

WebTransactions V7.5

Anschluss an openUTM-Anwendungen über UPIC

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

Copyright © Fujitsu Technology Solutions GmbH 2010.

Alle Rechte vorbehalten.

Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

Inhalt

1	Einleitung	7
1.1	Charakterisierung des Produkts	7
1.2	Architektur von WebTransactions for openUTM	9
1.3	Dokumentation zu WebTransactions	11
1.4	Konzept und Zielgruppe dieses Handbuchs	14
1.5	Neue Funktionen	14
1.6	Darstellungsmittel	15
2	WebTransactions installieren	17
2.1	Installation	17
2.1.1	Windows	18
2.1.1.1	Installation über die Bedienoberfläche	18
2.1.1.2	Bedienerlose Installation	19
2.1.2	Solaris	21
2.1.3	Linux	22
2.1.4	BS2000/OSD	23
2.1.5	Installation von WebLab	23
2.2	Lizenzierung	24
3	Beispielsitzung	25
3.1	WebTransactions-Server verwalten	25
3.1.1	Browser einstellen	26
3.1.2	Administrationsprogramm starten	27
3.1.3	Lizenzen eingeben	28
3.1.4	Benutzer anlegen	31
3.1.5	Pool anlegen	32

3.1.6	Pool dem Benutzer zuweisen	34
3.2	Host-Anwendung an das WWW anbinden	35
3.2.1	Projekt anlegen	35
3.2.1.1	Basisverzeichnis anlegen	36
3.2.2	Projekt speichern	39
3.2.3	Templates aus FHS-Formaten erzeugen	41
3.2.3.1	Mit IFG2FLD Beschreibungsdatei erzeugen	41
3.2.3.2	Aus Beschreibungsdatei Templates und Felddateien generieren	42
3.2.4	Lokale Host-Anwendungsnamen festlegen	45
3.2.5	Sitzung starten	46
3.3	Templates bearbeiten	52
3.3.1	Drop-Down-Liste zur Länderauswahl einfügen	53
3.3.2	Kommando-Eingabe durch Knöpfe ersetzen	59
3.3.3	Verweissensitive Grafik einfügen	61
3.4	WebTransactions starten	63
3.4.1	Start-Template erzeugen	63
3.4.2	Sitzung starten mit WebLab	67
3.4.3	Alternative Möglichkeiten zum Start einer WebTransactions-Anwendung	68
4	Basisverzeichnis anlegen	69
4.1	Basisverzeichnis anlegen mit WebLab	69
4.2	Struktur eines Basisverzeichnisses	71
5	Templates generieren	73
5.1	Templates aus FHS-Formaten generieren	74
5.1.1	Anwendung von IFG2FLD	75
5.1.2	Template und FLD-Datei mit WebLab aus der Formatbeschreibungsquelle generieren	77
5.2	Templates aus FORMANT-Formaten generieren	80
5.2.1	Template und FLD-Datei mit WebLab generieren	80
5.3	Aufbau der Felddateien (FLD-Dateien)	82
5.4	Aufbau der generierten Templates	87

6	Templates bearbeiten	95
6.1	Master-Template UTM.wmt	96
6.2	Templates gestalten	97
6.2.1	Globales Layout festlegen	97
6.2.2	Oberfläche individuell gestalten	99
6.2.3	Ablauf gestalten	99
6.3	Besonderheiten bei FHS/FORMANT-Teilformaten	100
6.3.1	Kommunikationsablauf	100
6.3.2	Aufbau des Master-Templates UTMpartial.wmt	102
6.3.3	Gestalten mit Teilformat-Templates	112
6.4	Unterstützung des openUTM-Zeilenmodus	115
6.5	Binärdaten-Unterstützung	120
7	Kopplung konfigurieren	121
7.1	WebTransactions und Host abstimmen	123
7.2	WebTransactions-Seite konfigurieren	128
7.2.1	Datei localapps	128
7.2.2	Adressierung der openUTM-Anwendung über Systemattribute	129
7.2.3	Datei upicfile	130
7.2.4	Server-Rechner bekannt machen	133
7.3	openUTM-Seite (Host) konfigurieren	134
7.3.1	openUTM-Generierung anpassen	134
7.3.2	Client-Rechner bekannt machen	137
7.4	BCMAP-Einträge (BS2000/OSD)	138
8	Kommunikation steuern	141
8.1	openUTM-spezifische Attribute des Systemobjekts	141
8.1.1	Übersicht	142
8.1.2	Zusammenspiel: Systemobjekt-Attribute und Aktionen/Methoden	149
8.2	Host-Objekte und Attribute	155
8.2.1	Host-Objekte für die einzelnen Format-Felder (Host-Datenobjekte)	155
8.2.2	Host-Steuerobjekt WT_HOST_MESSAGE	162
8.2.3	Host-Steuerobjekt WT_HOST_GLOBALS	166
8.2.4	Host-Steuerobjekte \$FIRST und \$NEXT	167

8.3	Unterstützung von Terminal-Funktionen durch den Browser	168
8.3.1	Unterstützte Terminal-Funktionen	168
8.3.2	Zuordnung der Tasten in wtKeysUTMFHS.js und wtKeysUTMFormant.js	171
8.3.3	Zusammenspiel von wtCommonBrowserFunctions.js und wt<browser>BrowserFunctions.js	176
8.3.4	Verwendung des Objekts WT_BROWSER	180
8.4	Start-Templates für openUTM	182
8.4.1	openUTM-spezifisches Start-Template des Start-Template-Sets (wtstartUTMV4.htm)	183
8.4.2	WTBean wtcStartUPIC.wtc zur Generierung eines Start-Templates	188
8.5	Neues openUTM-Kommunikationsobjekt anlegen (wtcUPIC)	190
8.6	Security durch openUTM-Benutzerkonzept	192
8.7	RESTART - Automatischer Wiederanlauf	194
8.8	BADTAC - Simulation des Event-Service BADTAC	197
8.9	Automatische Vorgangsverknüpfung	198
8.10	Simulation der Funktionstasten	199
8.11	Unterstützung von KDCSCUR	200
8.12	Gezieltes Anmelden über bestimmte LTERMs	201
	Fachwörter	203
	Abkürzungen	223
	Literatur	225
	Stichwörter	227

1 Einleitung

Bei den meisten IT-Anwendern ist über die Jahre hinweg eine heterogene System- und Anwendungslandschaft entstanden: Mainframes stehen neben Unix- und Windows-Systemen, PCs neben Terminals. Unterschiedliche Hardware, Betriebssysteme, Netze, Datenbanken und Anwendungen werden parallel betrieben. Auf den Mainframe-Systemen und auch auf Unix- oder Windows-Servern existieren oft komplexe und funktional mächtige Anwendungen. Sie sind meist mit erheblichen Investitionen entwickelt worden und stellen in der Regel zentrale Geschäftsprozesse dar, die nicht ohne weiteres durch neue Software ersetzt werden können.

Die Integration vorhandener heterogener Anwendungen in ein einheitliches und transparentes IT-Konzept ist die zentrale Herausforderung der modernen Informationstechnik. Flexibilität, Investitionsschutz und Offenheit für neue Technologien sind dabei von entscheidender Bedeutung.

1.1 Charakterisierung des Produkts

Mit dem Produkt WebTransactions bietet Fujitsu Technology Solutions einen best-of-breed Web-Integration-Server, mit dem eine breite Palette geschäftsrelevanter Anwendungen in kürzester Zeit Browser- und Portal-fähig gemacht werden können. WebTransactions ermöglicht einen schnellen und kostengünstigen Zugang über Standard-PCs und mobile Endgeräte wie Tablet PCs, PDAs (Personal Digital Assistant) und Mobile Phones.

WebTransactions deckt alle Facetten ab, die typischerweise in einem Web-Integrationsprojekt auftreten: von der automatischen Bereitstellung der ursprünglichen „Legacy Oberfläche“ über die grafische Aufbereitung und die Anpassung der Arbeitsabläufe bis hin zu einer umfassenden Frontend-Integration mehrerer Anwendungen. WebTransactions bietet eine hoch-skalierbare Laufzeitumgebung und eine komfortable grafische Entwicklungsumgebung.

Sie können in einer ersten Integrationsstufe folgende Anwendungen und Inhalte über WebTransactions in einer direkten Umsetzung an das WWW anbinden und so Ihren Nutzern intern und extern einfacher zur Verfügung stellen:

- Dialoganwendungen im BS2000/OSD
- MVS- bzw. z/OS-Anwendungen
- systemübergreifende Transaktionsanwendungen auf Basis von openUTM
- dynamische Web-Inhalte

Der Benutzer greift im Internet oder Intranet mit einem Web-Browser seiner Wahl auf die Host-Anwendung zu.

Durch Nutzung modernster Technologie bietet WebTransactions als zweite Integrationsstufe an, die - oftmals noch alphanumerische - Oberfläche der bestehenden Host-Anwendung durch eine attraktive grafische Oberfläche zu ersetzen oder zu ergänzen. Außerdem kann die Host-Anwendung mit WebTransactions auch funktional erweitert werden, ohne dass Eingriffe auf der Host-Seite erforderlich wären (Dialog-Reengineering).

In einer dritten Integrationsstufe können Sie unter der einheitlichen Oberfläche des Browsers unterschiedliche Host-Anwendungen miteinander verknüpfen. Dabei ist es möglich, beliebige vormals heterogene Host-Anwendungen, beispielsweise MVS- oder OSD-Anwendungen miteinander zu verknüpfen oder mit beliebigen dynamischen Web-Inhalten zu kombinieren. Welche Datenquelle ursprünglich die Daten liefert, ist für den Endnutzer nicht mehr sichtbar.

Zusätzlich können Sie den Leistungsumfang und die Funktionalität von WebTransactions-Anwendungen durch eigene Clients beliebig erweitern. Dazu stellt Ihnen WebTransactions ein offenes Protokoll und Schnittstellen (APIs) bereit.

Parallel zum Zugriff über WebTransactions kann weiterhin auch über „herkömmliche“ Terminals oder Clients auf die Host-Anwendungen oder dynamische Web-Inhalte zugegriffen werden. So können Sie eine Host-Anwendung schrittweise ans Web anschließen und die Wünsche und Bedürfnisse unterschiedlicher Nutzergruppen berücksichtigen.

1.2 Architektur von WebTransactions for openUTM

Folgende Abbildung zeigt die Architektur von WebTransactions for openUTM:

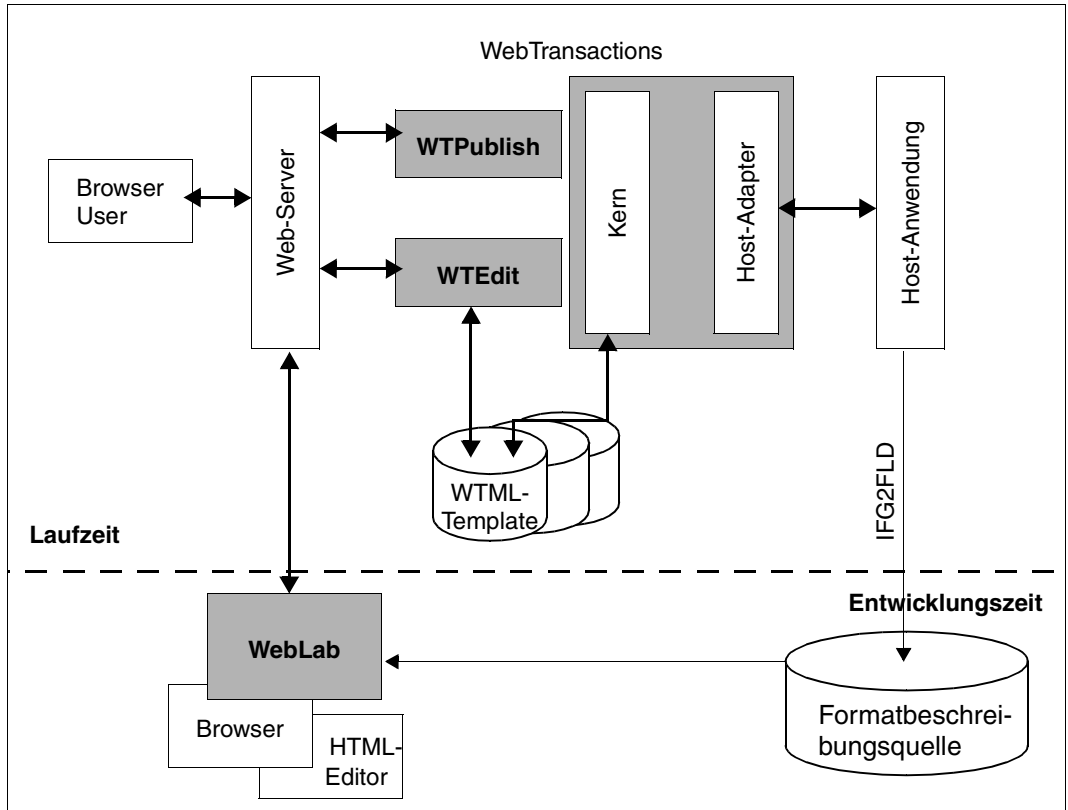


Bild 1: Architektur von WebTransactions for openUTM

Host-Adapter mit Integriertem UPIC-Protokoll

WebTransactions for openUTM benutzt zur Lauf- und zur Entwicklungszeit das UPIC-Protokoll, das in den Host-Adapter integriert ist und über das die Kommunikation zwischen dem Kern von WebTransactions und der Host-Anwendung abwickelt.

WebLab

WebLab ist die Entwicklungsumgebung von WebTransactions, mit der Sie alle Schritte von der Anbindung einer Host-Anwendung, der Erzeugung und Nachbearbeitung der format-spezifischen Templates bis zum Test der Anwendung ausführen können. Mit Hilfe von WebLab können Sie aus bestehenden FHS- und FORMANT-Formaten Templates generieren. Diese formatspezifischen Templates bearbeiten Sie mit WebLab nach.

WebLab muss nicht auf dem Rechner installiert sein, auf dem WebTransactions abläuft. Sie können WebLab auf einem anderen Rechner mit Windows-Betriebssystem benutzen. Alle, zum Ablauf einer WebTransactions-Anwendung benötigten, Daten werden auf dem Rechner verwaltet, auf dem WebTransactions abläuft.

Ablauf

Zur Laufzeit steuern die Templates die grafische Umsetzung und die Bearbeitung der Formate.

Unicode-Unterstützung

Der Host-Adapter in WebTransactions for openUTM kann Daten an der Schnittstelle UPIC auch als Unicode-Zeichen interpretieren.

Das BS2000/OSD-Programm `IFG2FLD` liest Formatbeschreibungen aus einer IFG-Bibliothek und legt sie in einer Formatbeschreibungsource ab. Die Felder in einer IFG-Bibliothek können das Attribut `Unicode` enthalten.

Aus der Formatbeschreibungsource können Sie mit WebLab Templates und Felddateien (FLD-Dateien) generieren. Bei dieser Umsetzung wird ab IFG2FLD Version 8.3 das Unicode-Kennzeichen automatisch berücksichtigt (siehe [Kapitel „Templates generieren“ auf Seite 73](#)).

Weitere Informationen dazu entnehmen Sie dem Abschnitt [„Unicode-Unterstützung“ auf Seite 122](#).

