2 Appender konfigurieren

Sie passen die Konfiguration eines Appenders mit Hilfe der Management Console an:

1. Öffnen Sie die Log4j-Konfigurationsdatei wie folgt:

   BeanConnect Resource Adapter:

   ► Klicken Sie im Navigationsbaum des Resource Adapters auf den Knoten **Log4j Configuration**.

   BeanConnect Proxy:

   ► Starten Sie, falls erforderlich, den MC-CmdHandler für den entfernten Proxy.

   ► Klicken Sie im Navigationsbaum des Proxys auf die Knoten **Advanced Features** - **Diagnosis** - **Configuration** - **Proxy Container Log4j**.

   Die Management Console zeigt die Konfiguration im Arbeitsbereichsfenster **Logging Configuration** an.

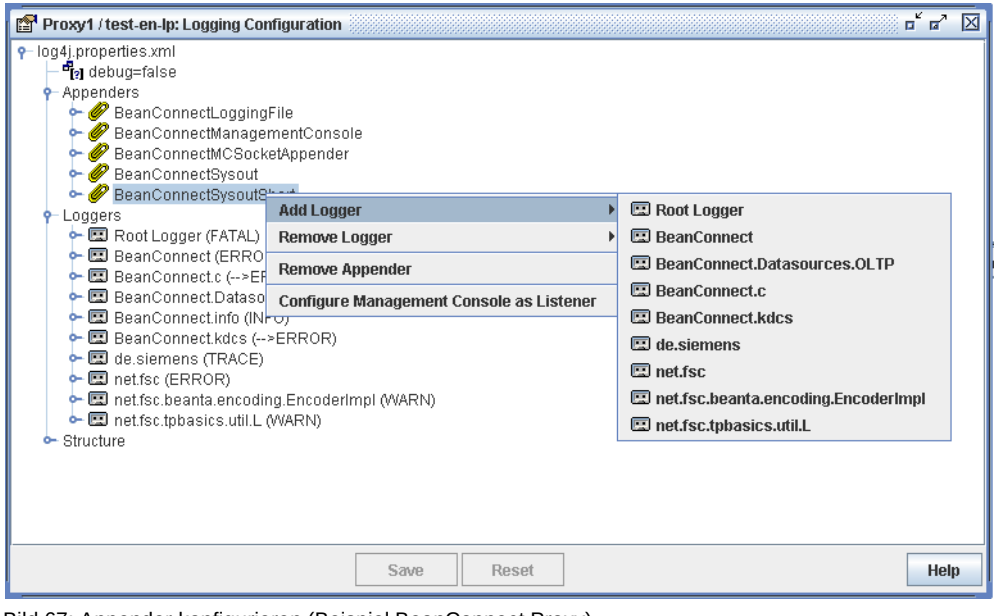


Bild 67: Appender konfigurieren (Beispiel BeanConnect Proxy)

Die Appender werden in einer Baumstruktur angezeigt. Alle vorhandenen Appender sind unter dem Knoten **Appenders** aufgelistet.

2. Um die Konfiguration zu ändern, klicken Sie mit der rechten Maustaste auf einen Appender oder einen seiner untergeordneten Knoten und wählen Sie den entsprechenden Befehl aus dem Kontextmenü.

   Wenn Sie dem Appender beispielsweise einen Logger hinzufügen möchten, wählen Sie den Befehl **Add Logger** aus dem Kontextmenü des Appenders. Die Liste zeigt alle Logger an, die dem Appender noch hinzugefügt werden können.

Detaillierte Informationen zu den Arbeitsbereichsfenstern und den Kommandos der Kontextmenüs finden Sie in der Online-Hilfe der Management Console.

13.3.2 Log4j-Konfigurationsdatei mit der BeanConnect Management Console bearbeiten

Mit der Management Console können Sie beliebige Log4j-Konfigurationsdateien im XML-Format bearbeiten:

1. Wählen Sie den Befehl **Extras - Edit Log4j Configuration File** in der Management Console.

   Das Dialogfeld **Choose Log4j Configuration File (XML)** wird angezeigt.

2. Wählen Sie in diesem Dialogfeld eine Konfigurationsdatei im XML-Format.

   Die Management Console zeigt die in dieser Datei enthaltene Konfiguration im Arbeitsbereichsfenster **Edit Log4j Configuration File** an. Sie bearbeiten jede beliebige Konfigurationsdatei genau wie die Log4j-Konfigurationsdatei eines Resource Adapters oder Proxys (siehe Abschnitt Abschnitt „Logger konfigurieren“ auf Seite 524 und Abschnitt „Appender konfigurieren“ auf Seite 525).

Detaillierte Informationen zu den Arbeitsbereichsfenstern und den Befehlen der Kontextmenüs finden Sie in der Online-Hilfe der Management Console.

13.3.3 BeanConnect Management Console als Log4j-Socket-Reader konfigurieren

Die Management Console kann als Log4j-Socket-Reader für einen Resource Adapter oder einen Proxy fungieren. Entsprechend können Logging-Ausgaben direkt an ein Fenster in der Management Console geschickt werden (siehe Abschnitt „Logging-Ereignisse auf der BeanConnect Management Console anzeigen“ auf Seite 528).

Um die Management Console entsprechend zu konfigurieren, gehen Sie wie folgt vor:

1. Wählen Sie den Befehl **Configure Management Console as Listener** aus dem Kontextmenü eines beliebigen Knotens im Arbeitsbereichsfenster **Logging Configuration**.
2. Soll die Management Console vom Start an bei allen nachfolgenden Sessions am angegebenen Listener-Port auf Logging-Ereignisse warten, aktivieren Sie die Option **Automatically Begin to Listen** auf der Registerkarte **General** im Arbeitsbereichsfenster **General Diagnostic Info Configuration**.
3. BeanConnect Resource Adapter:
   Passen Sie den Appender `BeanConnectMCSocketAppender` in der Datei `BeanConnect.log4j.properties.xml` des Resource Adapters an:
   - Beim Parameter `RemoteHost` muss der Rechner angegeben werden, auf dem die Management Console läuft.
   - Beim Parameter `Port` muss der Listener-Port angegeben werden, an dem die Management Console auf Logging-Ereignisse des Proxy-Containers wartet.

Die Management Console stellt eine Liste aller Logging Listener zur Verfügung. Sie ist über den Knoten **Log4j Logging Listeners** auf der höchsten Ebene des Navigationsbaums zugreifbar.

Die Konfiguration besteht für den Proxy aus den folgenden Schritten:

1. Wenn für diesen Fall in den Eigenschaften des betreffenden Proxys noch kein Socket Listener Port eingetragen wurde, fragt die Management Console zunächst diesen Port ab.

Der Port ist ein lokaler Listener-Port und darf deshalb mit keinem anderen bereits auf dem Rechner der Management Console verwendeten lokalen Listener-Port identisch sein. Die Management Console kann jedoch nur ihre eigenen Daten überprüfen, um die Eindeutigkeit der eingegebenen Ports sicherzustellen und zu erzwingen. Listener-Ports, die von anderen Anwendungen auf dem Rechner verwendet werden, können von der Management Console nicht überprüft werden. Hier müssen Sie als Anwender selbst die Eindeutigkeit sicherstellen.

2. Sobald Sie einen Port eingeben, prüft die Management Console, ob die folgenden Appender in der Logging-Konfiguration des Proxys enthalten sind:
   - ein Async Appender mit dem Namen `BeanConnectManagementConsole`
   - ein Socket Appender mit dem Namen `BeanConnectMCSocketAppender`

   Fehlen diese Appender, so werden sie von der Management Console erstellt. Sind sie bereits vorhanden, werden bestimmte Parameter der Appender überprüft und entsprechend eingestellt.
3. Die Management Console weist den Loggern `BeanConnect` und `de.siemens` (sofern vorhanden) den Appender `BeanConnectManagementConsole` zu. Entsprechend werden Logging-Ereignisse auf der Management Console über diese beiden Logger oder über Logger ausgegeben, die die Appender von diesen beiden Loggern erben.
4. Die Management Console aktiviert das Logging für die aktuelle Sitzung und wartet am angegebenen Listener-Port auf Logging-Ereignisse.

Listener-Port wechseln

Um den Socket-Listener-Port zu wechseln, gehen Sie wie folgt vor:

- Klicken Sie im Navigationsbaum des Proxys auf die Knoten **Advanced Features** - **Diagnosis** - **Configuration** - **General Diagnostic Info** und geben Sie den **MC Logging Listener Port** ein.

Diese Änderung tritt erst in Kraft, wenn der Proxy das nächste Mal gestartet oder gespeichert und neu geladen wird (Befehl **Save/Restart** - **Save & Restart** im Kontextmenü des Proxys).

13.3.4 Logging-Ereignisse auf der BeanConnect Management Console anzeigen

Wenn Sie die Management Console entsprechend konfiguriert haben (siehe Abschnitt „BeanConnect Management Console als Log4j-Socket-Reader konfigurieren“ auf Seite 527), erhält sie die an den Socket Appender des betreffenden Resource Adapters bzw. Proxys ausgegebenen Logging-Ereignisse.

Um die Logging-Ereignisse in der Management Console anzuzeigen, verfahren Sie wie folgt:

- Starten Sie den betreffendenProxy (siehe Abschnitt „Starten eines Proxy“ auf Seite 273).

- Prüfen Sie, ob die Option **Automatically Begin to Listen** im Arbeitsbereichsfenster **General Diagnostic Info Configuration** aktiviert ist. Nur wenn diese Option aktiviert ist, wartet die Management Console am angegebenen Listener-Port vom Start an auf Logging-Ereignisse.

  Alternativ können Sie auch den Befehl **Start Listening** im Kontextmenü des Knotens **Proxy Container Log4j** unterhalb des Knotens **Configuration** verwenden.

- Klicken Sie im Navigationsbaum des Proxys auf die Knoten **Advanced Features** - **Diagnosis** - **Output** - **Proxy Container Log4j**.

  Die Management Console zeigt die eingegangenen Meldungen im Arbeitsbereichsfenster **Logging Output** an.

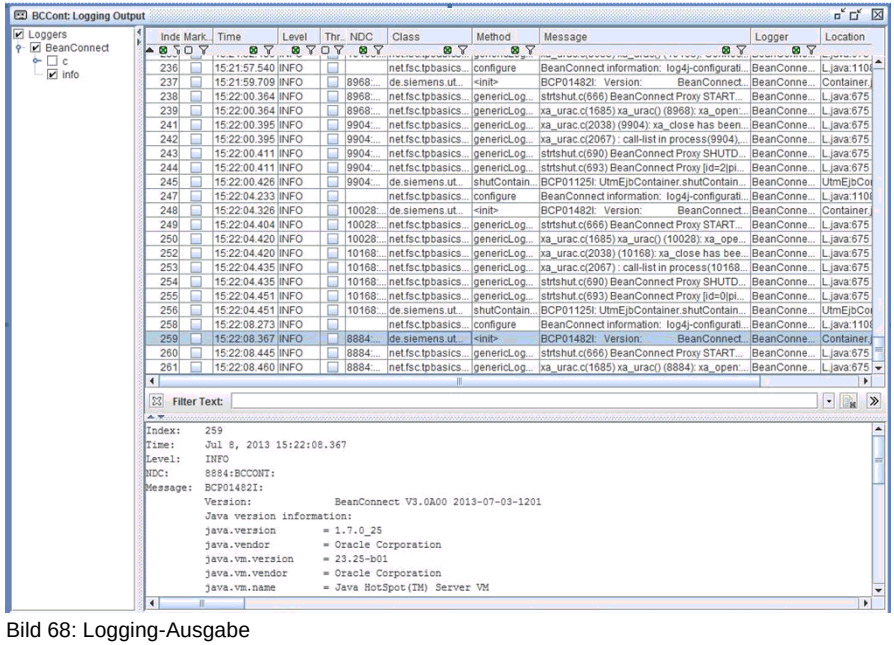


Bild 68: Logging-Ausgabe

Das Arbeitsbereichsfenster **Logging Output** umfasst folgende Bereiche:

- **Logger-Baum**: In diesem Bereich werden alle Logger angezeigt, für die Meldungen in der Management Console vorhanden sind. Logger, deren Pfad in der Struktur rot dargestellt ist, haben mindestens eine Meldung mit dem Logging-Level `WARN`.

- **Logging-Ereignisliste**: In diesem Bereich werden die in der Management Console verfügbaren Logging-Ereignisse angezeigt, die den aktuellen Einstellungen im Arbeitsbereichsfenster **Logging Output** entsprechen. Rot angezeigte Meldungen haben mindestens den Logging-Level `WARN`.

  Wenn Sie ein Ereignis aus der Ereignisliste anklicken, wird es im Bereich unterhalb der Ereignisliste ausführlich angezeigt.

Detaillierte Informationen zu den Arbeitsbereichsfenstern und den Befehlen der Kontextmenüs finden Sie in der Online-Hilfe der Management Console.

Layout des Arbeitsbereichsfensters verändern

Standardmäßig wird der Logger-Baum in der linken und die Meldungsliste in der rechten Hälfte des Arbeitsbereichsfensters angezeigt. Diese Einstellung können Sie ändern, so dass der Logger-Baum und die Meldungsliste jeweils auf einer eigenen Registerkarte angezeigt werden:

► Klicken Sie im Arbeitsbereichsfenster **Logging Output** auf die Schaltfläche **Properties**.
Daraufhin wird das Dialogfeld **Logging Properties** angezeigt.

► Wählen Sie unter **Display Mode** den Anzeigemodus.

- **split**
  Der Logger-Baum und die Logging-Ereignisliste werden in der linken und rechten Hälfte des Arbeitsbereichsfensters angezeigt.
- **tabbed**
  Der Logger-Baum und die Logging-Ereignisliste werden auf zwei verschiedenen Registerkarten angezeigt.

Anzeige der Logger aktivieren/deaktivieren

Die Ereignisliste zeigt nur die Ereignisse von Loggern an, die im Logger-Baum aktiviert sind.

So aktivieren oder deaktivieren Sie die Anzeige eines Loggers:

Klicken Sie auf den betreffenden Logger-Knoten oder das zugehörige Kontrollkästchen. Dieser Knoten, alle Knoten im Pfad des Knotens nach oben bis zum Logger-Knoten und alle zugehörigen Kindknoten werden aktiviert/deaktiviert.

Meldungsanzeige ändern

Folgende Kriterien sind für die Anzeige von Ereignissen in der Ereignisliste ausschlaggebend:

- Es werden nur die Meldungen der Logger angezeigt, die im Logger-Baum aktiviert sind (siehe obigen Abschnitt).
- Es werden nur Meldungen angezeigt, die derzeit nicht unterdrückt werden.

Mit der Schaltfläche **Clear** im Arbeitsbereichsfenster **Logging Output** werden sämtliche Logging-Ereignisse der betreffenden Proxys, die derzeit in der Management Console vorhanden sind, ausgeblendet. Die Meldungen werden nicht mehr im Arbeitsbereichsfenster ausgegeben, bleiben jedoch in der Management Console erhalten. Sie können sie wieder einblenden, indem Sie auf die Schaltfläche **Consider All** klicken.

- Es werden nur die Meldungen angezeigt, deren Logging-Level mindestens dem minimalen Logging-Level entspricht, der der Liste zugewiesen ist.

  Den Logging-Level für die Meldungsliste stellen Sie entweder mit Hilfe der Schaltflächen im Arbeitsbereichsfenster **Logging Output** oder über die Option **Minimum Displayed Logging Level** im Dialogfeld **Logging Properties** ein.

- Wenn Sie im Dialogfeld **Logging Properties** eine Filtermaske **(Filter Mask)** eingeben, werden nur die Meldungen angezeigt, die in mindestens einer Spalte den Filtertext enthalten (dabei werden Groß- und Kleinschreibung unterschieden).

13.3.5 Logging-Datei von Log4j in der BeanConnect Management Console anzeigen

In der Management Console können Sie die Log4j Logging-Dateien im XML-Format offline anzeigen.

Eine Logging-Datei im XML-Format wird beispielsweise erstellt, wenn Sie einem vordefinierten File Appender einen XML File Appender hinzufügen (Befehl **Add XML File Appender** im Kontextmenü des betreffenden File Appenders).

So zeigen Sie die erstellte Logging-Datei an.

1. Wählen Sie den Befehl **Extras - Open Log4j Logging File (XMC)** in der Management Console.

   Das Dialogfeld **Choose a Log4j Logging File (XMC)** wird angezeigt.

2. Wählen Sie in diesem Dialogfeld eine Logging-Datei im XML-Format. Standardmäßig hat eine Logging-Datei im XML-Format die Dateinamenserweiterung `.xmc`.

   Die Management Console zeigt den Inhalt der Logging-Datei offline im Arbeitsbereichsfenster **Log4j Logging File** an. Dieses Arbeitsbereichsfenster ist genauso aufgebaut wie das Arbeitsbereichsfenster **Logging Output**, in dem Meldungen online auf der Management Console ausgegeben werden (siehe Abschnitt „Logging-Ereignisse auf der BeanConnect Management Console anzeigen“ auf Seite 528).

13.4 Logwriter für Connection Factory

Der Application Server stellt LogWriter zur Verfügung, auf den der BeanConnect Resource Adapter bei bestimmten Ereignissen System-Level-Informationen zu ManagedConnectionFactories und zu ManagedConnections schreibt. Diese Informationen sind für den Administrator des Application Servers bestimmt und dienen nicht der Diagnose von Problemen in den Anwendungsprogrammen.

Ereignisse und Ereignisklassen

Die Ereignisse werden in folgende Klassen unterteilt, zu denen bestimmte Ereignisse gehören:

- Fehler
- Transaktionen:
  - Beginn einer Transaktion für eine Connection
  - Commit/Rollback einer Transaktion für eine Connection
- Lifecycle:
  - Erzeugen einer ManagedConnection
  - Anfordern eines Connection Handle für eine ManagedConnection
  - Switch einer Connection Handle für eine ManagedConnection
  - Freigabe einer Connecton Handle und Hinzufügen der ManagedConnection zum ConnectionPool des Application Servers
  - Entnahme einer ManagedConnection aus dem ConnectionPool des Application Servers
  - Freigabe einer ManagedConnection
  - Application Exceptions, die für eine Connection Handle an eine Anwendung geworfen werden
  - System Exceptions, die für eine ManagedConnection geworfen werden

LogWriter im Application Server konfigurieren

Für Oracle WebLogic Server konfigurieren Sie den LogWriter für eine ManagedConnectionFactory in der Datei `weblogic-ra.xml` wie folgt:

- Logging-Level

  Der Logging-Level bestimmt in welchem Granulat BeanConnect Meldungen zu einer ManagedConnectionFactory auf den LogWriter ausgibt. Den Logging-Level konfigurieren Sie bei Oracle WebLogic Server in der Datei `weblogic-ra.xml` mit der Property `logLevel`. Er ist für jede ManagedConnectionFactory getrennt einstellbar, es gibt die vier Log-Levels NONE, ERROR, INFO und ALL.

  Details finden Sie Abschnitt „Konfigurations-Properties für OSI TP / LU6.2 definieren“ auf Seite 111, Abschnitt „Konfigurations-Properties für UPIC definieren“ auf Seite 128. Beispiele finden Sie in Beispiel 6 auf Seite 120 und Beispiel 8 auf Seite 135.

- Logging Attribute

  Weitere Logging Attribute setzen Sie in der Definition einer ConnectionFactory im Sub-Element `<logging>`. Details zum Subelement `<logging>` finden Sie in der Dokumentation für den Oracle Weblogic Server in der Schema-Beschreibung für die Datei `weblogic-ra.xml`.

  Im Attribut `<log-filename>` geben Sie den Pfadnamen der Datei für die Logging-Ausgabe ein.

  Mit dem Attribut `<logging-enabled>` schalten Sie das Logging ein oder aus.

  Mit anderen Attributen können Sie u.a. steuern, wie groß eine Log-Datei werden darf oder wieviele Log-Dateien maximal für eine ManagedConnectionFactory angelegt werden sollen, falls `file rotation` konfiguriert ist.

  Beispiel

```
<logging>
    <log-filename>C:/temp/log/BeanConnect/echo.log</log-filename>
    <logging-enabled>true</logging-enabled>
    <rotation-type>bySize</rotation-type>
    <number-of-files-limited>true</number-of-files-limited>
    <file-count>3</file-count>
</logging>
```

  Für verschiedene Connection Factories sollten Sie unterschiedliche Dateien angeben. Andernfalls kann es zu Konflikten beim Schreiben der Dateien kommen, die zu verstümmelten Log-Sätzen führen können.

Beispiel

```
<connection-instance>
   <jndi-name>eis/beanconnect_oltp_echo</jndi-name>
      <connection-properties>
         <logging>
            <log-filename>C:/temp/log/BeanConnect/echo.log</log-filename>
            <logging-enabled>true</logging-enabled>
            <rotation-type>bySize</rotation-type>
            <number-of-files-limited>true</number-of-files-limited>
            <file-count>3</file-count>
         </logging>
         <properties>
            <property>
               <name>ConnectionURL</name>
               <value>oltp://echo</value>
            </property>
            <property>
                <name>displayName</name>
                <value>sample application/echo</value>
            </property>
            <property>
                <name>logLevel</name>
                <value>ALL</value>
            </property>
         </properties>
      </connection-properties>
    </connection-instance>
```

Format der Logging-Sätze

Alle Sätze, die BeanConnect auf den LogWriter schreibt, haben folgenden Aufbau:

```
BeanConnect:<date-time> <identifier> message
```

`<date-time>`	gibt Datum und Uhrzeit an, an dem der Satz geschrieben wurde. Format (Beispiel): 2015-07-17 08:30:26.810+0100.
`<identifier>`	gibt den Identifier an. Für eine ManagedConnectionFactory ist dies der Wert, der in der Property `DisplayName` der Datei `weblogic-ra.xml` konfiguriert ist, siehe Abschnitt „Konfigurations-Properties für OSI TP / LU6.2 definieren“ auf Seite 111 und Abschnitt „Konfigurations-Properties für UPIC definieren“ auf Seite 128. Für eine ManagedConnection hat der Identifier die Form `BCUnnnnn`, wobei jedes n für eine Ziffer steht. Für eine Connection Handle hat der Identifier die Form `BCUnnnnn.i`, wobei jedes n für eine Ziffer und i für eine Zahl steht.
`message`	Meldung, die der Resource Adapter ausgibt.

Beispiel 25 Einträge in der LogWriter Datei

1. Für Lifecycle-Ereignisse werden Datum, Uhrzeit und Identifier der ManagedConnection protokolliert:

```
BeanConnect:2015-07-17 08:30:51.225+0100 <sample application/echo>:
Managed connection with id <BCU00002> destroyed
```

2. Für Ereignisse, die sich auf eine Exception beziehen, wird auch die Exception protokolliert:

```
BeanConnect:2015-07-17 08:33:35.198+0100 <sample application/echo>:
rcvString(): Exception thrown for connection <BCU00003.2>:
net.fsc.jca.communication.EISConnectionException:
net.fsc.jca.communication.EISConnectionException:
exceptionShortageOfResources: shortage of resources (4OZ,KD10): no
connection to partner; partner: (SMPOSICL,gssbwrit), Dialog, error code:
undefined error code [EC_UNDEFINED:0], connectionId: , error code:
undefined error code [EC_UNDEFINED:0], connectionId: BCU00003.2, proxy:
MYPROXY:30004/BCU30004, userId: BCU00003; diagnostic string:
```

3. Bei einer Kommunikation mit Transaktionen werden im Allgemeinen sechs Logging-Sätze erzeugt:

```
BeanConnect:2015-07-17 08:30:27.138+0100 <sample application/echo>:
Managed connection with id <BCU00002> taken from pool
BeanConnect:2015-07-17 08:30:27.138+0100 <sample application/echo>:
Connection handle with id <BCU00002.1> created
BeanConnect:2015-07-17 08:30:27.653+0100 <sample application/echo>:
Transaction started for managed connection "BCU00002" with xid:
formatID=48801, gtrid=002157A9 D15A3057 A4BD, bqual=6569732F 6265616E
636F6E6E 6563745F 6F6C7470 5F656368 6F
BeanConnect:2015-07-17 08:30:43.815+0100 <sample application/echo>:
Transaction committed for managed connection "BCU00002" with xid:
formatID=48801, gtrid=002157A9 D15A3057 A4BD, bqual=6569732F 6265616E
636F6E6E 6563745F 6F6C7470 5F656368 6F
BeanConnect:2015-07-17 08:30:43.908+0100 <sample application/echo>:
Connection handle with id <BCU00002.1> released
BeanConnect:2015-07-17 08:30:44.267+0100 <sample application/echo>:
Managed connection with id <BCU00002> returned for pooling
```

4. Bei einer Kommunikation ohne Transaktionen werden im Allgemeinen vier Logging-Sätze erzeugt:

```
BeanConnect:2015-07-17 08:50:41.117+0100 <sample application/echo>:
Managed connection with id <BCU00005> taken from pool
BeanConnect:2015-07-17 08:50:41.117+0100 <sample application/echo>:
Connection handle with id <BCU00005.4> created
BeanConnect:2015-07-17 08:50:53.753+0100 <sample application/echo>:
Connection handle with id <BCU00005.4> released
BeanConnect:2015-07-17 08:50:54.112+0100 <sample application/echo>:
Managed connection with id <BCU00005> returned for pooling
```

13.5 Diagnose des BeanConnect Resource Adapters

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Überblick zum Logging des BeanConnect Resource Adapters
- Vordefinierte Logging-Konfiguration eines Resource Adapters
- Logging von User Interface Aufrufen

13.5.1 Überblick zum Logging des BeanConnect Resource Adapters

Dieses Kapitel beschreibt das Logging des BeanConnect Resource Adapters. Das Logging findet standardmäßig mit Log4j statt. Der Umfang des Loggings wird über Konfigurationsdateien gesteuert. In der Installation des BeanConnect Resource Adapters sind folgende Konfigurationsdateien enthalten:

- Die Datei `config/BeanConnect.log4j.properties.xml`

  Basis-Logging-Daten (Standard)

- Die Datei `config/BeanConnect.log4j.properties_default.xml`

  Entspricht der Datei `config/BeanConnect.log4j.properties.xml`

- Die Datei `config/BeanConnect.log4j.properties_debug.xml`

  detaillierte Logging-Daten

- Die Datei `config/BeanConnect.log4j.properties_error.xml`

  Es werden nur Fehlermeldungen protokolliert.

Je nach Ihren Anforderungen können Sie eine der Konfigurationsdateien verwenden.

Informationen dazu, wie Sie die Logging-Daten manuell steuern, finden Sie in Abschnitt „Logging mit Log4j“ auf Seite 518 und Abschnitt „Diagnose des BeanConnect Resource Adapters“ auf Seite 536.

Standard-Logging des Resource Adapters

- Nach dem Deployment des Resource Adapters ist automatisch das Basis-Logging aktiviert.
- Die Logging-Dateien werden im Logging-Verzeichnis des Application Servers gespeichert (bei Oracle WebLogic Server in `<WebLogicServerDomainDirectory>/servers/<ServerName>/logs`).

- Es werden folgende Logging-Dateien erstellt:
  - `BeanConnect.logging.txt` (Informationen für den Benutzer)
  - `BeanConnect.extlogging.txt` (Logging-Daten, die im Fehlerfall an den Systemdienst geschickt werden müssen.)
  - `BeanConnect.extlogging.txt.xmc` (Zusätzliche Logging-Daten, die im Fehlerfall an den Systemdienst geschickt werden müssen.)

Logging des Resource Adapters anpassen

Kopieren Sie die in der Resource Adapter Installation enthaltenen Konfigurationsdateien in das Konfigurationsverzeichnis des Application Servers.
Bei Oracle WebLogic Server ist dies das Verzeichnis `<WebLogicServerDomainDirectory>/config`.

- Logging erweitern (ausführliche Ablaufprotokollierung):
  Benennen Sie die kopierte Datei `config/BeanConnect.log4j.properties_debug.xm` um in `BeanConnect.log4j.properties.xml`.
- Logging reduzieren (reine Fehlerprotokollierung)
  Benennen Sie die kopierte Datei `config/BeanConnect.log4j.properties_error.xml`um in `BeanConnect.log4j.properties.xml`
- Anwendungsspezifische Logging-Steuerung des Resource Adapters
  Passen Sie die Datei `config/BeanConnect.log4j.properties.xml` wie nachfolgend beschrieben Ihren Anforderungen an (siehe Abschnitt „Logging mit Log4j“ auf Seite 518 und Abschnitt „Diagnose des BeanConnect Resource Adapters“ auf Seite 536) und benennen Sie sie um in `BeanConnect.log4j.properties.xml`.

13.5.2 Vordefinierte Logging-Konfiguration eines Resource Adapters

Das Logging des Resource Adapters ist mit den folgenden Standardwerten in der Datei `BeanConnect.log4j.properties.xml` vordefiniert (siehe Abschnitt „Logging für BeanConnect Resource Adapter und Proxy konfigurieren“ auf Seite 523).

Appender

Name	Beschreibung und Empfehlung
`BeanConnectSysoutShort`	Console Appender mit dem Ziel `System.out` und kurzem Ausgabeformat, d.h. ohne Angaben zum Meldungsursprung. BeanConnect gibt auf diesen Appender Meldungen zur Konfiguration des Resource Adapters sowie Warnings und Error-Meldungen aus. **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden.
`BeanConnectShortLoggingFile`	Rolling File Appender für das Logging in eine Datei mit mehreren Backup-Dateien (siehe Abschnitt „Funktionsweise des Rolling File Appenders“ auf Seite 520) und kurzer Ausgabe. Auf diesen Appender wird die gleiche Information ausgegeben wie auf den Appender `BeanConnectSysoutShort`. Auf diesen Appender werden von BeanConnect Meldungen zur Konfiguration des Resource Adapters, sowie Warnings und Error-Meldungen ausgegeben. Dieser Appender hat drei Parameter: – `File` (Relativer) Dateiname der aktuellen Logging-Datei. Standard: `log/BeanConnect.logging.txt` – `MaxNbrBackupFiles` Maximale Anzahl der Backup-Dateien. Wenn Sie hier 0 angeben, werden keine Backup-Dateien angelegt. Standard: 10 – `MaxFileSizePerProcessKB` Die maximale Größe in KB, die die Ausgabedatei erreichen darf, bevor auf eine Backup-Datei umgeschaltet wird. Standard: 1024 **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden. Ein Anwender sollte allenfalls für den Logger `BeanConnect.ui` den Level erhöhen (d.h. von WARN auf INFO, DEBUG oder TRACE).

Name	Beschreibung und Empfehlung
`BeanConnectLoggingFil`	Rolling File Appender für das Logging in eine Datei mit mehreren Backup-Dateien (siehe Abschnitt „Funktionsweise des Rolling File Appenders“ auf Seite 520) und mehr technischen Details als bei `BeanConnectShortLoggingFile`. Dieser Appender dient der Systemdiagnose; auf diesen Appender werden von BeanConnect bei Bedarf detaillierte Diagnose-Loggings geschrieben. Dieser Appender hat drei Parameter: – `File` (Relativer) Dateiname der aktuellen Logging-Datei. Standard: `log/BeanConnect.extlogging.txt` – `MaxNbrBackupFiles` Maximale Anzahl der Backup-Dateien. Wenn Sie hier 0 angeben, werden keine Backup-Dateien angelegt. Standard: 10 – `MaxFileSizePerProcessKB` Die maximale Größe in KB, die die Ausgabedatei erreichen darf, bevor auf eine Backup-Datei umgeschaltet wird. Standard: 1024 **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden. Ein Anwender sollte von sich aus, d.h. ohne Rücksprache mit der Systemdiagnose, allenfalls für den Logger `BeanConnect.ui` den Level erhöhen (d.h. von WARN auf INFO, DEBUG oder TRACE, siehe Abschnitt „Logging-Datei von Log4j in der BeanConnect Management Console anzeigen“ auf Seite 531).