Bereits existierende Templates können unverändert bestehen bleiben, falls die Umstellung auf Unicode-Felder die einzige Änderung in einem Format war. Allerdings muss die Zuweisung des Wertes `UTF-8` auf das globale Systemobjekt-Attribut `CHARSET` eingefügt werden (siehe [Abschnitt „Host-Steuerobjekt WT_HOST_MESSAGE“ auf Seite 162](#)).

Welche Regeln Sie für neue Templates einhalten müssen, entnehmen Sie dem Abschnitt [„Unicode-Unterstützung“ auf Seite 122](#).

1.3 Dokumentation zu WebTransactions

Zusätzlich zum vorliegenden Handbuch enthält die Dokumentation zu WebTransactions folgende Einheiten:

- Ein einführendes Handbuch, das für alle Liefereinheiten gilt:

Konzepte und Funktionen

Das Handbuch beschreibt alle zentralen Konzepte von WebTransactions:

- die unterschiedlichen Einsatzmöglichkeiten von WebTransactions.
 - das Konzept von WebTransactions und die Bedeutung der Objekte in WebTransactions, ihre wesentlichen Eigenschaften und Methoden, ihr Zusammenspiel und ihre Lebensdauer.
 - den dynamischen Ablauf einer WebTransactions-Anwendung.
 - die Administration von WebTransactions.
 - die Entwicklungsumgebung WebLab.
- Ein Referenz-Handbuch, das für alle Liefereinheiten gilt und die WebTransactions Template-Sprache WTML beschreibt:

Template-Sprache

Nach einem Überblick über WTML finden Sie

- die lexikalischen Elemente, die in WTML verwendet werden.
- die klassenunabhängigen globalen Funktionen, wie z.B. `escape()` oder `eval()`.
- die eingebauten Klassen und Methoden, wie z.B. die Klassen `Array` oder `Boolean`.
- die WTML-Tags, die die WebTransactions-spezifischen Funktionen enthalten.
- die WTScrip-Anweisungen, die Sie in den WTScrip-Bereichen angeben können.
- die Klassen-Templates, mit denen Sie die Auswertung gleichartiger Objekte automatisieren können.
- die Master-Templates, die von WebTransactions als Schablone verwendet werden und für ein einheitliches Layout sorgen.
- eine Beschreibung der Java-Integration, mit der Sie eigene Java-Klassen in WebTransactions instanzieren und der Userexits, mit denen Sie eigene C/C++-Funktionen integrieren können.
- die mit WebTransactions fertig ausgelieferten UserExits.

- die XML-Konvertierung für die portable Darstellung von Daten für die Kommunikation mit externen Anwendungen über XML-Nachrichten und die Konvertierung von WTSript-Datenstrukturen in XML-Dokumente.
- Jeweils ein Benutzerhandbuch für jeden Host-Adapter mit speziellen Informationen, zugeschnitten auf den Typ der Partneranwendung:

Anschluss an OSD-Anwendungen

Anschluss an MVS-Anwendungen

Alle Handbücher zu den Host-Adaptern enthalten eine ausführliche Beispielsitzung. Sie beschreiben

- die Installation von WebTransactions mit dem jeweiligen Host-Adapter.
 - das Einrichten und Starten einer WebTransactions-Anwendung.
 - die Umsetzungs-Templates für die dynamische Umsetzung der Formate auf die Oberfläche eines Web-Browsers.
 - die Bearbeitung von Templates.
 - die Steuerung der Kommunikation zwischen WebTransactions und den Host-Anwendungen über verschiedene Attribute des Systemobjekts.
 - die Behandlung asynchroner Nachrichten und die Druckfunktionen von WebTransactions.
- Ein Benutzerhandbuch, das für alle Liefereinheiten gilt und die Möglichkeiten des HTTP-Host-Adapters beschreibt:

Zugriff auf dynamische Web-Inhalte

Das Handbuch beschreibt

- wie Sie mit WebTransactions auf HTTP-Server zugreifen und deren Ressourcen nutzen.
- die Einbettung des SOAP-Protokolls (Simple Object Access Protocol) in WebTransactions und den Anschluss von Web-Services über SOAP.

- Ein Benutzerhandbuch, das für alle Liefereinheiten gilt und das offene Protokoll und die Schnittstellen für die Client-Entwicklung für WebTransactions beschreibt:

Client-APIs für WebTransactions

Das Handbuch beschreibt

- das Konzept der Client-Server-Schnittstelle von WebTransactions.
 - die Klasse `WT_RPC` und die Schnittstelle `WT_REMOTE`. Ein Objekt der Klasse `WT_RPC` repräsentiert eine Verbindung zu einer fernen WebTransactions-Anwendung, die auf der Server-Seite über die Schnittstelle `WT_REMOTE` abgewickelt wird.
 - Das Java-Package `com.siemens.webta`, das für die Kommunikation mit WebTransactions ausgeliefert wird.
- Ein Benutzerhandbuch, das für alle Liefereinheiten gilt und das Web-Frontend von WebTransactions beschreibt, das den Zugriff auf allgemeine Web-Services ermöglicht:

Web-Frontend für Web-Services

Das Handbuch beschreibt

- das Konzept des Web-Frontends für objektorientierte Backend-Systeme.
- die Generierung von Templates für den Anschluss von allgemeinen Web-Services an WebTransactions.
- den Test und die Weiterentwicklung des Web-Frontends für allgemeine Web-Services.

1.4 Konzept und Zielgruppe dieses Handbuchs

Diese Dokumentation wendet sich an Benutzer, die mit WebTransactions openUTM-Dialoganwendungen an das Web anschließen wollen.

Die einzelnen Kapitel beschreiben die hierfür notwendigen Schritte. Wenn Sie noch nicht mit WebTransactions for openUTM gearbeitet haben, sollten Sie sich zuerst das Kapitel [3](#) mit der Beispielsitzung durchlesen, die Ihnen einen ersten Einblick in die Arbeit mit WebTransactions geben soll.

Das Handbuch ergänzt das einführende WebTransactions-Handbuch „Konzepte und Funktionen“ und das WebTransactions-Referenzhandbuch „Template-Sprache“ um alle openUTM-spezifischen Informationen.

Gültigkeit der Beschreibung

WebTransactions for openUTM ist auf den Systemplattformen BS2000/OSD, Solaris, Linux sowie Windows ablauffähig. Diese Dokumentation gilt für alle WebTransactions-Plattformen. Falls sich eine Information speziell auf eine bestimmte WebTransactions-Plattform bezieht, wird jeweils ausdrücklich darauf hingewiesen.



1.5 Neue Funktionen

In diesem Abschnitt werden nur die openUTM-spezifischen Neuerungen der WebTransactions-Version 7.5 genannt. Einen allgemeinen Überblick über die Neuerungen finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

Art der Neuerung	Beschreibung
Neues Host-Datenobjekt-Attribut <code>Unicode</code>	Seite 157
Neues Attribut <code>Unicode</code> an <code>WT_HOST_MESSAGE</code>	Seite 163

1.6 Darstellungsmittel

Diese Dokumentation verwendet die folgenden Darstellungsmittel:

Auszeichnung	Bedeutung
dicktengleiche Schrift	festе Teile, die genau in dieser Form ein- oder ausgegeben werden, wie z.B. Schlüsselwörter, URLs, Dateinamen
<i>kursive Schrift</i>	variable Teile, für die Sie konkrete Angaben einsetzen müssen
fette Schrift	Zitate, die genauso am Bildschirm oder in der grafischen Oberfläche angezeigt werden, sowie Menübefehle
[]	optionale Angaben. Die eckigen Klammern selbst dürfen Sie nicht angeben.
{ <i>alternative1</i> <i>alternative2</i> }	alternative Angaben. Einen der Ausdrücke innerhalb der geschweiften Klammern müssen Sie auswählen. Die einzelnen Ausdrücke sind durch senkrechte Striche voneinander getrennt. Die geschweiften Klammern selbst dürfen Sie nicht angeben
...	optionale ein oder mehrmalige Wiederholung des vorhergehenden Elements
	wichtige Hinweise und weiterführende Informationen
▶	Aufforderungszeichen, wenn Sie etwas tun sollen.
	verweist auf weiterführende Informationen

2 WebTransactions installieren

Die WebTransactions-Installationsdateien stehen im Web zum Download zur Verfügung.



Detaillierte Informationen zu den Hardware- und Software-Voraussetzungen finden Sie in der mit dem Produkt ausgelieferten Freigabemitteilung.

2.1 Installation

WebTransactions for openUTM besteht aus dem Host-Adapter, über den die Kommunikation mit UTM-Host-Anwendungen läuft, dem WebTransactions Laufzeitsystem und dem Host-Adapter für dynamische Web-Inhalte.

WebTransactions for openUTM enthält das Installationspaket für die Entwicklungsumgebung WebLab, mit der Sie eine Host-Anwendung an das WWW anbinden sowie die Host-Formate optisch aufbereiten und funktional erweitern können. WebLab müssen Sie ggf. explizit auf Ihrem Entwicklungsrechner installieren (siehe [Abschnitt „Installation von WebLab“ auf Seite 23](#)).



Stellen Sie vor der Installation von WebTransactions sicher, dass der Web-Server und gegebenenfalls Java bereits installiert sind.

Notieren Sie sich das Installationsverzeichnis von Java und aus der Konfiguration des Web-Servers folgende Informationen:

- Root-Verzeichnis für Web-Seiten (=Dokumentenverzeichnis)
- CGI-Verzeichnis
- URL-Präfix für CGI-Programme

2.1.1 Windows

Für Windows steht WebTransactions nach dem Download als Windows Installationspaket (msi-Datei) `WebTransactionsUTM75.msi` zur Verfügung.

2.1.1.1 Installation über die Bedienoberfläche

Für die Installation benötigen Sie die Windows-Administratorberechtigung. Sie haben verschiedene Möglichkeiten, die Installation zu starten:

- Über den Befehl **Einstellungen/Systemsteuerung** im Start-Menü.
- Über den Windows-Explorer
Sie klicken dazu doppelt auf die msi-Datei oder Sie klicken mit der rechten Maustaste auf die msi-Datei und wählen im Kontextmenü den Befehl **Installieren**.

Einstellungen für den Web-Server festlegen

Nach dem Start von `WebTransactionsUTM75.msi` werden eine Reihe von Dialogfeldern angezeigt, in denen Sie das Installationsverzeichnis und die Werte Ihres Web-Servers angeben müssen:

- Root-Verzeichnis für Web-Seiten (=Dokumentenverzeichnis)
- CGI-Verzeichnis und URL-Präfix
- ISAPI-Verzeichnis und ISAPI-Präfix (optional)
- Verzeichnis der Java2-Bibliothek `jvm.dll` für die Java-Integration (optional)

Wenn Sie die Werte angegeben haben, wird die Installation gestartet und die gewünschten Komponenten installiert. Wenn Sie WebTransactions mit einem weiteren Host-Adapter auf dem selben System installieren, werden diese Werte in die neue Installation übernommen.

Komponenten auswählen

Im folgenden Verlauf können Sie die Komponenten auswählen, die installiert werden sollen. Im Dialogfeld **Select Installation Type** wählen Sie dazu einen der folgenden Einträge:

Typical oder **Complete**

Alle Komponenten von WebTransactions werden installiert.

Custom

Das Installations-Programm bietet folgende Komponenten an:

- WebTransactions Laufzeitsystem
- WebTransactions Demo Applikationen

2.1.1.2 Bedienerlose Installation

Für eine bedienerlose Installation (silent installation) verwenden Sie den Windows Installer `Msiexec.exe`. Die ausführliche Beschreibung dieses Kommandos finden Sie in der Online-Hilfe zu Windows. Für die Installation mit `Msiexec.exe` benötigen Sie die Windows-Administratorberechtigung.

Verwenden Sie das Kommando `Msiexec.exe` mit folgender Syntax:

```
Msiexec.exe /I "package" /q  
[INSTALLDIR="install-dir"]  
[DOCUMENTROOTDIR="documentroot-dir"]  
[HTTPSCRIPTSDIR="cgi-dir"]  
[JAVA2SYS="java-dir"]  
[ISPREFIX="isapi-prefix"]  
[URLPREFIX="cgi-prefix"]  
[ISAPICHECK="isapicheck"]  
[JAVA2CHECK="java2check"]
```

Die Parameter haben folgende Bedeutung:

package

Pfad für das zu installierende Paket (z.B. `C:\tmp\WebTransactionsUTM75.msi`).

install-dir

Installationsverzeichnis von WebTransactions.

Standardwert: `C:\Programme\WebTransactionsV75` bzw.

`C:\Program Files\WebTransactionsV75`

documentroot-dir

Dokumentenverzeichnis des Web-Servers.

Standardwert: `C:\InetPub\wwwroot`

cgi-dir CGI-Verzeichnis des Web-Servers.

Standardwert: `C:\InetPub\scripts`

java-dir

Verzeichnis der Java2-Bibliothek `jvm.dll`. Diese Angabe ist nur notwendig, wenn die Unterstützung für die Java-Schnittstelle installiert werden soll.

isapi-prefix

URL-Präfix für ISAPI.

Standardwert: `scripts`

cgi-prefix

URL-Präfix für CGI.

Standardwert: `scripts`

isapicheck

gibt an, ob die ISAPI Schnittstelle von WebTransactions installiert werden soll.

Mögliche Werte: Yes | No

Standardwert: No

java2check

gibt an, ob die Unterstützung für die Java-Schnittstelle installiert werden soll.

Mögliche Werte: Yes | No

Standardwert: No

Beispiel

```
Msiexec.exe /I "C:\tmp\WebTransactionsUTM75.msi" /q  
INSTALLDIR="D:\Programme\WebTransactionsV75"  
DOCUMENTROOTDIR="C:\Programme\Apache Group\Apache\htdocs"  
HTTPSCRIPTSDIR="C:\Programme\Apache Group\Apache\cgi-bin"  
JAVA2SYS="D:\Programme\Java\jdk1.6.0_13\jre\bin\client"  
URLPREFIX="cgi-bin" JAVA2CHECK="Yes"
```

2.1.2 Solaris

Für die Installation von WebTransactions nutzen Sie wie gewohnt das Installationsverfahren `pkgadd` unter `root`-Berechtigung. Geben Sie dabei den absoluten Dateinamen der entpackten Produktdatei an:

```
pkgadd -d /absoluter_pfad/dateiname
```

Im Lauf der Installation werden folgende Fragen gestellt:

1. Should WebTransactions demos be installed?

Wenn Sie `y` (yes) eingeben, werden die Demo-Anwendungen von WebTransactions mit installiert.

2. Your Web Server has a 'document default directory'
Where is this directory?

Geben Sie hier den entsprechenden Pfadnamen an.

3. The server uses an URL prefix to access WebTransactions CGI program.
URL prefix:

Geben Sie das URL-Präfix an, das auf Ihrem Web-Server für CGI-Programme eingestellt ist.

4. Your Web Web Server has a `cgi-bin` directory,
in which you install WebTransactions CGI-Program.
Where is this directory?

Hier geben Sie den absoluten Pfad zu dem CGI-Verzeichnis an, das bei Ihrem Web-Server konfiguriert ist.