Name	Beschreibung und Empfehlung
`BeanConnectLoggingFileXML`	Rolling File Appender für das Logging in eine Datei mit mehreren Backup-Dateien (siehe Abschnitt „Funktionsweise des Rolling File Appenders“ auf Seite 520). Die Ausgabe dieses Appenders verfügt über ein XML-ähnliches Layout. Die Datei wird als Eingabedatei für die Management Console bereitgestellt (siehe auch Abschnitt „Logging-Datei von Log4j in der BeanConnect Management Console anzeigen“ auf Seite 531). Auf diesen Appender wird die gleiche Information ausgegeben wie auf den Appender `BeanConnectLoggingFile`. Dieser Appender dient der Systemdiagnose; auf diesen Appender werden von BeanConnect bei Bedarf detaillierte Diagnose-Loggings geschrieben. Dieser Appender hat drei Parameter: – `File` (Relativer) Dateiname der aktuellen Logging-Datei. Standard: `log/BeanConnect.extlogging.txt.xmc` – `MaxNbrBackupFiles` Maximale Anzahl der Backup-Dateien. Wenn Sie hier 0 angeben, werden keine Backup-Dateien angelegt. Standard: 10 – `MaxFileSizePerProcessKB` Die maximale Größe in KB, die die Ausgabedatei erreichen darf, bevor auf eine Backup-Datei umgeschaltet wird. Standard: 1024 **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden. Ein Anwender sollte von sich aus, d.h. ohne Rücksprache mit der Systemdiagnose, allenfalls für den Logger `BeanConnect.ui` den Level erhöhen (d.h. von WARN auf INFO, DEBUG oder TRACE, siehe Abschnitt „Logging von User Interface Aufrufen“ auf Seite 542).

Name	Beschreibung und Empfehlung
`BeanConnectMCSocketAppender`	Hilfs-Appender für das Logging auf die Management Console. Dieser Appender hat vier Parameter: – `RemoteHost` Rechner, auf dem die Management Console läuft. Standard: `localhost` – `Port` Listener-Port der Management Console. Standard: 31015 – `LocationInfo` Immer aktiviert (true). – `ReconnectionDelay` Positive Ganzzahl, die die Anzahl der Millisekunden angibt, die nach einem fehlgeschlagenen Verbindungsversuch zum Server vor einem erneuten Versuch gewartet werden soll. Standard: 10000 **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden.
`BeanConnectManagementConsole`	Appender für das Logging auf die Management Console. **Empfehlung:** Diesem Appender sollten Logger hinzugefügt werden, die auf der Management Console angezeigt werden sollen..

Logger

Es wird empfohlen, die Einstellungen der Logger nicht zu verändern - mit Ausnahme des Loggers `BeanConnect.ui`.

Name	Beschreibung
`BeanConnect`	Parent-Logger aller anderen BeanConnect Logger. Level: `WARN`
`BeanConnect.in`	Logger für die BeanConnect Inbound-Kommunikation.
`BeanConnect.info`	Runtime-Logger. Level: `INFO`
`BeanConnect.out`	Logger für die BeanConnect Outbound-Kommunikation.

Name	Beschreibung
BeanConnect.ui	Logger für die Aufrufe der BeanConnect Bedienoberfläche. Level: WARN In der Standardkonfiguration werden die Ausgaben des Loggers BeanConnect.ui auf die Appender BeanConnectShortLoggingFile, BeanConnectLoggingFile, BeanConnectLoggingFileXML ausgegeben.
de.siemens net.fsc	Parent-Logger aller klassenspezifischen BeanConnect Logger, die für die Ausgabe von Debug-Traces verwendet werden. Level: WARN
net.fsc.beanta.encoding	Logger für die Codierungsunterstützung von BeanConnect. Level: WARN
net.fsc.tpbasics.util.L	Logger für die Log4j-Unterstützung.

13.5.3 Logging von User Interface Aufrufen

Aufrufe einer Anwendung (EJB) an BeanConnect werden auf eigene Logger (BeanConnect.ui und dessen Child-Logger) protokolliert. Diese Ausgaben können Anwendern bei der Diagnose von Problemen in ihrer Anwendung helfen.

Ausgaben auf den Logger BeanConnect.ui erfolgen mit den Levels INFO, DEBUG und TRACE. Eine Übersicht gibt die folgende Tabelle:

Liste der BeanConnect.ui Logger, ihre Levels und ihre Bedeutung

Logger	Level	Bedeutung
BeanConnect.ui.out	INFO	Logging der Outbound Interface Aufrufe ohne Daten
BeanConnect.ui.oltpmsg	INFO	Logging der OltpMessage-Interface Aufrufe ohne Daten
BeanConnect.ui.data.api	DEBUG	Logging aller am User Interface übergebenen Daten
BeanConnect.ui.data.net	TRACE	Logging aller am User Interface übergebenen Daten, in der Form, wie sie im Netz übertragen werden, d.h. ggf. umcodiert.
BeanConnect.ui.in	INFO	Logging der an und von der MessageEndpoint-Anwendung übergebenen OltpMessage Objekte

Beispiel 26 Steuerung der Logging-Ausgaben

- Setzen von Logger `BeanConnect.ui` auf Level `INFO`:

  Es werden alle User Interface Aufrufe protokolliert ohne Daten.

- Setzen von Logger `BeanConnect.ui` auf Level `DEBUG`:

  Es werden alle User Interface Aufrufe protokolliert und zusätzlich die bei diesen Aufrufen übergebenen Daten.

- Setzen von Logger `BeanConnect.ui` auf Level `TRACE`:

  Es werden alle User Interface Aufrufe protokolliert, die bei diesen Aufrufen übergebenen Daten, sowie die Daten codierter Form.

- Setzen von Logger`BeanConnect.ui.out` auf Level `INFO` und von `BeanConnect.ui.data.net` auf Level `TRACE`:

  Es werden alle Outbound Interface Aufrufe protokolliert und zusätzlich die Daten codierter Form. Nicht protokolliert werden die Aufrufe am OltpMessage-Interface sowie die am Outbound Interface übergebenen Daten.

13.6 Diagnose des BeanConnect Proxy-Containers

Es steht eine Reihe von Informationen für die Diagnose eines BeanConnect Proxy-Containers zur Verfügung. Diese Informationen sind auf verschiedene Dateien im Home-Verzeichnis des Containers verteilt.

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Vordefinierte Logging-Konfiguration eines Proxys
- Logging-Dateien des BeanConnect Proxy-Containers
- Traces des BeanConnect Proxy-Containers

13.6.1 Vordefinierte Logging-Konfiguration eines Proxys

Das Logging der einzelnen Proxys ist nach der Installation mit folgenden Standardwerten vorkonfiguriert:

Appender

Name	Beschreibung und Empfehlung
`BeanConnectSysoutShort`	Console Appender mit dem Ziel `System.out` und kurzem Ausgabeformat, d.h. ohne Angaben zum Meldungsursprung. BeanConnect gibt auf diesen Appender Meldungen allgemeiner Art sowie Warnings und Error-Meldungen aus. **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden.
`BeanConnectSysout`	Console Appender mit dem Ziel `System.out` und ausführlicherem Ausgabeformat zur Erleichterung der Systemdiagnose. Auf diesen Appender werden von BeanConnect Fatal-Meldungen ausgegeben, für die kein anderer Appender konfiguriert ist. **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden.

Name	Beschreibung und Empfehlung
`BeanConnectLoggingFile`	Rolling File Appender für das Logging in eine Datei mit mehreren Backup-Dateien (siehe Abschnitt „Funktionsweise des Rolling File Appenders“ auf Seite 520). Dieser Appender dient der Systemdiagnose; auf diesen Appender werden von BeanConnect bei Bedarf detaillierte Diagnose-Loggings geschrieben. Dieser Appender hat drei Parameter: – `File` (Relativer) Dateiname der aktuellen Logging-Datei Standard: `logs/logging.txt` – `MaxNbrBackupFiles` Maximale Anzahl der Backup-Dateien. Wenn Sie hier 0 angeben, werden keine Backup-Dateien angelegt. Standard: 10 – `MaxFileSizePerProcessKB` Prozess-spezifischer Grenzwert für den Dateiwechsel in KB. Wenn die Dateigröße durch das Schreiben eines Logging-Ereignisses diesen prozess-spezifischen Grenzwert erreicht, wird die Datei umgeschaltet. Die Größe einer Logging-Datei beim Umschalten liegt daher zwischen den Werten <MaxFileSizePerProcessKB> und <Anzahl_Prozesse>* <MaxFileSizePerProcessKB>. Standard: 500 **Empfehlung:** Diesem Appender sollten keine weiteren Logger hinzugefügt werden.
`BeanConnectLoggingFileXML`	Rolling File Appender für das Logging in eine Datei mit mehreren Backup-Dateien (siehe Abschnitt „Funktionsweise des Rolling File Appenders“ auf Seite 520). Die Ausgabe dieses Appenders verfügt über ein XML-ähnliches Layout. Die Datei kann mit der Management Console ausgewertet werden (siehe auch Abschnitt „Logging-Datei von Log4j in der BeanConnect Management Console anzeigen“ auf Seite 531). Dieser Appender ist vorkonfiguriert, wird aber in der Standard-konfiguration von BeanConnect nicht verwendet. Er dient der Systemdiagnose; auf diesen Appender werden von BeanConnect bei Bedarf detaillierte Diagnose-Loggings geschrieben. **Empfehlung:** Bei Bedarf bzw. auf Anforderung durch den Systemdienst sollten diesem Appender die gleichen Logger zugewiesen werden wie dem Appender BeanConnectLoggingFile.

Logger

Es wird empfohlen, die Einstellungen der Logger nicht zu verändern.

Name	Beschreibung
`Root-Logger`	Level: `FATAL`.
`BeanConnect`	Parent aller anderen BeanConnect Logger. Level: `ERROR`
`BeanConnect.c`	Logger für Debug-Traces der C-Komponenten von BeanConnect. Level: `ERROR`
`BeanConnect.info`	Logger für Laufzeitdaten. Level: `INFO`
`BeanConnect.Datasources.OLTP`	Logger für OLTP-Datensourcen. Level: `ERROR`
`BeanConnect.kdcs`	Logger für Debug-Traces der KDCS-Aufrufe der Proxy-Container-Anwendung. Level: `ERROR`

`de.siemens`
und
`net.fsc`
	Parent-Logger aller klassenspezifischen BeanConnect Logger, die für die Ausgabe von Debug-Traces verwendet werden.
	Level: `ERROR`

`net.fsc.tpbasics.util.L`	Logger für die Log4j-Unterstützung. Level: `WARN`
`net.fsc.beanta.encoding.EncoderImpl`	BeanConnect Logger für die Codierungsunterstützung. Level: `WARN`

13.6.2 Logging-Dateien des BeanConnect Proxy-Containers

Der Proxy-Container basiert auf openUTM. Es gibt verschiedene Logging- und Diagnosedateien mit Daten von openUTM. Dieser Abschnitt enthält Informationen zu folgenden Themen:

- stdout-/stderr-Log
- System-Logging-Datei SYSLOG
- Dumps und Diagnose-Dumps
- Anwendungsprotokoll auf Windows-Systemen

13.6.2.1 stdout-/stderr-Log

Meldungen des Proxy-Containers an `stdout/stderr` werden in Dateien im Home-Verzeichnis des Proxy-Containers protokolliert. Die Dateien stehen standardmäßig zur Verfügung. Beim Starten des Proxy Containers werden die Dateien `utmp.err.<suffix>` und `utmp.out.<suffix>` erstmals angelegt, wobei der Suffix YY-MM-DD.HHMMSS dem Startzeitpunkt entspricht.

Protokolldateien umschalten

Zur Laufzeit ist die Proxy Container Anwendung so eingestellt, dass jeweils um Mitternacht die `stdout/stderr` Ausgaben auf neue Dateien umgeschaltet werden. Der Suffix `YY-MM-DD.HHMMSS` dieser Dateien entspricht dabei dem Umschaltzeitpunkt.

Zusätzlich können Sie die `stdout/stderr` Dateien bei Bedarf auch manuell mit der Management Console umschalten. Verwenden Sie dazu den Befehl **Switch Protocol Files** im Kontextmenü des Proxys bzw. des Proxy Clusters.

Protokolldateien anzeigen

Sie zeigen den Inhalt dieser Dateien an wie folgt:

- Klicken Sie in der Management Console unter dem Knoten des Proxy-Containers auf die Knoten **Advanced Features** - **Diagnosis** - **Output** - **General Diagnostic Info**, bei einem Proxy Cluster müssen Sie zusätzlich noch den Proxy auswählen.

  Wählen Sie den Eintrag **Container STDERR Diagnostics** oder **Container STDOUT Diagnostics**. Klicken Sie dann auf die Schaltfläche **Show File**. Daraufhin bietet Ihnen die Management Console sämtliche Dateien `utmp.out.*` oder `utmp.err.*` zur Auswahl an. Markieren Sie die gewünschten Dateien und klicken Sie auf die Schaltfläche **OK**. Die Management Console überträgt die ausgewählten Dateien in den lokalen Diagnosepfad des Proxys und zeigt sie im Arbeitsbereichsfenster **Text File** an.

Einzelheiten dazu finden Sie in der Online-Hilfe der Management Console.

- Sie können diese Logging-Dateien auch direkt im Home-Verzeichnis des Proxy-Containtes mit einem Standardeditor anzeigen und auswerten.
- Auf Windows-Systemen können Sie zusätzlich die Dateien `utmp.err*` oder `utmp.out*` mit den Befehlen
  Advanced - **Show STDERR Diagnostics** oder
  Advanced - **Show STDOUT Diagnostics** in der Programmgruppe des Proxy-Containers anzeigen.

Protokolldateien sichern

Die Dateien werden beim nächsten Start des Proxy-Containers in das Proxy-Container Verzeichnis `out-err` gesichert. Vor der Sicherung werden sämtliche Dateien im Verzeichnis `out-err` gelöscht. Wenn Sie diese Dateien zu Diagnosezwecken verwenden möchten, müssen Sie sie daher sichern, bevor Sie einen Restart des Proxys anstoßen.

Wenn der Proxy-Container auf Windows-Systemen als Dienst ausgeführt wird (siehe Abschnitt „Proxy-Container als Windows-Dienst starten“ auf Seite 274), wird seine erste Ausgabe an `stdout` in die Datei `utmp.out` und seine erste Ausgabe an `stderr` in die Datei `utmp.err` geschrieben.
Bei einem Protokolwechsel heissen die Dateien dann:
`utmp.err.<suffix>` bzw. `utmp.out.<suffix>`.

Die Protokolle des letzten Anwendungslaufs werden automatisch im Verzeichnis `out-err` gesichert.

13.6.2.2 System-Logging-Datei SYSLOG

Die System-Logging-Datei erfasst wichtige Ereignisse (in Form binärer Meldungen) vom Lauf des Proxy-Containers. Sie enthält wichtige Informationen, die für die Fehlerdiagnose verwendet werden können. Die Datei wird im Dateiverzeichnis `SYSLOG` im Home-Verzeichnis des Proxy-Containers gespeichert. Die Datei steht standardmäßig zur Verfügung.

Sie kann wie folgt angezeigt werden:

- Klicken Sie in der Management Console unter dem Knoten des Proxy-Containers auf die Knoten **Advanced Features** - **Diagnosis** - **Output** - **General Diagnostic Info** und wählen Sie den Eintrag **Container Syslog Files**. Klicken Sie dann auf die Schaltfläche **Show File**.

  Falls mehr als eine Datei des gewählten Typs vorhanden ist, wird das Dialogfeld **Select Diagnostic File To Show** geöffnet. Es werden die entsprechenden Dateien auf dem Rechner angezeigt. Wählen Sie die Diagnosedatei, die Sie anzeigen möchten.

Die Management Console überträgt die konvertierte Textdatei `syslog.<number>.txt` in den lokalen Diagnosepfad des Proxys und zeigt sie im Arbeitsbereichsfenster **Text File** an.

Einzelheiten dazu finden Sie in der Online-Hilfe der Management Console.

- Auf Solaris- und Linux-Systemen: Wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript wie folgt auf:

  ```
  shsc/syslog.sh
  ```

  Die verfügbaren Logging-Dateien werden aufgelistet und Sie werden nacheinander gefragt, ob Sie die jeweilige Datei anzeigen möchten. Wenn Sie eine Datei auswählen, werden die formatierten Daten in die Datei `slogout` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

- Auf Windows-Systemen: Wählen Sie den Befehl **Advanced** - **Show Syslog** in der Programmgruppe des Proxy-Containers.

  Es wird ein Befehlszeilenfenster angezeigt, in dem die verfügbaren Logging-Dateien aufgelistet sind. Wählen Sie die Nummer der Datei, die Sie anzeigen möchten. Die formatierten Daten werden in die Datei `slog.out` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

- Auf Windows-Systemen (cmdline): Wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript wie folgt auf:

  ```
  shsc\syslog.cmd
  ```

  Die formatierten Daten werden in die Datei `slog.out` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

13.6.2.3 Dumps und Diagnose-Dumps

Wenn der Proxy-Container abstürzt oder ein schwerwiegender Fehler bei der Ausführung des Proxy-Containers auftritt, wird ein openUTM-Dump erstellt und im Unterverzeichnis `DUMP` des Home-Verzeichnisses des Proxy-Containers gespeichert.

Diese Dumps sollten von BeanConnect-Spezialisten ausgewertet werden.

13.6.2.4 Anwendungsprotokoll auf Windows-Systemen

Wenn der Proxy-Container als Dienst gestartet wird und beim Start Probleme auftreten, kann das Anwendungsprotokoll zusätzliche Informationen liefern (siehe Abschnitt „Proxy-Container als Windows-Dienst starten“ auf Seite 274):

1. Wählen Sie
   Start - Einstellungen - Systemsteuerung - Verwaltung - Ereignisanzeige.
2. Klicken Sie in der Ereignisanzeige auf **Anwendung** und wählen Sie die Quelle **openUTM** aus.

13.6.3 Traces des BeanConnect Proxy-Containers

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- OSS-Trace
- BCAM-Trace
- CMX-Trace

13.6.3.1 OSS-Trace

Diese Funktion protokolliert Aktivitäten im Proxy-Container, die sich auf eine OSI TP Verbindung zur openUTM-Partneranwendung bzw. zum openUTM-LU62 Gateway beziehen.

Standardmäßig ist der OSS-Trace deaktiviert. Sie aktivieren den OSS-Trace eines Proxy-Containers über die Management Console:

1. Wählen Sie die folgenden Knoten im Navigationsbaum des Proxys:
   Advanced Features - **Diagnosis** - **Configuration** - **General Diagnostic Info**.

   Die Registerkarte **General** des Arbeitsbereichsfensters
   General Diagnostic Info Configuration wird angezeigt.
2. Wählen Sie die Option **Activate OSS Trace**.

Bei laufendem Proxy-Container tritt die Änderung dynamisch in Kraft, wenn Sie den Proxy speichern. Läuft der Proxy-Container nicht, tritt die Änderung in Kraft, wenn Sie den Proxy das nächste Mal starten.

Die Traces werden in einem Binärformat in Dateien mit dem Namen `OSST.*` im Home-Verzeichnis des Proxy-Containers gespeichert. So zeigen Sie die Trace-Daten an und werten sie aus:

- Klicken Sie in der Management Console im Navigationsbaum des Proxys auf die Knoten
  Advanced Features - **Diagnosis** - **Output** - **General Diagnostic Info** und wählen Sie den Eintrag **Container OSS Traces**. Klicken Sie dann auf die Schaltfläche **Show File**.

  Die Management Console überträgt die konvertierte Textdatei `osstrac.txt` in den lokalen Diagnosepfad des Proxys (Standard: `diag/<proxy_cont_name>` im Home-Verzeichnis der Management Console) und zeigt die Datei im Arbeitsbereichsfenster **Text File** an.

- Auf Solaris- und Linux-Systemen: Wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript wie folgt auf:

  ```
  shsc/ositrace.sh
  ```

  Die generierten Traces werden in die Datei `osstrac.txt` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

- Auf Windows-Systemen: Wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript wie folgt auf:

  ```
  shsc\ositrace.cmd
  ```

  Die generierten Traces werden in die Datei `osstrac.txt` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

Zur weiteren Diagnose schicken Sie den aufbereiteten Trace an den Diagnosedienst.

13.6.3.2 BCAM-Trace

Diese Funktion protokolliert alle verbindungsbezogenen Aktivitäten innerhalb des Proxy-Containers.

Standardmäßig ist der BCAM-Trace deaktiviert. So aktivieren Sie den BCAM-Trace eines Proxy-Containers über die Management Console:

1. Wählen Sie die folgenden Knoten im Navigationsbaum des Proxys:
   Advanced Features - **Diagnosis** - **Configuration** - **General Diagnostic Info**.

   Die Registerkarte **General** des Arbeitsbereichsfensters
   General Diagnostic Info Configuration wird angezeigt.

2. Wählen Sie die Option **Activate BCAM Trace**.

Bei laufendem Proxy-Container tritt die Änderung dynamisch in Kraft, wenn Sie den Proxy speichern. Läuft der Proxy nicht, tritt die Änderung in Kraft, wenn Sie den Proxy das nächste Mal starten.

Die Traces werden in einem Binärformat in Dateien mit dem Namen `KDCBTRC.*` im Home-Verzeichnis des Proxy-Containers gespeichert. So zeigen Sie die Trace-Daten an und werten sie aus:

- Klicken Sie in der Management Console auf die Knoten
  Advanced Features - **Diagnosis** - **Output** - **General Diagnostic Info** und wählen Sie den Eintrag **Container BCAM Traces**. Klicken Sie dann auf die Schaltfläche **Show File**.

  Die Management Console überträgt die konvertierte Textdatei `btrc.txt` in den lokalen Diagnosepfad des Proxys (Standard: `diag/<proxy_cont_name>` im Home-Verzeichnis der Management Console) und zeigt die Datei im Arbeitsbereichsfenster **Text File** an.

- Auf Solaris- und Linux-Systemen: Wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript wie folgt auf:

  ```
  shsc/nettrace.sh
  ```

  Die generierten Traces werden in die Datei `btrc.txt` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

- Auf Windows-Systemen: Wechseln Sie in das Home-Verzeichnis des Proxy-Containers und rufen Sie das Skript wie folgt auf:

  ```
  shsc\nettrace.cmd
  ```

  Die generierten Traces werden in die Datei `btrc.txt` im Home-Verzeichnis des Proxy-Containers geschrieben und im Standardeditor des Systems angezeigt.

Zur weiteren Diagnose schicken Sie das aufbereitete Trace an den Diagnosedienst.

13.6.3.3 CMX-Trace

Diese Funktion protokolliert Aktivitäten der Transportschicht im Proxy-Container, die sich auf Verbindungen zur openUTM-Partneranwendung bzw. zum openUTM-LU62 Gateway beziehen.

Standardmäßig ist der CMX-Trace deaktiviert. So aktivieren Sie den CMX-Trace eines Proxy-Containers über die Management Console:

1. Wählen Sie die folgenden Knoten im Navigationsbaum des Proxys:
   Advanced Features - **Diagnosis** - **Configuration** - **General Diagnostic Info**.

   Die Registerkarte **General** des Arbeitsbereichsfensters
   General Diagnostic Info Configuration wird angezeigt.

2. Wählen Sie die Option **Activate CMX Trace**.

Diese Änderung tritt in Kraft, wenn Sie den Proxy das nächste Mal starten.

Solaris- und Linux-Systeme

Die CMX-Traces des Proxy-Containers werden in einem Binärformat in Dateien mit dem Namen `CMX*` im Unterverzeichnis `cmxt` des Home-Verzeichnisses des Proxy-Containers gespeichert.

Windows-Systeme

Die CMX-Traces des Proxy-Containers werden in einem Binärformat in Dateien mit dem Namen `<number>.CMX` geschrieben. Diese Dateien werden im konfigurierten Trace-Pfad von CMX gespeichert. Den Trace-Pfad können Sie mit Hilfe des PCMX-32-Tools **Trace Control** angeben. Wählen Sie den Befehl **Options** - **Trace Path** und geben Sie den Pfad an. Um das Tool **Trace Control** zu starten, wählen Sie den Befehl **Trace Control** aus der Programmgruppe PCMX-32. Diese Programmgruppe steht nach der Installation von PCMX-32 zur Verfügung.

CMX-Traces über die BeanConnect Management Console anzeigen

Sie können die CMX-Traces über die Management Console auswerten und anzeigen.

Klicken Sie in der Management Console unter dem Knoten des Proxy-Containers auf die Knoten **Advanced Features** - **Diagnosis** - **Output** - **General Diagnostic Info** und wählen Sie den Eintrag **Container CMX Traces**. Klicken Sie dann auf die Schaltfläche **Show File**.

Die Management Console überträgt die konvertierte Textdatei mit dem Suffix `.txt` an den lokalen Diagnosepfad des Proxys (Standard: `diag/<proxy_cont_name>` im Home-Verzeichnis der Management Console) und zeigt sie im Arbeitsbereichsfenster **Text File** an.

Zur weiteren Diagnose schicken Sie den aufbereiteten Trace an den Diagnosedienst.

13.7 Diagnose der BeanConnect Management Console

Die Management Console verwendet Log4j für die Ausgabe ihrer eigenen Meldungen und Debug-Traces.

Die Traces werden in die Datei `logging.txt` im Unterverzeichnis `logs` der Management Console geschrieben. Die Dateien sind standardmäßig immer vorhanden. Ihr Inhalt kann in einem beliebigen Texteditor angezeigt werden.

Die Trace-Ausgabe wird über die Datei `log4j.properties.xml` im Unterverzeichnis `config` der Management Console gesteuert.

Für das Command Line Interface MC-CLI gibt es zwei spezifische Logger:

- `name="com.fujitsu.ts"` für die Java-Klassen des MC-CLI und
- `name="mccli"` für die Jython-Module des MC-CLI

Detaillierte Informationen zum Tracing-Mechanismus der Management Console finden Sie in der Online-Hilfe der Management Console.

13.8 Diagnose der BeanConnect Tools

MC-CmdHandler

Das Logging für den MC-CmdHandler findet standardmäßig mit Log4j statt. Der Umfang des Loggings wird über Konfigurationsdateien gesteuert. In der Installation des BeanConnect Resource Adapters sind folgende Konfigurationsdateien enthalten:

- Die Datei `mccmdhandler.log4j.properties.xml`

  Basis-Logging-Daten (Standard)

- Die Datei `mccmdhandler.log4j.properties_debug.xml`

  detaillierte Logging-Daten

13.9 Diagnose des openUTM-LU62 Gateways

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Traces und Protokolle des openUTM-LU62 Gateways
- Diagnosedaten des openUTM-LU62 Gateways

13.9.1 Traces und Protokolle des openUTM-LU62 Gateways

Das openUTM-LU62 Gateway schreibt Meldungen in Protokolldateien und stellt optional Trace-Daten für die Diagnose zur Verfügung. Es gibt zwei Arten von Traces:

- Instance-Traces
- XAP-TP-Traces

Sie können verschiedene Trace-Ebenen definieren, um den Umfang der Instance-Traces zu konfigurieren.

13.9.1.1 Traces aktivieren/deaktivieren

Standardmäßig sind die Instance-Traces und die XAP-TP Traces deaktiviert. Sie können die Traces des openUTM-LU62 Gateways über die Management Console aktivieren.

Wählen Sie das openUTM-LU62 Gateway unter dem Punkt **openUTM-LU62 Gateways** im Navigationsbaum oder **openUTM-LU62 Gateways** im Navigationsbaum des Proxys aus und wählen Sie im Kontextmenü den Befehl **Edit Properties**. Die Tabelle **Edit Properties of openUTM-LU62 Gateway** zeigt die eingestellten Werte an, die nun geändert werden können.

- Trace Level
  Gibt die Ebene der Instance-Traces an, die das openUTM-LU62 Gateway protokolliert.
- Activate XAP-TP Trace
  Aktivieren Sie diese Option, damit das openUTM-LU62 Gateway XAP-TP-Traces protokolliert. Die Funktion protokolliert die Aktivitäten der Komponenten XAP-TP-Provider und OSS in Bezug auf Verbindungen zum Proxy-Container.

Bei laufendem openUTM-LU62 Gateway treten die Tracing-Einstellungen beim Speichern des openUTM-LU62 Gateways dynamisch in Kraft.

Bei aktiviertem Tracing werden die Traces in die folgenden Dateien geschrieben:

- Auf Solaris- und Linux-Systemen:
  - Instance-Trace-Datei
    `/opt/lib/utmlu62/PROT/inlog.<lu_name>.<suff>`
  - XAP-TP-Trace-Datei
    `/opt/lib/utmlu62/PROT/xaplog.<lu_name>.<suff1>.<suff2>`
- Auf Windows-Systemen:
  - Instance-Trace-Datei
    `<gateway_home>\PROT\inlog.<lu_name>.<suff>`
  - XAP-TP-Trace-Datei
    `<gateway_home>\PROT\xaplog.<lu_name>.<suff1>.<suff2>`

Hierbei steht `<lu_name>` für den Aliasnamen einer lokalen LU, `<suff>`/`<suff1>`/`<suff2>` sind numerische Suffixe und `<gateway_home>` gibt das Verzeichnis an, in dem das openUTM-LU62 Gateway installiert ist.

13.9.1.2 Traces und Protokolle auswerten

Die Traces des openUTM-LU62 Gateways sind im Binärformat geschrieben.

Sie können sämtliche Traces und Protokolldateien des openUTM-LU62 Gateways konvertieren und über die Management Console anzeigen:

1. Wählen Sie die folgenden Knoten im Navigationsbaum des Proxys:
   Advanced Features - **Diagnosis** - **Output** - **General Diagnostic Info**.
2. Wählen Sie einen der folgenden Einträge:
   - **LU62 Gateway Instance Traces**
   - **LU62 Gateway Instance Protocol Flow**
   - **LU62 Gateway XAP-TP Traces**
   - **LU62 Gateway Protocol Files**
3. Klicken Sie auf die Schaltfläche **Show File**.

Die Management Console überträgt die konvertierte Textdatei in den lokalen Diagnosepfad und zeigt die Datei im Arbeitsbereichsfenster **Text File** an.

Bei der Konvertierung eines binären Instance-Traces werden zwei Dateien erzeugt:

- generierter Instance-Trace
- Instance-Protokollfluss

Zusätzlich zu den Trace-Dateien schreibt das openUTM-LU62 Gateway Meldungen in folgende Protokolldateien:

- Auf Solaris- und Linux-Systemen:

  `/opt/lib/utmlu62/PROT/prot.<lu_name>`

- Auf Windows-Systemen:

  `<gateway_home>\PROT\prot.<lu_name>.txt`

Hierbei steht `<lu_name>` für den Aliasnamen einer lokalen LU und `<gateway_home>` gibt das Verzeichnis an, in dem das openUTM-LU62 Gateway installiert ist.

Die Namen der Protokolldateien und der generierten Traces im lokalen Diagnosepfad lauten wie folgt:

- Instance-Traces: `inlog.*.txt`
- Instance-Protokollfluss: `inlog.*.flow.txt`
- XAP-TP-Traces: `xaplog.*.txt`
- Protokolldateien: `prot.*.txt`

Instance-Protokollfluss

In den Instance-Protokollflussdateien finden Sie eine kurze Beschreibung des Protokollflusses (LU6.2- und OSI TP Protokoll).

Das openUTM-LU62 Gateway nutzt die APPC-Schnittstelle für die Kommunikation über das LU6.2-Protokoll und die XAP-TP-Schnittstelle für die Kommunikation über das OSI TP Protokoll.

Auf der Seite des LU6.2-Protokolls werden für jede Meldung folgende Informationen im Protokollfluss angezeigt:

- Name des APPC-Aufrufs
- TP-ID, die von SNAP-IX oder vom IBM Communications Server zugewiesen wird
- Verarbeitungsrichtung
- zusätzliche Parameter

Die Verarbeitungsrichtung ist durch einen Pfeil gekennzeichnet. Ein linksgerichteter Pfeil kennzeichnet Meldungen, die vom openUTM-LU62 Gateway gesendet wurden, während ein rechtsgerichteter Pfeil die vom openUTM-LU62 Gateway empfangenen Meldungen kennzeichnet.

Beim Start des openUTM-LU62 Gateways oder bei einem Verbindungsfehler werden die Administrationsdaten zwischen dem openUTM-LU62 Gateway und dem LU6.2-Partner ausgetauscht. Hierfür wird der Transaktionscode X'06F2' verwendet. Diese Protokollflüsse sind durch einen einfachen Pfeil gekennzeichnet (-->). Ein Protokollfluss von einem Anwendungsprogramm ist mit einem Doppelpfeil markiert (==>).

Jede Meldung enthält eine Zuordnungsnummer, um die Zuordnung einer LU6.2-Conversation und einer parallelen Verbindung über XAP-TP zu erleichtern. Protokollflüsse, die keiner Conversation zugeordnet sind, erhalten die Zuordnungsnummer Null.

Zusätzlich wird jede Meldung mit der Zeit und der zugehörigen Zeilennummer in der ursprünglichen Ausgabedatei ausgegeben. Der Protokollfluss enthält keine Benutzerdaten, diese sind nur in der ursprünglichen Ausgabedatei enthalten.

13.9.2 Diagnosedaten des openUTM-LU62 Gateways

Folgende Informationen sind für die Fehlerdiagnose erforderlich:

- Der Status des openUTM-LU62 Gateways. Prüfen Sie den Status über die Management Console, indem Sie den Befehl **Check Availability** im Kontextmenü des openUTM-LU62 Gateways auswählen.

  Das openUTM-LU62 Gateway und seine Verfügbarkeit werden in einer Tabelle angezeigt. Ist das openUTM-LU62 Gateway verfügbar, können Sie per Doppelklick auf den Eintrag oder mit der Schaltfläche **Result Details** Details zu diesem Eintrag anzeigen.

  Folgende Informationen sind interessant:

  – **LLU-NAME**: Aliasname der lokalen LU, über die der EIS Partner identifiziert wird. Er besteht aus

    – dem Wert, der in der Management Console im Feld **Prefix** auf der Registerkarte **General** des Eigenschaftsfelds **Edit Properties of EIS Partner** für den EIS Partner angegebenen wurde

    – einem weiteren generierten Namensbestandteil, um die Eindeutigkeit von Namen zu erreichen.

  – **atot**: Die Anzahl der aufgebauten parallelen Verbindungen zwischen dem Proxy-Container und dem openUTM-LU62 Gateway.

  – **stot**: Die Anzahl der aufgebauten Verbindungen (Sessions) zwischen dem openUTM-LU62 Gateway und der CICS-Anwendung.

  Außerdem wird die Anzahl der Control Sessions ausgegeben. Ist die Anzahl der Control Sessions 0, so liegt ein Konfigurationsfehler vor. Hat **atot** den Wert 0, so liegt ein Konfigurationsfehler vor oder der Proxy Container ist nicht gestartet. Hat **stot** den

Wert 0, so liegt ein Konfigurationsfehler vor oder der Communication Service wurde nicht gestartet oder der EIS Partner läuft nicht. Zeigt **stot** den Wert 2 an, so handelt es sich um einen Konfigurationsfehler, der angegebene Mode ist in VTAM am z/OS nicht bekannt.

- Die Beschreibung der Fehlersituation
- Sämtliche verfügbaren Diagnosedateien:
  - Auf Solaris- und Linux-Systemen:
    - Instance-Trace-Dateien
      `/opt/lib/utmlu62/PROT/inlog.<lu_name>.<suff>`
    - XAP-TP-Trace-Dateien
      `/opt/lib/utmlu62/PROT/xaplog.<lu_name>.<suff1>.<suff2>`
    - Protokolldateien
      `/opt/lib/utmlu62/PROT/prot.<lu_name>`
  - Auf Windows-Systemen:
    - Instance-Trace-Dateien
      `<gateway_home>\PROT\inlog.<lu_name>.<suff>`
    - XAP-TP-Trace-Dateien
      `<gateway_home>\PROT\xaplog.<lu_name>.<suff1>.<suff2>`
    - Protokolldateien
      `<gateway_home>\PROT\prot.<lu_name>.txt`

Hierbei steht `<lu_name>` für den Aliasnamen einer lokalen LU, `<suff>`, `<suff1>` und `<suff2>` sind numerische Suffixe und `<gateway_home>` gibt das Verzeichnis an, in dem das openUTM-LU62 Gateway installiert ist.

Wenn Sie das openUTM-LU62 Gateway neu starten, werden folgende Diagnosedateien im Unterverzeichnis `<gateway_home>/PROT` gelöscht:

- `in.dump.<lu_name>`
- `xaplog.<lu_name>.*`
- `xap.dump.<lu_name>.*`
- `prot.<lu_name>.old`
- `prot.<lu_name>.*.old`
- `core.<lu_name>`

Daher müssen Sie die Diagnosedateien sichern, bevor Sie die das openUTM-LU62 Gateways neu starten.

Die Dateien `prot.<lu_name>` und `inlog.<lu_name>.*` werden mit dem Suffix `.old` gespeichert. Auf Windows-Systemen hat die Datei `prot.<lu_name>` das zusätzliche Suffix `.txt`.

13.10 Diagnose von SNAP-IX auf Solaris-Systemen

Für die Diagnose von SNAP-IX-Problemen stehen Logging-Dateien mit unterschiedlichen Meldungsarten und verschiedene Trace-Optionen zur Verfügung.

Meldungen in Logging-Dateien

SNAP-IX unterscheidet drei Meldungstypen in Logging-Dateien:

- Problem

  Meldungen des Typs `Problem` kennzeichnen schwerwiegende und unerwartete Ereignisse. Sie werden immer protokolliert.

- Exception

  Meldungen des Typs `Exception` kennzeichnen Ereignisse, die die Systemleistung beeinträchtigen oder zukünftig Probleme verursachen bzw. die Leistung beeinträchtigen werden.

- Audit

  Meldungen des Typs `Audit` kennzeichnen normale Ereignisse während der Ausführung von SNAP-IX.

SNAP-IX-Traces

SNAP-IX bietet verschiedene Trace-Optionen für die Diagnose von SNAP-IX-spezifischen Problemen (Line-Tracing, API-Tracing, Client-Server-Tracing, TN-Server-Tracing und Internal Tracing).

13.10.1 Diagnose mit der Management Console

Mit der Management Console können Sie das Logging und die Traces für SNAP-IX konfigurieren und die Meldungsprotokolle und Trace-Dateien anzeigen und auswerten.

Logging und Traces konfigurieren

Wählen Sie einen Communication Service unter dem Punkt **Communications Services** im Navigationsbaum oder **Communication Service** im Navigationsbaum des Proxys aus, dann im Kontextmenu des Communication Service (in diesem Fall SNAP-IX) **Edit Properties**. Die Tabelle **Edit Properties of Communication Service Instance** zeigt die eingestellten Werte an, die nun geändert werden können.

Aktivieren Sie die entsprechenden Optionen, um das Logging und die Traces zu aktivieren bzw. zu deaktivieren. Zusätzlich können Sie detaillierte Versionen des Audit-Logging (über die Option **Verbose Audits**) und des Problem- und Exception-Logging (über die Option **Verbose Errors**) einschalten.

Bei laufendem SNAP-IX treten die Änderungen in Kraft, wenn Sie den Communication Service speichern.

Logging und Traces auswerten

SNAP-IX schreibt die folgenden Dateien in das Verzeichnis `/var/opt/sna/`:

- Die Logging-Datei `sna.aud`, die Audit-Meldungen enthält.
- Die Logging-Datei `sna.err`, die Fehlermeldungen (Problem- und Exception-Logging) enthält.
- Die Trace-Dateien `sna1.trc` and `sna2.trc`, die Line-Traces in Binärform enthalten.

Zur Anzeige dieser Dateien wählen Sie die folgenden Knoten im Navigationsbaum des Proxys:
Advanced Features - **Diagnosis** - **Output** - **General Diagnostic Info**.
Wählen Sie in der Tabelle einen der Einträge **SNAP-IX Audit Log**, **SNAP-IX Error Log** oder **SNAP-IX Line Trace**. Klicken Sie dann auf die Schaltfläche **Show File**.

Die Management Console konvertiert die ausgewählte Trace-Datei und überträgt die konvertierte Textdatei oder Meldungsprotokoll-Datei in den lokalen Diagnosepfad und zeigt sie im Arbeitsbereichsfenster **Text File** an.

Detaillierte Informationen finden Sie in der Dokumentation zu SNAP-IX.

13.11 Diagnose des IBM Communications Server auf Linux-Systemen

Für die Diagnose von Problemen des IBM Communications Server stehen Logging-Dateien mit unterschiedlichen Meldungstypen und verschiedene Trace-Optionen zur Verfügung.

Meldungen in Logging-Dateien

Der IBM Communications Server unterscheidet drei Meldungstypen in Logging-Dateien:

- Problem

  Meldungen des Typs `Problem` kennzeichnen schwerwiegende und unerwartete Ereignisse. Sie werden immer protokolliert.

- Exception

  Meldungen des Typs `Exception` kennzeichnen Ereignisse, die die Systemleistung beeinträchtigen oder zukünftig Probleme verursachen bzw. die Leistung beeinträchtigen werden.

- Audit

  Meldungen des Typs `Audit` kennzeichnen normale Ereignisse während der Ausführung des IBM Communications Server.

IBM Communications Server Traces

Der IBM Communications Server bietet verschiedene Trace-Optionen für die Diagnose spezifischer Probleme des IBM Communications Server an (Line-Tracing, API-Tracing, Client-Server-Tracing, TN-Server-Tracing und Internal Tracing).

13.11.1 Diagnose mit der Management Console

Mit der Management Console können Sie das Logging und die Traces für den IBM Communications Server konfigurieren und die Meldungsprotokolle und Trace-Dateien anzeigen und auswerten.

Logging und Traces konfigurieren

Wählen Sie einen Communication Service unter dem Punkt **Communications Services** im Navigationsbaum oder **Communication Service** im Navigationsbaum des Proxys aus, dann im Kontextmenu des Communication Service (in diesem Fall IBM Communications Server for Linux) **Edit Properties**. Die Tabelle **Edit Properties of Communication Service Instance** zeigt die eingestellten Werte an, die nun geändert werden können.

Aktivieren Sie die entsprechenden Optionen, um das Logging und die Traces zu aktivieren bzw. zu deaktivieren. Zusätzlich können Sie detaillierte Versionen des Audit-Logging (über die Option **Verbose Audits**) und des Problem- und Exception-Logging (über die Option **Verbose Errors**) einschalten.

Bei laufendem IBM Communications Server treten die Änderungen in Kraft, wenn Sie den Communication Service speichern.

Logging und Traces auswerten

Der IBM Communications Server schreibt die folgenden Dateien in das Verzeichnis `/var/opt/ibm/sna/`:

- Die Logging-Datei `sna.aud`, die Audit-Meldungen enthält.
- Die Logging-Datei `sna.err`, die Fehlermeldungen (Problem- und Exception-Logging) enthält.
- Die Trace-Dateien `sna1.trc` und `sna2.trc`, die Line-Traces in Binärform enthalten.

Zur Anzeige dieser Dateien wählen Sie die folgenden Knoten im Navigationsbaum des Proxys: **Advanced Features** - **Diagnosis** - **Output** - **General Diagnostic Info**. Wählen Sie in der Tabelle einen der Einträge **Communications Server (Linux) Audit Log**, **Communications Server (Linux) Error Log** oder **Communications Server (Linux) Line Trace**. Klicken Sie dann auf die Schaltfläche **Show File**.

Die Management Console konvertiert die ausgewählte Trace-Datei und überträgt die konvertierte Textdatei oder Meldungsprotokoll-Datei in den lokalen Diagnosepfad und zeigt sie im Arbeitsbereichsfenster **Text File** an.

Detaillierte Informationen finden Sie in der Dokumentation zum IBM Communications Server.

13.12 Diagnose des IBM Communications Server auf Windows-Systemen

Auf Windows-Systemen stellt der IBM Communications Server einen **Log Viewer** und eine **Trace Facility** zur Verfügung. Die Tools bieten eine grafische Benutzeroberfläche und können über die entsprechenden Programmgruppen gestartet werden.

Detaillierte Informationen finden Sie in der Dokumentation zum IBM Communications Server.

13.12.1 Diagnose mit der Management Console

Mit der Management Console können Sie das Logging und die Traces für den IBM Communications Server konfigurieren und die Meldungsprotokolle und Trace-Dateien anzeigen und auswerten.

Logging und Traces konfigurieren

Wählen Sie einen Communication Service unter dem Punkt **Communications Services** im Navigationsbaum oder **Communication Service** im Navigationsbaum des Proxys aus, dann im Kontextmenu des Communication Service (in diesem Fall IBM Communications Server for Linux) **Edit Properties**. Die Tabelle **Edit Properties of Communication Service Instance** zeigt die eingestellten Werte an, die nun geändert werden können.

Aktivieren Sie die entsprechenden Optionen, um das Logging und die Traces zu aktivieren bzw. zu deaktivieren. Zusätzlich können Sie detaillierte Versionen des Audit-Logging (über die Option **Verbose Audits**) und des Problem- und Exception-Logging (über die Option **Verbose Errors**) einschalten.

Bei laufendem IBM Communications Server treten die Änderungen in Kraft, wenn Sie den Communication Service speichern.

Logging und Traces auswerten

Wählen Sie die folgenden Knoten im Navigationsbaum des Proxys:
Advanced Features - **Diagnosis** - **Output** - **General Diagnostic Info** und wählen Sie den Eintrag **Communications Server (Windows) Message Log** oder **Communications Server (Windows) Trace Logs**. Klicken Sie dann auf die Schaltfläche **Show File**.

Die Management Console konvertiert die ausgewählte Trace-Datei und überträgt die konvertierte Textdatei oder Meldungsprotokoll-Datei in den lokalen Diagnosepfad des Proxys im Home-Verzeichnis der Management Console und zeigt sie im Arbeitsbereichsfenster **Text File** an.

13.13 Diagnosedaten sammeln

BeanConnect ermöglicht das Abrufen sämtlicher verfügbarer Diagnosedaten im Proxy mit einem Mausklick.

Klicken Sie dazu in der Management Console auf die Knoten **Advanced Features - Diagnosis - Output - General Diagnostic Info** im Navigationsbaum des Proxys. Klicken Sie dann auf die Schaltfläche **Get All Files** oder wählen Sie den Befehl **Get All Files** aus dem Kontextmenü eines beliebigen Eintrags in der Tabelle.

Daraufhin wird das Dialogfeld **Select File** angezeigt, in dem Sie das Zielverzeichnis für die Traces und Logging-Dateien festlegen. In diesem Dialogfeld wird ein Unterverzeichnis des konfigurierten lokalen Diagnosepfads vorgeschlagen. Der Name dieses Verzeichnisses setzt sich aus dem aktuellen Datum und Zeitpunkt zusammen (`<local-diag-path>/<date-time>`).

Nach der Auswahl des Zielverzeichnisses beginnt die Management Console mit dem Sammeln aller verfügbaren Diagnosedaten des BeanConnect Proxys und der Proxy-Komponenten. Falls erforderlich werden diese Dateien aus dem Binärformat in das Textformat konvertiert und anschließend in das Zielverzeichnis kopiert. Ein Aktions-Dialogfeld informiert über den Fortschritt und das Ergebnis der Aktion.

13.14 Fehlermeldungen des BeanConnect Proxy-Containers

Dieser Abschnitt enthält Informationen zu folgenden Themen:

- Konfigurations-Fehlermeldungen
- Laufzeit-Fehlermeldungen

13.14.1 Konfigurations-Fehlermeldungen

openUTM, auf dem der BeanConnect Proxy-Container basiert, wird mit dem Konfigurations-Tool KDCDEF konfiguriert. Der Ablauf und der Input für dieses Tool werden über die Management Console gesteuert. KDCDEF läuft in der Regel ohne Benutzereingaben ab.

Wenn Sie in den Konfigurationsprozess eingreifen und beispielsweise die Input-Dateien für KDCDEF ändern, kann eine erfolgreiche Konfiguration nicht mehr garantiert werden.

Jede Ausführung von KDCDEF wird anhand von Meldungen in der Datei `kdcdef.out` im Unterverzeichnis `def` des Home-Verzeichnisses des Proxy-Containers protokolliert. Sie finden die Fehlermeldungen gegebenenfalls in dieser Datei.

Die Konfiguration war erfolgreich, wenn der Prozess mit folgender Meldung beendet wird:

- K450 KDCFILE generated; KAA size: `&KAASIZE` K

Wurden Fehler in den Input-Dateien entdeckt oder ist ein sonstiger interner Fehler aufgetreten, wird der Prozess mit einer der folgenden Fehlermeldungen beendet:

- K448 KDCFILE generated with warnings; KAA size: `&KAASIZE` K
- K449 There was at least one ERROR. Generation aborted.

Wenn eine der obigen Fehlermeldungen auftritt und Sie diese nicht auf eine manuelle Veränderung der Input-Dateien zurückführen können, informieren Sie den Systemdienst.

13.14.2 Laufzeit-Fehlermeldungen

Diese Meldungen weisen im Allgemeinen auf Probleme zwischen dem Resource Adapter und dem Proxy-Container oder zwischen dem Proxy-Container und dem EIS Partner bzw. openUTM-LU62 Gateway hin. Benutzerfehler oder interne Probleme in BeanConnect oder im openUTM-LU62 Gateway könnten ein Grund für diese Probleme sein.

13.14.2.1 Meldungstypen

Bei BeanConnect gibt es drei Meldungsgruppen:

Gruppe 1	Meldungen, die normales Verhalten protokollieren
Gruppe 2	Meldungen, die Probleme und Fehler protokollieren. Auf diese Meldungen können sie reagieren. Der Hinweis bzw. die Maßnahme sind in diesem Dokument beschrieben.
Gruppe 3	Interne Meldungen von BeanConnect.

Dieses Kapitel enthält alle Meldungen der Gruppen 2 und 3, die zur Laufzeit des Proxy-Containers angezeigt werden können, in alphabetischer Reihenfolge. Sie kann zur Laufzeit des Proxy-Containers angezeigt werden. Meldungen der Gruppe 1 sind nicht beschrieben.

Die Laufzeitmeldungen werden in den Dateien `utmp.out.<suffix>` und `utmp.err.<suffix>` aufgezeichnet, `<suffix>` bezeichnet den Datum- und Zeitstempel. Diese Dateien werden im Home-Verzeichnis des Proxy-Containers gespeichert.

Jede Meldung beginnt mit einer eigenen ID. Ein „&“-Zeichen vor dem Namen kennzeichnet ein Insert. Die Beschreibung einer Meldung gibt die Bedeutung der Inserts an, die für den Benutzer von BeanConnect erforderlich sind. Alle anderen Inserts werden vom Systemdienst für die Diagnose benötigt. Manche Inserts enthalten Informationen zu Fehlercodes bei der Dateiverarbeitung (DMS-Fehlercodes) oder zu Systemfehlercodes. Diese Inserts werden im Abschnitt „Systemfehlercodes“ auf Seite 614 beschrieben.

Bedeutung der openUTM-spezifischen Begriffe im BeanConnect Proxy-Container, die in den Meldungen verwendet werden:

DMS	Dateizugriff
UTM	openUTM-Komponente
UTM-D	Komponente für die verteilte Kommunikation
XAP-TP	Komponente für den OSI TP-Protokoll-Stack

Bitte überprüfen Sie folgende Punkte, bevor Sie sich an den Systemdienst wenden:

- Entspricht die Konfiguration des Proxy-Containers in der Management Console der Konfiguration des Resource Adapters, z.B. in den Werten für **proxyURL** oder **inboundListenerPort** (siehe Konfiguration im Resource Adapter im Abschnitt „Allgemeine Eigenschaften des Resource Adapters konfigurieren“ auf Seite 95 und Konfiguration in der Management Console im Abschnitt „BeanConnect Resource Adapter konfigurieren“ auf Seite 202)?
- Entspricht die Konfiguration des Proxy-Containers in der Management Console der Konfiguration des EIS Partners bzw. des openUTM-LU62 Gateways, z.B. in den Werten für **Host** oder **Port**?
- Wurden der Proxy-Container, alle Proxy-Komponenten und der EIS Partner gestartet und stehen sie zur Verfügung? Die Verfügbarkeit können Sie über die Management Console prüfen (siehe Abschnitt „Verfügbarkeit von BeanConnect Proxys überprüfen“ auf Seite 287).

Im Folgenden finden Sie eine Liste der Meldungen, die von BeanConnect ausgegeben werden. Den Beschreibungen wurden zusätzliche Informationen hinzugefügt, um die Maßnahmen (Reaktionen) bezüglich der Meldungen zu erläutern:

- K-Meldungen
- P-Meldungen
- U-Meldungen

K-Meldungen und U-Meldungen werden standardmäßig ausgegeben.

P-Meldungen werden nur während der Kommunikation über das OSI TP Protokoll ausgegeben, d.h. bei Kommunikation zwischen Proxy-Container und dem EIS Partner bzw. zwischen Proxy-Container und openUTM-LU62 Gateway.

Wenn eine K-Meldung ausgegeben wird, sollten Sie sich ggf. auch die entsprechenden P- oder U-Meldungen anschauen.

13.14.2.2 K-Meldungen

BCSYSEX K009 Der Transaktionscode &TAC ist ungültig(&RCDC) - Bitte Eingabe

Ungültiger Service-Name, der von einem EIS unter Verwendung einer UPIC-, Socket- oder RFC1006-Verbindung aufgerufen wurde.
Maßnahme: Die folgende Tabelle enthält die möglichen Fehlercodes, die Fehlerursache sowie mögliche Maßnahmen zur Behebung des Fehlers. Bei allen Fehlercodes, die in der Liste nicht enthalten sind, informieren Sie den Systemdienst.

Fehlercode &RCDC

&RCDC	Bedeutung
KM01	Der Service wurde nicht generiert. Maßnahme: Konfigurieren Sie einen Inbound Service, dem Sie diesen Namen zuordnen oder ändern Sie das Client-Programm.

BCSYSEX K017 Vorgang &TCVG durch openUTM beendet (&RCCC/&RCDC &RCF2A) - Bitte Eingabe

Eine Inbound Transaktion wurde zurückgesetzt.

Maßnahme: Normales Verhalten, wenn &RCCC gleich 70Z und &RCDC gleich K306 ist. Sonst informieren Sie den Systemdienst.

K036 Verbindungsaufbau: &PTRM/&PRNM/&BCAP/&LTRM &RSLT, &REA1

Die Meldung wird beim Aufbau einer Verbindung vom BeanConnect Resource Adapter zum BeanConnect Proxy-Container ausgegeben.

&RSLT	Bedeutung
Y	Verbindung aufgebaut.
N	Verbindung wurde nicht aufgebaut; die Ursache wird in &REA1 angegeben.

&REA1	Bedeutung
X' 00'	Verbindung aufgebaut.
X' 0A'	SHUTDOWN des BeanConnect Proxy-Containers. Maßnahme: BeanConnect Proxy neu starten.
X' 0C'	Verbindungsabbau in Bearbeitung. Maßnahme: Anfrage wiederholen.
X' 12'	Kein weiterer freier Eintrag im Terminal-Pool vorhanden. Maßnahme: Abhängig vom Insert &LTERM.
X' 1B'	Die IP-Adresse des EIS Partners konnte nicht ermittelt werden. Maßnahme: EIS Partner Host prüfen bzw. Netzverwaltung informieren.

&REA1	Bedeutung
X' 2E'	Die Verbindung ist noch nicht vollständig abgebaut. Maßnahme: Anfrage wiederholen.

&LTRM	Bedeutung
BCUP	Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Number of Parallel Inbound UPIC Connections**.
BCSO	Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Number of Parallel Inbound Socket Connections**.
BCAP	Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Number of Parallel Inbound RFC1006 Connections**.

K040 Die Warnungsstufe &WLEV fuer &PGPOOL wurde unterschritten

Diese Meldung wird nur ausgegeben, wenn zuvor die Meldung K041 ausgegeben wurde und bedeutet, dass die dort empfohlenen Maßnahmen zur Zeit nicht mehr notwendig sind.

K041 Die Warnungsstufe &WLEV fuer &PGPOOL wurde überschritten

Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Proxy Container Storage Area Size (Page Pool)** und arbeiten Sie die Todo-Themen ab, die Ihnen in der Todo-Themenliste der Management Console angezeigt werden.

K043 DMS Fehler &DMSE fuer Datei &FNAM aufgetreten

Im Insert &DMSE wird der DMS-Fehlercode ausgegeben.

Maßnahme: Die möglichen Fehlercodes sind im Abschnitt „Fehlercodes“ auf Seite 613 beschrieben. Oder informieren Sie den Systemdienst.

K049 Fehler = &RCCC2 bei Start der Anwendung aufgetreten

Der BeanConnect Proxy-Container gibt die Meldung K049 aus, wenn der Start eines Proxy-Container-Prozesses wegen eines Fehlers abgebrochen wurde, der Fehlercode &RCCC2 zeigt die Ursache des Fehlers an.
Maßnahme: Die folgende Tabelle enthält die möglichen Fehlercodes, die Fehlerursache sowie mögliche Maßnahmen zur Behebung des Fehlers. Bei allen Fehlercodes, die in der Liste nicht enthalten sind, informieren Sie den Systemdienst.

Fehlercode &RCCC2

Code	Fehlerursache	Maßnahme
20	Wegen Adressraummangel gelingt es dem ersten Prozess des BeanConnect Proxy-Containers nicht, den benötigten Shared Memory einzurichten.	System-Generierung prüfen.
21	Wegen Adressraummangel gelingt es dem ersten Prozess des BeanConnect Proxy-Containers nicht, den benötigten Shared Memory einzurichten oder der BeanConnect Proxy-Container wird bereits beendet.	Wie 20 oder normales Verhalten.
22	Dateizugriffs-Fehler.	Siehe DMS-Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613.
24	Dateizugriffs-Fehler.	Siehe DMS-Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613.
32	Es gelang nicht, die Prozess-Lock-Börse zu erzeugen. Dieser Fehler tritt auf, wenn zu wenig Semaphoreinträge für die BeanConnect Prozesse zur Verfügung stehen. Das kann vorkommen, wenn sich das Beenden und das Neustarten eines Prozesses überlappen.	Mit größerer Anzahl von Semaphoren neu starten (siehe Abschnitt „Anzahl der Semaphore im Proxy-Container“ auf Seite 516).
33	Der Start eines weiteren Prozesses für den BeanConnect Proxy-Container wird abgelehnt, weil der BeanConnect Proxy-Container schon beendet wird (normale Beendigung oder Abbruch).	Normales Verhalten.
35	Beim Restart eines Proxy-Container Prozesses wird festgestellt, dass der BeanConnect Proxy-Container abnormal beendet wird.	Normales Verhalten.
50	Siehe 20.	System-Generierung prüfen.
55	DMS-Fehler bei KDCA-Datei.	Siehe DMS-Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613.
56	DMS-Fehler bei Pagepool-Datei.	Siehe DMS-Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613.
57	DMS-Fehler bei Restart-Datei.	Siehe DMS-Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613.

Code	Fehlerursache	Maßnahme
58	Fehler im Zusammenhang mit der SYSLOG Datei des BeanConnect Proxy. Mögliche Ursachen: Das Verzeichnis <Proxy_home>/SYSLOG ist nicht korrekt.	Bereitgestelltes SYSLOG-Directory ggf. löschen und im Verzeichnis <Proxy_home> mit folgenden Kommandos neu erzeugen: – Solaris- und Linux-Systeme: . ./initenv.sh $UTMPATH/ex/kdcslog . 20 – Windows-Systeme: initenv.cmd %UTMPATH%\ex\kdcslog . 20
59	Fehler beim Öffnen der SYSLOG Datei.	Siehe DMS-Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613.
79	Eine Prozess des BeanConnect Proxy-Containers fordert Speicher an, bekommt ihn aber nicht.	System-Generierung prüfen.
84	Speichermangel.	System-Generierung prüfen.
85	Speichermangel.	System-Generierung prüfen.
91	Fehler beim Start des BeanConnect Proxy-Containers. Der Fehler wird in der Meldung K124 näher beschrieben.	Siehe Meldung K124 auf Seite 582.

K055 Asynchron-Vorgang &ATAC1 durch openUTM beendet; KCRCCC=&RCCC; KCRCDC=&RCDC;Benutzer=&USER; LTERM=&LTRM

Eine asynchrone Inbound Transaktion wurde zurückgesetzt und eine asynchrone Inbound-Nachricht wird nochmals ausgeliefert (falls notwendig).

Maßnahme: Normales Verhalten wenn KCRCCC gleich 000 und KCRCDC gleich 0000 oder KCRCCC gleich 70Z und KCRCDC gleich K306. Sonst informieren Sie den Systemdienst.

K060 Der Anwendungslauf wurde abgebrochen; die Ursache ist &TRMA

BeanConnect erzeugt einen Speicherauszug, wenn eine Anwendung des Proxy-Container abnormal beendet oder ein Speicherabzug angefordert wird, und zwar für jeden Work-Prozess des Proxy-Containers.

Die Spalte **Grp.** (Gruppe) in der folgenden Tabelle beschreibt, welcher Ursachengruppe der Abbruch-Fehlercode (&TRMA) angehört. Es gibt folgende Gruppen:

A	Ursache ist ein Anwenderfehler, z.B. ein Fehler beim – Konfigurieren und administrieren von Proxy-Container Anwendungen mit der Management Console. – Generieren des Systems (z.B. Aufteilung des Adressraums).
D	Der Dump wurde für Diagnosezwecke erzeugt.
F	Es handelt sich um einen Folgefehler, ein anderer Prozess hat die abnormale Beendigung des BeanConnect Proxy-Containers veranlasst.
M	Ursache ist ein Speicherengpass.
U, S, X	Ursache ist ein interner Fehler bei BeanConnect.

Bei allen Fehlercodes, die nicht in der folgenden Tabelle aufgelistet sind, informieren Sie den Systemdienst.

Code	**Grp.**	**Ursache**
ALGxxx	ASU	Speicherengpass. Maßnahme: System Kernel tunen.
ASIS99	D	Administrative abnormale Beendigung des BeanConnect Proxy-Containers.
BRSREM	F	Administrative abnormale Beendigung des BeanConnect Proxy-Containers.
CACHT1 bis CACHT6	F	Nachdem ein Prozess die abnormale Beendigung des BeanConnect Proxy-Containers eingeleitet hat, hat sich ein anderer Prozess des BeanConnect Proxy-Containers abnormal beendet (= Folge-Dump). Maßnahme: Abhängig von der Ursache der abnormalen Beendigung des BeanConnect Proxy-Containers.
DIAGDP	D	Zur Diagnose wurde administrativ ein Dump erzeugt. Der BeanConnect Proxy-Container läuft normal weiter.
ENDE14	F	Siehe CACHT1.
ENDPET	A	Der BeanConnect Proxy-Container kann nicht normal beendet werden, weil es noch verteilte Transaktionen gibt, die im Zustand PTC (prepare to commit) sind. Die Transaktion kann erst nach einem Restart des BeanConnect Proxy-Containers beendet werden. Außerdem muss eine Verbindung zu den EIS Partnern bestehen, die an den verteilten Transaktionen beteiligt sind. Maßnahme: Restart des BeanConnect Proxy-Containers.