Im Verlauf der Installation bekommen Sie dann die URL angezeigt, mit der Sie die Demo-Anwendungen starten können.

2.1.3 Linux

WebTransactions wird als komprimiertes Archiv zum Herunterladen bereitgestellt mit der Endung `.gz` (z.B. `webtransUTMV75.tar.gz`). Diese Datei müssen Sie zuerst dekomprimieren, mit dem Kommando:

```
gunzip -d webtransUTMV75.tar
```

Beachten Sie, dass Sie die Endung `.gz` nicht angeben dürfen. Danach holen Sie die Installationsdateien mit dem `tar`-Kommando aus dem Archiv:

```
tar -xvf webtransUTMV75.tar
```

Starten Sie dann das Installationsverfahren `doinstall` unter der `root`-Berechtigung:

```
./doinstall
```

Im Lauf der Installation werden folgende Fragen gestellt:

```
You can install WebTransactions into any directory.  
Where is this directory ? [/opt]
```

Geben Sie nur einen anderen Pfadnamen an, wenn WebTransactions nicht im voreingestellten Pfad `/opt` installiert werden soll.

```
Your Web Server has a directory for CGI programs.  
Where is this directory ? [/usr/local/httpd/cgi-bin]
```

Geben Sie hier den entsprechenden Pfadnamen an.

```
Your Web Server uses an URL prefix to access the CGI programs in  
/usr/local/httpd/cgi-bin  
What is this prefix ? [cgi-bin]
```

Geben Sie das URL-Präfix an, das auf Ihrem Web-Server für CGI-Programme eingestellt ist.

```
Are this settings OK ? [y]
```

Bestätigen Sie Ihre Angaben, um die Installation abzuschließen.

2.1.4 BS2000/OSD

Die Standard-Installation erfolgt durch das Verfahren SOLIS. Ist im Ausgangssystem das Produkt IMON (Installations MONitor) gestartet, können Sie die Standard-Installation auch mit IMON ausführen.

Für die Installation in POSIX steht Ihnen das POSIX-Installationstool zur Verfügung.

2.1.5 Installation von WebLab

Bei der Installation von WebTransactions wird auf jeder Plattform die msi-Datei für die Installation von WebLab auf Windows (`WebLab75.msi`) im Dokumentenverzeichnis des Web-Servers unter dem Verzeichnis `webtav75` abgelegt.

Installationspaket auf den Entwicklungsrechner übertragen

Das WebLab-Installationspaket kann über einen Browser-Aufruf mit folgender URL auf den gewünschten Entwicklungsrechner heruntergeladen werden:

`http://web-server/webtav75/wtdownload.htm`

WebLab auf Windows installieren

Nachdem Sie das WebLab-Installationspaket auf Ihren Entwicklungsrechner heruntergeladen haben, installieren Sie die msi-Datei wie gewohnt über die Bedienoberfläche (siehe [Seite 18](#)) oder mit `Msiexec.exe` (siehe [Seite 19](#)).

Sie müssen in beiden Fällen nur das Installationsverzeichnis für WebLab angeben.

2.2 Lizenzierung

Nach der Installation müssen Sie noch die Anzahl der vorhandenen Lizenzen und den maschinenspezifischen Aktivierungsschlüssel konfigurieren. Dazu rufen Sie das Administrations-Programm von WebTransactions auf und wählen dort den Menüpunkt **Licences**. Weitere Informationen zum Administrationsprogramm finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

3 Beispielsitzung

In diesem Kapitel lernen Sie die Möglichkeiten von WebTransactions sowie einige grundlegende Regeln für die Arbeit mit WebTransactions kennen. Diese Beispielsitzung ist als exemplarische Vorgangsbeschreibung gedacht, die Ihnen zeigen soll, wie Sie ohne viel Programmieraufwand eine Host-Anwendung an das WWW anbinden.

In dieser Beispielsitzung werden Sie zunächst mit dem Administrationsprogramm die nötigen Voraussetzungen für die Arbeit mit WebLab und WebTransactions schaffen. Anschließend werden Sie mit WebLab die Host-Anwendung an das Web anbinden. Danach werden Sie die Möglichkeiten von globalen und formatspezifischen Änderungen im Template kennenlernen.



Beachten Sie, dass bei allen Pfadangaben davon ausgegangen wird, dass WebTransactions im voreingestellten Verzeichnis installiert wurde.

3.1 WebTransactions-Server verwalten

Nachdem Sie WebTransactions for openUTM auf einem Rechner installiert haben (siehe hierzu auch [Seite 17](#)), müssen Sie zuerst die Voraussetzungen schaffen, um mit WebTransactions und WebLab zu arbeiten. Diese Voraussetzungen schaffen Sie mit dem Administrationsprogramm, das im WebTransactions-Handbuch „Konzepte und Funktionen“ beschrieben ist.

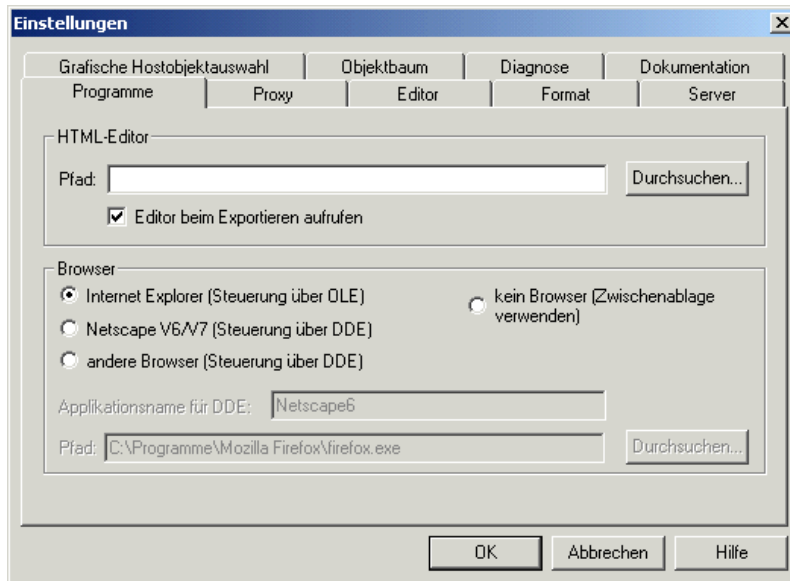
Als ersten Schritt stellen Sie in WebLab den Browser ein, den WebLab für die Bedienung der WebTransactions-Anwendung verwenden soll. Die Arbeit mit dem Administrationsprogramm gliedert sich dann in folgende Schritte:

1. Lizenzen eingeben
2. Benutzer anlegen
3. Pool anlegen
4. Pool dem Benutzer zuweisen

3.1.1 Browser einstellen

Bevor Sie beginnen zu arbeiten, sollten Sie in WebLab den Browser einstellen, den WebLab für die Bedienung der WebTransactions-Anwendung verwenden soll. Dieser Schritt ist nur erforderlich, wenn Sie zum ersten Mal mit WebLab arbeiten.

- ▶ Starten Sie WebLab mit dem Befehl **Start/Programme/WebTransactions 7.5/ WebLab**. Das Hauptfenster von WebLab wird am Bildschirm eingeblendet. Eine genaue Beschreibung des Hauptfensters und seiner Elemente finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“ und in der Online-Hilfe.
- ▶ Wählen Sie dann in WebLab den Befehl **Optionen/Einstellungen**. Das Dialogfeld **Einstellungen** mit der ersten Registerkarte **Programme** wird am Bildschirm eingeblendet.



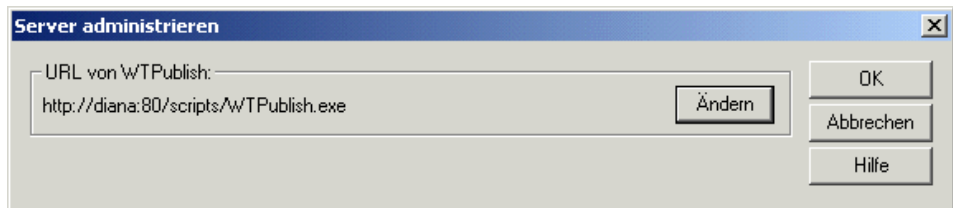
- ▶ Wählen Sie im unteren Bereich **Browser** den Browser aus, der auf Ihrem Rechner installiert ist, und wie er von WebLab verwendet werden soll.
- ▶ Bestätigen Sie Ihre Einstellung mit **OK**.

3.1.2 Administrationsprogramm starten

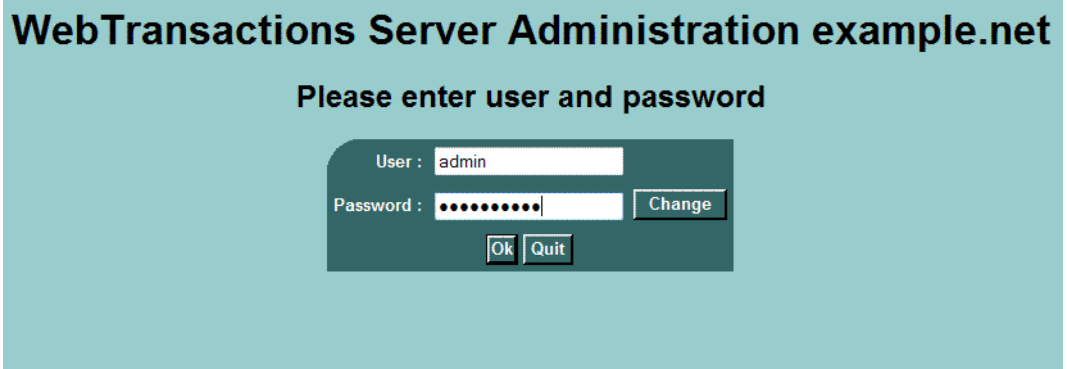
- ▶ Wählen Sie den Befehl **Administrieren/Server**, um zuerst das Administrationsprogramm zu starten. Das Dialogfeld **Server administrieren** wird am Bildschirm eingeblendet.
- ▶ Klicken Sie unter **Url von WTPublish** auf **Ändern**. Das Dialogfeld **Url von WTPublish** wird angezeigt.
- ▶ Wählen Sie das **Protokoll**, das für die Verbindung verwendet werden soll.
- ▶ Geben Sie in den anderen Feldern die entsprechenden Werte für Ihren Rechner an:

Server Rechner, auf dem WebTransactions läuft
Port zugehörige Portnummer
CGI-Pfad Pfad für das CGI-Programm `WTPublish`
Programm CGI-Programm

- ▶ Bestätigen Sie mit **OK**. Die Werte werden dann in das Dialogfeld **Server administrieren** übernommen.



- ▶ Bestätigen Sie mit **OK**. Das Administrationsprogramm wird gestartet und das erste Fenster im Browser angezeigt.



- ▶ Loggen Sie sich als Benutzer `admin` ein. Dieser Benutzer wird bei der Installation von WebTransactions ohne Passwort eingerichtet. Damit wird automatisch die Lizenzierungsseite angezeigt.



Wenn Sie zum erstenmal mit dem Administrationsprogramm arbeiten, sollten Sie zur Sicherheit nach der Anmeldung für den Benutzer `admin` ein Passwort vergeben.

3.1.3 Lizenzen eingeben

FUJITSU Administration for server: **example.net**, 0 active sessions, 3 licenses installed

Licenses

Users

Pools


Applications

Sessions

Tools

Clusters

Print



Registration

Type	Description	Action
Online	To set the number of licenses you need an activation key from the WebTransactions license server. Use the button on the right to register online.	Register
Help Desk	Call our support team +49 (1805) 4040 Use the Info button on the right to get the required informations.	Info

Licenses

Info

Server-Id: **693619ac**

Key: **Invalid**

Licensed Servers: **1**

Licensed concurrent users: **3**

On demand user licenses: **0**

License logging:

Action

Enter new license

Servers:

Licenses:

On Demand Licenses:

Days:

Key:

[Set](#)

[Save](#) [Load](#) [Refresh](#) [Exit](#)

- ▶ Klicken Sie auf die Schaltfläche **Register**.

Die Registrierungsseite wird geöffnet:

The screenshot shows a web form for license registration. At the top, there are two radio buttons for 'Type of License': 'Single Server' (selected) and 'Cluster'. Below this is a 'Server-Id' field containing 'fe7b19ac'. The 'Number of licenses' field is empty, and there is an unchecked checkbox for 'On demand licenses:'. The 'Email address' field is empty, with a note below it stating 'Key will be sent to this address!'. The 'Platform' is a dropdown menu set to 'Please select...'. Under 'Installed adapters', there is a dropdown menu set to 'Change of registered licenses'. The bottom section contains several text input fields: 'Name', 'Company', 'Department', 'Street', 'PC/City' (split into two boxes), 'Country', and 'Sales Representative'. A larger 'Remarks' field is a text area. At the bottom center is a 'Request Key' button.

- ▶ Um Lizenzen für einen Standalone-Server zu registrieren, aktivieren Sie unter **Type of license** die Option **Single Server**.
- ▶ Geben Sie die Anzahl der zu lizenzierenden Server in das Feld **Number of licences** ein.
- ▶ Geben Sie Ihre eMail-Adresse und ggf. weitere Parameter an.

- ▶ Schicken Sie das Formular mit **Request Key** ab.
Der Lizenz-Schlüssel wird Ihnen nach kurzer Zeit an die angegebene Mail-Adresse gesendet.
- ▶ Tragen Sie auf der Lizenzierungsseite in die Felder **Licenses** und **Key** die Anzahl der erworbenen Lizenzen bzw. den gültigen Lizenz-Schlüssel ein, der Ihnen per eMail zugesendet wurde.
- ▶ Bestätigen Sie mit **Set** und anschließend mit **Save**.
Die Lizenzen sind aktiviert. Die neue Anzahl Lizenzen wird in der Statusleiste angezeigt.

3.1.4 Benutzer anlegen

- Klicken Sie dann auf den Menüeintrag **Users**, um neue Benutzer einzutragen. Das Fenster **Users** wird im Browser angezeigt.

Administration for server: **example.net**, 0 active sessions, 3 licenses installed

Users

Username	Comment	Action
admin	Administrator	Change Password
frank		Change Password Remove
webtaman		Change Password Remove
<input type="text"/>	<input type="text"/>	Add

Click on user name to access the user's properties

Save Load Refresh Exit

- Geben Sie im Arbeitsbereich in das Eingabefeld **Username** den Namen des neuen Benutzers ein.
- Geben Sie gegebenenfalls eine Beschreibung oder Erläuterung zum Benutzer im Eingabefeld **Comment** ein und klicken Sie auf die Schaltfläche **Add**. Damit wird der neue Benutzer für WebTransactions und die Arbeit mit WebLab eingetragen. Allerdings hat der neue Benutzer noch keine Rechte. Die müssen Sie ihm erst zuweisen.
- Klicken Sie aber zuerst auf den Knopf **Change Password** und geben Sie für den neuen Benutzer ein Passwort ein. Genauso kann für die Kennung `admin` ein Passwort vergeben werden.

3.1.5 Pool anlegen

- Klicken Sie dann auf den Menüeintrag **Pools**, um einen Pool anzulegen, unter dem Basisverzeichnisse angelegt werden dürfen. Das Fenster **Pools** wird im Browser angezeigt.