Code	Grp.	Ursache
FMMM10	A	Eine Eingabenachricht kann nicht abgespeichert werden, da der Pagepool voll ist. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Proxy Container Storage Area Size (Page Pool)**.
IOyxxx	ASU	Ein Fehler bei der Dateibearbeitung ist aufgetreten und konnte nicht korrigiert werden, yxxx = DMS-Fehlercode, siehe Abschnitt „Fehlercodes“ auf Seite 613.
IPC035	A	Fehler beim Sperren des IPC Shared Memory Segments, Ursache kann die Ausführung des `remove` Skripts bzw. `Forced Clear` während des normalen Laufs des BeanConnect Proxy-Containers sein.
IPC037	FU	Nachdem ein Prozess die abnormale Beendigung des BeanConnect Proxy-Containers eingeleitet hat, hat sich ein anderer Prozess des BeanConnect Proxy-Containers abnormal beendet (= Folge-Dump).
IPCEND	FU	Nachdem ein Prozess die abnormale Beendigung des BeanConnect Proxy-Containers eingeleitet hat, hat sich ein anderer Prozess des BeanConnect Proxy-Containers abnormal beendet (= Folge-Dump).
IPCREM	F	BeanConnect Proxy-Container wurde mit `remove` Skript bzw. `Forced Clear` abnormal beendet.
ISLPT1 bis ISLPT4	F	Siehe CACHT1.
LATCT1	F	Siehe CACHT1.
JVMABT		Die JVM hat die abnormale Beendigung des BeanConnect Proxy-Containers eingeleitet. Siehe auch Datei `hs_err*.log` im Container Verzeichnis.
LCACT1	F	Siehe CACHT1.
LKAA04	SU	Siehe CACHT1.
LKAAT1	F	Siehe CACHT1.
LKLCT1 bis LKLCT4	F	Siehe CACHT1.
LPCMT1	F	Siehe CACHT1.
OSAFT2	F	Siehe CACHT1.
OSTM07	A	Eine Log-Record für eine verteilte Transaktion kann nicht gesichert werden, da der Pagepool voll ist. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Proxy Container Storage Area Size (Page Pool)**.

Code	Grp.	Ursache
PCMM05	AU	Der Pagepool ist voll. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Proxy Container Storage Area Size (Page Pool)**.
PEND02	A	Es kann keine weiteren Daten mehr geschrieben werden, da der Pagepool voll ist. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Proxy Container Storage Area Size (Page Pool)**.
PENDER	ADU	Nach der abnormalen Beendigung einer Transaktion wurde ein Dump erzeugt. Zuvor wurde eine K017 oder eine K055 ausgegeben. Der BeanConnect Proxy-Container läuft normal weiter.
PENDT1 PENDT2	F	Siehe CACHT1.
PUTR01	AU	Fehler beim Schreiben auf Datei. Mögliche Ursache: Plattenspeicherengpass.
PWRT03	AMU	Speicherengpass. Maßnahme: Speicherbedarf und Betriebssystem-Generierung überprüfen.
PWRT06	F	Siehe CACHT1.
RESTRT	D	Bei einem Restart des BeanConnect Proxy-Containers nach einer vorhergehenden abnormalen Beendigung wird ein Diagnose-Dump geschrieben.
SACT28	M	Speicherengpass. Maßnahme: Ändern Sie den Wert `OSI-SCRATCH-AREA` im Proxy Container und aktualisieren Sie die Konfigurationsdateien des Proxy-Containers. Sonst informieren Sie den Systemdienst. Detaillierte Informationen siehe Kapitel „Hoch-Verfügbarkeit und Skalierbarkeit“ auf Seite 507.
SHM002	A	Ein Shared Memory Key ist auf dem Rechner nicht eindeutig. Eventuell Folgefehler zu K078 OSS 03. Maßnahme: Proxy-Container Generierung prüfen: Die verwendeten Shared Memory Keys stehen in der Datei <Proxy_home>/def/input.system.txt Parameter: MAX *SHMKEY Sie sind abhängig von der generierten Portnummer. Eventuell muss der Proxy-Container mit einer anderen Portnummer generiert werden.
SLOG09	SU	Das Schreiben des Message-Puffers in die aktuelle SYSLOG-Datei ist misslungen. Maßnahme: Der DMS-Fehlercode in der vorausgegangenen K043-Meldung weist eventuell auf den Fehler hin. Sonst informieren Sie den Systemdienst.
SLOG10	SU	Der Versuch eines Folge-Workprozesses, auf die gerade aktuelle SYSLOG-Dateigeneration umzuschalten, ist fehlgeschlagen. Maßnahme: Beachten Sie eventuell vorausgegangene K043-Meldungen. Sonst informieren Sie den Systemdienst.

Code	Grp.	Ursache
SMSG03	ASU	Probleme beim Schreiben des Meldungs-Puffers in die aktuelle SYSLOG Datei. Maßnahme: Der DMS-Fehlercode in der vorausgegangenen K043-Meldung weist eventuell auf den Fehler hin. Sonst informieren Sie den Systemdienst.
STnnnn	ADSU	Fehler in der Start-Phase eines Proxy Container-Prozesses. Dabei ist nnnn die Nummer, die in der Meldung „K049 Error nnnn during application startup“ die Fehlerursache anzeigt. Bereits aktive Prozesse des BeanConnect Proxy-Containers laufen normal weiter.
WAITT1 WAITT2	F	Siehe CACHT1.
XATT02	F	Siehe CACHT1.
XFGE01	F	Siehe CACHT1.

K065 Netzmeldung: &PTRM/&PRNM/&BCAP/&LTRM &FIL1B &FIL2B

Maßnahme: Abhängig von Wert in FIL1B und FIL2B.

Die Inserts &FIL1B und &FIL2B haben folgende Bedeutung:

FIL1B	Bedeutung
X' F0' - X' FF'	Normales Verhalten.
Sonst	Siehe Maßnahme zu FIL2B.

FIL2B	Bedeutung	Maßnahme
00000008	Ungültige Parameter.	Informieren Sie den Systemdienst.
00000014	Connection letter zu lang.	Informieren Sie den Systemdienst.
00000030	Interner Fehler.	Informieren Sie den Systemdienst.
0400001C	Betriebsmittel-Engpass.	Informieren Sie den Systemdienst.
04000020	BeanConnect Proxy-Container ist nicht angemeldet.	Informieren Sie den Systemdienst.
30000020	Fehler beim Anmelden.	Informieren Sie den Systemdienst.
other	Verbindungsspezifische Ereignisse.	Normales Verhalten.

K075 Programmaustausch von Prozess &PID abgebrochen; &CTYP &PROG &PVER

Maßnahme: Informieren Sie den Systemdienst.

K078 ffffffff yyyyyyyyy

Bedeutung der Parameter:

ffffffff	Enthält eine Kurzkennung des aufgetretenen Fehlers (siehe nachfolgende Tabelle).
yyyyyyyyy	Spezifische, kontextabhängige Fehlermeldung.

Bei allen Fehlercode ffffffff, die nicht in der folgenden Tabelle aufgelisteten sind, informieren Sie den Systemdienst

ffffffff	**Fehlerursache**	**Maßnahme**
ALME	Speicherengpass beim Starten der Anwendung.	Speicherbedarf überprüfen, Betriebssystem tunen.
ATEXIT 00 - 04	Prozessbeendigung.	Informationsmeldung.
DIAG 01 - 07	Angaben zum Prozess Environment	Informationsmeldung.
ENV 00 - 04	Informationen zur Prozessumgebung.	Informationsmeldung.
IPC 02	Speicherengpass beim Einrichten des IPC Shared Memories.	System Kernel Parameter tunen.
IPC 03 - 07	Angaben zur Dimensionierung des IPC Shared Memories.	Informationsmeldung.
MEM 01	Speicherengpass beim Starten der Anwendung. Vollständiger Text: K078 MEM 01 in utmwork nn Bytes not available.	Speicherbedarf überprüfen, Betriebssystem tunen.
MSG 01 - 02	Angaben zum Prozess Environment	Informationsmeldung.
NET 01	Angaben zum Prozess Environment	Informationsmeldung.
OSS 03	Fehler beim Laden des OSS Shared Memories in den Adressraum.	Speicherbedarf prüfen, Betriebssystem tunen.
PIPE 01	Angaben zum Prozess Environment	Informationsmeldung
SEM 01	Fehler während der Erzeugung eines Semaphors. Ein Semaphor ist nicht eindeutig im Rechner.	Überprüfen Sie die Generierung des Proxy Containers. Die Semaphor-Keys sind von der generierten Port-Nummer abhängig. Die verwendeten Semaphor-Keys werden in die Datei `<Proxy_home>/def/input.system.txt` geschrieben (`MAX SEMARRAY`). Ändern Sie den ersten Wert in der Klammer oder generieren Sie den Proxy Container mit einer anderen Port-Nummer.
SIGNAL nn	Signalbehandlung im utmwork-Prozess.	Informationsmeldung.
SYSPROT 01 - 02	Angaben zum Prozess Environment	Informationsmeldung

K104 UTM-D TIMEOUT (&RCVDANNO) : &LSES, &LPAP, &AGUS;
alter Status: (&OCVST, &OTAST); Aktion: &ACTION; neuer Status: (&NCVST, &NTAST).

Ursache: Zeitüberschreitung bei der Kommunikation zwischen dem Proxy-Container und dem EIS Partner.

Maßnahme: Abhängig vom Wert bei RCVDANNO.

ACTION	**Bedeutung**
ASYNCH	Timeout bei asynchroner Kommunikation: Entweder lief der Associationbelegungs-Timer oder der Reply Timer ab.
STPROG	Timeout bei dialog-orientierter Outbound-Kommunikation ohne Einfluss auf die Transaktion.
COMMIT RESET	Timeout bei Outbound-/Inbound-Kommunikation: Die Transaktion wird vor- (COMMIT) oder zurückgesetzt (RESET).

RCVDANNO (Byte 1-2)	**Bedeutung**
X'F331	Ablauf eines Timers bei Inbound-Kommunikation nach dem Senden einer Antwort an das EIS. Maßnahme: Mit Hilfe der Management Console den Wert für **Transaction Communication Timer** im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Timer Settings** erhöhen.
X'F332'	Ablauf eines Timers bei Outbound-Kommunikation, nachdem alle an der verteilten Transaktion beteiligten EIS Partner zum Einleiten des Transaktionsendes aufgefordert wurden. Maßnahme: Mit Hilfe der Management Console den Wert für **Reply Timer** im Eigenschaftsfeld **Edit Properties of an Outbound Service** aller an der Transaktion beteiligten Outbound Services überprüfen und ggf. erhöhen.
X'F333'	Ablauf des Ready-Timers bei Inbound-Kommunikation über OSI TP. Maßnahme: Mit Hilfe der Management Console den Wert für **Prepare to Commit Timer** im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Timer Settings** erhöhen.
X'F400'	Ablauf des OSI TP Associationbelegungs-Timer für einen Dialog-Auftrag bei Outbound-Kommunikation.
X'F520'	Ablauf des OSI TP Associationbelegungs-Timer für einen Asynchron-Auftrag (interner Timer von 60 Sekunden) bei Outbound-Kommunikation.
X'F522'	Ablauf des Timers, der bei Outbound-Kommunikation den Empfang der Quittung für eine über eine OSI TP Association gesendete Asynchron-Nachricht überwacht..
X'F534'	Ablauf des Antwort-Timers bei Outbound-Kommunikation für einen Dialog-Auftrag. Maßnahme: Mit Hilfe der Management Console den Wert für **Reply Timer** im Eigenschaftsfeld **Edit Properties of an Outbound Service** des Outbound Services erhöhen..

K119 OSI-TP Fehlerinformation: &OSLPAP, &USER, &TAC, &DIA1, &DIA2, &DIA3

Die Inserts &DIA1, &DIA2, &DIA3 enthalten die Ursache für die Meldung K119.

Ursache bei openUTM-Partner: Der Dialog mit dem EIS Partner wurde beendet.

Ursache bei CICS-Partner: Der Dialog mit der Proxy-Komponente openUTM-LU62 Gateway wurde beendet.

Maßnahme: Siehe Tabelle.

&DIA1	&DIA2	&DIA3	Bedeutung
01	06	02	Keine freie Verbindung vorhanden. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of an EIS Partner** in der Registerkarte **General** die Anzahl der Verbindungen (**Connections**).
02	02	01	Outbound-Kommunikation: Der Outbound Service ist im EIS nicht bekannt. Maßnahme: Im Outbound Service spezifizierten Partner Service Namen mit dem im EIS generierten Service Namen abstimmen.
02	03	18	Outbound-Kommunikation: Der EIS Partner hat die Authentisierung mit den angegebenen Security Credentials abgelehnt. Maßnahme: Konfiguration bzw. Programmierung überprüfen.
02	03	19	Outbound-Kommunikation: Der EIS Partner hat den OSI TP Dialog abgelehnt, weil das Partner LPAP, das den Proxy-Container im EIS Partner repräsentiert, gesperrt oder im QUIET-Zustand ist.
02	03	21	Outbound-Kommunikation: Speicher-Engpass beim EIS Partner.
02	03	22	Outbound-Kommunikation: Der Outbound Service ist beim EIS Partner nicht bekannt oder gesperrt oder die Berechtigung fehlt. Mögliche Maßnahme: Im Outbound Service spezifizierten Partner Service Namen mit dem im EIS generierten Service Namen abstimmen.
02	03	23	Outbound-Kommunikation: Der Outbound Service ist beim EIS Partner bekannt, kann aber nicht aufgerufen werden.
02	03	24	Outbound-Kommunikation: Für den asynchronen Outbound Service ist beim EIS Partner der Queue Level erreicht.
02	03	25	Outbound-Kommunikation: Der dialogorientierte Outbound Service ist im EIS Partner als asynchroner Service konfiguriert.
03	03	18	Inbound-Kommunikation: Der vom EIS Partner für den Dialog verwendete User ist nicht im Proxy-Container konfiguriert oder das verwendete Passwort ist falsch. Maßnahme: Konfiguration überprüfen.

&DIA1	&DIA2	&DIA3	Bedeutung
03	03	21	Inbound-Kommunikation: Der Proxy-Container Speicherbereich (Pagepool) ist voll. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** bei **Proxy Container Storage Area Size** den Wert von **Pagepool**.
04	02	00 01	Der EIS Partner hat den Dialog beendet. Maßnahme: Siehe Meldungen im EIS Partner
09 09	03 03	21 31	Der Proxy-Container Speicherbereich (Pagepool) ist voll. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** bei **Proxy Container Storage Area Size** den Wert von **Pagepool**.
10	11 12 13 15 16	xx xx xx xx xx	Keine freie Verbindung vorhanden. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of an EIS Partner** in der Registerkarte **General** die Anzahl der Verbindungen (**Connections**).
10	17	xx	Für jede Verbindung zum EIS Partner ist ein Inbound-Dialogprozess aktiv und es trifft ein weiterer Inbound-Auftrag auf einer zwischenzeitlich abgebauten und wieder neu aufgebauten Verbindung ein. Maßnahme: Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of an Outbound Service** den **Reply Timer**. Der Wert muss größer sein als der Reply Timer im EIS Partner.
Sonstige Werte			Informieren Sie den Systemdienst.

K124 Fehler: &RCXAPTP beim Start von XAP-TP in Phase: &PHAXAPTP aufgetreten

Maßnahme:

Bei RCXAPTP = 106: mögliche Ursache: Workprozess konnte nicht gestartet werden, da der Proxy schon beendet wird.

Sonst: Informieren Sie den Systemdienst.

K128 UTM-D Auftrag abgelehnt: &CON/&PRNM/&BCAP/&LPAP &LSES &REA1 &RCDC &TAC

Bedeutung der Inserts bei openUTM:

Insert	Bedeutung
&CON	OSI-CON-Name
&PRNM	Acht Leerzeichen
&BCAP	ACCESS-POINT-Name
&LTRM	OSI-LPAP-Name

Das Insert &REA1 beinhaltet die Ursache für die Meldung K128.

&REA1	Bedeutung	Maßnahme
X' 01'	Ungültiger Service-Name wurde durch ein EIS aufgerufen, das eine OSI TP Verbindung verwendet. - Recipient-tpsu-title in TP-BEGIN-DIALOGUE-RI &RCDC enthält einen Fehlercode KCRCDC.	Bei &RCDC=KM01: Konfigurieren Sie einen Inbound Message Endpoint, dem Sie diesen Service-Namen zuordnen oder ändern Sie den OSI TP Auftraggeber. Sonst informieren Sie den Systemdienst.
X' 02'	Falsch generierter Service-Name. - Recipient-tpsu-title in TP-BEGIN-DIALOGUE-RI	Informieren Sie den Systemdienst.
X' 03'	Ein asynchroner Service soll gestartet werden, der dem Message Endpoint zugeordnete Service ist als dialogorientierter Service konfiguriert. - Empfang eines TP-END-DIALOGUE-RI Protokollelements.	Prüfen Sie, ob die OLTP Message-Driven Bean das asynchrone Interface implementiert und eventuell den Inbound Message Endpoint mit asysnchronen Service neu eintragen oder OSI TP Auftraggeber ändern.
X' 04'	Ein dialog-orientierter Service soll gestartet werden, der dem Message Endpoint zugeordnete Service ist als asynchroner Service konfiguriert.	Prüfen Sie, ob die OLTP Message-Driven Bean ein dialog-orientiertes Interface implementiert und eventuell den Inbound Message Endpoint mit dialog-orientierten Service neu eintragen oder OSI TP Auftraggeber ändern.
X' 05'	Ein asynchroner Service soll gestartet werden, aber die Message Queue des Service hat den generierten Schwellwert erreicht. - Die Verbindung wird abgebaut.	Informieren Sie den Systemdienst.

K135 UPIC-Meldung: &PTRM/&PRNM/&BCAP/&LTRM/&UPCREAS/&UPCSTAT/&UPCPROT/ &UPVENC1/&UPPENC2

Maßnahme bei UPCREAS = 0D:
Erhöhen Sie in der Management Console im Eigenschaftsfeld **Edit Properties of Proxy** in der Registerkarte **Performance Settings** die **Proxy Container Storage Area Size (Pagepool)**.

Maßnahme bei UPCREAS ≠0D : Informieren Sie den Systemdienst.

K139 Fehler beim Umschalten der SYSLOG-Datei! Es wird weiterhin die Datei &FNAM benutzt

Maßnahme: Die Grund für den Fehler beim Umschalten kann evtl. dem DMS-Fehlercode der vorangegangenen Meldung K043 entnommen werden. Sonst informieren Sie den Systemdienst.

K147 Anmeldung fuer &USRTYPE User &USER nicht erfolgreich &PTRM/&PRNM/&BCAP/ &LTRM Grund: &REA7

Ursache: Abhängig von Wert in &USER.

&USER	**Ursache**
BCURAxxx	Outbound-Kommunikation in Cluster-Konfiguration oder Multi-Resource Adapter. Bei REA7=U3: normales Verhalten Andernfalls: Probleme bei der Kommunikation zwischen dem Resource Adapter und dem Proxy Container
BCUxxxxx	Outbound-Kommunikation: Probleme bei der Kommunikation zwischen dem Resource Adapter und dem Proxy-Container.
BCADMIN	Administration: Probleme bei der Kommunikation mit dem Proxy-Container über die Management Console.
Sonst	Inbound-Kommunikation: Probleme bei der Kommunikation zwischen dem EIS Partner und dem Proxy-Container.

Maßnahme: Abhängig von Wert in &REA7.

&REA7	**Bedeutung**
U1	Inbound: Der vom EIS Partner für den Dialog verwendete USER ist nicht im Proxy-Container konfiguriert. Siehe Abschnitt „Inbound-Kommunikation konfigurieren“ auf Seite 240. Outbound: Interner Fehler. Informieren Sie den Systemdienst.
U3	Inbound: Es wurde versucht, mehrere parallele Dialoge ohne Commit-Funktionalität mit demselben USER zu starten. Outbound: Interner Fehler. Informieren Sie den Systemdienst.

&REA7	Bedeutung
U4	Inbound: Das vom EIS Partner für den Dialog verwendete Passwort ist nicht im Proxy-Container konfiguriert. Siehe Abschnitt „Inbound-Kommunikation konfigurieren“ auf Seite 240. Outbound: Interner Fehler. Informieren Sie den Systemdienst. Administration: Innerhalb der Management Console wurde für den Zugriff auf den Proxy-Container ein falsches Passwort angegeben.
U17	Der Administrator des Proxy-Containers hat bereits die Beendigung des Containers eingeleitet. Deshalb ist der Start eines Dialogs nicht mehr möglich.
Sonst	Informieren Sie den Systemdienst.

K152 Heuristikmeldung: &COND &MTYPE &OLPAP &USER &LTAC &AAIS &AAID

Maßnahme: Abhängig von Wert in &COND.

&COND	Bedeutung
MIX	Inbound: Der Application Server und der EIS Partner sind bzgl. der Transaktionssicherung nicht synchronisiert. Eine Dateninkonsistenz ist möglich. Um diese zu beheben, sind manuelle Eingriffe in eine der beteiligten Datenbanken notwendig. Outbound: Der EIS Partner hat einen Heuristik MIX erkannt.
HAZ	Der Application Server hat die Transaktion zurückgesetzt. Die Art der Transaktionsbeendigung (commit/rollback) im EIS Partner ist unbekannt. Eine Dateninkonsistenz ist möglich. Um diese zu beheben, sind manuelle Eingriffe in eine der beteiligten Datenbanken notwendig.

K160 Die &TACNTR. Transaktion des Vorgang &TCVG wurde durch &RBCAUSER zurueckgesetzt (&RCCC/&RCDC); (pid: &TASK).

Bei &TCVG = KDCGIOPU handelt es sich um einen Prozess für Outbound-Kommunikation.

Bei Inbound-Kommunikation enthält &TCVG den Service-Namen, der dem Message Endpoint über die Management Console zugeordnet wurde.

Maßnahme bei openUTM: Normales Verhalten, siehe Ablauf im Resource Adapter bzw. EIS Partner.

Maßnahme bei CICS: Normales Verhalten, siehe Ablauf im Resource Adapter bzw. in der Proxy-Komponente openUTM-LU62 Gateway.

K204 XA (&TNSPID) Precommit erfordert generelles Ruecksetzen; Ursache: &XATXT TA= &INTTAID

Maßnahme: Systemdienst informieren, falls der Wert des Inserts &XATXT ungleich einer der folgenden Werte ist:

&XATXT	**Bedeutung**
XA_RBROLLBACK	Rollback aus nicht näher spezifiziertem Grund
XA_RBCOMMFAIL	Rollback wegen eines internen Kommunikationsfehlers im Resource Manager
XA_RBDEADLOCK	Rollback wegen Deadlocks
XA_RBINTEGRITY	Rollback wegen Ressourcen-Inkonsistenz
XA_RBOTHER	Rollback aus nicht näher spezifiziertem Grund
XA_RBPROTO	Rollback wegen eines internen Protokollfehlers beim Resource Manager
XA_RBTIMEOUT	Rollback wegen Zeitüberschreitung der Transaktionsdauer
XA_RBTRANSIENT	Rollback wegen eines vorübergehenden Fehlers

K210 XA (&TNSPID) Returncode: &XATXT Open RM &TEXT32, &INSTNUM

Maßnahme: Systemdienst informieren.

K211 XA (&TNSPID) Returncode: &XATXT Close RM &TEXT32, &INSTNUM

Maßnahme: Systemdienst informieren.

K212 XA (&TNSPID) xa_start(&XAFLAG) - Returncode &XATXT TA= &INTTAID

Maßnahme: Systemdienst informieren.

K213 XA (&TNSPID) xa_end(&XAFLAG) - Returncode &XATXT TA= &INTTAID

Maßnahme: Systemdienst informieren.

K214 XA (&TNSPID) xa_commit() - Returncode &XATXT TA= &INTTAID

Ursache: normales Verhalten falls keine abnormale Beendigung des Proxy-Containers folgt.

Maßnahme: Systemdienst informieren bei abnormaler Beendigung.

K215 XA (&TNSPID) xa_rollback() - Returncode &XATXT TA= &INTTAID

Ursache: normales Verhalten falls keine abnormale Beendigung des Proxy-Containers folgt.

Maßnahme: Systemdienst informieren bei abnormaler Beendigung.

K216 XA (&TNSPID) Returncode &XATXT , Recover PTC-Liste, RM: &TEXT32,&INSTNUM

Ursache: normales Verhalten falls keine abnormale Beendigung des Proxy-Containers folgt.

Maßnahme:

- eine abnormale Beendigung während der Startphase kann durch eine fehlerhafte Verbindung zum Resource Adapter verursacht sein (z.B. falsche Portnummer oder auch ungültiges Protokoll zwischen Proxy-Container und dem Resource Adapter). Die Ursache wird in Meldungen in den Protokolldateien `utmp.err.*` und `utmp.out.*` näher beschrieben, ebenso in der Logging-Datei des Proxy-Containers. Wurde dem Resource Adapter eine falsche Portnummer zugeordnet, kann diese mit der Management Console richtig gestellt werden. Ein Neustart des Proxy-Containers kann durchgeführt werden.

  Stimmen die Versionen von BeanConnect Proxy-Container und Resource Adapter nicht überein, baut der Resource Adapter die Verbindung ab, der Proxy-Container wird als Folge davon noch in der Startphase beendet. Der Hinweis auf die fehlerhafte Versionsbezeichnung wird im Logging des Resource Adapters protokolliert.

- Wenn die Ursache für eine abnormale Beendigung während der Startphase nicht wie oben beschrieben ermittelt werden kann, dann infomieren Sie bitte den Systemdienst.

K217 XA (&TNSPID) xa_prepare() - Returncode &XATXT TA= &INTTAID

Ursache: normales Verhalten falls keine abnormale Beendigung des Proxy-Containers folgt.

Maßnahme: Systemdienst informieren bei abnormaler Beendigung.

K218 XA (&TNSPID) xa_forget() - Returncode &XATXT TA= &INTTAID

Ursache: normales Verhalten falls keine abnormale Beendigung des Proxy-Containers folgt.

Maßnahme: Systemdienst informieren bei abnormaler Beendigung.

K220 XA (&TNSPID) Fehler: xa_switch Definition fuer spezifizierten RM nicht gefunden: &TEXT32

Maßnahme: Systemdienst informieren.

K221 XA (&TNSPID) Fehler: Startparameter fuer definierten RM nicht gefunden: &TEXT32

Maßnahme: Systemdienst informieren.

K222 XA (&TNSPID) Fehler: Gebundener RM ist nicht &XASPEC kompatibel: &TEXT32

Maßnahme: Systemdienst informieren.

K223 XA (&TNSPID) Syntaxfehler in Startparameter

Maßnahme: Systemdienst informieren.

K224 XA (&TNSPID) &XACALL - Returncode &XASTAT der RM-Instanz &INSTNUM, &TEXT32 entspricht nicht der XA(CAE)-Spezifikation

Maßnahme: Systemdienst informieren.

K225 XA (&TNSPID) rekursiver Aufruf: &XADBC1 - Fehler/Signal im DB/XA-Anschluss bei &XADBC2

Maßnahme: Systemdienst informieren.

K230 XA (&TNSPID) Int. Fehler: &TEXT32

Maßnahme: Systemdienst informieren.

K231 XA (&TNSPID) Int. Fehler: PETA wird nicht unterstützt

Maßnahme: Systemdienst informieren.

K232 XA (&TNSPID) Int. Fehler: DBSTAT sekundaerer Operationscode inkonsistent

Maßnahme: Systemdienst informieren.

13.14.2.3 P-Meldungen

Alle Meldungen des OSI TP Protocol Stacks beginnen mit dem Buchstaben “P”.

Mit openUTM treten P-Meldungen zwischen dem Proxy-Container und dem EIS Partner auf.

Mit CICS können P-Meldungen im Proxy-Container oder im openUTM-LU62 Gateway auftreten. Wenn in der Erklärung der P-Meldung der Begriff „Partner-Proxy-Komponente“ steht, ist damit der Proxy-Container oder das openUTM-LU62 Gateway gemeint, abhängig davon, in welcher Komponente die Meldung aufgetreten ist.

P001 Fehler beim OSS Aufruf (&XPFUNC): &ACPNT, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Fehler bei der OSI TP Kommunikation zwischen dem Proxy-Container und dem EIS Partner oder zwischen den Proxy-Komponenten.

Falls es sich um einen vom Transportsystem gemeldeten Fehler handelt, wird zusätzlich die Meldung P012 ausgegeben.

Maßnahme: Informieren Sie den Systemdienst.

P002 Fehler beim Associationaufbau (&XPFUNC): &ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Fehler beim OSI TP Verbindungsaufbau zwischen dem Proxy-Container und dem EIS Partner oder zwischen den Proxy-Komponenten.

Falls es sich um einen vom Transportsystem gemeldeten Fehler handelt, wird zusätzlich die Meldung P012 ausgegeben

Maßnahme: Informieren Sie den Systemdienst.

P003 Association abgelehnt (a_assin()): &ACPNT, Grund: &XPRJCT, Laenge: &XPLTH

Maßnahme: Informieren Sie den Systemdienst.

P004 Association abgelehnt (a_assin()): &ACPNT, &OSLPAP, Grund: &XPRJCT

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association vom EIS Partner abgelehnt wird.

Maßnahme: Siehe Tabelle.

XPRJCT	Maßnahme
1 - 16	Informieren Sie den Systemdienst.
17	Unbekannte Partneranwendung. Maßnahme: Generierung im Proxy-Container und im EIS-System openUTM bzw. in den Proxy-Komponenten überprüfen. Eventuell ist der von der Management Console erzeugte Generierungsinput bei einem der beiden Partner oder bei den Komponenten nicht aktiviert worden.
18 - 33	Informieren Sie den Systemdienst.
34 - 35	Konfigurationsproblem. Auf beiden Seiten wurde eine unterschiedliche Anzahl von Verbindungen generiert. Maßnahme: Generierung im Proxy-Container und im EIS-System openUTM bzw. in den Proxy-Komponenten überprüfen. Eventuell ist der von der Management Console erzeugte Generierungsinput bei einem der beiden Partner oder bei den Komponenten nicht aktiviert worden.
36 - 39	Informieren Sie den Systemdienst.
40	Zeitüberschreitung beim Verbindungsaufbau. Der Verbindungsaufbau wurde begonnen, kann aber nicht in der vorgegebenen Zeit vollendet werden. Maßnahme bei openUTM-Partner: Siehe Meldungen des EIS Partners. Oder informieren Sie den Systemdienst. Maßnahme bei CICS-Partner: Siehe Meldungen des EIS Partners und der Proxy-Komponenten. Versuchen Sie herauszufinden, welche Komponente nicht antwortet. Oder informieren Sie den Systemdienst.
41 - 46	Informieren Sie den Systemdienst.

P005 Association abgelehnt (a_assin()): &ACPNT, Grund: Partner unbekannt,
N-SEL: &XPNSEL, T-SEL: &XPTSEL
S-SEL: (&XPLSSEL,&XPCSSEL,&XPHSSEL)
P-SEL: (&XPLPSEL,&XPCPSEL,&XPHPSEL)

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association von außen abgelehnt wird, weil der entfernte Partner in der lokalen Anwendung nicht bekannt ist.

Maßnahme: Generierung im Proxy-Container und im EIS-System openUTM bzw. in den Proxy-Komponenten überprüfen. Eventuell ist der von der Management Console erzeugte Generierungsinput bei einem der beiden Partner oder bei den Komponenten nicht aktiviert worden.

P006 Association abgelehnt (a_assin()): &ACPNT &OSLPAP, Grund: Falscher Application Context Name (&XP0OBID, &XP1OBID, &XP2OBID, &XP3OBID, &XP4OBID, &XP5OBID, &XP6OBID, &XP7OBID, &XP8OBID, &XP9OBID)

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association von außen abgelehnt wird. Der Application Context Name für den entfernten Partner stimmt nicht mit dem in der lokalen Anwendung für diesen Partner generierten Application Context Namen überein.

Maßnahme: Generierung im Proxy-Container und im EIS-System openUTM bzw. in den Proxy-Komponenten überprüfen. Eventuell ist der von der Management Console erzeugte Generierungsinput bei einem der beiden Partner oder bei den Komponenten nicht aktiviert worden.

P007 Fehler beim Associationaufbau (a_assrs()): &ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn die Antwort auf einen Associationsaufbau von außen einen Fehler liefert. Handelt es sich um einen vom Transportsystem gemeldeten Fehler, wird zusätzlich die Meldung P012 ausgegeben.

Maßnahme: Informieren Sie den Systemdienst.

P008 Association (&XPOSAS) aufgebaut: &ACPNT, &OSLPAP

Diese Meldung wird ausgegeben, wenn eine Association aufgebaut wurde.

Maßnahme: Normales Verhalten.

P009 Association (&XPOSAS) abgelehnt (a_asscf()): &ACPNT, &OSLPAP, Grund: &XPRJCT, Laenge: &XPLTH

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association abgelehnt wird, weil die Bestätigung des Kommunikations-Partners nicht akzeptiert werden kann.

Maßnahme: Informieren Sie den Systemdienst.

P010 Association abgelehnt (a_asscf()): &ACPNT, &OSLPAP, Grund: Partner unbekannt
N-SEL: &XPNSEL, T-SEL: &XPTSEL
S-SEL: (&XPLSSEL,&XPCSSEL,&XPHSSEL)
P-SEL: (&XPLPSEL,&XPCPSEL,&XPHPSEL)

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association abgelehnt wird, weil der entfernte Partner bei der Bestätigung des Association-Aufbaus mit einer Adresse antwortet, die der lokalen Anwendung nicht bekannt ist.

Maßnahme: Informieren Sie den Systemdienst.

P011 Association (&XPOSAS) abgelehnt (a_asscf()): &ACPNT &OSLPAP, Grund: Falscher Application Context Name (&XP0OBID, &XP1OBID, &XP2OBID,&XP3OBID, &XP4OBID, &XP5OBID, &XP6OBID, &XP7OBID, &XP8OBID, &XP9OBID)

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association abgelehnt wird, weil der entfernte Partner bei der Bestätigung des Associationsaufbaus mit einem Application Context Namen antwortet, der nicht mit dem in der lokalen Anwendung für diesen Partner konfigurierten Application Context Namen übereinstimmt.

Maßnahme: Generierung im Proxy-Container und im EIS-System openUTM bzw. in den Proxy-Komponenten überprüfen. Eventuell ist der von der Management Console erzeugte Generierungsinput bei einem der beiden Partner oder bei den Komponenten nicht aktiviert worden.

P012 CMX Diagnoseinformation: &XPCTYPE, &XPCCLS, &XPCVAL

Maßnahme:

XPCVAL > 15: Fehlercode des Transportsystems. Informieren Sie den Systemdienst.

XPCVAL <anderer Wert>: Informieren Sie den Systemdienst.

P013 Association (&XPOSAS) abgelehnt (a_asscf()): &ACPNT, &OSLPAP, Grund: &XPCRES, &XPSRC, &XPNDIA,
CCR V2 = &XP1BOOL, Version Incompatibility = &XP2BOOL,
ContWin Assignment rejected = &XP3BOOL,
Bid mandatory rejected = &XP4BOOL, No reason = &XP5BOOL

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Verbindung zum EIS-System bzw. zwischen den Proxy-Komponenten vom entfernten Partner abgelehnt wird.

Maßnahme:

– Falls XP1BOOL, XP2BOOL oder XP4BOOL den Wert TRUE haben: Informieren Sie den Systemdienst.

– Falls XP3BOOL den Wert TRUE hat: Temporärer Engpass bzgl. Anzahl der Verbindungen.

– Falls XP5BOOL den Wert TRUE hat: Maßnahme abhängig vom Wert XPNDIA (siehe Tabelle).

XPNDIA	Maßnahme
1	Informieren Sie den Systemdienst.
2 - 10	Konfigurationsproblem. Maßnahme: Generierung im Proxy-Container und im EIS-System openUTM bzw. in den Proxy-Komponenten überprüfen. Eventuell ist der von der Management Console erzeugte Generierungsinput bei einem der beiden Partner oder bei den Komponenten nicht aktiviert worden. Bei openUTM-Partner: Überprüfen Sie, ob die Angabe des Rechner-Namens in der Management Console für den EIS Partner mit dem Rechner-Namen übereinstimmt, der im Netz beim Verbindungsaufbau verwendet wird. Bei einem Fehler wird in der U315-Meldung beim Proxy-Container der richtige Name ausgegeben. Wird in der U315 bei dem Rechner-Namen der Wert „*ANY“ ausgegeben, dann müssen Sie den Rechner-Namen der openUTM-Partneranwendung in die Host-Datei des Systems aufnehmen.
11 - 24	Informieren Sie den Systemdienst.

P014 Fehler beim Associationabbau (&XPOSAS) (&XPFUNC): &ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Maßnahme: Informieren Sie den Systemdienst.

P015 Association (&XPOSAS) abgebaut (&XPFUNC): &ACPNT, &OSLPAP, &XPLNK, &XPSRC, &XPNDIA, &XPINI, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn eine OSI TP Association abgebaut wird.

Maßnahme: Siehe Tabelle.

XPINI	Bedeutung
0	Normales Verhalten.
401	Das lokale Transportsystem hat die Verbindung abgebaut. Die nachfolgende P012-Meldung enthält den detaillierten CMX-Returncode.

XPINI	Bedeutung
402	openUTM-Partner: Das Transportsystem des EIS Partners hat die Verbindung abgebaut. CICS-Partner: Das Transportsystem der Proxy-Komponente openUTM-LU62 hat die Verbindung abgebaut. `&XP1INFO` enthält den Grund des Verbindungsabbaus. Nachfolgend einige ausgewählte Werte von `&XP1INFO`: 0 (T_USER) Der Abbau erfolgte durch den Kommunikationspartner, u. U. auch durch einen Benutzerfehler auf der Partnerseite. 258 (T_RSAP_NOTATT) Der Partner hat die Verbindung abgebaut, weil der adressierte Communication Endpoint dort nicht angemeldet ist. 482 (T_RLNOCONN) Weil keine Netzverbindung verfügbar ist, konnte das lokale Transportsystem den Verbindungsaufbau nicht durchführen.
403 - 406	informieren Sie den Systemdienst.
407	Der Verursacher ist die lokale Proxy-Komponente.
408	Der EIS Partner oder die Partner Proxy-Komponente openUTM-LU62 hat den Verbindungsabbau initiiert.

P016 Association (&XPOSAS) abgebaut (a_relin()): &ACPNT, &OSLPAP, &XPLNK, &XPNDIA

Diese Meldung wird ausgegeben, wenn eine Association abgebaut wird, weil eine „release indication“ empfangen wurde.

Maßnahme: Siehe Tabelle.

XPNDIA	Bedeutung
0	NO_REASON_GIVEN
21 - 23	Die Association wird vom EIS Partner oder von der Partner Proxy-Komponente mit „release“ abgebaut.

P017 OSS Dekodierfehler: &XPDU, &XP1DIA, &XP2DIA, &XP3DIA

Maßnahme: Informieren Sie den Systemdienst.

P018 FSM Protokollfehler: &ACPNT, &OSLPAP, &XPPTYP, &XPFSMN

Maßnahme: Informieren Sie den Systemdienst.

P019 APDU enthält ungültigen Wert: &ACPNT, &OSLPAP, &XPAPDU, &XP3INFO

Maßnahme: Informieren Sie den Systemdienst.

13.14.2.4 U-Meldungen

U184 &OBJ1 DMS Fehler &DMSE fuer Datei &FNAM aufgetreten

Maßnahme: Siehe DMS Fehlercode, Abschnitt „Fehlercodes“ auf Seite 613 bzw. informieren Sie den Systemdienst.

U185 kdcdef während Anwendungslauf nicht erlaubt

Maßnahme: Anwenderfehler. Sie dürfen die Konfiguration des Proxy-Containers erst nach Beendigung des Proxy-Containers ändern.

U189 &OBJ1 (&PTRM, &PRNM): IPC Engpass &IPCOBJ &IPCREAS

Maßnahme: Die Bedeutung der Inserts &IPCOBJ und &IPCREAS finden Sie in der nachfolgenden Tabelle. Bei Inserts, deren Bedeutung in der Tabelle nicht enthalten sind, informieren Sie den Systemdienst.

&IPCOBJ	&IPCREAS	Bedeutung	Maßnahme
TSAP	tsapname	openUTM Netzprozess noch nicht gestartet. Anmelden an Transportsystem war nicht erfolgreich	Normales Verhalten bei Anwendungsstart. sonst: openUTM-Generierung bezüglich BCAMAPPL und ACCESS-POINT überprüfen.
EXTP EXTP EXTP	WORK USED NET USED EXTP USED	Prozessverwaltungstabelle vollständig belegt	UTM-Generierung bzgl. Semaphore überprüfen
NET	PROC DEAD	openUTM Netzprozess hat sich beendet.	Siehe vorangegangene Meldung U3nn.
LETT	IPC FULL	Datenbereich ist bereits zu voll um noch Daten für diese Verbindung aufzunehmen.	openUTM Generierung bzw. Umgebungsvariable `UTM_IPC_LETTER` überprüfen (siehe Hinweis).
LETT	EXTP FULL	Datenbereich für Empfangsnachrichten ist bereits vollständig belegt.	openUTM Generierung bzw. Umgebungsvariable `UTM_IPC_LETTER` überprüfen (siehe Hinweis).
LETT	USED	Datenbereich ist belegt.	openUTM Generierung bzw. Umgebungsvariable `UTM_IPC_LETTER` überprüfen (siehe Hinweis).
LETT LETT	MAX ILETT MAX OLETT	Maximaler Datenbereich pro Verbindung belegt.	openUTM Generierung bzw. Umgebungsvariable `UTM_IPC_LETTER` überprüfen (siehe Hinweis).
SEMA	USED	Maximale Anzahl der Semaphoren belegt.	openUTM Generierung bzw. Umgebungsvariable `UTM_IPC_LETTER` überprüfen (siehe Hinweis).

Die Meldungen mit `&IPCOBJ=LETT` beziehen sich auf den Datenbereich im Shared Memory, der zum Nachrichtenaustausch zwischen den Proxy-Container-Prozessen genutzt wird.

Erhöhen Sie die Werte der Umgebungsvariablen `UTM_IPC_LETTER` (Voreinstellung: 1600) und `UTM_IPC_EXTP_LETTER` (Voreinstellung: 32) in der folgenden Datei:

– Solaris- und Linux-Systeme:
`<Proxy_home>/shsc/startcontainer.sh`

– Windows-Systeme:
`<Proxy_home>\shsc\startcontainer.cmd`

Um die Werte zu übernehmen, müssen Sie einen Restart des Proxys anstoßen. Nähere Informationen finden Sie im Abschnitt „Shared Memory im Proxy-Container“ auf Seite 507.

U190 &OBJ1 SHM Fehler (Schluessel: &SHMKEY, Laenge: &SHMLTH): &UERRNO

Das Insert &UERRNO hat folgende Bedeutung:

&UERRNO	Bedeutung	Maßnahme
1	Shared Memory kann in der angeforderten Größe nicht eingerichtet werden.	Anwenderfehler. Maßnahme: Eventuell muss das System getunt werden. Hinweise hierzu entnehmen Sie bitte der Freigabemitteilung.
2 - 5	Shared Memory kann nicht eingerichtet werden.	Informieren Sie den Systemdienst.

U205 utmtimer: Fehler &UERRNO während utmtimer-Ablauf

Das Insert &UERRNO hat folgende Bedeutung:

&UERRNO	Bedeutung	Maßnahme
1 - 11 und 13 - 21	Interner Fehler.	Informieren Sie den Systemdienst.
12, 22, 32	Proxy-Container wird gerade beendet.	Normales Verhalten.
29 - 31, 40	Interner Fehler.	Informieren Sie den Systemdienst.

U206 utmtimer: Empfangene Nachricht hat falschen Typ

Maßnahme: Informieren Sie den Systemdienst.

U207 utmtimer: Vergroeserung der Timerliste von &DIA1 auf &DIA2 Elemente

Maßnahme: Informieren Sie den Systemdienst.

U223 &OBJ1 Laut internen Status läuft die UTM-Anwendung &APPL, appfile: &OBJ3

Es wurde versucht, den Proxy-Container mehrfach zu starten.

U227 &OBJ1 UTM-Anwendung &APPL beendet durch kdcrem

Maßnahme: `Remove` Skript bzw. `Forced Clear` wurde aktiviert.

U228 utmmain: Fehler beim Lesen aus der Pipe, errno: &ERRNO

Maßnahme: Informieren Sie den Systemdienst.

U229 utmmain: &OBJ1-Prozess gestorben, pid: &PID, &SIGEXIT (&STRTIME)

Maßnahme: Informieren Sie den Systemdienst.

U231 utmmain: utmwork-Prozess gestorben, pid: &PID
Unerwarteter exitcode: &EXTCODE &SIGEXIT (&STRTIME)

Maßnahme: Informieren Sie den Systemdienst.

U304 &OBJ1 (pid: &PID, &TNSNAME): &NETFCT Aufruf: Fehler &NETERR

Fehler während der Aktivierung des Communication Endpoint &NETFCT.

Maßnahme: Probleme mit der Konfiguration (siehe Meldung U305) oder informieren Sie den Systemdienst.

U305 &OBJ1 (pid: &PID): CMX-Anwendung &BCAP wurde bereits angemeldet

Fehler während der Aktivierung des Communication Endpoint &BCAP. Dieser Communication-Endpoint ist auf dem Rechner nicht eindeutig.

Maßnahme: Die Communication Endpoints sind abhängig vom Container-Name und der generierten Container-Portnummer. Die verwendeten Communication-Endpoints werden in die Datei `<Proxy_home>/def/input.system.txt` (`BCAMAPPL <communication endpoint>`) geschrieben. Sie müssen den Proxy-Container mit einem anderen Namen oder anderen Portnummer installieren.

U306 &OBJ1 (pid: &PID, &TNSNAME): Fehler &UERRNO während Prozess-Ablauf

Maßnahme: Siehe Tabelle.

Das Insert &UERRNO hat folgende Bedeutung:

&UERRNO	**Bedeutung**	**Maßnahme**
1 - 11	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.
12	Workprozess beim Empfangen nicht vorhanden.	Normales Verhalten bei Beendigung des Proxy-Containers.
13	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.

&UERRNO	Bedeutung	Maßnahme
14	Konkurrierender Verbindungsabbau.	Normales Verhalten.
19 - 21	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.
22	Work-Prozess beim Senden nicht vorhanden, da Anwendungsende.	Normales Verhalten bei Beendigung des Proxy-Containers.
23 - 24	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.
25	Verbindung wurde vom Proxy-Container abgebaut.	Normales Verhalten.
26	Beim Senden wurde Verbindungsabbau erkannt.	Normales Verhalten.
27 - 36	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.
37 - 38	Anmeldung für lokale Kommunikationsendpunkte misslungen (Fehler beim Aktivieren des Communication Endpoints)	Anwenderfehler: Ändern Sie die Konfiguration, siehe auch U304/U305.
41 - 43	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.
44	Empfänger des Verbindungs-aufbauwunsches konnte nicht ermittelt werden.	Anwenderfehler: Ändern Sie die Konfiguration.
45	Verbindung bereits abgebaut.	Normales Verhalten.
46	Ungültige Portnummer beim Verbindungsaufbauwunsch von openUTM	Anwenderfehler: Generierung falsch oder unvollständig
47 - 58	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.
82	Absender des Verbindungsaufbauwunsches konnte nicht ermittelt werden.	Anwenderfehler: Ändern Sie die Konfiguration.
83 - 283	Interner Fehler bei Proxy-Container Netzanbindung.	Informieren Sie den Systemdienst.

U307 &OBJ1 (pid: &PID, &TNSNAME): ungültiges Ereignis &EVENT

Maßnahme: Informieren Sie den Systemdienst.

U309 &OBJ1 (pid: &PID, &TNSNAME): KDCSTRMA mit Grund &TRMA aufgerufen (&STRTIME)

Maßnahme: Informieren Sie den Systemdienst.

13.15 Fehlermeldungen des openUTM-LU62 Gateways

Dieser Abschnitt beschreibt Meldungen des openUTM-LU62 Gateways:

- Fehlermeldungen des openUTM-LU62 Gateways beim Start
- Fehlermeldungen des openUTM-LU62 Gateways zur Laufzeit
- Fehlermeldungen des openUTM-LU62 Gateways bei Statusabfrage
- Fehlermeldungen des openUTM-LU62 Gateways beim Administrieren
- Fehlermeldungen des openUTM-LU62 Gateways beim Konfigurieren

13.15.1 Fehlermeldungen des openUTM-LU62 Gateways beim Start

Beim Start des openUTM-LU62 Gateways gibt das openUTM-LU62-Dienstprogramm `u62_start` Meldungen nach `stdout` aus.

Alle Meldungen beginnen mit der Zeichenfolge `u62_start <nn>`, hierbei bezeichnet `<nn>` die Meldungsnummer.

- Die Meldungen mit den folgenden Nummern sind normales Verhalten:
  `17`, `18`, `24`, `25`, `26`, `28`, `29`
- Zu den Meldungen mit den Nummern `05`, `06` und `07` siehe unten.
- Bei allen anderen `u62_start <nn>` Meldungen, die nicht aufgelistet sind, informieren Sie den Systemdienst.

05 Das Directory &DIRNAME kann nicht gelesen werden, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

06 Die Konfigurationsdatei &FILENAME kann nicht geöffnet werden, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

07 Fehler beim Einlesen der Konfigurationsdatei &FILENAME, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

13.15.2 Fehlermeldungen des openUTM-LU62 Gateways zur Laufzeit

Alle Meldungen des OSI TP Protocol Stacks bei Verbindungen zwischen dem Proxy-Container und dem openUTM-LU62 Gateway beginnen mit dem Buchstaben “P”. Diese Meldungen sind im Abschnitt „P-Meldungen“ auf Seite 589 beschrieben.

Alle anderen Meldungen des openUTM-LU62 Gateways beginnen mit der Zeichenfolge `u62_tp[<komp>]<nnn>`.

Hierbei bezeichnet `<komp>` die Teil-Komponente von openUTM-LU62 Gateway und `<nnn>` die Meldungsnummer.

004 Fehler bei Signalbehandlung, errno &ERRNO (&ERRTEXT)

Maßnahme: Informieren Sie den Systemdienst.

006 Die Konfigurationsdatei &FILENAME kann nicht geöffnet werden, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

010 Die Instanz fuer den lokalen LU-Namen &LUNAME läuft bereits.

Maßnahme: Informieren Sie den Systemdienst.

012 Interner Fehler aufgetreten

Maßnahme: Informieren Sie den Systemdienst.

015 Fehler beim Erzeugen der PID-Datei &PIDFILE, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

016 Fehler beim Schreiben in die PID-Datei &PIDFILE, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

026 Abbruch von openUTM-LU62 im Modul &MODULNAME

Maßnahme: Informieren Sie den Systemdienst. Sichern Sie den Inhalt des folgenden Verzeichnisses:

- Solaris- und Linux-Systeme: `/opt/lib/utmlu62/PROT`
- Windows-Systeme: `<gateway_home>\utmlu62\PROT`

Wenn möglich, sollten Sie den Fehlerfall mit eingeschaltetem Instance-Trace nachstellen.

027 Abbruch durch den XAP-TP-Provider: &REASON

Maßnahme: Informieren Sie den Systemdienst. Sichern Sie den Inhalt des folgenden Verzeichnisses:

- Solaris- und Linux-Systeme: `/opt/lib/utmlu62/PROT`
- Windows-Systeme: `<gateway_home>\utmlu62\PROT`

Wenn möglich, sollten Sie den Fehlerfall mit eingeschaltetem XAP-TP-Trace nachstellen.

036 Fehler beim Anlegen eines Shared Memory der Laenge &LEN, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

040 Fehler in der Konfiguration der lokalen LU &LLUNAME oder der Partner-LU &RLUNAME

Maßnahme: Generierungsfehler in einer der folgenden BeanConnect Proxy-Komponenten:

- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

048 Der Knoten ist deaktiviert: Kein Conversationaufbau möglich.

Der Knoten des SNAP-IX oder IBM Communications Server ist vermutlich durch Eingriff des Administrators gestoppt worden. Conversations zum EIS Partner können jetzt nicht mehr aufgebaut werden.

Maßnahme: Überprüfen Sie die folgende BeanConnect Proxy-Komponente:

- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

302 Conversation wird abgelehnt, TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber: Restart der Kontrollinstanz ist noch nicht abgeschlossen

Dieser Fall kann auftreten, wenn nach einem Absturz die Verbindung zum EIS Partner wieder aufgebaut wurde, die Verbindung zwischen openUTM-LU62 Gateway und dem Proxy-Container aber noch nicht wiederhergestellt wurde.
Maßnahme: Auftrag wiederholen.

303 Ankommende Conversation (TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber) liefert Initialisierungsdaten, die jedoch von openUTM-LU62 verworfen werden.

Maßnahme: Informieren Sie den Systemdienst.

304 Conversation wird abgelehnt, TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber: Security-Daten werden vom OSI-TP-Partner nicht unterstützt oder enthalten ungültige Zeichen.

Maßnahme: Informieren Sie den Systemdienst.

305 Conversation wird abgelehnt, TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber: Der Alias-Name &ALIAS der Partner-LU stimmt nicht mit dem konfigurierten Namen &CONFALIAS ueberein.

Maßnahme: Überprüfen Sie die Generierung des openUTM-LU62 Gateway.

306 Conversation wird abgelehnt, TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber: Keine Unterstützung von Service TPs auf mapped (!) Conversations!

Der LU6.2-Partner hat versucht, eine Conversation mit einem nicht abdruckbaren TP-Namen (d.h. Transaktionscode zwischen X'00' und X'3F') zu starten. openUTM-LU62 Gateway unterstützt außer dem Resync-TP X'06F2' keine derartigen so genannten Service-TPs. Diese Meldung erscheint beispielsweise bei der Verwendung der Funktion EXEC CICS START, die im openUTM-LU62 Gateway das Service-TP X'02' anspricht.

307 Conversation wird abgelehnt, TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber: Konfigurationsfehler: Der Netzname der lokalen und/oder der entfernten LU ist umkonfiguriert worden!

Maßnahme: Überprüfen Sie die Generierung von openUTM-LU62 Gateway.

308 Conversation wird abgelehnt, TP-Name &TPNAME, LU6.2 -Seite ist Auftraggeber: Fehler beim Austausch der Log-Namen zwischen der lokalen und der entfernten LU!

Der LU6.2-Partner hat zum openUTM-LU62 Gateway eine Conversation mit Sync-Level 2 aufgebaut, obwohl zu diesem Zeitpunkt die Log-Namen zwischen den beiden LUs noch nicht ausgetauscht waren oder es beim letzten Austausch der Log-Namen zu einem schwerwiegenden Fehler gekommen ist. Dieser Protokoll-Verstoß wird vom openUTM-LU62 Gateway mit dem sofortigen Abbau der Conversation geahndet.
Maßnahme: Der Administrator der transaktionalen Ressourcen muss überprüfen, welche Aktionen zurückgesetzt werden müssen. Der Systemdienst kann dies nicht überprüfen.

309 RECEIVE_ALLOCATE wegen Konfigurationsfehler zurückgewiesen:
Der lokale LU-Alias-Name &LUNAME ist nicht konfiguriert!

Maßnahme: Überprüfen Sie die Generierung von

- openUTM-LU62 Gateway
- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

310 RECEIVE_ALLOCATE fehlgeschlagen: LU6.2-Basis-Software läuft nicht!

Maßnahme: Starten Sie die folgende Proxy-Komponente:

- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

311 RECEIVE_ALLOCATE fehlgeschlagen: Systemfehler (errno &ERRNO) aufgetreten: &ERRTEXT

Fehler

- Solaris-Systeme: in SNAP-IX
- Linux- und Windows-Systeme: in dem IBM Communications Server

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

312 Aufbau einer Conversation zum LU6.2-Partner zurueckgewiesen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber.

Maßnahme: Überprüfen Sie die Generierung von

- openUTM-LU62 Gateway
- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

313 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: LU6.2-Basis-Software läuft nicht!

Bedeutung: Die Proxy-Komponente ist nicht gestartet.

Maßnahme: Starten Sie die folgende Proxy-Komponente:

- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

314 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Systemfehler (errno &ERRNO) aufgetreten: &ERRTEXT

Fehler

- Solaris-Systeme: in SNAP-IX
- Linux- und Windows-Systeme: in dem IBM Communications Server

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

315 Bereits aufgebaute Conversation zum LU6.2-Partner (TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber) wegen Konfigurationsfehler wieder abgebaut:
Der Netzname der lokalen und/oder entfernten LU ist umkonfiguriert worden!

Maßnahme: Überprüfen Sie die Generierung von

- openUTM-LU62 Gateway
- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

320 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Allokationsfehler (code = &ERRTEXT)

Im Moment lässt sich keine Conversation zum TP &TPNAME des LU6.2-Partners aktiv aufbauen. (&TPNAME ist in diesem Fall z. B. der CICS-Transaktionscode.) Der Fehlercode &ERRTXT enthält dann im Klartext den Returncode des entsprechenden LU6.2-Aufrufs.

Maßnahme: Starten Sie die CICS-Anwendung.

321 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Inkonsistenz in den Konfigurationen von openUTM-LU62 und der LU6.2-Basis-Software

Maßnahme: Überprüfen Sie die Generierung von

- openUTM-LU62 Gateway
- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

322 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Security-Daten ungültig

Maßnahme: Informieren Sie den Systemdienst.

332 Conversation zum LU6.2-Partner (TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber) durch Ausfall der LU6.2-Basis-Software abgebrochen!

Maßnahme: Überprüfen Sie die Diagnose-Informationen und starten Sie neu:

- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

333 Conversation zum LU6.2-Partner (TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber) durch Ausfall der LU6.2-Basis-Software abgebrochen!

Maßnahme: Überprüfen Sie die Diagnose-Informationen und starten Sie neu:

- Solaris-Systeme: SNAP-IX
- Linux- und Windows-Systeme: IBM Communications Server

334 Conversation zum LU6.2-Partner (TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber) wegen Conversations-Fehler abgebrochen!

Eventuell verursacht durch administrativen Abbau aller Sessions.
Maßnahme: Informieren Sie den Administrator oder den Systemdienst.

335 Conversation zum LU6.2-Partner (TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber) wegen Conversations-Fehler abgebrochen!

Eventuell verursacht durch administrativen Abbau aller Sessions.
Maßnahme: Informieren Sie den Administrator oder den Systemdienst.

336 Auf Conversation zum LU6.2-Partner (TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber) User Control Daten empfangen => Abbruch der Conversation durch openUTM-LU62!

Maßnahme: Informieren Sie den Systemdienst.

337 Auf Conversation zum LU6.2-Partner (TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber) User Control Daten empfangen => Abbruch der Conversation durch openUTM-LU62!

Maßnahme: Informieren Sie den Systemdienst.

338 Auf Conversation zum LU6.2-Partner (TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber) wurde der Returncode AP_STATE_CHECK gemeldet => Abbruch der Conversation durch openUTM-LU62!

Maßnahme: Informieren Sie den Systemdienst.

339 Auf Conversation zum LU6.2-Partner (TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber) wurde der Returncode AP_STATE_CHECK gemeldet => Abbruch der Conversation durch openUTM-LU62!

Maßnahme: Informieren Sie den Systemdienst.

340 Ankommende Conversation zum LU6.2-Partner abgelehnt, TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber: Ablauf des Timers zur Überwachung des Assoziationsaufbaus

Diese maximale Wartezeit ist in openUTM-LU62 Gateway mit dem Parameter `ALLOC-TIME` konfigurierbar, der Standardwert beträgt 30 Sekunden. Erscheint diese Meldung häufiger, so sollte entweder das Sessionlimit erhöht oder die Anzahl der parallelen Verbindungen von openUTM-LU62 Gateway verringert werden. Ansonsten informieren Sie den Systemdienst.

341 Ankommende Conversation zum LU6.2-Partner abgelehnt, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Ablauf des Timers zur Überwachung des Assoziationsaufbaus

Diese maximale Wartezeit ist in openUTM-LU62 Gateway mit dem Parameter `ALLOC-TIME` konfigurierbar, der Standardwert beträgt 30 Sekunden. Erscheint diese Meldung häufiger, so sollte entweder das Sessionlimit erhöht oder die Anzahl der parallelen Verbindungen von openUTM-LU62 Gateway verringert werden. Ansonsten informieren Sie den Systemdienst.

350 XAP-TP-Provider lehnt Assoziationswunsch ab (TP-Name &TPNAME): result =&RES, source = &SRC, reason = &RSN

Proxy-Container lehnt Association ab.
Maßnahme: Siehe Meldungen im Proxy-Container.

351 XAP-TP-Provider meldet Verlust der Assoziation (TP-Name &TPNAME): source =&SRC, reason = &RSN, event = &EVT

Proxy-Container lehnt Association ab.
Maßnahme: Siehe Meldungen im Proxy-Container.

352 OSI-TP-Partner (XAP-TP User) hat Dialogwunsch abgelehnt: TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

Proxy-Container lehnt Dialogwunsch ab.
Maßnahme: Siehe korrespondierende Meldung im Proxy-Container.

353 OSI-TP-Partner (XAP-TP User) hat Dialogwunsch abgelehnt: TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Reason = &RSN

Proxy-Container lehnt Dialogwunsch ab.
Maßnahme: Siehe korrespondierende Meldung im Proxy-Container bzw. informieren Sie den Systemdienst.

354 Diagnose-Information in den Initialisierungsdaten von AP_TP_BEGIN_DIALOGUE_CNF: 0xhhhh (d,d)

wobei `hhhh` Hexadezimal-Werte und `d,d` die entsprechenden Dezimalwerte darstellen.

Bei openUTM als OSI TP Partner enthalten die Initialisierungsdaten genauere Informationen über den Grund der Ablehnung des Dialogs. Der erste Wert zeigt an, ob es sich um einen transienten (1) oder permanenten (2) Fehler handelt.

Der zweite Wert gibt den genauen Ablehnungsgrund an.

Die Bedeutung dieser Werte siehe Beschreibung DIA3 in Meldung K119 bei DIA1=2 im Abschnitt „K-Meldungen“ auf Seite 571.

361 Ankommender Dialogwunsch abgelehnt, OSI-TP-Seite ist Auftraggeber:
Lokaler TPSU Title kann nicht dekodiert werden

Maßnahme: Informieren Sie den Systemdienst.

373 Ankommender Dialogwunsch abgelehnt, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Restart der Kontrollinstanz ist noch nicht abgeschlossen

Maßnahme: Auftrag wiederholen.