FUJITSU Administration for server: **example.net**, 0 active sessions, 3 licenses installed

Licenses

Users

Pools

Applications

Sessions

Tools

Clusters

Print

Pools

Directory	Virtual Path	Comment	Action
c:/inetpub/wwwroot/webtav75	webta		Remove Edit
d:/basedirs/manual	manual		Remove Edit
d:/basedirs/v71	basev71		Remove Edit
d:/basedirs/v75	basedirs/v75		Remove Edit
d:/home/cvs/v75/webta	regtest		Remove Edit
<input type="text" value="d:/basedirs/manual"/>	<input type="text" value="manual"/>	<input type="text" value="Beispiel"/>	Add <small>Check here to create it</small>

Click on directory to access the pool's properties

Powered by WebTransactions

- Geben Sie im Arbeitsbereich in das Eingabefeld **Directory** den Namen des Verzeichnisses ein. Beachten Sie, dass dieses Verzeichnis entweder existieren muss oder Sie wählen die Option, um das Verzeichnis anzulegen.
- Geben Sie in das Eingabefeld **Virtual Path** den Namen für ein Verzeichnis unterhalb des Dokumentenverzeichnisses des Web-Servers an, das dem neuen Pool zugeordnet ist. Dieses Verzeichnis entspricht dem Anfang des virtuellen Pfades, mit dem vom Web-Server direkt, also ohne WebTransactions dafür aufrufen zu müssen, auf Dateien (z.B. Bilder, Aufrufseite, ...) von WebTransactions-Anwendungen in diesem Verzeichnis zugegriffen wird.



Falls Sie in unterschiedlichen Pools Basisverzeichnisse mit gleichen Namen verwenden wollen, müssen die dem Pool zugehörigen Werte bei **Virtual Path** unterschiedlich angegeben werden.

- ▶ Geben Sie gegebenenfalls eine Beschreibung oder Erläuterung zum Pool im Eingabefeld **Comment** ein und klicken Sie auf die Schaltfläche **Add**. Damit wird der neue Pool für WebTransactions und die Arbeit mit WebLab eingetragen. Je nach Bedarf können Sie weitere Pools eintragen.

Unter Pools, die auf diese Weise angelegt wurden, können nun mit WebLab die Basisverzeichnisse angelegt werden. Allerdings kann bisher noch kein Benutzer mit WebLab auf diesen Pool zugreifen, da er noch keinem Benutzer zugewiesen wurde.

3.1.6 Pool dem Benutzer zuweisen

- Klicken Sie In der Tabelle der Pools auf den Pool, den Sie gerade angelegt haben. Das Fenster **Pool** wird mit dem neu angelegten Pool im Browser angezeigt.

FUJITSU

Administration for server: **example.net**, 0 active sessions, 3 licenses installed

Licenses

Users

Pools

Applications

Sessions

Tools

Clusters

Print

Pool d:/basedirs/manual

Users
(who can create applications here) Action

No users allowed yet

frank	
webtaman	Add

Click on user name to access the user's properties

Save Load Refresh Exit

Powered by WebTransactions

In diesem Fenster werden die Benutzer angezeigt, die auf den neuen Pool zugreifen dürfen. Momentan ist für diesen Pool noch kein Benutzer zugewiesen. In einer Liste werden alle Benutzer angezeigt, die auf diesem Rechner mit WebTransactions arbeiten dürfen.

- Wählen Sie aus dieser Liste mit einem Klick den Benutzer, den Sie neu angelegt haben und klicken Sie auf den Knopf **Add**. Der ausgewählte Benutzer wird für den Zugriff auf diesen Pool eingetragen. Damit haben Sie die notwendigen Vorarbeiten für die Arbeit mit WebTransactions erledigt.
- Speichern Sie die aktuelle Konfiguration von WebTransactions mit dem Knopf **Save**.

- ▶ Beenden Sie das Administrationsprogramm mit dem Knopf **Exit**.
- ▶ Beenden Sie den Browser.

3.2 Host-Anwendung an das WWW anbinden

Nachdem Sie die Voraussetzungen für die Arbeit mit WebTransactions und WebLab geschaffen haben, können Sie die Host-Anwendung mit WebLab, der Entwicklungsumgebung für WebTransactions, an das WWW anbinden. Dazu sind folgende Schritte erforderlich:

1. Projekt anlegen
 - Basisverzeichnis anlegen
2. Projekt speichern
3. Templates aus FHS-Formaten erzeugen
4. Lokale Anwendungsnamen festlegen
5. Sitzung starten

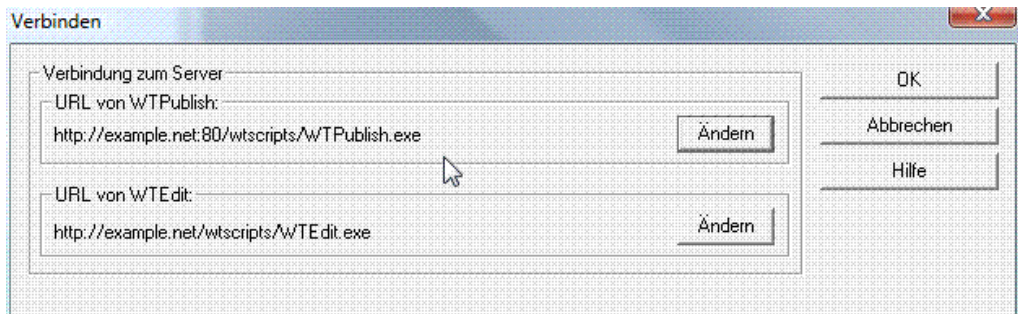
3.2.1 Projekt anlegen

Im Projekt sind die wichtigsten Daten gespeichert, die WebLab zum Erzeugen und Bearbeiten einer WebTransactions-Anwendung benötigt, z.B. Daten des WebTransactions-Server.

- ▶ Um ein Projekt anzulegen, wählen Sie den Befehl **Projekt/Neu....**
- ▶ Im folgenden Dialogfeld werden Sie gefragt, ob Sie ein Basisverzeichnis erzeugen wollen. Klicken Sie auf **Ja**. Das Dialogfeld **Verbinden** wird geöffnet, siehe nachfolgender Abschnitt.

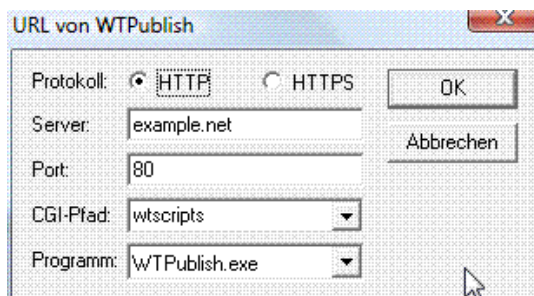
3.2.1.1 Basisverzeichnis anlegen

Das Basisverzeichnis ist die Grundlage für die Web-Anbindung einer Host-Anwendung mit WebTransactions. Hier befinden sich alle notwendigen Dateien und Links auf Programme, die eine WebTransactions-Anwendung ausmachen.



Das Basisverzeichnis muss immer auf dem Rechner angelegt werden, auf dem WebTransactions läuft. Im Dialogfeld **Verbinden** geben Sie diesen WebTransactions-Server an und die Pfade zu den CGI-Programmen `WTPublish.exe` und `WTEdit.exe`.

- `WTEdit.exe` nimmt alle Anforderungen von WebLab entgegen. Es führt alle notwendigen Arbeiten stellvertretend für WebLab (das ggf. auf einem anderen Rechner abläuft) auf dem WebTransactions-Server aus (z.B. das Erstellen eines Basisverzeichnisses) und ermöglicht WebLab den Zugriff auf eine laufende WebTransactions-Sitzung.
- `WTPublish.exe` nimmt alle Anforderungen vom Browser an. Es startet neue WebTransactions-Sitzungen oder stellt die Verbindung zu einer bereits laufenden Sitzung für jeden folgenden Dialogschritt wieder her.
- ▶ Klicken Sie unter **Verbindung zum Server - URL von WTPublish** auf **Ändern**. Das Dialogfeld **URL von WTPublish** wird angezeigt.



- ▶ Wählen Sie das **Protokoll**, das für die Verbindung verwendet werden soll, hier **HTTP**.

- ▶ Geben Sie im Feld **Server** den Rechner an, auf dem WebTransactions läuft, hier *example.net*.
- ▶ Geben Sie im Feld **Port** die zugehörige Portnummer an, hier *80*.
- ▶ Geben Sie den Pfad für das CGI-Programm WTPublish an, hier *wtscripts*.
- ▶ Geben Sie das CGI-Programm an, hier *WTPublish.exe*, und bestätigen Sie mit **OK**. Diese Werte werden dann in das Dialogfeld **Verbinden** übernommen.
- ▶ Wiederholen Sie den Vorgang für den Eintrag unter **URL von WTEdit**. Geben Sie auch hier die entsprechenden Werte für Ihren Rechner an, in diesem Fall:

Server *example.net*

Port *80*

CGI-Pfad *wtscripts*

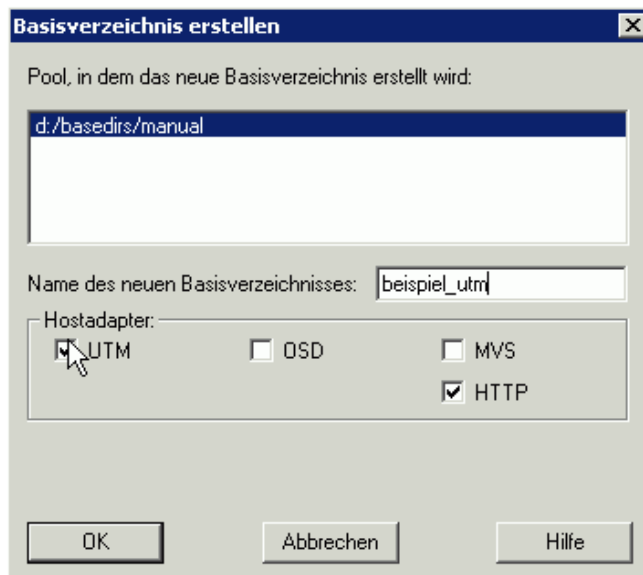
Programm *WTEdit.exe*

- ▶ Klicken Sie anschließend im Dialogfeld **Verbinden** auf **OK**. Die Verbindung zum WebTransactions-Rechner wird mit den angegebenen Werten aufgebaut.

Zuerst müssen Sie sich aber bei WebTransactions anmelden. Dazu wird das Dialogfeld **Name und Passwort** am Bildschirm eingeblendet.



- ▶ Geben Sie Benutzernamen und Passwort ein, die Sie in [Abschnitt „Benutzer anlegen“ auf Seite 31](#) angelegt haben.
- ▶ Bestätigen Sie Ihre Angaben mit **OK**. Das Dialogfeld **Basisverzeichnis erstellen** wird am Bildschirm eingeblendet.



In der oberen Liste dieses Dialogfeldes werden die Pools angezeigt, unter denen der angemeldete Benutzer auf dem WebTransactions-Server Basisverzeichnisse angelegen darf.

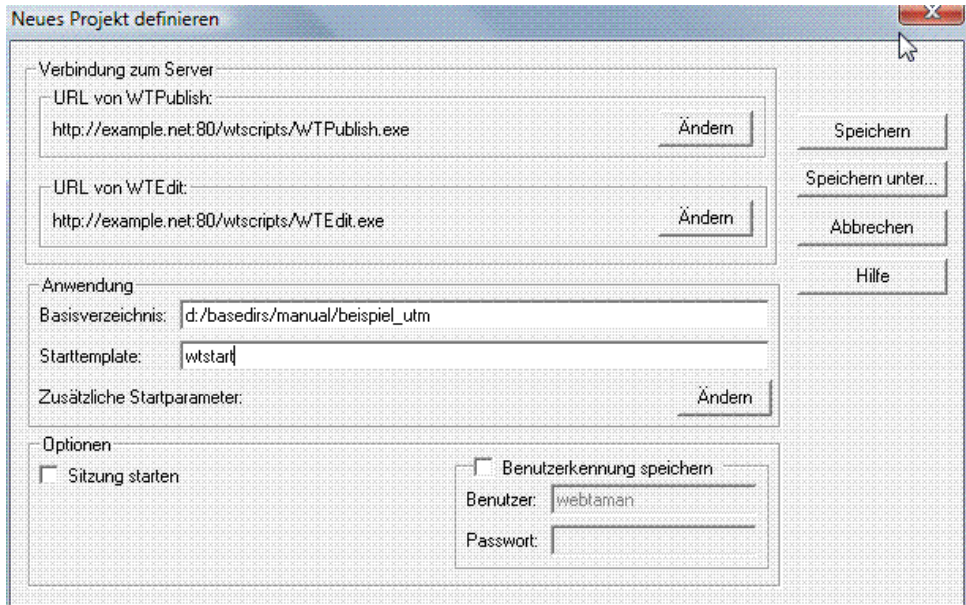
- ▶ Wählen Sie mit einem Klick aus der Liste den Pool, den Sie in [Abschnitt „Pool anlegen“ auf Seite 32](#) angelegt haben.
- ▶ Geben Sie dann im Eingabefeld **Name des neuen Basisverzeichnisses** einen Namen an, hier *beispiel_utm*.
- ▶ Wählen Sie dann den passenden Host-Adapter, über den WebTransactions mit der Host-Anwendung kommuniziert, hier *UTM*. Es sind nur die Host-Adapter auswählbar, die Sie auch installiert haben. Der Host-Adapter für HTTP ist schon voreingestellt.
- ▶ Bestätigen Sie Ihre Eingaben mit **OK**. Das Dialogfeld **Basisverzeichnis erstellen** wird geschlossen und das Basisverzeichnis wird mit den angegebenen Werten am WebTransactions-Server generiert. Im WebLab-Hauptfenster wird unterhalb des Arbeitsbereichs ein Meldefenster geöffnet, in dem der Arbeitsfortschritt angezeigt wird.

Das Dialogfeld **Neues Projekt definieren** wird geöffnet, siehe folgender Abschnitt.

Ein Start-Template, das den Benutzer direkt zum ersten Format der Host-Anwendung führt, wird am Ende der Beispielsitzung erstellt (siehe [Abschnitt „Start-Template erzeugen“ auf Seite 63](#)).

3.2.2 Projekt speichern

Im Dialogfeld **Neues Projekt definieren** legen Sie die Einstellungen für das neu angelegte Projekt fest.