374 Ankommender Dialogwunsch abgelehnt, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Der entfernte Application Process Title stimmt nicht mit der Konfiguration ueberein

Maßnahme: Überprüfen Sie die Generierung von Proxy-Container und openUTM-LU62 Gateway.

375 Ankommender Dialogwunsch abgelehnt, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber: Der entfernte Application Entity Qualifier stimmt nicht mit der Konfiguration ueberein

Maßnahme: Überprüfen Sie die Generierung von Proxy-Container und openUTM-LU62 Gateway.

390 OSI-TP-Dialog durch Partner (XAP-TP Provider) abgebrochen, TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

Maßnahme: Siehe entsprechende Meldung im Proxy-Container bzw. informieren Sie den Systemdienst.

391 OSI-TP-Dialog durch Partner (XAP-TP Provider) abgebrochen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber

Maßnahme: Siehe entsprechende Meldung im Proxy-Container bzw. informieren Sie den Systemdienst.

392 OSI-TP-Dialog durch Partner (XAP-TP User) abgebrochen, TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

Maßnahme: Siehe entsprechende Meldung im Proxy-Container bzw. informieren Sie den Systemdienst.

393 OSI-TP-Dialog durch Partner (XAP-TP User) abgebrochen, TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber

Maßnahme: Siehe entsprechende Meldung im Proxy-Container bzw. informieren Sie den Systemdienst.

408 OSI-TP-Partner meldet heuristischen Mix. TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber

Die Transaktion zwischen Proxy-Container und openUTM-LU62 Gateway ist vermutlich inkonsistent.
Maßnahme: Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbank-Sätze wieder zu koordinieren.

409 OSI-TP-Partner meldet heuristischen Hazard. TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber

Die Transaktion zwischen Proxy-Container und openUTM-LU62 Gateway ist vermutlich inkonsistent.
Maßnahme: Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbank-Sätze wieder zu koordinieren. Informieren Sie den Systemdienst.

410 Syncpoint-Request vom LU6.2-Auftragnehmer mit TP-Name &TPNAME erhalten. Transaktion wird abgebrochen.

Ein Auftragnehmerprogramm auf der CICS-Seite hat einen Syncpoint angefordert (z.B. `EXEC CICS SYNCPOINT`), ohne vom Proxy-Container aufgefordert zu sein. Das ist verboten, daher wird die Transaktion abgebrochen. &TPNAME gibt den Transaktionscode auf der CICS-Seite an.
Maßnahme: Ändern Sie das CICS-Anwendungsprogramm.

420 Fehler beim Resynchronisieren einer Transaktion. LU6.2 Partner akzeptiert den Zustand &STATE nicht.

Eine Transaktion wurde wegen eines Verbindungsverlustes auf der LU6.2-Seite in der Commit-Phase unterbrochen. Beim Resynchronisieren mit dem LU6.2-Partner trat ein Fehler auf. Die Transaktion kann weder beendet noch zurückgesetzt werden.
Maßnahme: Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbank-Sätze wieder zu koordinieren. Der Administrator muss die Transaktion manuell beenden.

421 Log-Name Fehler beim Wiederanlaufen einer Transaktion.

Eine Transaktion wurde wegen eines Verbindungsverlustes auf der LU6.2-Seite in der Commit-Phase unterbrochen. Beim Resynchronisieren mit dem LU6.2-Partner trat ein Fehler auf. Die Transaktion kann weder beendet noch zurückgesetzt werden.
Maßnahme: Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbank-Sätze wieder zu koordinieren. Der Administrator muss die Transaktion manuell beenden.

423 Protokollfehler des LU6.2-Partners: Compare States &LUWSTATE &RRI im Zustand &STATE1/&STATE2 erhalten.

Fehlverhalten des EIS Partners.
Maßnahme: Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbank-Sätze wieder zu koordinieren. Informieren Sie den Administrator des EIS Partners.

424 LU6.2-Partner meldet Protokollfehler. Zustand: &STATE1/&STATE2, TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.

Maßnahme: Informieren Sie den Systemdienst.

425 Fehlerhafter Log-Record (Fehlertyp &ERROR). Transaktion wurde gelöscht.

Beim Warmstart von openUTM-LU62 Gateway wurde ein fehlerhafter Log-Record eingelesen. Die in diesem Log-Record beschriebene Transaktion kann deshalb nicht resynchronisiert werden.
Maßnahme: Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbank-Sätze wieder zu koordinieren. Der Administrator muss die Transaktion manuell beenden.

426 Problem an der XAP-TP-Schnittstelle. &EVENT erhalten.

Maßnahme: Informieren Sie den Systemdienst.

427 Fehler &ERRNO beim Aufruf von &CALL.

Bei einem Solaris- oder Linux-Systemaufruf ist ein Fehler aufgetreten. Dies kann zu einer abnormalen Beendigung von openUTM-LU62 Gateway führen.
Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

510 Die LU (Alias-Name = &ALIASNAME, Netzname = &NETNAME) hat eine negative Antwort auf eine Exchange Logname Nachricht gesendet.

Maßnahme: Informieren Sie den Systemdienst.

511 Die LU (Alias-Name = &ALIASNAME, Netzname = &NETNAME) hat einen Kaltstart versucht, die lokale LU hat jedoch noch LUWs (logical units of work) von einer früheren Aktivierung zu resynchronisieren.

Maßnahme: Informieren Sie den Systemdienst.

13.15.3 Fehlermeldungen des openUTM-LU62 Gateways bei Statusabfrage

Wenn der Status des openUTM-LU62 Gateways in der Management Console abgefragt wird (Check Availability), dann gibt das openUTM-LU62-Dienstprogramm `u62_sta` Meldungen nach `stdout` aus.

Alle Meldungen beginnen mit der Zeichenfolge `u62_sta <nn>`, hierbei bezeichnet `<nn>` die Meldungsnummer.

- Zu den Meldungen mit den Nummern `06`, `09`, `12` und `13` siehe unten.
- Bei allen anderen `u62_sta <nn>` Meldungen, die nicht aufgelistet sind, informieren Sie den Systemdienst.

06 Instanz &INST: Instanz befindet sich noch in der Initialisierungsphase oder bereits in der Endebehandlung.

Die Proxy-Komponente openUTM-LU62 Gateway wird gerade gestartet oder beendet.
Maßnahme: Wiederholen Sie die Statusabfrage.

09 Instanz &INST: Instanz reagiert nicht in angemessener Zeit.

Verzögerung, die bei höherer Last und großen Konfigurationen auftreten kann.
Maßnahme: Starten Sie die Überprüfung noch einmal. Ansonsten erhöhen Sie die eingestellte Wartezeit von 20 Sekunden im folgenden Skript:

- Solaris- und Linux-Systeme: `<Proxy_home>/shsc/checkgateway.sh`
- Windows-Systeme: `<Proxy_home>\shsc\checkgateway.cmd`

12 Keine openUTM-LU62-Instanz aktiv.

Die Proxy-Komponente openUTM-LU62 Gateway ist noch nicht gestartet.
Maßnahme: Starten Sie openUTM-LU62 Gateway.

13 Die angegebene Instanz ist nicht aktiv.

Die Proxy-Komponente openUTM-LU62 Gateway ist noch nicht gestartet.
Maßnahme: Starten Sie openUTM-LU62 Gateway.

13.15.4 Fehlermeldungen des openUTM-LU62 Gateways beim Administrieren

Beim Administrieren des openUTM-LU62 Gateways durch die Management Console (Beenden, Aktivieren von Traces) gibt das openUTM-LU62-Dienstprogramm `u62_adm` Meldungen nach `stdout` aus.

Alle Meldungen beginnen mit der Zeichenfolge `u62_adm <nn>`, hierbei bezeichnet `<nn>` die Meldungsnummer.

- Die Meldungen mit den folgenden Nummern sind normales Verhalten: `20` bis `45`, `56`
- Zu den Meldungen mit den Nummern `06`, `09`, `12`, `13`, `52`, `53` und `59` siehe unten.
- Bei allen anderen `u62_adm <nn>` Meldungen, die nicht aufgelistet sind, informieren Sie den Systemdienst.

06 Instanz &INST: Instanz befindet sich noch in der Initialisierungsphase oder bereits inder Endebehandlung.

Die Proxy-Komponente openUTM-LU62 Gateway wird gerade gestartet oder beendet.
Maßnahme: Wiederholen Sie die Aktion.

09 Instanz &INST: Instanz reagiert nicht in angemessener Zeit.

Verzögerung, die bei höherer Last und großen Konfigurationen auftreten kann.
Maßnahme: Wiederholen Sie die Aktion.

12 Keine openUTM-LU62-Instanz aktiv.

Die Proxy-Komponente openUTM-LU62 Gateway ist noch nicht gestartet.
Maßnahme: Starten Sie openUTM-LU62 Gateway.

13 Die angegebene Instanz ist nicht aktiv.

Die Proxy-Komponente openUTM-LU62 Gateway ist noch nicht gestartet.
Maßnahme: Starten Sie openUTM-LU62 Gateway.

52 Fehler beim Oeffnen der Tracedatei &FILENAME, errno &ERRNO (&ERRTXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

53 Fehler beim Lesen der Tracedatei &FILENAME, errno &ERRNO (&ERRTXT)

Die angegebene Datei ist keine Tracedatei vom openUTM-LU62 Gateway.
Maßnahme: Geben Sie die richtige Datei an.

59 Das Extrahieren des Protokoll-Trace kann einige Zeit dauern. Bitte warten ...

Maßnahme: Warten Sie bis die Trace-Aufbereitung beendet ist.

13.15.5 Fehlermeldungen des openUTM-LU62 Gateways beim Konfigurieren

Beim Konfigurieren des openUTM-LU62 Gateways mit der Management Console (Update Configuration) gibt das openUTM-LU62-Dienstprogramm `u62_gen` Meldungen nach `stdout` aus.

Alle Meldungen beginnen mit der Zeichenfolge `u62_gen <nn>`, hierbei bezeichnet `<nn>` die Meldungsnummer.

- Die Meldung mit der Nummer `u62_gen 60` ist normales Verhalten.
- Zu den Meldungen `51`, `58`, und `59` siehe unten.
- Bei allen anderen `u62_gen <nn>` Meldungen, die nicht aufgelistet sind, informieren Sie den Systemdienst.

51 Datei &FILENAME kann nicht geöffnet werden, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

58 Fehler beim Schreiben in die Datei &FILENAME, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

59 Fehler beim Lesen aus der Datei &FILENAME, errno &ERRNO (&ERRTEXT)

Maßnahme: ERRNO siehe Abschnitt „Systemfehlercodes“ auf Seite 614 bzw. informieren Sie den Systemdienst.

13.16 Fehlercodes

Dieses Kapitel informiert über folgende Themen:

- Fehlercodes bei Dateibearbeitung (DMS-Fehlercodes)
- Systemfehlercodes

13.16.1 Fehlercodes bei Dateibearbeitung (DMS-Fehlercodes)

Im Zusammenhang mit der Dateibearbeitung treten im Fehlerfall Returncodes der Form yxxx auf. Diese werden auch DMS-Fehler genannt und haben folgende Bedeutung:

y	Das erste Zeichen y bezeichnet die Funktion, bei der der Fehler aufgetreten ist. y kann folgende Werte annehmen: A Fehler beim Laden eines Shared Memories in den Adressraum C Fehler beim close-Aufruf D Fehler beim Abmelden von einem Shared Memory F Fehler beim fstat/stat-Aufruf G Fehler beim Einrichten eines Shared Memory L Fehler beim lseek-Aufruf O Fehler beim open-Aufruf R Fehler beim read-Aufruf W Fehler beim write-Aufruf X Fehler beim create-Aufruf
xxx	Die drei Zeichen xxx stellen die abdruckbare Fehlernummer dar, die vom Betriebssystem in der externen Variablen 'errno' hinterlegt wird. Die Bedeutung der einzelnen Fehlernummern ist in den Betriebsystem-Handbüchern und in der Header-Datei errno.h beschrieben. Eine Auswahl der häufigsten Fehlercodes finden Sie in Abschnitt „Systemfehlercodes“ auf Seite 614.

Zusätzlich gibt es noch die Fehler CONS, LERR, OERR, REND, RERR, WERR, SXDE, SDDE und SDFE. Sie haben folgende Bedeutung:

CONS Der Dateiinhalt ist inkonsistent.
LERR lseek konnte nicht auf die gewünschte Stelle positioniert werden.
OERR Es wurde versucht, ein Dateiverzeichnis als normale Datei zu öffnen.
REND Beim Lesen aus einer Datei wurde das Dateiende erreicht.
RERR Es konnten nicht genügend Bytes gelesen werden.
WERR Es konnten nicht genügend Bytes geschrieben werden.
SXDE Ein Dateiverzeichnis konnte nicht eingerichtet werden.
SDDE Ein Dateiverzeichnis konnte nicht gelöscht werden.
SDFE Eine Datei konnte nicht gelöscht werden.

13.16.2 Systemfehlercodes

Eine Reihe von Meldungen des BeanConnect Proxy-Containers und des openUTM-LU62 Gateway enthalten als Insert eine Fehlernummer, die vom Betriebssystem zurückgegeben wird. Die Maßnahmen zu diesen Meldungen hängen vom Fehlercode ab.

Die folgende Tabelle beschreibt die häufigsten Fehlerursachen. Alle anderen, nicht in der Tabelle aufgeführten Ursachen sind schwerwiegende Fehler, bei denen der Systemdienst zu informieren ist.

ERRNO	Bedeutung
2	Datei oder Verzeichnis nicht vorhanden.
12	Speicherengpass im System.
13	Zugriffsrechte auf Datei oder Verzeichnis nicht vorhanden.
17	Datei existiert bereits.
28	Plattenengpass.
36	Betriebsmittel wurde gelöscht.

14 Cobol2Java

BeanConnect-Clients können über eine Anzahl unterschiedlicher Protokolle mit einer COBOL-Anwendung auf einem BS2000-System kommunizieren. Für diesen Informationsaustausch werden Byte-Arrays oder Strings verwendet. Die Informationsstruktur wird durch die COBOL-Anwendung definiert. Folglich gestaltet sich für Anwendungsentwickler der Datenaustausch mit dem Legacy-Service schwierig, da dies eine sehr genaue Kenntnis der Datenstruktur des COBOL-Programms erfordert. Das Tool Cobol2Java vereinfacht die Integration von BS2000-COBOL-Anwendungen und BeanConnect-Clients.

Dieses Kapitel enthält Informationen zu folgenden Themen:

- COBOL-Datentypen auf Java-Klassen abbilden
- COBOL-Datentypen konvertieren
- Programmier-Referenz
- Beispiel
- Fehlermeldungen und Fehlerbehandlung

14.1 COBOL-Datentypen auf Java-Klassen abbilden

Das Tool Cobol2Java ermöglicht die objektorientierte Abbildung von COBOL-Datenstrukturen auf Java-Klassen.

Cobol2Java setzt sich aus folgenden Teilen zusammen:

- Cobol2XML, ein BS2000-Tool, das XML-Beschreibungen zu einer BS2000 COBOL-Struktur generiert
- ein Framework mit Cobol2Java-Konvertierungsklassen
- ein Programmgenerator für die Generierung von anwendungsspezifischen Klassen auf der Grundlage dieses Frameworks

Der Programmgenerator verwendet zur Generierung anwendungsspezifischer Java-Klassen ein XSLT-Stylesheet. Dieses wird auf der Grundlage einer XML-Beschreibung des COBOL-Services ausgeführt, die Sie mit Hilfe des Cobol2XML-Tools in BS2000-Systemen generieren können.

Die Cobol2Java-Klassen können die Byte-Arrays eines COBOL-Programms verarbeiten oder ein Byte-Array generieren, das vom COBOL-Service ausgewertet werden kann. Für den Zugriff auf die einzelnen Datenfelder innerhalb eines Byte-Arrays stellt das Tool Cobol2Java die Zugriffsmethoden bereit, die zu dem Datentyp passen, der in der COBOL-Struktur definiert ist.

Die Abbildung unten zeigt den Generierungsablauf.

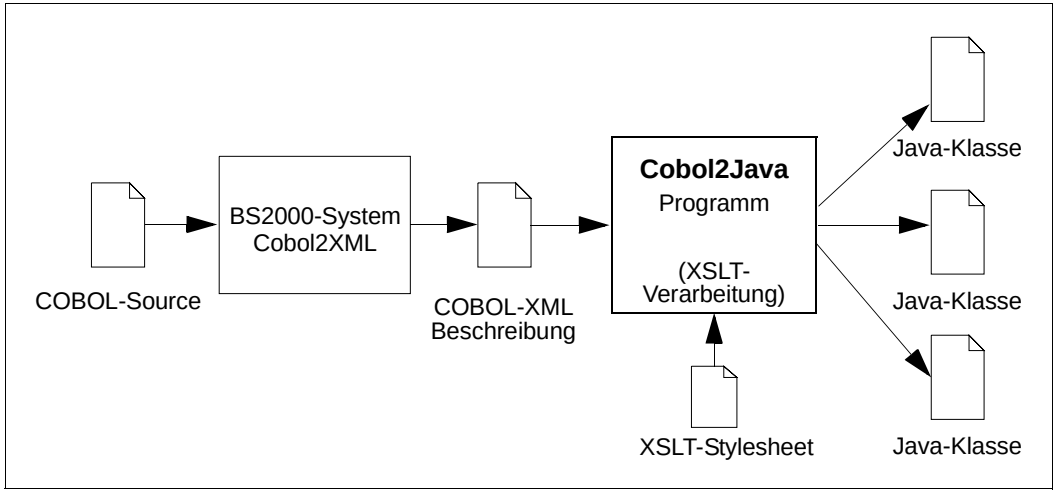


Bild 69: Cobol2Java Generierungsschritte

Mit dem Tool Cobol2XML kann in einem BS2000-System aus der BS2000-COBOL-Source eine XML-Datei generiert werden. Anschließend verwendet Cobol2Java diese Datei als Grundlage für die Erstellung von Java-Klassen.

Darüber hinaus kann Cobol2Java auch mit COBOL-Programmen verwendet werden, die für andere Plattformen, wie z.B. Unix- oder Linux-Systeme, entwickelt wurden. Voraussetzung hierfür ist, dass die XML-Beschreibung auf einem BS2000-System generiert wurde. Allerdings kann die volle Funktionalität nicht garantiert werden, da die Speicherzuweisung vom COBOL-Compiler abhängt.

14.1.1 Systemanforderungen

Folgende Betriebssysteme werden als Cobol2Java-Zielplattformen unterstützt:

- Solaris-, Linux- und andere Unix-Systeme
- Windows-Systeme

Folgende Programme müssen installiert sein, damit Sie mit Cobol2Java arbeiten können:

- Java SDK
- Ant (im Lieferumfang enthalten)

Cobol2XML muss auf einem BS2000-System installiert sein.

14.1.2 Installation

Die für Cobol2Java erforderlichen Komponenten werden zusammen mit den BeanConnect-Tools zur Verfügung gestellt

Die Installation von Cobol2Java ist im Abschnitt „BeanConnect Tools installieren“ auf Seite 78 beschrieben.

Die folgende Verzeichnisstruktur wird bei der Installation erstellt:

Verzeichnis	Dateien	Bedeutung
/	build.xml	Ant-Skript
	cobol2java.properties	Konfigurationsdatei
	runAnt.sh	Ant-Skript (Unix-/Linux-System)
	runAnt.bat	Ant-Skript (Windows-System)
	xml2java.sh	Skript (Unix-/Linux-System)
	xml2java.cmd	Skript (Windows-System)
/api		JavaDoc
/BS2Files/ftp	COB2XML.LIB	BS2000-Tool Cobol2XML für Dateitransfer mit FTP
/BS2Files/openFT	COB2XML.LIB	BS2000-Tool Cobol2XML für Dateitransfer mit openFT
/Docs	Copyright.htm	Copyright Hinweise der verwendeten openSource Produke
	Readme.pdf	Freigabemitteilung
/lib	ant.jar ant-launcher.jar	Ant
	BeanConnectCob2java.jar	Transfer-Tool XML nach Java

Verzeichnis	Dateien	Bedeutung
	BeanConnectCob2java_ext.jar	notwendig für die Ausführung der generierten Cobol2Java-Klassen, wenn Cobol2Java in einem eigenen Programm ohne BeanConnect verwendet wird
	BeanConnectEncoding.jar	Code-Konvertierung
	functions.xslt	XSLT-Stylesheet
	mkcob2java.xslt	XSLT-Stylesheet
	newformat.dtd	Document Type Definition
/sample	*.xml	XML-Beispieldateien, unter Verwendung von Cobol2XML im BS2000-System generiert
	newformat.dtd	Document Type Definition

14.2 COBOL-Datentypen konvertieren

Die Abbildung von COBOL-Datentypen auf Java-Klassen umfasst folgende Schritte:

- XML-Beschreibung für COBOL-Programm im BS2000-System erstellen
- Java-Klassen auf Unix-, Linux- oder Windows-Systemen generieren

14.2.1 XML-Beschreibung für COBOL-Programm im BS2000-System erstellen

Mit dem BS2000-Tool Cobol2XML wird ein COBOL-Programm oder COPY-Element in XML konvertiert. Folgende Schritte sind erforderlich:

- Transferieren der LMS-Bibliothek `COB2XML.LIB` zum BS2000-System.
- Konvertieren der Datenstrukturen von COBOL-Programmen oder COPY-Elementen mit Hilfe der Prozeduren `D.XMLPROG` oder `D.XMLCOPY` in XML.
- Transferieren der XML-Beschreibungen in Textformat zur weiteren Verarbeitung mit Cobol2Java auf Unix-, Linux- oder Windows-Systeme.

14.2.1.1 LMS-Bibliothek nach BS2000-System transferieren

Cobol2XML ist ein BS2000-Tool. Daher müssen Sie die Bibliothek `COB2XML.LIB` an eine BS2000-Benutzerkennung transferieren. Je nach Transfermodus können Sie entweder die Datei `COB2XML.LIB` unter `cobol2java/BS2Files/ftp` oder die Datei `COB2XML.LIB` unter `cobol2java/BS2Files/openFT` transferieren.

Verwenden Sie für ftp-Übertragungen den Modus `binary`.
Wenn Sie mit openFT transferieren, verwenden Sie die Dateiart `binary`.
Verfügt das BS2000-System nicht über ausreichenden Speicherplatz, könnten Sie bei der ftp-Übertragung die Fehlermeldung `DD33` (ausgegebene Datei nicht vorhanden) erhalten.

Benennen Sie die Bibliothek im BS2000-System nicht um, da die Konvertierungsprozedur auf Elemente dieser Bibliothek zugreift.

14.2.1.2 Datenstrukturen konvertieren

Die Konvertierungsprozedur kann nur unter der Kennung aufgerufen werden, unter der die Bibliothek COB2XML.LIB gespeichert ist.

Weisen Sie vor dem Aufrufen der Konvertierungsprozedur die zur Kompilierung der Source erforderlichen COPY-Bibliotheken einem Link-Namen COBLIB<n> (<n>= 1, .., 9) zu. Weisen Sie z.B. für ein openUTM COBOL-Programm der Bibliothek einen Link-Namen zu, die die UTM-COPY-Elemente enthält, um sicher zu stellen, dass diese gefunden werden. Verwenden Sie dazu den Befehl ADD-FILE-LINK:

```
/ADD-FILE-LINK COBLIB1, <copy_library>
```

(Näheres hierzu erfahren Sie im Handbuch „COBOL2000 (BS2000), COBOL-Compiler“).

Der Linkname COBLIB wird intern verwendet.

Starten Sie anschließend die Prozeduren D.XMLPROG oder D.XMLCOPY aus der Bibliothek COB2XML.LIB. Die Parameter werden in den folgenden Abschnitten beschrieben.

14.2.1.3 D.XMLPROG

Auf der Grundlage der entsprechenden Programm-Source generiert D.XMLPROG eine XML-Beschreibung der Datenstrukturen eines COBOL-Programms.

So starten Sie D.XMLPROG mit dem Befehl CALL-PROCEDURE:

```
CALL-PROC
FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=COB2XML.LIB,ELEM=D.XMLPROG)
,PROCEDURE-PARAMETERS=(
{SRC=FILE,TSTNAM=<cobol_source>|SRC=LIB,LIB=<cobol_lib>,
TSTNAM=<cobol_source>},
[XMLOUT=<ouptput_file>,]
[COBRUN=<compiler_options>] )
```

Erläuterung der Parameter:

SRC=FILE,TSTNAM=<cobol_source>
: Das COBOL-Programm, für dessen Datenstrukturen eine XML-Beschreibung erzeugt werden soll, befindet sich in der Datei <cobol_source>.

SRC=LIB,LIB=<cobol_lib>, TSTNAM=<cobol_source>
: Das COBOL-Programm, für dessen Datenstrukturen eine XML-Beschreibung erzeugt werden soll, befindet sich im Element <cobol_source> in der LMS-Bibliothek <cobol_lib>.

XMLOUT=<output_file>
: Name der Datei, in die die Datenstrukturen im XML-Format geschrieben werden. Dieser Name muss das Suffix .xml haben. Wenn dieser Parameter nicht angegeben und der Linkname XMLLINK nicht definiert ist, wird die XML-Beschreibung in die Datei XMLFIL.COBOL.<progID>.XML geschrieben, wobei <progID> für den in PROGRAM-ID definierten Programmnamen steht.

COBRUN=<compiler_options>
: Festlegung von Compiler-Optionen in Form eines Strings aus maximal 121 Zeichen. Trennen Sie die einzelnen Optionen durch Kommata. Sie können z. B. folgendes angeben:

COMMENT=YES/NO
: Es werden keine Kommentarzeilen ausgegeben, wenn die Option COMMENT=NO eingestellt ist.

DTD=YES/NO
: Es wird kein Verweis auf die Document Type Definition ausgegeben, wenn die Option DTD=NO eingestellt ist.

14.2.1.4 D.XMLCOPY

Auf der Grundlage des entsprechenden COPY-Elements generiert D.XMLCOPY eine XML-Beschreibung der Datenstrukturen dieses COBOL-Elements.

So starten Sie D.XMLCOPY mit dem Befehl CALL-PROCEDURE:

```
CALL-PROC
FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=COB2XML.LIB,ELEM=D.XMLCOPY)
,PROCEDURE-PARAMETERS=(
LIB=<cobol_lib>,
ELEM=<cobol_copy>,
[XMLOUT=<output_file>,]
[COBRUN=<compiler_options>] )
```

Erläuterung der Parameter:

LIB=<cobol_lib>, ELEM=<cobol_copy>
: Das COBOL-COPY-Element, für dessen Datenstrukturen eine XML-Beschreibung erzeugt werden soll, ist als Element <cobol_copy> in der LMS-Bibliothek <cobol_lib> verfügbar.

XMLOUT=<output_file>
: Name der Datei, in die die Datenstrukturen im XML-Format geschrieben werden. Dieser Name muss das Suffix .xml haben. Wenn dieser Parameter nicht angegeben und der Linkname XMLLINK nicht definiert ist, wird die XML-Beschreibung in die Datei XMLFIL.COBOL.<cobol_copy>.XML geschrieben, wobei <cobol_copy> für den Namen des COPY-Elements steht.

`COBRUN=<compiler_options>`
: Spezifikationen von Compiler-Optionen in Form eines Strings aus maximal 121 Zeichen. Trennen Sie die einzelnen Optionen durch Kommata. Sie können z. B. folgendes angeben:

`COMMENT=YES/NO`
: Es werden keine Kommentarzeilen ausgegeben, wenn die Option `COMMENT=NO` eingestellt ist.

`DTD=YES/NO`
: Es wird kein Verweis auf die Document Type Definition ausgegeben, wenn die Option `DTD=NO` eingestellt ist.

14.2.1.5 Beispielaufruf

```
/PROC A
/REMARK is necessary for converting UTM programs
/ADD-FILE-LINK COBLIB1,$TSOS.SYSLIB.UTM.062.COB
/ADD-FILE-LINK COBLIB2,COBOLCOPY.LIB
/CALL-PROC F-F=*LIB(LIB=COB2XML.LIB,ELEM=D.XMLPROG),P-P=(SRC=FILE -
/  ,TSTNAM=COBTAC.CBL -
/  ,XMLOUT=COBTAC.XML),LOG=N
/ENDP
```

14.2.1.6 Generierte Dateien

Die Ausgabe von `D.XMLPROG` und `D.XMLCOPY` befindet sich entweder in der Datei,

- die zuvor mit dem Befehl `ADD-FILE-LINK` dem Link-Namen `XMLLINK` zugewiesen wurde oder
- die im Parameter `XMLOUT` festgelegt ist.

Wenn Sie keine dieser Möglichkeiten in Anspruch nehmen, wird die Ausgabe von `D.XMLPROG` und `D.XMLCOPY` in die Datei `XMLFIL.COBOL.<progID>.XML` geschrieben. Hier steht `<progID>` für den in `PROGRAM-ID` (`D.XMLPROG`) definierten Programmnamen oder für den Namen des COPY-Elements (`D.XMLCOPY`).

Sie müssen die generierten Dateien mit der XML-Beschreibung im Textformat zur weiteren Verarbeitung mit Cobol2Java auf Unix-, Linux- oder Windows-Systeme transferieren.

14.2.2 Java-Klassen auf Unix-, Linux- oder Windows-Systemen generieren

Java-Klassen können Sie mit oder ohne Ant generieren.

14.2.2.1 Java-Klassen mit Ant generieren

Java-Klassen werden mit Hilfe eines Ant-Skripts generiert. Die Parameter sind in der Datei `cobol2java.properties` wie folgt enthalten:

```
xml.file=<xml_file>
cobol.struct=<list_of_structure_names>
package.name=<package_name>
doc.dir=<doc_directory>
jar.dest=<jarfile_name>
code.convention={java|cobol}
undef.pic9=<undef_value>
```

Erläuterung der Parameter:

`xml.file=<xml_file>`

Der Parameter `xml.file` gibt den Namen der Datei an, die die XML-Beschreibung der zu verarbeitenden Cobol-Struktur enthält.

Stellen Sie sicher, dass die DTD `newformat.dtd` im selben Verzeichnis liegt wie die XML-Datei. Eine Kopie der `newformat.dtd` finden Sie in den Verzeichnissen `lib` und `sample`.

`cobol.struct=<list_of_structure_names>`

Geben Sie im Parameter `cobol.struct` eine Liste von COBOL-Strukturen an, die durch Leerzeichen voneinander getrennt werden. Cobol2Java durchsucht alle Elemente (Datensätze wie Felder) in der Input-Datei nach den in `<list_of_structure_names>` enthaltenen Namen. Für jedes gefundene Element wird eine Java-Klasse generiert. Geben Sie mindestens einen Strukturnamen an.
Für den Parameter `cobol.struct` sind nicht alle Angaben sinnvoll. Daher sollten Sie folgende Einschränkungen beachten:

- Geben Sie in einem Aufruf keine zwei ineinander geschachtelte Strukturen an. Grund: Die Klasse der tieferliegenden Struktur wird zweimal erzeugt (als root level und als sub level class). Die später erzeugte Klasse überschreibt die erste und es gibt Fehler bei der Übersetzung bzw. bei der Verwendung der Klassen.
- Geben Sie keine Struktur- oder Feldnamen an, die mehrfach vorhanden sind (occurs, Array). Grund: Über die erzeugte Klasse kann nur auf das erste Element des Arrays zugegriffen werden. Statt dessen sollten übergeordnete Strukturen angegeben werden.
- Die Angabe von Struktur- oder Feldnamen, die mehrfach vorhanden sind, ist nicht gestattet. Es wird eine Meldung ausgegeben und keine Java-Klassen erzeugt.