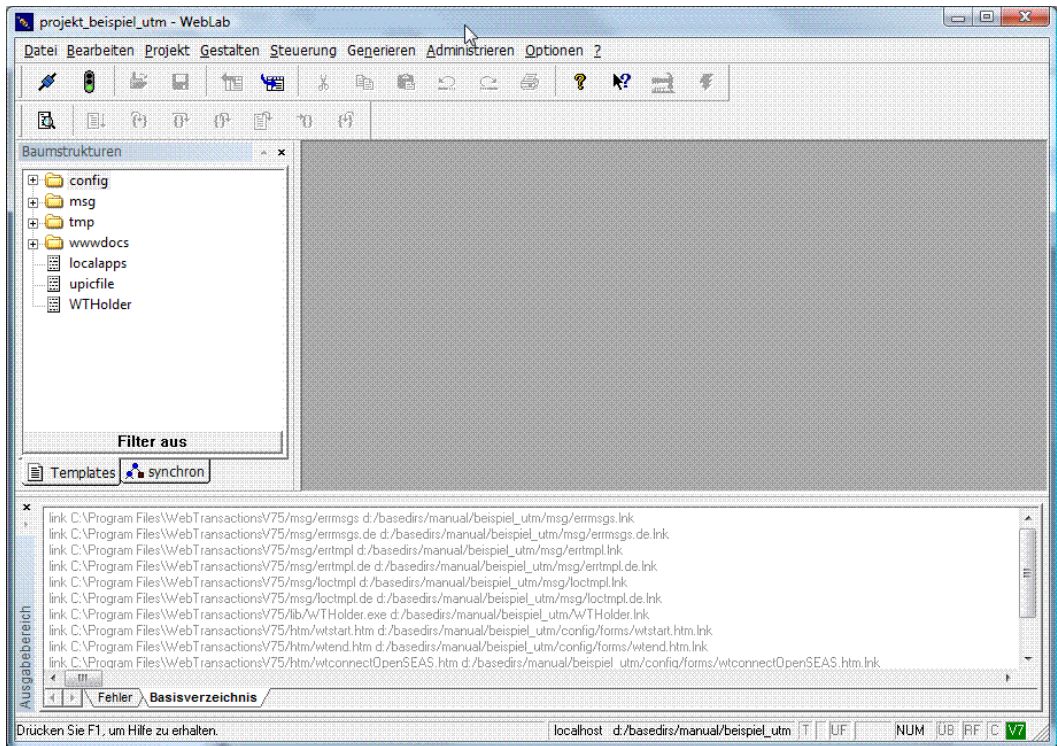


- ▶ In diesem Beispiel akzeptieren Sie alle Voreinstellungen und speichern das Projekt mit **Speichern unter...**

Das Dialogfeld **Datei speichern unter** wird geöffnet.

- ▶ Wählen Sie in diesem Dialogfeld das Verzeichnis, in dem Sie das Projekt speichern möchten und geben Sie den Namen für die Projektdatei an.
- ▶ Klicken Sie auf **Speichern**.

Die Projektdatei wird mit dem Suffix `.wtp` im ausgewählten Verzeichnis angelegt. Der Name der Projektdatei wird in der Titelleiste von WebLab angezeigt.



Anschließend sind Sie mit Ihrem neuen Basisverzeichnis verbunden. Eine Übersicht über die angelegten Verzeichnisse finden Sie im [Abschnitt „Struktur eines Basisverzeichnisses“](#) auf Seite 71.

3.2.3 Templates aus FHS-Formaten erzeugen

Um aus FHS-Formaten Templates zu generieren, erzeugen Sie zunächst unter OSD mit dem Programm `IFG2FLD` aus der IFG-Bibliothek eine Formatbeschreibungsource. Die Formatbeschreibungsource enthält alle Feldnamen, die der Entwickler der Host-Formate ursprünglich mit dem Formatierungssystem definiert hat. Danach übertragen Sie die Formatbeschreibungsource auf den WebLab-Rechner und erstellen mit WebLab daraus die Templates und Felddateien.

3.2.3.1 Mit IFG2FLD Beschreibungsdatei erzeugen

- ▶ Übertragen Sie eine der folgenden LMS-Bibliotheken aus dem WebTransactions-Installationsverzeichnis `lib` binär auf den OSD-Rechner zu der Kennung, unter der die IFG-Bibliothek liegt und nennen Sie diese Bibliothek auf dem OSD-Rechner `WTIFG2FLD.LMS`.

Bibliothek	Bedeutung
<code>WTifg2f1dFTP.lms</code>	für die Übertragung mit FTP
<code>WTifg2f1dopenFT.lms</code>	für die Übertragung mit openFT

- ▶ Loggen Sie sich am OSD-Rechner unter dieser Kennung ein.
- ▶ Lenken Sie mit folgendem SDF-Kommando die Ausgabe `SYSLST` auf eine Datei um.
`/ASSIGN=SYSLST beschreibungdatei`
- ▶ Starten Sie `IFG2FLD` mit:
`/START=PROGRAM FROM=FILE=*PHASE(LIBRARY=WTIFG2FLD.LMS,ELEMENT=IFG2FLD)`
- ▶ Geben Sie im `IFG2FLD`-Kommando `MAPFILE` die Bibliothek mit den zu konvertierenden Formaten an:
`MAPFILE LIB.EUROSI.FORMATS`
- ▶ Falls in der IFG-Bibliothek unterschiedliche Benutzerprofile definiert sind, müssen Sie mit dem `PROFILE`-Kommando das gleiche Profil auswählen, das bei der Einsatzvorbereitung für die openUTM-Anwendung verwendet wurde:
`PROFILE benutzerprofil`
- ▶ Mit dem Operanden `*ALL` des `CONVERT`-Kommandos können Sie alle Formate der Bibliothek für die Konvertierung auswählen:
`CONVERT *ALL`

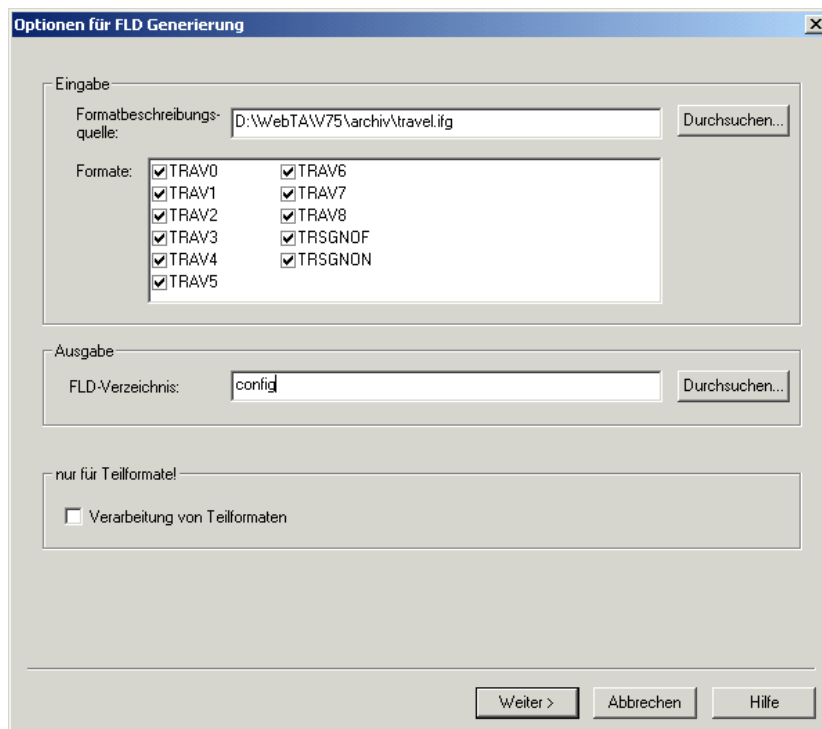
Sie können alternativ hierzu aber auch gezielt Formate für die Konvertierung auswählen:

```
CONVERT TRAVO
CONVERT TRAV1
...
```

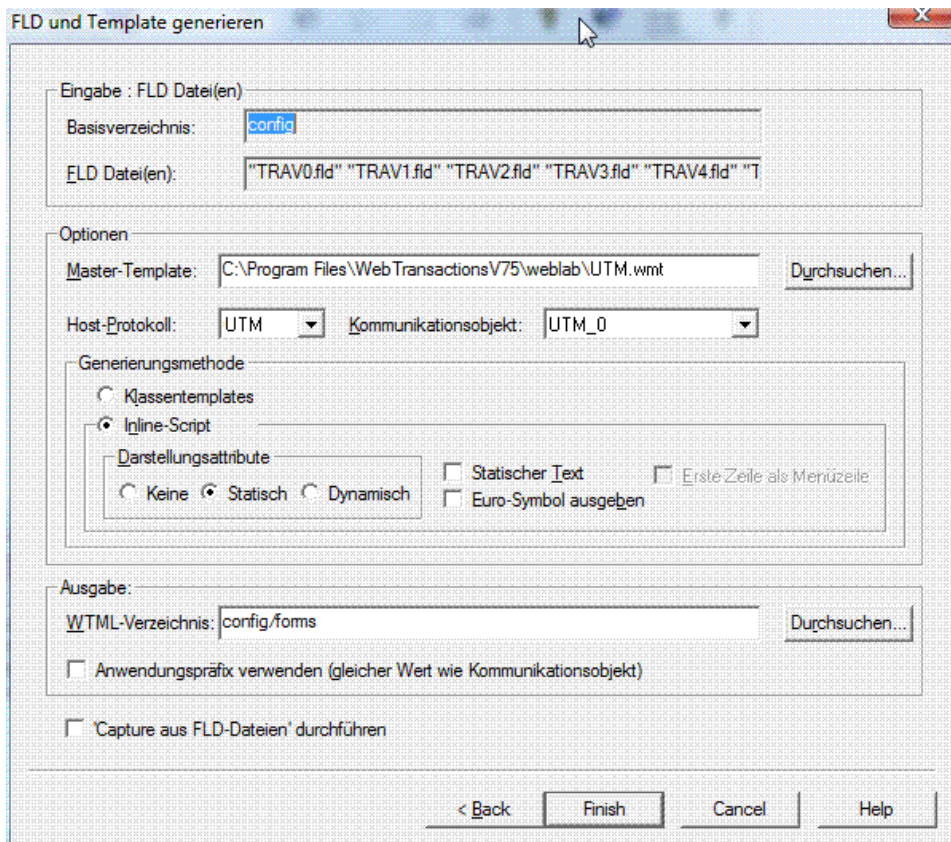
- ▶ Mit dem END-Kommando lösen Sie die Erzeugung der Beschreibungsdatei aus und beenden IFG2FLD:
END
- ▶ Stellen Sie mit dem folgenden SDF-Kommando die Ausgabe SYSLST wieder auf die Voreinstellung des System zurück:
/ASSIGN-SYSLST *P
- ▶ Übertragen Sie die erzeugte Formatbeschreibungsquelle im Textmodus auf den Rechner, in dem WebLab installiert ist. In diesem Beispiel wird die Datei unter *D:\WebTA\W75\archiv\travel.ifg* abgespeichert.

3.2.3.2 Aus Beschreibungsdatei Templates und Felddateien generieren

- ▶ Rufen Sie auf dem Entwicklungsrechner WebLab auf.
- ▶ Wählen Sie den Befehl **Generieren/Templates/aus IFG-Bibliothek**. Das Dialogfeld **Optionen für FLD Generierung** wird am Bildschirm eingeblendet.



- ▶ Geben Sie im Eingabefeld **Formatbeschreibungquelle** den Namen der Formatbeschreibungquelle ein, hier *D:\WebTAV75\archiv\travel.iff*. Danach werden in der Liste **Formate** die Namen der Formate angezeigt, deren Beschreibung WebLab in der angegebenen Formatbeschreibungquelle lesen konnte.
- ▶ Wählen Sie aus der Liste durch Anklicken die Formate, für die Templates und FLD-Dateien generiert werden sollen.
- ▶ Bestimmen Sie im Eingabefeld **FLD-Verzeichnis** das Verzeichnis, in das die neu generierten FLD-Dateien abgelegt werden sollen.
- ▶ Bestätigen Sie Ihre Angaben mit **Weiter >**. Das Dialogfeld **FLD und Template generieren** wird am Bildschirm eingeblendet.



In diesem Dialogfeld können Sie mit den Optionen die Generierung der Templates und damit die spätere Umsetzung beeinflussen.

- ▶ Überprüfen Sie die Voreinstellungen für **Master-Template** (UTM.wmt) und **Host-Protokoll** (UTM).
- ▶ Geben Sie dem **Kommunikationsobjekt** einen Namen, in diesem Beispiel wird die Voreinstellung UTM_0 verwendet. Wenn Sie in einer Sitzung mehrere Verbindungen öffnen wollen, ist es jedoch sinnvoll, das Kommunikationsobjekt individuell zu benennen. Den Namen des Kommunikationsobjektes müssen Sie im Start-Template wieder angeben.



Beachten Sie, dass die Groß-/Kleinschreibung unterschieden wird.

- ▶ Wählen Sie als **Generierungsmethode** *Inline-script*.
- ▶ Geben Sie im Eingabefeld **WTML-Verzeichnis** das Verzeichnis des Unterverzeichnisses forms an, in das die neu generierten Templates abgelegt werden sollen.
- ▶ Wählen Sie **Fertig stellen**, um die FLD-Dateien und die Templates zu erzeugen.

Das Verzeichnis `beispiel_utm\config` enthält danach die erzeugten Felddateien (*.fld), das Verzeichnis `beispiel_utm\config\forms` die generierten Templates (*.htm).

Die Templates sind bereits ablauffähig und können getestet werden.

Im [Abschnitt „Aufbau der generierten Templates“ auf Seite 87](#) finden Sie ein entsprechend generiertes Template.

3.2.4 Lokale Host-Anwendungsnamen festlegen

In der Datei `localapps` müssen Sie die lokalen Host-Anwendungsnamen definieren. Legen Sie in der Datei `beispiel_utm\localapps` folgende Einträge an:

```
*file: localapps
FREE          UPICPTR0
FREE          UPICPTR1
FREE          UPICPTR2
...
FREE          UPICPTR9
```

Erklärung

Es werden 10 freie lokale Anwendungsnamen für die openUTM-Anwendung festgelegt. Damit können bis zu 10 WebTransactions-Sitzungen gleichzeitig mit der openUTM-Anwendung verbunden sein.

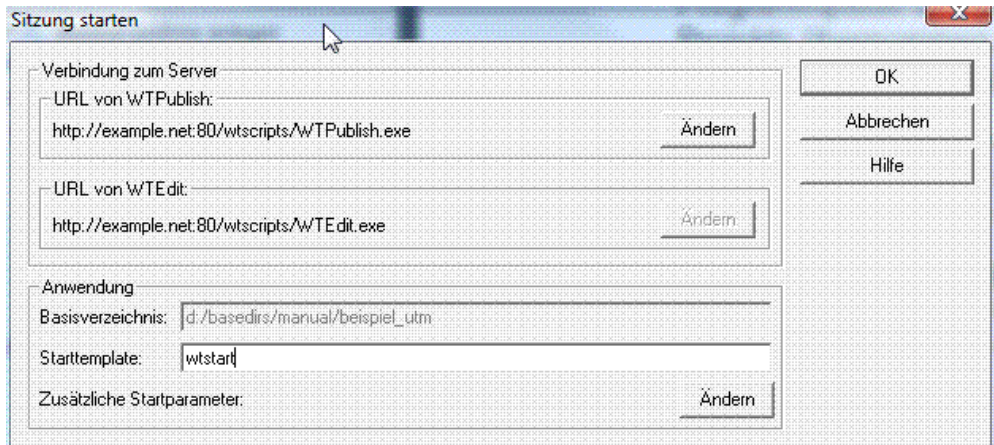


Die Konfigurationsdaten für die openUTM-Anwendung geben Sie in diesem Beispiel beim Sitzungsstart ein. Alternativ dazu können Sie die Daten auch in die `upicfile` eintragen. Näheres siehe [Abschnitt „Datei upicfile“ auf Seite 130](#).