`package.name=<package_name>`
Der Parameter `package.name` enthält den Namen des Pakets, unter denen die generierten Java-Klassen zusammengefasst werden sollen.

`doc.dir=<doc_directory>`
`doc.dir` enthält den Verzeichnisnamen für JavaDoc.

`jar.dest=<jarfile_name>`
`jar.dest` enthält den Namen der JAR-Datei, die generiert werden soll.

`code.convention={java|cobol}`
`code.convention` gibt die für Java-Klassen verwendete Namenskonvention an. Bei Angabe der Bezeichnung `cobol` werden, wann immer möglich, alle Namen aus dem COBOL-Programm übernommen. Andernfalls werden alle Namen den Konventionen angepasst, die in Java für die Benennung von Klassen, Variablen, Methoden und Objekten festgelegt sind (siehe auch Abschnitt „Namenskonventionen“ auf Seite 629).

`undef.pic9=<undef-value>`
`undef.pic9` gibt an, welcher ausgezeichnete Wert für Pic9-Felder "undefiniert" bedeuten soll. Dabei kann `<undef-value>` folgende Form haben:

`0x<nn>`	wobei `<nn>` der hexadezimale Bytewert ist, oder
`"'<char>'"`	wobei `<char>` ein abdruckbares Zeichen mit einer 1-Byte-Darstellung ist. Um in der Java-Klasse den Wert `'<char>'` zu erhalten, muss der Parameter zusätzlich in Hochkommata (") angegeben werden, z.B. `undef.pic9="' '"`

Wird `undef.pic9` nicht angegeben, ist '0' der vordefinierte Wert für undefinierte Pic9-Felder.
Achtung! Bei Angabe des Bytewertes und aktiviertem Encoding ist ggf. die Codeumsetzung zu beachten. Siehe dazu auch Abschnitt „Datenfeld lesen“ auf Seite 632.

Wenn Sie generierte Java-Klassen kompilieren, ist es wichtig, dass die Klassen- und Dateinamen einander hinsichtlich der Groß- und Kleinschreibung entsprechen. Da das Windows-Dateisystem nicht zwischen Groß- und Kleinschreibung unterscheidet, meldet der Java-Compiler während der Kompilierung eventuell folgenden Fehler, wenn er z.B. eine Klasse `Benid` in einer Datei `BENID.java` findet:
`"[javac] BENID.java:19:class Benid is public, should be declared in a file named Benid.java"`

Dieser Fehler tritt auf, wenn Sie dieselbe COBOL-Struktur zunächst mit `code.convention=cobol` und dann noch einmal mit `code.convention=java` erstellen. Wenn Sie `code.convention` ändern möchten, müssen Sie zuerst die Sourcen löschen, die zuvor im Verzeichnis `src` erstellt wurden.

Starten Sie das Programm mit dem Skript `runAnt.sh` (Unix-/Linux-System) oder `runAnt.bat` (Windows-System). Dies startet die Generierung unter Verwendung des Ant-Programms, das mit Cobol2Java ausgeliefert wird. Als Ergebnis dieses Generierungsprozesses werden die Java-Sourcen im Verzeichnis `src` angelegt, kompiliert und in der in `jar.dest` angegebenen JAR-Datei gespeichert. Wenn Sie das mitgelieferte Ant-Skript nicht verwenden möchten, rufen Sie Ant aus dem Verzeichnis auf, das die Datei `build.xml` enthält.

Einige der hier verwendeten Tools benötigen sehr viel Arbeitsspeicher. Nehmen Sie daher mit Hilfe der Optionen `-Xss (stack size)` und `-Xmx (heap size)` die für große Datenstrukturen erforbderlichen Anpassungen vor. Auf Unix-/Linux-Systemen muss ggf. auch das `stack size limit` mit dem Kommando `ulimit` angepasst werden.

Damit `runAnt` korrekt ausgeführt wird, müssen Sie die Java-Programme `javac` und `javadoc` aufrufen können, d.h. das Java SDK-Programmverzeichnis muss in der Umgebungsvariablen `PATH` vorhanden sein.

Sie müssen bei der Angabe der Pfadnamen für die Dateien `xml.file`, `doc.dir` und `jar.dest` die Trennzeichen „/“ oder „\\“ verwenden, da das Ant -Skript „\“ und das nachfolgende Zeichen als Steuerzeichen interpretiert.

Beispiel 27 Beispiel für eine cobol2java.properties-Datei

```
# Properties to set for Cobol2Java Program
# used by Ant

# Name of Source XML generated by the BS2000 COBOL Compiler
# Make sure the DTD File is available!!
xml.file=cobkb.xml

# Name of the COBOL Records
# Space separated list:
# cobol.struct=RECORD1 RECORD3 will create Java classes for
# RECORD1 and RECORD3
cobol.struct=MPUT-MSG

# Name of the package to be generated
package.name=de.siemens.cob2java.cobkb
#

# Directory for JavaDoc
doc.dir=doc/cobkb

# Jar file name
jar.dest=cobkb.jar
```

```
# Determines what code convention the generated code will use
#
# code.convention=java
# code.convention=cobol

code.convention=java

# defines a non numeric value which marks a PIC9 field as undefined
#
undef.pic9=0x20
```

14.2.2.2 Java-Klassen ohne Ant generieren

Das oben beschriebene Generierungsverfahren ist für Batch-Verarbeitung und große Datenmengen nur begrenzt verwendbar.

Aus dem Cobol2Java-Home-Verzeichnis können Sie folgenden Befehl aufrufen, um eine größere Anzahl von Java-Klassen auf der Grundlage Ihrer eigenen Spezifikationen zu generieren:

```
java -Xss8m -Xmx512m -classpath lib/BeanConnectCob2java.jar
de.siemens.cob2java.Cob2Java [-undef.pic9=<default-value>] <xml_file>
<package_name>
<code_convention>
<cobol_struct1> [<cobol_struct2> ...]
```

Erläuterung der Parameter:

`undef.pic9=<default-value>`
: `undef.pic9` gibt an, welcher ausgezeichnete Wert für Pic9-Felder "undefiniert" bedeuten soll. Details siehe Beschreibung von `undef.pic9` in Abschnitt „Java-Klassen mit Ant generieren“ auf Seite 623.

`<xml_file>`
: Name der XML-Datei, die die Beschreibung der COBOL-Datenstruktur enthält.

`<package_name>`
: Paketname für die generierten Klassen.

`<code_convention>`
: Von den generierten Klassen verwendete Konvention: `cobol` oder `java`.
Wenn Sie diese Parameter verwenden, beachten Sie bitte die Kommentare zum Parameter `code.convention`<z_ignore> <Color>auf Seite 624.

`<cobol_struct>[1...<n>]`
: Durch Leerzeichen getrennte Liste der zu konvertierenden COBOL-Strukturen. Geben Sie mindestens einen Namen an.

Einige der hier verwendeten Tools benötigen sehr viel Arbeitsspeicher. Nehmen Sie daher mit Hilfe der Optionen `-Xss (stack size)` und `-Xmx (heap size)` die für große Datenstrukturen erforderlichen Anpassungen vor. Auf Unix-/Linux-Systemen muss ggf. auch das `stack size limit` mit dem Kommando `ulimit` angepasst werden.

Beispiel 28 Java-Klassen generieren

```
java -Xss8m -Xmx512m -classpath lib/BeanConnectCob2java.jar
de.siemens.cob2java.Cob2Java cobkb.xml de.cobol java MPUT-MSG BENID
```

Dieses Programm generiert nur die Java-Klassen im Unterverzeichnis `src`. Es führt weder Kompilierung durch noch erstellt es JAR-Dateien oder Dokumentationen.

Sie finden ein Beispiel im Skript `xml2java.sh` auf Unix-/Linux-Systemen oder `xml2java.cmd` auf Windows-Systemen.

Stellen Sie sicher, dass die DTD `newformat.dtd` im selben Verzeichnis wie die XML-Datei liegt. Eine Kopie der `newformat.dtd` finden Sie in den Verzeichnissen `lib` und `sample`.

14.3 Programmier-Referenz

Dieser Abschnitt enthält eine Darstellung des Frameworks, das für die Konvertierung von COBOL-Datentypen nach Java und umgekehrt verwendet wird.

14.3.1 Typzuweisung

Die folgende Tabelle bietet einen Überblick über die im Framework vorhandenen Java-Klassen samt einer kurzen Beschreibung der für die Klassen verwendeten COBOL-Typen.

Java-Klasse	**Beschreibung**
`Datentyp`	Basisklasse für alle Konvertierungsklassen
`CobolRecord`	Basisklasse für COBOL-Strukturen
`PicX`	Für alphabetische/alphanumerische COBOL-Typen: `PIC X(n)`
`PicN`	Für nationale COBOL-Typen: `PIC N(n)`
`Pic9`	Für positive und ganzzahlige COBOL-Typen: `PIC 9`
`Pic9COMP`	Für ganzzahlige COBOL-Typen: `PIC S9(n)` und `PIC 9(n) USAGE [COMP, BINARY, COMP-5]`
`PicU`	Für COBOL-Typen, für die Cobol2Java keine besonderen Datenkonvertierungsklassen anbietet. Die Daten werden im Java-Programm ohne Konvertierung als Byte-Array verfügbar gemacht.

Das Verzeichnis `api` enthält die Dokumentation zu diesen Klassen.

Die folgende Tabelle zeigt die unterstützten COBOL-Typen und -Anweisungen:

COBOL-Anweisung	**Java-Support**
Datenstrukturen mit Stufennummern	`CobolRecord`
Pic X (n) (alphanumerisch)	`PicX`
Kombinationen von A, X, 9 (nicht nur 9)	`PicX`
Pic 9 (n) (numerisch)	`Pic9`
Pic N (n) (national)	`PicN`

`BINARY, COMP, COMP-5`	`Pic9Comp` für `PIC9(1)`nach `PIC9(18)`
	Nicht unterstützt:
	`PIC 9(19)` nach `PIC 9(31)` wird abgebildet auf `PicU`

`COMP-1, COMP-2, COMP-3`	Nicht unterstützt.
Alphanumerisch, druckaufbereitet Alphabetisch, druckaufbereitet	Abgebildet auf `PicX`. Die Vorbereitungsmaske wird ignoriert.

COBOL-Anweisung	Java-Support
Numerisch, druckaufbereitet	Nicht unterstützt. Abgebildet auf `PicU`.
`BLANK WHEN ZERO`	Nicht unterstützt. Abgebildet auf `PicU`.
`INDEX`	Nicht unterstützt.
`POINTER; PROCEDURE POINTER, OBJECT REFERENCE`	Nicht unterstützt.
`JUSTIFIED RIGHT`	`PicX`, ignoriert.
`SYNCHRONIZED`	Unterstützt.
Ebene Nummer 77	Unterstützt.
`OCCURS`	Begrenzt unterstützt. – Dynamische Arrays (`OCCURS DEPENDING ON`) werden mit einer festen Länge unterstützt. – `OCCURS INDEXED BY`, `OCCURS KEY IS` wird nicht unterstützt.
`REDEFINES`	Unterstützt.
`RENAMES`	Unterstützt.

14.3.2 Namenskonventionen

In diesem Abschnitt ist beschrieben, wie die Namen in den Java-Klassen für die unterschiedlichen Code-Konventionen aufgebaut werden.

Wenn `cobol` als Code-Konvention verwendet wird, gelten folgende Abbildungsregeln:

- Der COBOL-Name wird wann immer möglich beibehalten.
- Bindestriche werden durch Unterstriche ersetzt.
- Alle Arrays werden mit 0 indiziert.

Wenn `java` als Code-Konvention verwendet wird, gelten folgende Abbildungsregeln:

- Alle Großbuchstaben werden in Kleinbuchstaben umgewandelt.
- Buchstaben, die auf einen Bindestrich folgen, werden groß geschrieben. Alle Bindestriche werden entfernt.
- Der erste Buchstabe eines Klassennamens wird groß geschrieben.
- Die Namen von `Get`-Methoden werden aus `get` sowie dem Attributnamen gebildet, wobei der erste Buchstabe des Attributnamens groß geschrieben wird.
- Die Namen von `Set`-Methoden werden aus `set` sowie dem Attributnamen gebildet, wobei der erste Buchstabe des Attributnamens groß geschrieben wird.
- Alle Arrays werden mit 0 indiziert.

Beispiel 29 Namenszuweisungen für unterschiedliche Code-Konventionen

Abhängig von der angewandten Code-Konvention werden die folgenden Namen für das COBOL-Feld `EMPLOYEE-RECORD` gebildet:

Code-Konvention	Java-Attributname	Java Get-/Set-Methodenname
`cobol`	`EMPLOYEE_RECORD`	`getEMPLOYEE_RECORD ()` `setEMPLOYEE_RECORD ()`
`java`	`employeeRecord`	`getEmployeeRecord()` `setEmployeeRecord()`

Wenn die Datenstruktur untergeordnete Strukturen mit dem Namen `FILLER` enthält, oder wenn das Programm mehrere untergeordnete Strukturen mit demselben Namen enthält, dann werden diese Strukturen bezüglich ihrer Reihenfolge in der xml-Eingabe-Datei laufend durchnummeriert. Beim Erzeugen der Java-Klassen wird diese Nummer `n` in der Form `_R_n` an den generierten Namen angehängt.

Wenn die Datenstruktur Felder mit dem Namen FILLER enthält, dann werden diese Felder bezüglich ihrer Reihenfolge in der xml-Eingabe-Datei laufend durchnummeriert. Beim Erzeugen der Java-Klassen wird diese Nummer `n` wird in der Form `_n` an den generierten Namen angehängt.

14.3.3 Auf COBOL-Felder zugreifen

Die hierarchische COBOL-Datenstruktur wird auf eine hierarchische Klassenstruktur abgebildet.

Folglich kann ein Feld <XXX> so adressiert werden:

Lesezugriff: `level01.getLevel02().getLevel03().get<XXX>()`

Schreibzugriff: `level01.getLevel02().getLevel03().set<XXX>()`

Mit dieser Konstruktion können Sie hoch-performanten Zugriff auf tief verschachtelte Felder erlangen.

Anstelle von

```
in.getArray2(1).getLineTab().getLinex(3).getKeyx().setData1
(1,"Value1");
in.getArray2(1).getLineTab().getLinex(3).getKeyx().setData2
(1,"Value2");
```

kann Ebene für Ebene auf die Felder zugegriffen werden:

```
Keyx keyx;
keyx = in.getArray2(1).getLineTab().getLinex(3).getKeyx();
keyx.setData1(1,"Value1");
keyx.setData2(1,"Value2");
```

14.3.3.1 Datenfeld schreiben

Damit ein Datenfeld <XXX> geschrieben werden kann, muss die Methode `setXXX()` aufgerufen werden. Ein Objekt des COBOL-Datenfeldtyps wird als Parameter übergeben.

```
EmployeeRecord out = new EmployeeRecord();
out.setLastName(new PicX("LastName"));
```

Für jeden COBOL-Datentyp gibt es zusätzliche Methoden mit Parametern des Typs `String` oder `int` und `long`:

für `PicX, PicN`: `setXXX(String): out.setLastName("LastName");`

für `Pic9`: `setXXX(int), setXXX(long)`

14.3.3.2 Datenfeld lesen

Damit ein Datenfeld `<XXX>` gelesen werden kann, muss die Methode `get<XXX>()` aufgerufen werden. Bei dem durch diese Methode zurückgelieferten Wert, handelt es sich um ein Objekt, dessen Typ dem des COBOL-Datenfeldes entspricht. Jedes Objekt besitzt Methoden, die an den Typ angepasst sind, der beim Extrahieren der Daten verwendet wird:

für `PicX`: `toString()`

für `Pic9`: `longValue()`

Bei COBOL kann es vorkommen, dass ein numerisches Datenfeld (`PIC 9(n)`) mit einem nicht-numerischen Wert initialisiert wird, z.B. Leerzeichen oder `X'00'`. Um beim Zugriff auf solche "undefinierten" Felder keine `NumberFormatException` zu erhalten, gibt es folgende Methoden zur Überprüfung des Inhaltes:

für `Pic9`: `isUndefined(), isUndefined(byte)`

Der Defaultwert für "undefiniert" kann beim Erzeugen der Java-Klassen angegeben werden. (siehe Abschnitt „Java-Klassen auf Unix-, Linux- oder Windows-Systemen generieren“ auf Seite 623).

Falls ein Datenfeld nur mit dem Defaultwert für "undefiniert" initialisiert ist (`isUndefined()` gibt `true` zurück) und dieser Defaultwert nicht numerisch ist, wird beim Lesen des Feldes der Wert 0 zurückgegeben.

14.3.3.3 Ersatzdatentyp PicU

Der Ersatzdatentyp `PicU` wird für nicht unterstützte Datenfelder verwendet. Damit ist ein transparenter Zugriff auf die Daten möglich. Der Anwendungsprogrammierer kann dann diese Daten mit eigenen Ressourcen bearbeiten. Die ungeänderten Daten der Partneranwendung werden als Byte-Array bereitgestellt.

Dieser Typ besitzt folgende Methoden:

Lesezugriff: `getBytes()`

Schreibzugriff: `setBytes()`

14.3.3.4 Daten für die gesamte Struktur festlegen und lesen (Senden und Empfangen)

Die Daten einer ganzen Datensatz- oder Datengruppe werden mit der Methode `getBytes()` gelesen und mit der Methode `setBytes()` geschrieben.

Mit den BeanConnect-Kommunikationsmethoden zum Senden und Empfangen von Daten (`sndRecord()`, `rcvRecord()` und den `call()`-Methoden mit dem Parameter `ByteContainer`) können Sie Java-Objekte auch direkt angeben, da alle Klassen, die die von Cobol2Java generierten COBOL-Strukturen repräsentieren, das Interface `ByteContainer` implementieren.

Beispiel 30 Daten senden und empfangen

```
// Java object which was created by Cobol2Java
EmployeeRecord emplRecord = new EmployeeRecord();
// Set encoding of the connection
emplRecord.setEncoding( connection.getEncoding() );
emplRecord.setEncodingActive( connection.isEncodingActive() );
// Connection object: sndRecord/rcvRecord method
connection.sndRecord(emplRecord);
connection.rcvRecord(emplRecord);
```

Mit dem Aufruf von `sndRecord()` aus `emplRecord` werden die Daten über die Verbindung geschickt und mit dem Aufruf von `rcvRecord()` werden die Daten aus der Verbindung in `emplRecord` gespeichert.

Weitere Einzelheiten zum Thema Codierung finden Sie in Kapitel „Zeichensatz-Konvertierung und Sprachunterstützung“ auf Seite 489.

Um ein Datenfeld `XXX` zu ändern, genügt es nicht, nur das über die Methode `get<XXX>` erhaltene Objekt zu ändern. Stattdessen muss das Feld über eine der `set<XXX>`-Methoden geändert werden.

Beispiel 31 Daten holen und speichern

```
// Data is received and stored in the EmployeeRecord
connection.rcvRecord(in);           (1)
PicX       lastName;
String     newName = "MyName";
lastName = in.getLastName();        (2)
lastName.setString(newName);        (3)
in.setLastName(lastName);           (4)
// or in.setLastName(newName);      (5)
// The data stored in EmployeeRecord is sent connection.sndRecord(in);
```

wobei:

(1) Java-Klasse, die von Cobol2Java aus einer COBOL-Datenstruktur erstellt wurde

(2) Datenfeld als `PicX`-Objekt zurückgeben

(3) `PicX`-Objekt ändern

(4) Datenfeld über die Methode `set` mit dem Parameter `PicX` ändern

(5) Datenfeld über die Methode `set` mit dem Parameter `String` ändern

14.3.4 Java/EBCDIC-Konvertierung

So gehen Sie bei der Konvertierung vor:

1. Erstellen Sie die Cobol2Java-Objekte mit einem leeren Konstruktor.

   ```
   cob2javaclass cob2java = new cob2javaclass();
   ```

2. Stellen Sie anschließend die Codierung der Verbindung ein:

   ```
   cob2java.setEncoding( connection.getEncoding() );
   cob2java.setEncodingActive( connection.isEncodingActive() );
   ```

3. Legen Sie die Cobol2Java-Objekte direkt in den Kommunikationsmethoden fest:

   ```
   sndRecord(ByteContainer), rcvRecord(ByteContainer),
   call(ByteContainer, ByteContainer)
   ```

Weitere Einzelheiten zum Thema Codierung finden Sie in Kapitel „Zeichensatz-Konvertierung und Sprachunterstützung“ auf Seite 489.

14.3.5 Unterstützung des formatierten Modus

Cobol2Java bietet begrenzte Unterstützung für den formatierten Modus. Die Verwendung der Klasse `Kcat` vereinfacht den Gebrauch der +Formate. Diese Klasse enthält zur Unterstützung des formatierten Modus Konstanten für `KDCS ATTRIBUTE`.

14.4 Beispiel

Dieser Abschnitt zeigt beispielhaft die Konvertierung von Datentypen eines COBOL-Programms in Java-Klassen. Es enthält Informationen darüber, wie in einem Standardfall ein einfaches COBOL-Programm konvertiert werden kann. Sie erhalten die folgenden Informationen:

- COBOL-Beispielprogramm
- XML-Beschreibung erstellen
- Java-Klassen generieren
- Verwendung der generierten Klassen

14.4.1 COBOL-Beispielprogramm

Das nachfolgende Programm `employee.cbl` wird als Beispiel verwendet.

```
IDENTIFICATION DIVISION.
   PROGRAM-ID. EMPLOYEE.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
   01 EMPLOYEE-RECORD
      05 EMPLOYEE-NUMBER   PIC X(08).
      05 FIRST-NAME        PIC X(20).
      05 LAST-NAME         PIC X(20).
      05 PAY-METHOD        PIC X.
      05 SALARY-INFO.
         10 ANNUAL-SALARY  PIC 9(5).
LINKAGE SECTION.
      COPY KCKBC.
      ...
      COPY KCPAC.
PROCEDURE DIVISION
...
```

14.4.2 XML-Beschreibung erstellen

Die XML-Beschreibung `employee.xml` kann im BS2000-System mit Cobol2XML auf der Grundlage des COBOL-Programms `employee.cbl` generiert werden. Anschließend muss die Datei `employee.xml` auf das Unix-, Linux- oder Windows-System transferiert werden.

Weitere Einzelheiten zu Cobol2XML finden Sie in Abschnitt „XML-Beschreibung für COBOL-Programm im BS2000-System erstellen“ auf Seite 619.

LMS-Bibliothek COB2XML.LIB nach BS2000-System transferieren

Senden Sie die Datei `COB2XML.LIB` an das BS2000-System (siehe Abschnitt „LMS-Bibliothek nach BS2000-System transferieren“ auf Seite 619).

Datenstrukturen vom COBOL-Programm in XML konvertieren

1. Melden Sie sich auf dem BS2000-System unter der Benutzerkennung an, unter der sich die Dateien `COB2XML.LIB` und `EMPLOYEE.CBL` befinden.
2. Weisen Sie den Link-Namen `COBLIB<n>` (`<n>`= 1, .., 9) den COPY-Bibliotheken zu, die für die Konvertierung der Source erforderlich sind:

   ```
   /ADD-FILE-LINK COBLIB1, $TSOS.SYSLIB.UTM.062.COB
   ```

3. Starten Sie die Prozeduren `D.XMLPROG` aus der Bibliothek `COB2XML.LIB`:

   ```
   CALL-PROC
   FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=COB2XML.LIB,ELEM
                              =D.XMLPROG)
   ,PROCEDURE-PARAMETERS=(
   SRC=FILE,TSTNAM=employee.cbl,
   XMLOUT=employee.xml)
   ```

   Die Prozedur `D.XMLPROG` generiert eine XML-Beschreibung der Datenstrukturen, die im COBOL-Programm in der Datei `employee.cbl` verwendet werden. Die XML-Beschreibung wird in der Datei `employee.xml` gespeichert.

XML-Beschreibungen auf ein Unix-, Linux- oder Windows-System transferieren

Transferieren Sie die XML-Beschreibung `employee.xml` im Textformat zur weiteren Verarbeitung auf das Unix-, Linux- oder Windows-System, z.B. in das Unterverzeichnis `sample` von Cobol2Java.

14.4.3 Java-Klassen generieren

Die Klasse `EmployeeRecord` wird von Cobol2Java auf Unix-, Linux- oder Windows-Systemen auf der Grundlage der XML-Datei `employee.xml` generiert. Die Klassen werden durch Ant generiert, kompiliert und in die JAR-Datei `employee.jar` gepackt, die als Grundlage für einen BeanConnect Client verwendet werden kann. Ant rufen Sie mit dem Skript `runAnt` auf.

Weitere Einzelheiten hierzu finden Sie in Abschnitt „Java-Klassen auf Unix-, Linux- oder Windows-Systemen generieren“ auf Seite 623.

Konfiguration definieren

So bearbeiten Sie die Parameter-Datei `cobol2java.properties` für Ant:

```
xml.file=sample/employee.xml
cobol.struct=EMPLOYEE-RECORD
package.name=de.siemens.cob2java.test
doc.dir=doc
jar.dest=employee.jar
code.convention=java
```

PATH Variable einstellen

Damit sich das Skript `runAnt` ausführen lässt, müssen die Java-Programme `javac` und `javadoc` aufgerufen werden können.

Fügen Sie Ihrer Umgebungsvariable `PATH` das Programmverzeichnis des Java-SDK hinzu.

Generierung starten

1. Wechseln Sie in das Home-Verzeichnis von Cobol2Java.
2. Rufen Sie `runAnt.sh` (Unix-/Linux-Systeme) oder `runAnt.bat` (Windows-Systeme) auf.

Während der Generierung werden die Java-Sourcen im Verzeichnis `src` erstellt, kompiliert und in der JAR-Datei `employee.jar` gespeichert. Die JavaDoc der generierten Klassen befindet sich im Verzeichnis `doc`.

14.4.4 Verwendung der generierten Klassen

Auf der Grundlage der generierten Klassen können Anwendungen erstellt werden. Die nachfolgende Klasse zeigt das Beispiel eines BeanConnect-Clients, dem die Klasse `EmployeeRecord` zugrunde liegt.

```
package net.fsc.jca.BeanConnect.qa;

import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

import de.siemens.cob2java.cobtypes.*; // Runtime of Cobol2Java
import de.siemens.cob2java.test.EmployeeRecord

public class EmployeeServiceBean implements SessionBean
{
   private net.fsc.jca.communication.EISConnectionFactory cf;
   public void ejbCreate() throws javax.ejb.CreateException
   {
      try {
          javax.naming.Context ic = new
```

```
        javax.naming.InitialContext();
        cf =(net.fsc.jca.communication.EISConnectionFactory)
             ic.lookup
        ("java:comp/env/eis/myEIS");
    } catch (javax.naming.NamingException ex) {
       throw new javax.ejb.CreateException
       ("NamingException:"+ex);
    }
}

public void ejbActivate()
{
}

public void ejbPassivate()
{
}

public void ejbRemove()
{
}

public void setSessionContext(SessionContext ctx)
{
}

public String addSalary(String employeeNr, int
                        salaryIncrease)
{
   String retValue = "";
   net.fsc.jca.communication.EISConnection con = null;
  try {
     con = cf.getConnection();
     con.setServiceName("EMPLOYEE");

     // Create EmployeeRecord and accept the encoding setting
     // of the connection
     EmployeeRecord employee = new EmployeeRecord();
     try
     {   employee.setEncoding( con.getEncoding() );  }
     catch (net.fsc.beanta.encoding.EncoderException encEx) {
         // todo  Error handling
     } // catch EncoderException
     employee.setEncodingActive( con.isEncodingActive() );

     // Fetch the required EmployeeData
     employee.setEmployeeNumber( employeeNr );
     con.sndRecord( employee );
     con.rcvRecord( employee );
```

```
            // Increase the salary and send modification
            int oldSalary =employee.getSalaryInfo().
                           getAnnualSalary().intValue();
            employee.getSalaryInfo().setAnnualSalary( oldSalary+
                           salaryIncrease );
            con.sndRecord( employee );
            con.rcvRecord( employee );

            retValue="Salary for "+employee.getLastName()+"
                     increased from "+oldSalary +" to "+
                     employee.getSalaryInfo().getAnnualSalary();

            con.close();

         } // try
         catch (net.fsc.jca.communication.EISConnectionException
                eisEx) {
            if (con != null) {
               try {    con.close();
               } catch(Throwable thr) { }
            }
            throw new javax.ejb.EJBException
                                 ("EISConnectionException:"+eisEx);
         } // catch EISConnectionException
         catch (de.siemens.cob2java.cobtypes.
                Cob2JavaException cobEx) {
            if (con != null) {
               try {    con.close();
               } catch(Throwable thr) { }
            }
            throw new javax.ejb.EJBException("Cob2JavaException:
                                             "+cobEx);
         } // catch Cob2JavaException
         return retValue;
      } // addSalary
}
```

14.5 Fehlermeldungen und Fehlerbehandlung

Die folgende Tabelle liefert eine Übersicht über die Fehlermeldungen, die das Tool Cobol2Java ausgeben kann:

Nr.	Fehler	Fehlerbehebung
1	`No compatible XSLT Processor found.`	Prüfen Sie, ob alle benötigten JAR-Dateien im Cobol2Java-Verzeichnis `lib` vorhanden sind.
2	`TransformerFactory-ConfigurationError`	Siehe 1.
3	`Could not create file <name>`	Bitte vergewissern Sie sich, dass Sie die nötige Datenzugriffs-Authentisierung für das Datenmedium besitzen.
4	`No Record/Field <name> found in specified XML document`	Der Name der Datenstruktur ist nicht korrekt. Bitte prüfen.
5	`WARNING! Multiple occurrence of <name>`	Das Dokument enthält mehrere Strukturen mit dem angegebenen Namen. In diesem Fall erfolgt keine Generierung.

Taucht ein Datenelement auf, das nicht von Cobol2Java unterstützt wird, so wird ein generisches Datenelement vom Typ `PicU` generiert. Entwickler können mit Hilfe der Methode `getBytes/setBytes` auf Informationen zugreifen, die dieses Element betreffen.

Die Fehlerbehandlung in den Cobol2Java-Klassen basiert auf COBOL und ist äußerst fehlertolerant. Eingabe-Daten, die für das Feld, in dem sie gespeichert werden sollen, zu lang sind, werden entsprechend der Länge des Zielfeldes abgeschnitten. Ungültige Eingaben, wie z.B. eine ungültige Zahl, haben die Erzeugung einer `NumberFormatException` zur Folge.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

ACID-Eigenschaften
ACID properties
: Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Advanced program-to-program communication (APPC)
advanced program-to-program communication
: Anderer Name für das *LU6.2-Protokoll* und die zugrunde liegende Architektur.

APPC
: Siehe *Advanced program-to-program communication (APPC)*.

Appender
appender
: *Log4j*-Meldungsziel. Die an einen *Logger* übergebenen Logging-Meldungen werden durch den oder die Appender ausgegeben, die dem Logger zugeordnet worden sind.

Application Entity
application entity
: Repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen globalen (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title*. Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Qualifier (AEQ)
application entity qualifier
: Identifiziert entsprechend dem OSI-Standard einen Dienstzugriffspunkt innerhalb der Anwendung.

Application Entity Title
application entity title
: global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title (APT)* des jeweiligen *Application Process* und dem *Application Entity Qualifier (AEQ)*.

Application Process
application process
Repräsentiert im *OSI-Referenzmodell* eine Anwendung. Ein Application Process wird durch den *Application Process Title (APT)* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (APT)
application process title
: Wird entsprechend dem OSI-Standard verwendet, um eine Anwendung global (d.h. weltweit) zu identifizieren.

Application Server
application server
: Ein Application Server (Java EE Server) ist die Basis-Komponente einer *EJB*-Architektur. Der Application Server stellt den EJB-Clients die Dienste der Enterprise-Anwendungen zur Verfügung.

Application Server Cluster
application server cluster
: Cluster, der aus mehreren *Application Server*n besteht. Damit können n Instanzen des Application Servers m Proxy Instanzen zugeordnet werden.

Association (OSI)
association
: Kommunikationsbeziehung zwischen zwei *Application Entity*s.

Asynchron-Auftrag
asynchronous job
: Auftrag, der vom Auftragnehmer später ausgeführt wird. Die Verarbeitung erfolgt zeitlich entkoppelt vom Auftraggeber.

Asynchron-Vorgang
asynchronous service
: *Service* in openUTM, der einen Hintergrundauftrag bearbeitet.

Auftraggeber-Vorgang
job-submitting service
: *Service*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (Auftragnehmer-Vorgang) anfordert.

Auftragnehmer-Vorgang
job-receiving service
: *Service*, der von einem Auftraggeber-Vorgang einer anderen Server-Anwendung gestartet wird.

Authentisierung
authentication
: Siehe *Zugangskontrolle*.

Autorisierung
authorization
: Siehe *Zugriffskontrolle*.

Basic Communication Access Method (BCAM)
: Basis des Datenkommunikationssystems für BS2000-Systeme oder im *BeanConnect Proxy-Container.*

Basic conversation
basic conversation
: Typ einer *APPC-Conversation*, bei dem die *CICS*-Anwendung für die Übermittlung zum Partner Steuer-Bytes zu den Anwendungsdaten hinzufügen muss.

BCAM
: Siehe *Basic Communication Access Method (BCAM)*.

BeanConnect Management Console
BeanConnect Management Console
: Die BeanConnect Management Console dient zur Verwaltung und Konfiguration der BeanConnect-Komponenten. Die BeanConnect Management Console bietet eine grafische Bedienoberfläche und ein Command Line Interface.

BeanConnect Proxy
BeanConnect Proxy
: Ein BeanConnect Proxy ist die BeanConnect-Komponente, die einerseits mit dem *Resource Adapter* innerhalb des *Application Server*s und andererseits mit dem *EIS* kommuniziert.
Ein BeanConnect Proxy besteht aus dem *Proxy-Container*, der auf dem Transaktionsmonitor openUTM basiert.
Bei Kommunikation mit einem EIS Partner vom Typ CICS besteht der BeanConnect Proxy zusätzlich aus einem *openUTM-LU62 Gateway* sowie dem *Communication Service*.

BeanConnect Proxy-Container
BeanConnect Proxy Container

Der Proxy-Container ist das Kernstück des *BeanConnect Proxy*. Er beinhaltet die Definitionen aller Objekte, die für *Outbound-Kommunikation* und *Inbound-Kommunikation* sowie für die Konfiguration der Kommunikationspartner (*Resource Adapter* und *EIS* Partner) notwendig sind.

BeanConnect Resource Adapter
BeanConnect Resource Adapter

Der BeanConnect Resource Adapter ist die BeanConnect-Komponente, die auf dem Application Server läuft und der die Schnittstellen gemäß JCA implementiert. Bei *Outbound-Kommunikation* über das *OSI TP* Protokoll und bei *Inbound-Kommunikation* wird zusätzlich der *BeanConnect Proxy* für die Verbindung zwischen EIS und Resource Adapter benötigt.

Benutzerkennung
user ID

Bezeichner für einen Benutzer, der in der Konfiguration der openUTM-/ CICS-Anwendung festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Client muss bei der Anmeldung an die *openUTM-/CICS*-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben.

CCI

Siehe *Common Client Interface (CCI)*.

CICS

Siehe *Customer Information Control System (CICS)*.

Cluster

Cluster bezeichnet eine Anzahl von vernetzten Rechnern, die von außen in vielen Fällen als ein Rechner gesehen werden können. In der Regel sind die einzelnen Elemente eines Clusters untereinander über ein schnelles Netzwerk verbunden. Ziel der Cluster-Bildung ist es, die Rechenkapazität und/oder die Verfügbarkeit zu erhöhen.
Siehe auch *Application Server Cluster* und *Proxy Cluster*.

CMX

Siehe *Communications Manager for Unix Systems (CMX)*.

Common Client Interface (CCI)
common client interface

CCI ist Bestandteil der *Java EE Connector-Architektur (JCA)*. CCI bietet eine *EIS*-unabhängige Client-Schnittstelle für den Zugriff auf ein EIS.

Communication Service

Service, den ein *BeanConnect Proxy* für die Kommunikation zwischen Proxy-Container und CICS-Partner benötigt. Der Communication Service arbeitet direkt mit dem *openUTM-LU62 Gateway* zusammen und wird abhängig von der Plattform durch den *IBM Communications Server* oder durch *SNAP-IX* erbracht. SNAP-IX bzw. der IBM Communications Server sind Software-Voraussetzungen von BeanConnect, falls mit CICS-Partnern kommuniziert wird. Sie werden jedoch nicht mit BeanConnect ausgeliefert.

Communications Manager for Unix Systems (CMX)

Grundlage für Kommunikations-Software, die auf Solaris- Betriebssystemen läuft. Zur Nutzung ist eine Lizenz notwendig.

Connection Factory

connection factory

Eine Connection Factory wird von einem *EJB* eines *Application Server*s benötigt, um eine Verbindung zu einer externen Datenquelle (*EIS*) aufzubauen. Die Connection Factory wird beim Start des Application Servers im JNDI-Namensverzeichnis zur Verfügung gestellt.

Connection Pooling

connection pooling

Connection Pooling verwaltet „teure“ Verbindungen, deren Auf- und Abbau hohen Ressourceneinsatz erfordert. Connection Pooling wird eingesetzt, um die Skalierbarkeit und Performance in einer Anwendungs-Umgebung zu verbessern.

Connector-Architektur

connector architecture

Siehe *Java EE Connector-Architektur (JCA)*.

container.properties

Konfigurationsdatei eines *BeanConnect Proxy*s. In dieser Datei werden viele Änderungen hinterlegt, die Sie mit Hilfe der *BeanConnect Management Console* durchführen.

Contention Winner / Contention Loser

contention winner / contention loser

Jede Verbindung zwischen zwei Partnern wird von einem der Partner verwaltet. Dieser Partner wird als Contention Winner bezeichnet, der andere als Contention Loser. Beide Partner können Aufträge initiieren. Wenn beide Partner gleichzeitig einen Auftrag abschicken, hat der Contention Winner die höhere Priorität.

Control Point (CP)
control point

Der Control Point ist verantwortlich für die Verwaltung des Endknotens und seiner Ressourcen in einem APPN-Netz (advanced peer-to-peer-networking).

Conversation
conversation

Logische Verbindung zwischen zwei *Transaktion*sprogrammen, die über eine LU6.2-*Session* kommunizieren. Eine Conversation beginnt mit dem Allocate und endet mit dem Deallocate. Eine Conversation belegt über ihre gesamte Lebensdauer eine Session und sperrt sie damit für andere Benutzer.

CP

Siehe *Control Point (CP)*.

Customer Information Control System (CICS)

Transaktionsmonitor von IBM. CICS gibt es auf verschiedenen Plattformen, z.B. CICS/ESA für das Betriebssystem z/OS. Eine OLTP-Anwendung, die auf CICS basiert, wird als CICS-Anwendung bezeichnet.

Data Link Control (DLC)
data link control

Data Link Control ist ein Dienst, der durch die Sicherungsschicht (Data Link Layer) zur Verfügung gestellt wird. Die Sicherungsschicht entspricht der Schicht 2 des OSI-Referenzmodells für die Kommunikation in Netzwerken (z.B. LAN, Enterprise Extender).

Deployment Descriptor
deployment descriptor

Datei im XML-Format, die für das Deployment von *EJB*s oder Resource Adaptern benötigt wird. Ein Deployment Descriptor stellt Konfigurations-Informationen zur Verfügung, welche nicht im Code der EJB oder des Resource Adapters stehen.

Dialogschritt
dialog step

Ein Dialogschritt beginnt mit dem Empfang einer Dialog-*Nachricht* durch die *openUTM-Anwendung*. Er endet mit der Antwort der openUTM-Anwendung.

Dialog-Service
dialog service
Service, der einen Auftrag im Dialog (zeitlich gekoppelt) mit dem Auftraggeber bearbeitet. Ein Dialog-Service verarbeitet Dialog-*Nachricht*en vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Service besteht aus mindestens einer *Transaktion*. Bei openUTM umfasst ein Dialog-Service in der Regel mindestens einen *Dialogschritt*.

Dienstzugriffspunkt
service access point
Im OSI-Referenzmodell stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen Selektor identifiziert. Bei der Kommunikation bindet sich die *openUTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Program Link (DPL)
distributed program link
DPL ist eine Programmschnittstelle. Es ermöglicht einem *CICS*-Programm, ein anderes CICS-Programm aufzurufen, das auf einem entfernten CICS-System liegen kann. DPL wirkt genauso wie der Aufruf eines Unterprogramms.

Distributed Transaction Processing (DTP)
distributed transaction processing
bei openUTM-Partnern:
*Transaktion*s-orientierte verteilte Verarbeitung mit globalen Transaktionen. Verteilte Verarbeitung bedeutet, dass Aufträge von mehreren verschiedenen Anwendungen bearbeitet werden.
bei CICS-Partnern:
Distributed Transaction Processing ist eine Programmierschnittstelle, die es einer *CICS-Transaktion* ermöglicht, eine andere CICS-Transaktion aufzurufen (ggf. auch in einem anderen CICS-System). DTP unterstützt das Arbeiten mit globalen Transaktionen. DTP entspricht dem Client-Server Programmiermodell.

DLC
Siehe *Data Link Control (DLC)*.

DPL
Siehe *Distributed Program Link (DPL)*.

DTP
Siehe *Distributed Transaction Processing (DTP)*.

Einschritt-Transaktion
single-step transaction
: *Transaktion*, die genau einen *Dialogschritt* umfasst.

EIS
: Siehe *Enterprise Information System (EIS)*.

EJB
: Siehe *Enterprise JavaBeans (EJB)*.

EJB-Container
EJB container
: Ablaufumgebung für *EJB*-Komponenten. Er ist in einen *Application Server* eingebettet.

Enterprise Information System (EIS)
: Bezeichnet externe Datenquellen wie z.B. ERP-Systeme (Enterprise Resource Planning Systems, z.B. SAP), OLTP-Anwendungen wie *openUTM-/CICS*-Anwendungen, oder Datenbanksysteme wie Oracle DB, SESAM, UDS.
Ein EIS, das mit dem *Application Server* mittels BeanConnect kommuniziert, wird als EIS Partner bezeichnet.

Enterprise JavaBeans (EJB)
: Enterprise JavaBeans™ (EJB) ist eine Komponententechnologie, die die Entwicklung von plattformübergreifenden, Multitier- und verteilten Server-Anwendungen ermöglicht.

Functional Unit Commit
functional unit commit
: Funktionsgruppe im OSI TP Protokoll, die zum Bilden von verteilten *Transaktion*en erforderlich ist. Ob die Functional Unit Commit verwendet werden darf, wird beim Aufbau einer Association zwischen den beiden Partnern ausgehandelt. Auf einer Association, für die die Functional Unit Commit vereinbart wurde, können OSI TP Dialoge mit oder ohne Functional Unit Commit laufen. Eine Association ist eine Kommunikations-Verbindung zwischen zwei Anwendungen.

Functional Unit Handshake
functional unit handshake
: Funktionsgruppe im OSI TP Protokoll, die von den Kommunikationspartnern genutzt werden kann, um die Verarbeitung eines Dialogs auf Anwendungsebene zu koordinieren. Diese Funktion ermöglicht es, Bestätigungen aus der

Verarbeitung anzufordern und positive oder negative Bestätigungen zu senden. Mit dieser Funktion ist kein anwendungsübergreifendes Transaktions-Management verbunden.

Globale Transaktion
global transaction

Transaktion, die sich über mehr als eine Anwendung erstreckt.

IBM Communications Server

Der IBM Communications Server ist ein IBM-Produkt, das Anwendungen in SNA-Netzen mit Anwendungen in TCP/IP-Netzen verbindet.
Der IBM Communications Server wird in BeanConnect auf Linux- und Windows-Systemen benötigt, um die Kommunikation eines *BeanConnect Proxy* mit einer *CICS*-Partneranwendung über das *LU6.2-Protokoll* zu ermöglichen.

IDE

Integrated Development Environment. Bei BeanConnect ist die Open Source Entwicklungsumgebung NetBeans IDE gmeint.

Inbound-Kommunikation
inbound communication

Kommunikation vom *EIS* zum Java EE *Application Server*.

Inbound Message Endpoint
inbound message endpoint

Endpunkt der *Inbound-Kommunikation* im Java EE *Application Server*.
Für jeden Inbound Message Endpoint im Application Server muss mit Hilfe der *BeanConnect Management Console* ein gleichnamiger Inbound Message Endpoint im
BeanConnect Proxy konfiguriert werden. Ein BeanConnect Proxy kann mehrere Inbound Message Endpoints haben.

Inbound Service
inbound service

Inbound Services stellen die Objekte dar, die von den EIS Partnern bei der *Inbound-Kommunikation* adressiert werden. Die Inbound Services, die in der Management Console bekannt sind, werden implizit über die Inbound Message Endpoints definiert.
Jedem Inbound Service ist genau ein Inbound Message Endpoint zugeordnet.

Inbound User
inbound user

Benutzername und Passwort, die bei der *Inbound-Kommunikation* vom EIS an den
BeanConnect Proxy übergeben werden können.

J2EE®

Ältere Bezeichnung für *Java EE*.

Java EE

Java Platform, Enterprise Edition, abgekürzt Java EE oder früher J2EE, ist die Spezifikation einer Softwarearchitektur für die transaktionsbasierte Ausführung von in Java programmierten Anwendungen.

Java EE Connector-Architektur (JCA)
Java EE Connector architecture

Definiert eine Standard-Architektur für die Verbindung der Java EE Plattform mit heterogenen Enterprise Information Systems.

Java Development Kit (JDK)

Standard-Entwicklungsumgebung von Sun Microsystems für die Entwicklung von Java-Anwendungen.

Java Naming and Directory Interface (JNDI)

Standard Java-Extension, die ein einheitliches API für den Zugriff auf Directory- und
Namensdienste verschiedener Hersteller zur Verfügung stellt.

JCA

Siehe *Java EE Connector-Architektur (JCA)*.

JDK

Siehe *Java Development Kit (JDK)*.

JMX

Java Management Extensions (JMX) ist eine vom Java Community Process (JSR-3) entwickelte Spezifikation zur Verwaltung und Überwachung von Java-Anwendungen.

JMX-Client

Client, der auf einen *JMX* Server zugreifen und dessen Dienste nutzen kann. Bei BeanConnect ist die Management Console die Implementierung eines JMX Clients.

JMX-Server

Instanz in einer Java Anwendung, welche Dienste zur Überwachung der Anwendung zur Verfügung stellt. In einer BeanConnect-Umgebung implementiert der Application Server die JMX Server Funktionalität.

JNDI-Name

Name eines Java-Objekts in einer Programmierumgebung, in der das *Java Naming and Directory Interface (JNDI)* verwendet wird.

Jython

Jython (früher JPython) ist eine reine Java-Implementierung der Programmiersprache Python und ermöglicht somit die Ausführung von Python-Programmen auf jeder Java-Plattform. Python ist eine universelle, üblicherweise interpretierte höhere Programmiersprache.

KDCA

Standard-Name der *KDCFILE*.

KDCDEF

openUTM Generierungs-Tool bzw. Generierungs-Tool des *BeanConnect Proxy-Container*s.

KDCFILE

Konfigurationsdatei eines *BeanConnect Proxy-Container*. Die Datei enthält die Daten, die die Container-Anwendung für den Ablauf benötigt. Die KDCFILE wird mit dem Generierungstool *KDCDEF* erstellt.
bei openUTM-Partnern:
Konfigurationsdatei einer openUTM-Partneranwendung. Die Datei enthält die Daten, die die openUTM-Partneranwendung für den Ablauf benötigt. Die KDCFILE wird mit dem openUTM-Generierungstool *KDCDEF* erstellt.

KDCS

Universelle openUTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen. Außerdem stellt KDCS Aufrufe zur verteilten Verarbeitung zur Verfügung. BeanConnect verwendet KDCS nur als interne Schnittstelle.

Komponente
component

Wiederverwendbare Software-Einheit mit genormten Schnittstellen, die in der Regel in einer Entwicklungsumgebung manipuliert werden kann.

Konfigurations-Property
configuration property

Wird für die Konfiguration eines *Resource Adapter*s verwendet. Eine Konfigurations-Property wird in einer Konfigurationsdatei mit Hilfe von XML-Tags gesetzt.

Log4j

BeanConnect verwendet für die Trace- und Logging-Funktionalität das Softwareprodukt Log4j. Log4j ist ein Bestandteil des Apache Jakarta Projekts. Log4j bietet Schnittstellen zum Protokollieren von Informationen (Ablauf-Informationen, Trace-Records,...) und zum Konfigurieren der Protokoll-Ausgabe. Log4j wird mit Hilfe der *BeanConnect Management Console* konfiguriert.

Logger
logger

Log4j-Meldungsquelle. Ein Programm, das Protokoll-Informationen schreiben soll, holt sich bei Log4j Logger-Objekte mit vorgegebenen Namen und gibt seine Meldungen über diese Objekte aus. Dabei ist es für das Programm transparent, wohin Log4j die übergebenen Meldungen ausgibt.

Logical Unit (LU)
logical unit

Logischer virtueller Port, der einem Benutzer den Zugriff zu den Netzwerkdiensten in einem SNA-Netz ermöglicht. Die logische Einheit korrespondiert mit dem *Control Point (CP)* und einer Partner-LU, die z.B. ein Anwender-Programm repräsentiert.

LU

siehe *Logical Unit (LU)*.

LU6.2-Protokoll
LU6.2 protocol

Das LU6.2-Protokoll ist Bestandteil der IBM-Netzwerkarchitektur. LU6.2 definiert Methoden für eine Programm-Programm-Kommunikation zwischen Anwendungen in verschiedenen Rechnern.

Mapped Conversation
mapped conversation

Typ einer *APPC-Conversation*, bei dem nur Benutzerdaten zu/von einer anderen APPC-Anwendung übergeben werden. Der Benutzer muss sich nicht um die internen Datenformate kümmern, die aufgrund der Architektur erforderlich sind.

Mapped Hostname
mapped host name
: UTM-Hostname der Partner-Anwendung. Der UTM-Hostname wird auf offenen Plattformen über die UTM-Hostname-Datei und im BS2000-System mit BS2000-internen Mechanismen auf einen realen Hostnamen abgebildet wird.

MBean
: Managed Bean, welche eine Ressource eines JMX Servers repräsentiert.

MC-CLI
MC-CLI
: Command Line Interface der Management Console.

MC-CLI Recording
MC-CLI recording
: Mitschneiden von Aktionen der Management Console. Die Mitschnitte werden in Form von *MC-CLI*-Funktionen erzeugt.

MC-CmdHandler
MC-CmdHandler
: Die BeanConnect-Komponente MC-CmdHandler wird benötigt, um eine andere, entfernte, BeanConnect-Komponente mit der *BeanConnect Management Console* zu administrieren, wobei der Funktionsumfang derselbe ist wie beim Administrieren einer lokalen Komponente.

Management Console Command Line Interface (MC-CLI)
Management Console command line interface (MC-CLI)
: Paket von *Jython*-Funktionen, mit dem Funktionen der *BeanConnect Management Console* aus einem Jython-Skript heraus gestartet werden können.

Message-Driven Bean
message-driven bean
: Bestandteil der Spezifikation „Enterprise JavaBeans“von Sun Microsystems. Message-Driven Beans sind Beans für den Empfang von *Nachricht*en. Message-Driven Beans werden nur vom *EJB-Container* aufgerufen und besitzen deshalb keine Home-Interfaces und Remote-Interfaces.

Message Endpoint
message endpoint
: Ein Message Endpoint ist eine in einem Application Server deployte Message-Driven Bean Anwendung.
The message endpoint is a message-driven bean application which is to be deployed on the application server.

Message Listener
message listener
: Message Consumer. Dem Message Listener-Objekt werden *Nachricht*en zugestellt, sobald sie verfügbar sind. Message-Driven Beans sind Message Listener.

Message Listener Interface
message listener interface
: Interface, das ein Message Listener implementieren muss. Inbound Resource Adapter stellen spezifische Message Listener Interfaces zu Verfügung, die ein Message Listener implementieren muss, wenn er Nachrichten von diesem Resource Adapter konsumieren will.

Mehrschritt-Transaktion
multi-step transaction
: *Transaktion*, die aus mehr als einem *Dialogschritt* besteht.

Mode-Name
mode name
: Symbolischer Name für eine Liste von *Session*-Eigenschaften. Der Mode Name ist bei allen Kopplungen in einem SNA-Netz nötig, er wird vom Initiator einer Sitzung benutzt.

Multi-Resource Adapter Betrieb
multi-resource adapter mode
: Konfiguration, in der ein *BeanConnect Proxy* mit mehreren *Resource Adapter*n zusammenarbeitet. Diese Resource Adapter können sich auf unterschiedlichen Application Servern befinden.

Nachricht
message
: Eine Nachricht ist ein Datenpaket, das aus einem Header und einem Body besteht. Der Header enthält Daten für die Adressierung, das Network-Routing und ggf. Daten über das Nachrichtenformat. Der Body enthält die eigentliche Nachricht in Form von Geschäftsdaten oder Systemmeldungen.

Naming
naming
: Abbildung von Namen auf Objektreferenzen. Die Abbildung erfolgt i.A. über einen Namensdienst.

Netzwerkname
network name
Name, der ein SNA-Netz identifiziert. Der Netzwerkname ist Bestandteil des LU-Namens, siehe *Logical Unit (LU)*.

OLTP Message-Driven Bean
OLTP message-driven bean
OLTP Message-Driven Beans sind *EJB*s, die Aufträge von OLTP-Anwendungen (openUTM/CICS-Anwendungen) empfangen und bearbeiten. Die OLTP-Anwendung adressiert die OLTP Message-Driven Bean über einen *Inbound Message Endpoint*.

openUTM
Transaktionsmonitor von Fujitsu Technology Solutions und Basis-Komponente des *BeanConnect Proxy-Container*s.

openUTM-Anwendung
openUTM application
OLTP-Anwendung, die auf dem Transaktionsmonitor openUTM von Fujitsu Technology Solutions basiert.

openUTM-LU62 Gateway
openUTM-LU62 gateway
openUTM-LU62 ist eine *BeanConnect Proxy*-Komponente, die die Kopplung mit Partneranwendungen ermöglicht, die das SNA-Protokoll LU6.2 unterstützen, insbesondere mit *CICS*-Anwendungen.

openUTM-Socket-Protokoll (USP)
openUTM socket protocol
Von openUTM-Partneranwendungen verwendetes Protokoll zum Umsetzen von Bytestreams in *Nachricht*en. USP setzt als Transportsystem TCP/IP voraus.

Oracle WebLogic Server
Java EE 6 konformer Application Server der Oracle Corporation.

OSI-LPAP-Partner
OSI-LPAP partner
OSI-LPAP-Partner sind die beim *BeanConnect Proxy-Container* generierten Adressen der OSI TP Partner. Für die verteilte Verarbeitung über das Protokoll *OSI TP* muss im Proxy-Container für jede Partneranwendung ein OSI-LPAP-Partner konfiguriert werden. Bei openUTM-Partnern spiegelt der OSI-LPAP-Partner im Proxy-Container die Partneranwendung und bei CICS-Partnern die openUTM-LU62 Gateway-Instanz wider. Bei der Kommunikation wird die

Partneranwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Open **S**ystem **I**nterconnection **T**ransaction **P**rocessing.
Von der ISO definiertes Kommunikationsprotokoll für die Verteilte Transaktionsverarbeitung.
Der Partner einer Anwendung, der mit dem *BeanConnect Proxy-Container* über das OSI TP Protokoll kommuniziert, wird als OSI TP Partner bezeichnet.
Bei openUTM-Partnern ist dies der *EIS* Partner, bei CICS-Partnern das *openUTM-LU62 Gateway*.
Bei CICS wird dieses Protokoll bei der Kommunikation zwischen dem *BeanConnect Proxy-Container* und dem *openUTM-LU62 Gateway* verwendet.

OSS

OSI **S**ession **S**ervice
OSS bildet im *BeanConnect Proxy-Container* die Basis für die Datenkommunikation über OSI TP.

Outbound-Kommunikation
outbound communication

Kommunikation vom *Java EE Application Server* zum *EIS*.

Outbound Communication Endpoint
outbound communication endpoint

Symbolischer Name, der einen Service des Partner-*EIS* repräsentiert.

Outbound Service
outbound service

Beschreibt einen Service (Transaktionscode) innerhalb des *EIS* Partners für die *Outbound-Kommunikation*.

PCMX

Grundlage für Kommunikations-Software, die auf den Betriebssystemen Solaris, Linux und Windows läuft.

Physical Unit (PU)

physical unit

Jeder Knoten in einem SNA-Netz enthält als adressierbare SNA Instanz eine Physical Unit (PU). Bevor zwei Logical Units (LUs) im SNA-Netz eine Kommunikationsbeziehung aufbauen können, muss zuerst eine Kommunikationsbeziehung zwischen den jeweiligen PUs aufgebaut werden.

Proxy Cluster

proxy cluster

Cluster, der aus mehreren *BeanConnect Proxy*s besteht und über die *BeanConnect Management Console* verwaltet wird.

Proxy-Container

proxy container

Siehe *BeanConnect Proxy-Container*.

PU

Siehe *Physical Unit (PU)*.

Resource Adapter

resource adapter

Resource Adapter (auch Connectoren genannt) koppeln einen *Application Server* mit einem *EIS*, siehe auch *BeanConnect Resource Adapter*.

Resource Manager

resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

Schema

siehe *XML Schema*.

Service
service

Services bearbeiten die Aufträge, die an eine Server-Anwendung geschickt werden. Services können von Clients oder anderen Vorgängen angefordert werden. Ein Service in einer *openUTM-Anwendung* (Vorgang) bzw. *CICS*-Anwendung setzt sich aus einer oder mehreren *Transaktion*en zusammen. Die erste Transaktion wird über den Vorgangs-*TAC* bzw. den transaction program name aufgerufen.
Bei openUTM gibt es Dialog-Vorgänge und Asynchron-Vorgänge. openUTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung.

Session
session

Unter einer Session versteht man eine Kommunikationsbeziehung zwischen zwei *LU*s, allgemeiner zwischen zwei adressierbaren SNA Instanzen.

SNA

Siehe *Systems Network Architecture (SNA)*.

SNAP-IX

SNAP-IX ist ein Produkt von Data Connection, das Anwendungen in SNA-Netzen mit Anwendungen in TCP/IP-Netzen verbindet.
SNAP-IX wird in BeanConnect auf Solaris-Systemen benötigt, um die Kommunikation eines *BeanConnect Proxy* mit einer *CICS*-Partneranwendung über das *LU6.2-Protokoll* zu ermöglichen.

Synchronization level (Sync-Level)
synchronization level

Bezeichnung bei LU6.2, die die Transaktionssicherheit bei verteilter Verarbeitung charakterisiert:

- Bei Sync-Level 0 (None) dürfen nur Nettodaten und Fehlermeldungen gesendet werden. Quittungen sind nicht zulässig.
- Bei Sync-Level 1 (Confirm) sind neben Nettodaten und Fehlermeldungen auch einfache Quittungen zulässig.
- Bei Sync-Level 2 (Syncpoint) ist die volle Transaktionssicherheit für verteilte Transaktionen eingeschaltet.

Synchronization point (Sync-Point)
synchronization point
: Logischer Punkt innerhalb des Ablaufs einer verteilten Verarbeitung, an der die gemeinsamen Ressourcen in einen definierten Zustand gebracht werden. Bei openUTM verwendet man statt dessen den Begriff „Transaktionsende“.

Systems Network Architecture (SNA)
: SNA ist die Bezeichnung für eine Reihe von Kommunikationsprotokollen, die von IBM definiert wurden.

TAC
: Siehe *Transaktionscode (TAC)*.

Transaktion
transaction
: Verarbeitungsabschnitt innerhalb eines Services, der die *ACID-Eigenschaften* aufweist. Von den in einer Transaktion beabsichtigten Änderungen der Anwendungsinformation werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt
(Alles-oder-Nichts Regel). Das Transaktionsende bildet einen Sicherungspunkt (siehe *Synchronization point (Sync-Point)*).

Transaktionscode (TAC)
transaction code
: Name, über den ein Service der *openUTM-Anwendung* aufgerufen werden kann.

USP
: Siehe *openUTM-Socket-Protokoll (USP)*.

UTM
: Siehe *openUTM*.

UTM-Anwendung
UTM application
: Siehe *openUTM-Anwendung*.

Verteilte Transaktion
distributed transaction
: Siehe *Globale Transaktion*.

Vorgang
: Siehe *Service*.

Virtual Telecommunications Access Method (VTAM)

Komponente in einem IBM-Hostsystem, die für die Datenfernverarbeitung zuständig ist.

Web Service Description Language (WSDL)

Die Web Services Description Language (WSDL) definiert eine plattform-, programmiersprachen- und protokollunabhängige XML-Spezifikation zur Beschreibung von Netzwerkdiensten (Web Services) zum Austausch von Nachrichten.

XML

XML (e**X**tensible **M**arkup **L**anguage) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

XML Schema

XML Schema ist eine Empfehlung des W3C zum Definieren von Strukturen für XML-Dokumente. Die Struktur wird in Form eines XML-Dokuments beschrieben. Darüber hinaus wird eine große Anzahl von Datentypen unterstützt.

Zugangskontrolle

system access control

Prüfung durch den *Application Server*, ob eine Benutzerkennung berechtigt ist, mit dem Application Server zu arbeiten.

Zugriffskontrolle

data access control

Prüfung durch den *Application Server*, ob ein Kommunikationspartner/Client berechtigt ist, auf eine bestimmte Business-Methode zuzugreifen. Die Zugriffsrechte werden als ein Bestandteil der Konfiguration festgelegt.

Zugriffspunkt

access point

Siehe *Dienstzugriffspunkt*.

Literatur

Die Handbücher finden Sie im Internet unter *http://manuals.ts.fujitsu.com*. Handbücher, die mit einer Bestellnummer angezeigt werden, können Sie auch in gedruckter Form bestellen.

PDF-Dateien von allen openUTM-Handbüchern sind sowohl auf der openUTM Enterprise Edition DVD für die offenen Plattformen als auch für BS2000-Systeme auf der openUTM WinAdmin-DVD enthalten.

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

openUTM
Anwendungen generieren
Benutzerhandbuch

openUTM
Einsatz von openUTM-Anwendungen unter Unix- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose in Unix- und Windows-Systemen
Benutzerhandbuch

openUTM
Einsatz von openUTM-Anwendungen unter BS2000-Systemen
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose in BS2000-Systemen
Benutzerhandbuch

openUTM
XML für openUTM

openUTM-Client
für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM WinAdmin
Grafischer Administrationsarbeitsplatz für openUTM
Beschreibung und Online-Hilfe

openUTM WebAdmin
Web-Oberfläche zur Administration von openUTM
Beschreibung und Online-Hilfe

openUTM, **openUTM-LU62**
Verteilte Transaktionsverarbeitung
zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

WS4UTM (Unix- und Windows-Systeme**)**
Web-Services für openUTM

openUTM
Masterindex

Stichwörter

C

L

P

T