3.2.5 Sitzung starten



Nachdem Sie das Basisverzeichnis angelegt haben, können Sie eine Sitzung zur Host-Anwendung starten.

- ▶ Wählen Sie den Befehl **Datei/Sitzung starten**. Das Dialogfeld **Sitzung starten** wird am Bildschirm eingeblendet.





In diesem Dialogfeld wurden die Verbindungsdaten wie Web-Server-Name und CGI-Programmpfade und der Name des Basisverzeichnisses aus den Einstellungen des Projekts übernommen. Erwartet wird nur noch der Name eines Start-Templates, mit dem die Host-Anwendung gestartet wird.

- ▶ Geben Sie im Eingabefeld **Starttemplate** den Namen eines Start-Templates an, hier `wtstart`. `wtstart.htm` ist ein mit ausgeliefertes Start-Template, das in das Basisverzeichnis kopiert wurde und für alle Host-Anwendungen verwendet werden kann.
- ▶ Starten Sie die Sitzung mit **OK**. Das Dialogfeld wird geschlossen. Der eingestellte Browser wird geöffnet und veranlasst, eine neue WebTransactions-Sitzung mit dem Start-Template `wtstart` aufzurufen. `wtstart` präsentiert im Browser-Fenster ein Formular.

 		main menu	
		[WebTransactions: test environment]	
status	base directory:	d:/base_dirs/manual/beispiel_utm	
workflow	PROTOCOL:	HTTP ▾	
	private WT_SYSTEM:	<input checked="" type="checkbox"/>	
	name of new communication object:	UTM_0	
	create new communication:	<input type="button" value="create"/>	
	connect webService	<input type="button" value="webService"/>	
	terminate session	<input type="button" value="quit"/>	
system parameters	STYLE:	<input type="text"/>	
	LANGUAGE:	<input type="text"/>	
	DEFAULT_FORMAT:	wtstart	
	TIMEOUT_APPLICATION:	120 (2 minutes) ▾	
	TIMEOUT_USER:	600 (10 minutes) ▾	
	COMMUNICATION_INTERFACE_VERSION:	3.0 or higher ▾	
	WTML_VERSION:	3.0 or higher ▾	
	SEARCH_HOST_OBJECTS:	<input type="checkbox"/>	


In diesem Formular des allgemeinen Start-Templates können Sie nun die Verbindungsparameter für WebTransactions angeben, um ein neues Kommunikationsobjekt einzurichten.

- ▶ Wählen Sie in der **PROTOCOL**-Auswahlliste den Eintrag *UTMV4*.
- ▶ Geben Sie den Namen des Kommunikationsobjekts an, hier *UTM_0*. Der Name des Kommunikationsobjekts muss mit dem Namen übereinstimmen, den Sie bei der Generierung der Templates verwendet haben.
- ▶ Klicken Sie nun auf den Knopf **create**, um ein neues Kommunikationsobjekt einzurichten. Ihre Angaben werden von WebTransactions verarbeitet. Das openUTM-spezifische Start-Template *wtstartUTMV4.htm* übernimmt die Kontrolle und zeigt das nächste Formular.

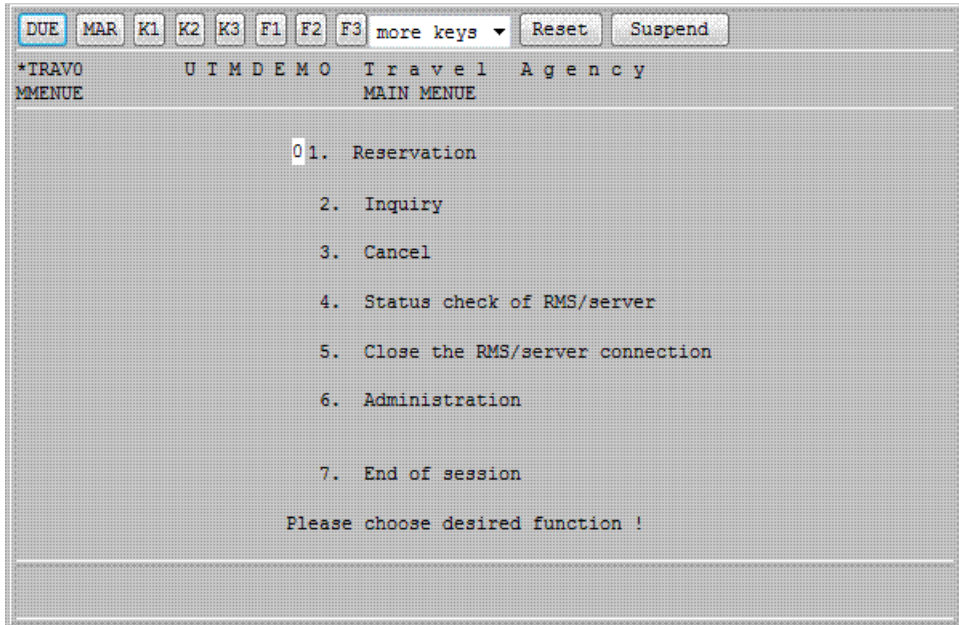
 		UTMV4 communication	
status	communication object:	WT_HOST.UTM_0	
	connection:	down	
workflow	destination:	main menu	<input type="button" value="go to"/>
	access host:	<input type="button" value="open"/>	<input type="button" value="open in linemode"/>
	parameters:	<input type="button" value="update"/>	<input type="button" value="reset"/>
connection parameters directly	APPLICATION_NAME:	VTV10TRV	
	HOST:	NAME: HOST001	
		or IP_ADDRESS:	<input type="text"/>
		with HOST_PORT:	<input type="text"/>
	TAC:	MMENUE	
	UPIC_CODE_CONVERSION:	<input checked="" type="checkbox"/>	
... or via upicfile	SYM_DEST:	<input type="text"/>	
additional connection parameters	APPLICATION_PREFIX:	<input type="checkbox"/>	
	CONVERSATION_TAC:	<input type="text"/>	
	CUT_TAC_FIELD:	<input checked="" type="checkbox"/>	
	HOST_CHAR_CODE:	ASCII	<input type="text"/>
	DISPLAY_EURO:	<input type="checkbox"/>	
	FLD:	TRAV0	<input type="text"/>
	BADTAC:	<input type="text"/>	
	LOCAL_APPLICATION:	<input type="text"/>	
	UPIC_TRACE:	<input type="checkbox"/>	
	UPIC_LIB:	<input type="text"/>	
	UTM_PATH:	<input type="text"/>	
	SECURITY_TYPE:	NONE	<input type="text"/>
	USER:	<input type="text"/>	
	PASSWORD:	<input type="text"/>	
NEW_PASSWORD:	<input type="text"/>		
RESTART:	<input type="checkbox"/>		

- ▶ Tragen Sie bei **APPLICATION_NAME** den Namen der openUTM-Anwendung ein, hier *VTV10TRV*.
- ▶ Geben Sie bei **HOST** unter **NAME** den Namen des Rechners ein, auf dem die openUTM-Anwendung läuft, hier *HOST0001*.

- ▶ Geben Sie bei **TAC** den Transaktionscode ein, mit dem der Vorgang in der openUTM-Anwendung gestartet wird, hier *MMENUE*.
- ▶ Wählen Sie in der **FLD**-Auswahlliste *TRAVO* als erste zu verwendende Formatbeschreibung. In den übrigen Feldern können Sie die gleichnamigen Attribute des Systemobjekts setzen.
- ▶ Klicken Sie nun auf **open**, um eine Verbindung zur Host-Anwendung herzustellen.

 UTMV4 communication	
status	communication object: WT_HOST.UTM_0
	connection: established
workflow	destination: main menu <input type="button" value="go to"/>
	access host: <input type="button" value="send"/> <input type="button" value="receive"/> <input type="button" value="close"/> <input type="button" value="enter dialog"/>
	parameters: <input type="button" value="update"/> <input type="button" value="reset"/>
connection parameters directly	APPLICATION_NAME: <input type="text" value="VTV10TRV"/>
	HOST: <input type="text" value="NAME: HO S T001"/> or IP_ADDRESS: <input type="text"/> with HOST_PORT: <input type="text"/>
	TAC: <input type="text" value="MMENUE"/>
	UPIC_CODE_CONVERSION: <input checked="" type="checkbox"/>
... or via upicfile	SYM_DEST: <input type="text"/>
additional connection parameters	APPLICATION_PREFIX: <input type="checkbox"/>
	CONVERSATION_TAC: <input type="text"/>
	CUT_TAC_FIELD: <input checked="" type="checkbox"/>
	HOST_CHAR_CODE: ASCII <input type="text"/>
	DISPLAY_EURO: <input type="checkbox"/>
	FLD: TRAV0 <input type="text"/>
	BADTAC: <input type="text"/>
	LOCAL_APPLICATION: <input type="text"/>
	UPIC_TRACE: <input type="checkbox"/>
	UPIC_LIB: <input type="text"/>
	UTM_PATH: <input type="text"/>
	SECURITY_TYPE: NONE <input type="text"/>
	USER: <input type="text"/>
	PASSWORD: <input type="text"/>
NEW_PASSWORD: <input type="text"/>	
RESTART: <input type="checkbox"/>	
global host attributes	Padding: <input type="text" value="OUT MSG"/>
	Detect: <input type="text" value="#f"/>
	UnDetect: <input type="text" value="#00"/>
	Read: Modified <input type="text"/>
	FieldLength: Effective length <input type="text"/>
	Update: Only <input type="text"/>
[UTM start parameters]	Cursor: ATTR <input type="text"/>

- ▶ Hier können Sie gegebenenfalls im Abschnitt **global host attributes** Attribute des Objekts `WT_HOST_GLOBALS` setzen.
- ▶ Wählen Sie nun **enter dialog**. Anschließend wird das erste Template des openUTM-Service (TRAVO) angezeigt.



Wenn Sie jetzt die Verbindung zum Host verlassen wollen, müssen Sie zuerst die Host-Anwendung beenden. In diesem Beispiel geben Sie den Wert **7** ein (für **End of Session**) und klicken auf **DUE** oder die Return-Taste, um den Bildschirm abzuschicken. Es wird wieder in das openUTM-spezifische Start-Template verzweigt (siehe auch [Abschnitt „openUTM-spezifisches Start-Template des Start-Template-Sets \(wtstartUTMV4.htm\)“ auf Seite 183](#)). Mit der Auswahl **main menu** und dem Knopf **go to** gelangen Sie in das allgemeine Start-Template zurück. Hier können Sie mit **quit** die WebTransactions-Anwendung verlassen.

In dieser Beispielsitzung fahren Sie wie folgt fort:

- ▶ Geben Sie **1** (für **Reservation**) ein.
- ▶ Schicken Sie den Bildschirm mit **DUE** oder der Return-Taste ab. Das nächste Format der Anwendung (`travel.htm`) wird im Browser angezeigt (siehe folgender Abschnitt).

3.3 Templates bearbeiten

In diesem Abschnitt werden am Beispiel von `trav1.htm` einige Möglichkeiten zur Verschönerung der Templates dargestellt.

Oberfläche des generierten Templates



Die Oberfläche des automatisch generierten Templates ist dem Aussehen und der Funktionalität des FHS-Formats nachempfunden.



Das generierte Template unterscheidet sich vom FHS-Format nur durch die Schaltflächen **DUE**, **MAR**, **K1**, ... **Suspend**. Diese werden immer generiert, da das Standard-Template `wtKeysUTM.htm` inkludiert wird. Eine zusammenhängende Gesamt-Darstellung eines Templates finden Sie im [Abschnitt „Aufbau der generierten Templates“ auf Seite 87](#).

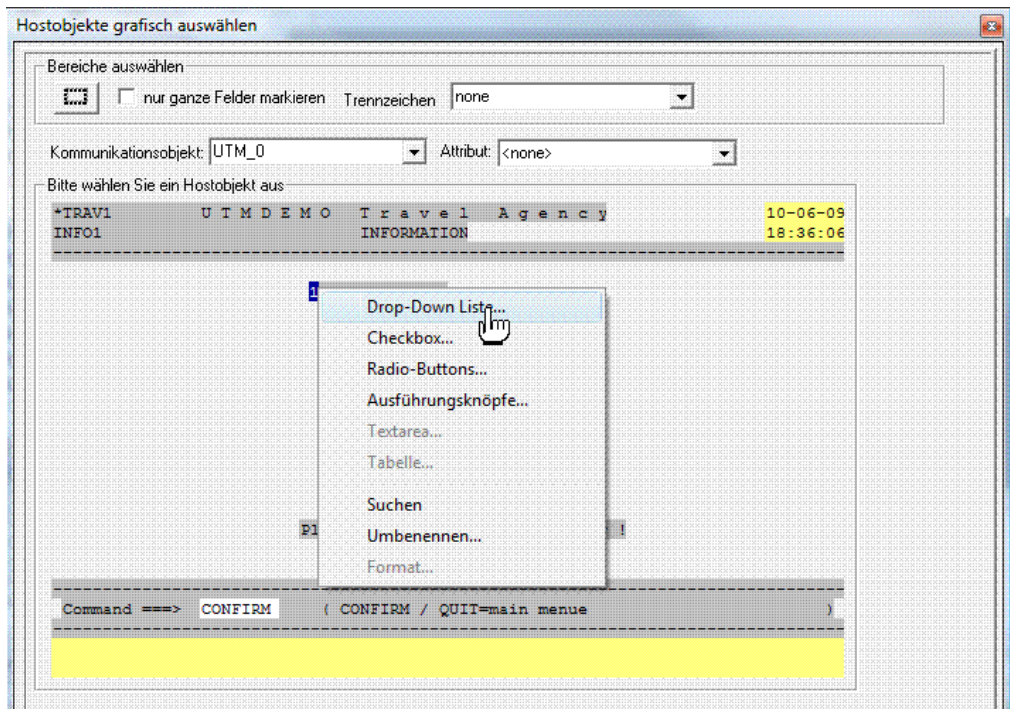
3.3.1 Drop-Down-Liste zur Länderauswahl einfügen

Als Beispiel für eine individuelle Gestaltung wird eine wertorientierte Auswahl (Auswahl durch Eingabe einer Zahl) auf eine Drop-Down-Liste abgebildet, wie in der folgenden Tabelle dargestellt:

Vorher	Nachher
	

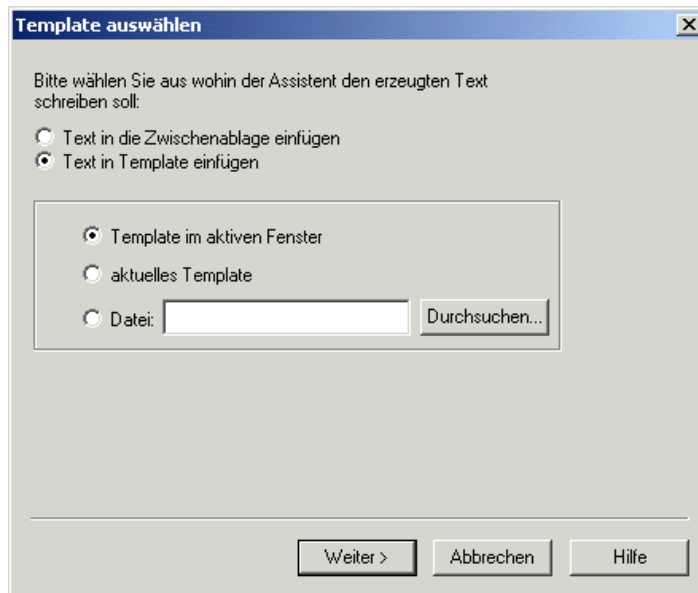
Im FHS-Format und im generierten Template wählt der Anwender das gewünschte Land durch Eingabe der entsprechenden Nummer. Da jedoch für Anwender, die grafische Oberflächen gewohnt sind, die Auswahl über eine Drop-Down-Liste vertrauter ist, soll in das Template eine solche Liste integriert werden.

- ▶ Wählen Sie dazu den Befehl **Datei/aktuelles Template öffnen**, um das formatspezifische Template *TRAVI* im Arbeitsbereich von WebLab zu öffnen.
- ▶ Wählen Sie dann den Befehl **Gestalten/Hostobjekte grafisch auswählen/ Aus einem Kommunikationsobjekt**. Das Dialogfeld **Hostobjekte grafisch auswählen** für das aktuelle Template wird am Bildschirm eingeblendet.



In diesem Dialogfeld wird das Format angezeigt, wie es auch in einer Emulation oder am Terminal dargestellt würde. Alle Ausgabefelder, die nicht veränderbar sind, sind gelb hinterlegt. Eingabefelder dieses Formats sind mit der Farbe weiß hinterlegt.

- ▶ Positionieren Sie den Mauszeiger auf das erste Eingabefeld (durch die Selektion wird das Feld blau) und öffnen Sie das Kontextmenü mit der rechten Maustaste.
- ▶ Wählen Sie aus dem Kontextmenü den Befehl **Drop Down Liste**. Das Dialogfeld **Template auswählen** wird am Bildschirm eingeblendet. Dieses Dialogfeld ist das erste eines Assistenten, der Sie beim Erstellen einer Liste unterstützt.



In diesem Dialogfeld legen Sie fest, in welches Template die Liste eingefügt werden soll. Die Option **Template im aktiven Fenster** ist bereits voreingestellt. Wenn Sie das aktive Template nicht vorher geöffnet haben, wählen Sie die Option **aktuelles Template**.

- ▶ Bestätigen Sie ansonsten die Voreinstellung mit **Weiter >**. Das zweite Dialogfeld **Werte zuordnen** wird am Bildschirm eingeblendet.

Werte zuordnen

Interner Wert:
9

Wert an der Oberfläche:
United Kingdom

Variable auswählen...

Hinzufügen Löschen

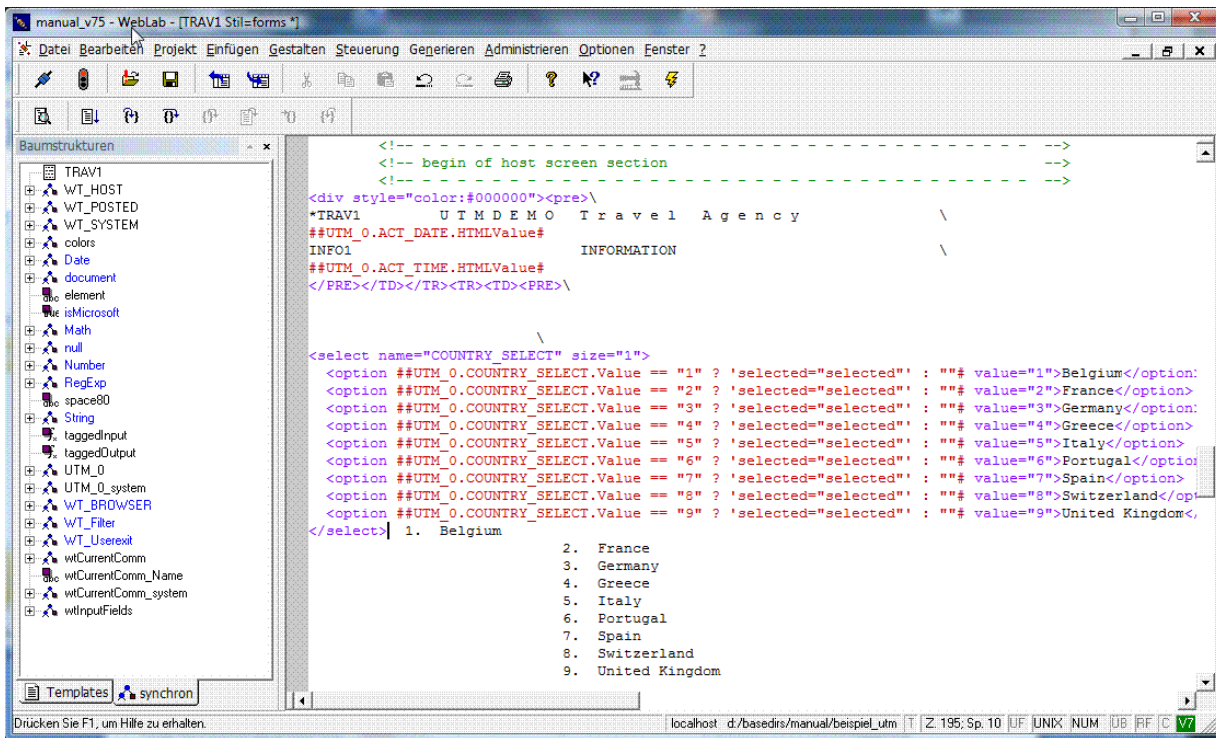
Intern	Wert an der Oberfläche
1	Belgium
2	France
3	Germany
4	Greece
5	Italy
6	Portuoaal

Auswahl nicht anzeigen, wenn der externe Wert leer ist

< Zurück Fertig stellen Abbrechen Hilfe

Der **interne Wert** entspricht in diesem Beispiel dem Zahlenwert, den der Benutzer zur Auswahl eines Menüpunktes im Eingabefeld eingeben muss. Der **Wert an der Oberfläche** ist die zum internen Wert passende Beschreibung und entspricht einem Eintrag in der Auswahlliste.

- ▶ Tragen Sie die internen Werte mit den entsprechenden Beschreibungen (siehe Abbildung auf [Seite 52](#)) in die Eingabefelder ein. Mit der Schaltfläche **Hinzufügen** übernehmen Sie ein solches Wertepaar in die Liste.
- ▶ Bestätigen Sie am Ende der Liste Ihre Eingaben mit **Fertig stellen**. Der entsprechende HTML-Code für die Umsetzung einer Liste wird mit den angegebenen Werten in das Template *TRAVI.htm* eingefügt.
- ▶ Um sich diese Ersetzung anzusehen, blättern Sie im Template *TRAVI*, bis Sie zum Host-Abschnitt gelangen. Dieser Abschnitt beginnt mit dem Kommentar `Begin of Host Screen Section`.



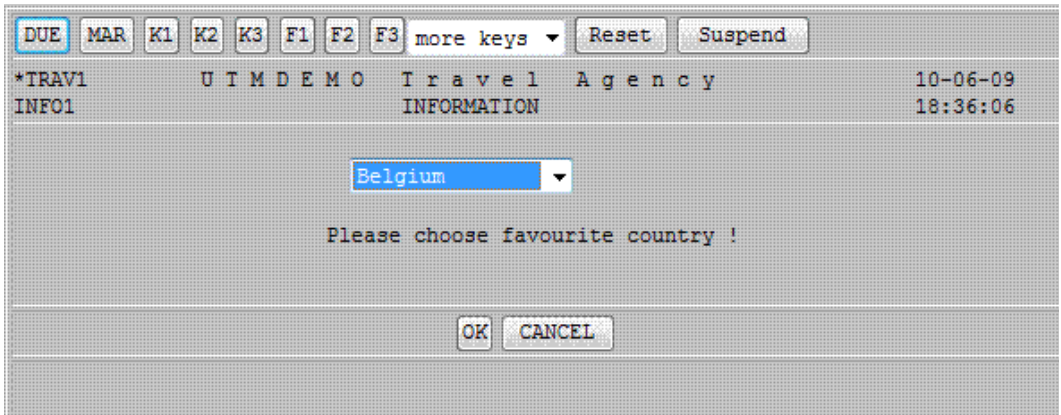
WebLab hat für das Eingabefeld folgenden Code generiert:

```
<select name="COUNTRY_SELECT" size="1">
<option ##UTM_0.COUNTRY_SELECT.Value == "1" ? "SELECTED" : ""# value="1">Belgium</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "2" ? "SELECTED" : ""# value="2">France</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "3" ? "SELECTED" : ""# value="3">Germany</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "4" ? "SELECTED" : ""# value="4">Greece</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "5" ? "SELECTED" : ""# value="5">Italy</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "6" ? "SELECTED" : ""# value="6">Portugal</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "7" ? "SELECTED" : ""# value="7">Spain</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "8" ? "SELECTED" : ""# value="8">Switzerland</option>
<option ##UTM_0.COUNTRY_SELECT.Value == "9" ? "SELECTED" : ""# value="9">United
Kingdom</option>
</select>
```

WebLab generiert also nicht nur eine Drop-Down-Liste mit der Abbildung der Oberflächenwerte auf die intern verwendeten Werte, sondern sorgt auch automatisch dafür, dass in der Liste der Wert voreingestellt ist, der von der Host-Anwendung ausgegeben wurde.

Hierzu enthält jeder OPTION-Tag einen Auswertungsoperator und dieser wiederum jeweils eine IF-Abfrage in JavaScript-Kurzschreibweise: Auf die Bedingung folgt ein Fragezeichen. Hinter dem Fragezeichen wird ein Wert angegeben, der aktuell ist, wenn die Bedingung erfüllt ist. Dahinter folgt ein Doppelpunkt, und dahinter ein Wert für den Fall, dass die Bedingung nicht erfüllt ist.

- ▶ Entfernen Sie nun den überflüssig gewordenen statischen Länderauswahl-Text.
- ▶ Mit dem Befehl **Im Browser aktualisieren** des WebLab-Menüs **Steuerung** können Sie sich nun das Ergebnis anzeigen lassen:



3.3.2 Kommando-Eingabe durch Knöpfe ersetzen

Im nächsten Bearbeitungsschritt soll das Textfeld zur Kommando-Eingabe durch entsprechende Knöpfe ersetzt werden. Hierzu gehen Sie genauso vor, wie beim Erzeugen der Drop-Down-Liste:

- ▶ Positionieren Sie den Cursor im Dialogfeld **Hostobjekte grafisch auswählen** in das Eingabefeld des Kommandos und öffnen Sie mit der rechten Maustaste das Kontextmenü.
- ▶ Wählen Sie den Befehl **Ausführungsknöpfe...** Das Dialogfeld **Template auswählen** wird am Bildschirm eingeblendet. Dieses Dialogfeld ist das erste eines Assistenten, der Sie beim Erstellen einer Liste unterstützt.
- ▶ Akzeptieren Sie die Voreinstellung im Dialogfeld **Template auswählen** mit **Weiter >**. Das Dialogfeld **Werte zuordnen** wird am Bildschirm eingeblendet.
- ▶ Hier geben Sie nun die Werte für die Knöpfe ein. Unter **Interner Wert:** tragen Sie wieder die Werte ein, die an die Host-Anwendung geschickt werden sollen. Im Feld **Wert an der Oberfläche:** bestimmen Sie den Text, den der Benutzer an der Oberfläche sieht.

Werte zuordnen

Interner Wert:
QUIT

Wert an der Oberfläche:
CANCEL Variable auswählen...

Hinzufügen Löschen

Intern	Wert an der Oberfläche
CONFIRM	OK

Auswahl nicht anzeigen, wenn der externe Wert leer ist

< Zurück Fertig stellen Abbrechen Hilfe

- ▶ Bestätigen Sie am Ende der Liste Ihre Eingaben mit **Fertig stellen**. Der entsprechende HTML-Code für die Umsetzung einer Liste wird mit den angegebenen Werten in das Template `TRAV1.htm` eingefügt.

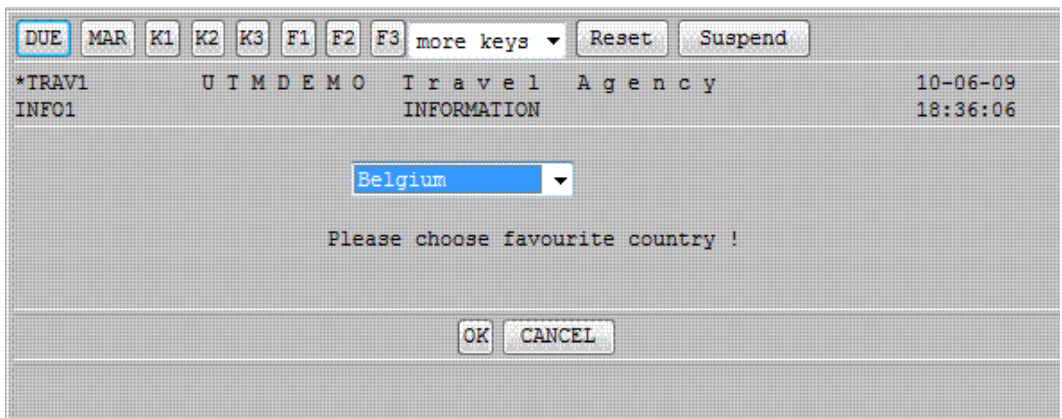
WebLab generiert dafür folgenden HTML-Code:

```
Command ===&#62; \
<input type="submit" name="COMMAND" value="OK">

<input type="submit" name="COMMAND" value="CANCEL">
  ( CONFIRM / QUIT=main menue )
```

Außerdem wird automatisch das entsprechende OnReceive- bzw. OnReceiveScript-Tag so angepasst, dass das Host-Datenobjekt mit dem der gedrückten Taste entsprechenden Wert versorgt wird.

- ▶ Entfernen Sie nun den überflüssig gewordenen statischen Text `Command ===>` und `(CONFIRM / QUIT=main menue)` und richten Sie die Knöpfe aus, z.B. durch Ergänzen mit Leerzeichen.
- ▶ Entfernen Sie aus dem Include-Template `wtKeysUTM.htm` die nicht benötigten Knöpfe (in diesem Beispiel wird davon ausgegangen, dass in der `openUTM`-Anwendung keine Funktionstasten für F1 bis F3 generiert sind). `wtKeysUTM.htm` befindet sich im Basisverzeichnis unter `config/forms`. Da die Wirkung dieser Tasten auch über die PC-Tastatur ausgelöst werden kann, sollten Sie die entsprechenden Zeilen in der Datei `wwwdocs/javascript/wtKeysUTMFHS.js` entfernen. In dieser Datei können Sie auch die Auswahlliste `more keys` anpassen. Mehr Informationen dazu finden Sie im [Abschnitt „Zuordnung der Tasten in wtKeysUTMFHS.js und wtKeysUTMFormant.js“ auf Seite 171](#).
- ▶ Mit dem Befehl **Im Browser aktualisieren** des WebLab-Menüs **Steuerung** können Sie sich nun das Ergebnis anzeigen lassen:



3.3.3 Verweissensitive Grafik einfügen

Im nächsten Bearbeitungsschritt soll für die Länderauswahl ein Bild eingesetzt werden, dessen sensitive Teilbereiche auf unterschiedliche HTML-Seiten verzweigen („clickable image“):



Im Folgenden wird das Template beschrieben, das diese Umsetzung bewirkt:

```
<html>
<head>
<title>Travel Agency - Country</title>
</head>
<wtinclude name="header">
<h2><font color="white">Click the country name you want to travel
to</font></h2>
```

Das Bild (`<input name="EUROPE" type="image">`) liefert zwei Name-Value-Paare zurück, nämlich die *x*- und *y*-Koordinaten der im Bild angeklickten Position. Diese stehen als Attribute des Posted-Objekts zur Verfügung und werden innerhalb eines OnReceive-Scripts von einem Userexit ausgewertet (siehe WebTransactions-Handbuch „Template-Sprache“).

```

<input name="EUROPE" type="image" border="0"
src="##WT_SYSTEM.WWWDOCS_VIRTUAL#/image/europe.gif"><p>
<p>
<p>
<input type="submit" name="COMMAND" value="Cancel">
<p><p><p>
<wtonreceivescript>
<!--
    host = WT_HOST[WT_SYSTEM.HANDLE];
    // Erzeugen eines Objekt der Klasse Userexit, die verwendete
    // Funktion wird in der Bibliothek WTUserexit.[dll|so] gesucht.
    // Sollte die Funktion in einer anderen Bibliothek liegen,
    // muss die Bibliothek im folgenden Konstruktor angegeben
    // werden.
    UserExit = new WT_Userexit();
    // Das Ergebnis des Userexits wird in einer lokalen Variablen
    // zwischengespeichert.
    country=UserExit.UXEurope(WT_Posted.EUROPE.x, WT_POSTED.EUROPE.y);
    // Falls der Benutzer ausserhalb eines sensitiven Bereichs
    // geklickt hat, liefert der Userexit "0", dann
    // bleibt der alte Wert des Hostobjekts erhalten.
    if (country != "0")
        host.COUNTRY_SELECT.Value = country;
    // Nun wird abgefragt, welcher Knopf angeklickt wurde.
    if (WT_POSTED.COMMAND == "Cancel")
        host.COMMAND.Value = "QUIT";
    else
        host.COMMAND.Value = "CONFIRM";
-->
</wtOnReceiveScript>

```

3.4 WebTransactions starten

Eine bearbeitete WebTransactions-Anwendung starten Sie mit WebLab auf die gleiche Weise wie in [Abschnitt „Sitzung starten“ auf Seite 46](#) beschrieben. Mit Hilfe von WebLab können Sie jedoch für die Anwendung ein eigenes Start-Template erstellen, das den Benutzer direkt zum ersten Format der Host-Anwendung bringt.

3.4.1 Start-Template erzeugen

Für das Erstellen eines anwendungs-spezifischen Start-Templates bietet WebLab ein spezielles WtBean an. Es handelt sich dabei um ein standalone WtBean.

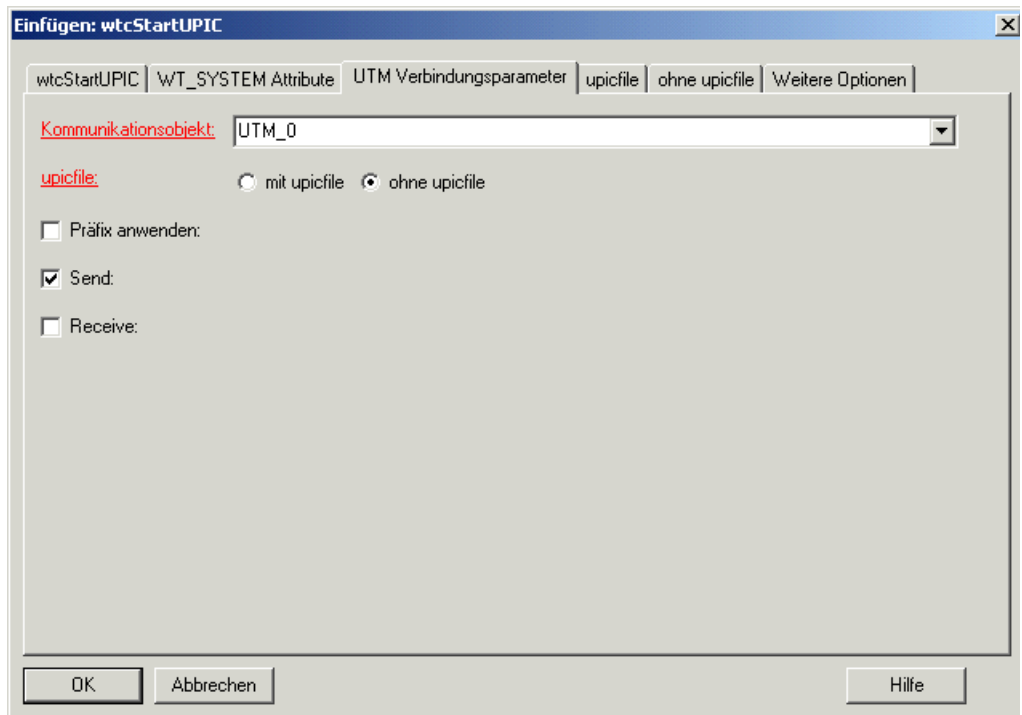


Damit Sie auf WtBeans zugreifen können, muss eine Verbindung zur WebTransactions-Anwendung bestehen.

- ▶ Wählen Sie den Befehl **Datei/Neu/wtcStartUPIC**, um das WtBean aufzurufen. Das Dialogfeld **Einfügen:wtcStartUPIC** wird eingeblendet. Hier können Sie auf vier Registerkarten die Eigenschaften des WtBean bearbeiten.

In der Registerkarte **wtcStartUPIC** legen Sie den Namen und das Verzeichnis für das Start-Template fest. Der Dateiname ist mit `config/forms/StartUPIC.htm` vorgelegt.

- ▶ Geben Sie unter **Dateiname** das Verzeichnis und den Namen des Start-Templates ein, hier *Start_Travel.htm*.
- ▶ Wählen Sie dann die Registerkarte **WT_SYSTEM Attribute**.
In der Registerkarte **WT_SYSTEM Attribute** legen Sie die wichtigsten Attribute des Systemobjekts fest. Für die Beispielsitzung sind die Voreinstellungen ausreichend.
- ▶ Wählen Sie die Registerkarte **UTM Verbindungsparameter**.



- ▶ Geben Sie den Namen für das Kommunikationsobjekt ein, hier *UTM_0*. Beachten Sie, dass der Name des Kommunikationsobjekts mit dem Namen übereinstimmen muss, den Sie im Template `wtstart` definiert haben.
- ▶ Aktivieren Sie für die Beispielanwendung die Optionen **ohne upicfile** und **Send**.
- ▶ Wählen Sie die Registerkarte **ohne upicfile**.

Einfügen: wtcStartUPIC

wtcStartUPIC | WT_SYSTEM Attribute | UTM Verbindungsparameter | upicfile | ohne upicfile | Weitere Optionen

Hostapplikation: VTV10RTV [Auswählen...]

Rechner: HOST0001 [Auswählen...]

Rechner IP Adresse: [Auswählen...]

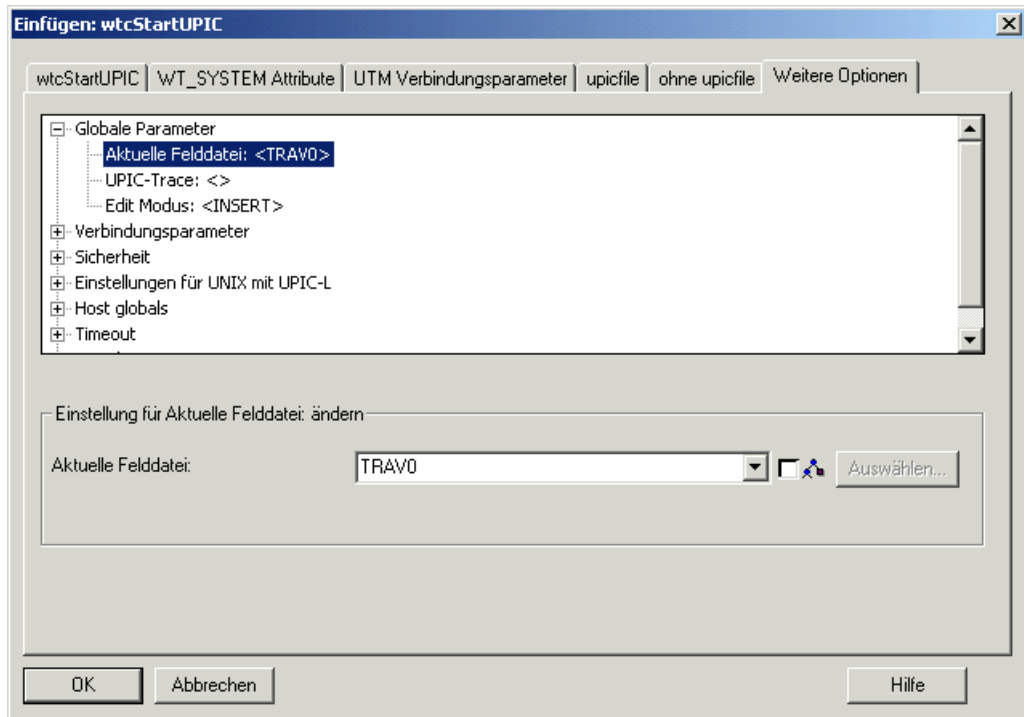
Rechnerport: [Auswählen...]

Start tac: MMENUE [Auswählen...]

UPIC Code Konvertierung:

OK Abbrechen Hilfe

- ▶ Geben Sie in das Feld **Hostapplikation** den Namen der openUTM-Anwendung ein, hier *VTV10TRV*.
- ▶ Tragen Sie in das Feld **Rechner** den Namen des Rechners ein, auf dem die openUTM-Anwendung läuft, hier *HOST0001*.
- ▶ Geben Sie bei **Start tac** den Transaktionscode ein, mit dem der Vorgang in der openUTM-Anwendung gestartet wird, hier *MMENUE*.
- ▶ Markieren Sie die Option **UPIC Code Konvertierung**.
- ▶ Wählen Sie die Registerkarte **Weitere Optionen**.



In dieser Registerkarte können Sie in einer Baumstruktur alle Eigenschaften für die Verbindung zur openUTM-Anwendung bearbeiten.

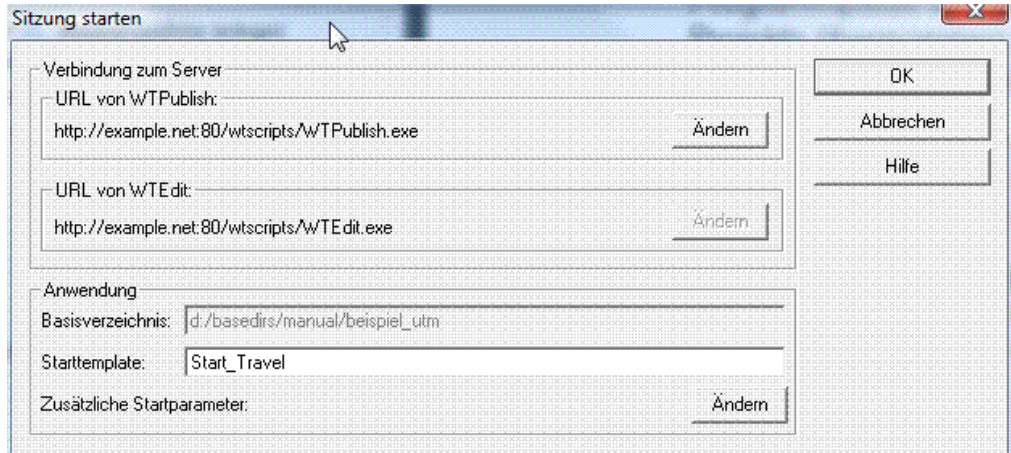
- ▶ Setzen Sie unter **Globale Parameter** den Eintrag **Aktuelle Felddatei** auf *TRAVO*.
- ▶ Für die Beispielanwendung sind keine weiteren Änderungen erforderlich.
- ▶ Wählen Sie **OK**. Das Dialogfeld **Einfügen: wtcStartUPIC** wird geschlossen. Das Start-Template *Start_Travel.htm* wird generiert und im Arbeitsbereich von WebLab angezeigt. Gespeichert wird das Start-Template im Basisverzeichnis unter `config/forms`.

Das Start-Template *Start_Travel.htm* wird nun bei jedem Aufruf die hier einmal vorgenommenen Einstellungen vornehmen. Sie müssen diese Einstellungen nicht mehr bei jedem Sitzungs-Start erneut eingeben, wie es im [Abschnitt „Sitzung starten“ auf Seite 46](#) mit `wtstart.htm` und `wtstartUTMV4.htm` beschrieben wurde.

3.4.2 Sitzung starten mit WebLab

Um eine WebTransactions-Sitzung mit dem anwendungs-spezifischen Start-Template von WebLab zu starten, haben Sie zwei Möglichkeiten:

- Sie wählen den Befehl **Datei/Sitzung starten**. In diesem Fall wird das Dialogfeld **Sitzung starten** eingeblendet.



- ▶ Geben Sie hier im Eingabefeld **Starttemplate** den Namen des anwendungs-spezifischen Start-Templates ein, hier *Start_Travel*.
 - ▶ Bestätigen Sie mit **OK**.
- Sie klicken im Template-Baum mit der rechten Maustaste auf das anwendungs-spezifische Start-Template und wählen im Kontextmenü den Befehl **Sitzung starten**.

In beiden Fällen startet WebLab die Sitzung sofort mit dem ausgewählten Template als Start-Template.

3.4.3 Alternative Möglichkeiten zum Start einer WebTransactions-Anwendung

In obigem Beispiel wird nur erklärt, wie eine WebTransactions-Anwendung während der Entwicklung von WebLab aus gestartet wird. Für den Produktivbetrieb stehen Ihnen andere Möglichkeiten zur Verfügung. Die ausführliche Beschreibung dazu finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

- Sie können die mit ausgelieferte Aufrufseite `wtadm.htm` mit dem Namen des Start-Templates versorgen und für den Benutzer bereit stellen.
- Sie können eine eigene Aufrufseite schreiben, in der WebTransactions über ein Formular oder einen Link gestartet wird.

Beispiel

```
<form method="post" action=
    "/cgi-prefix/WTPublish.exe/basedir?startTemplate">
    <input type="submit" value="Start WebTransactions">
</form>
```

- Sie könnten WebTransactions auch ohne Aufrufseite starten, durch unmittelbare Eingabe des URL:

```
http://WebServer/cgi-prefix/WTPublish.exe/basedir?startTemplate
```

Für *basedir* ist jeweils der absolute Pfad des Basisverzeichnisses anzugeben.

Beispiel

```
http://diana/scripts/WTPublish.exe/d:\webta\apps\
beispiel_utm?Start_Travel
```

4 Basisverzeichnis anlegen

Nach der Installation von WebTransactions auf dem WebTransactions-Server und von WebLab auf Ihrem persönlichen Windows-Rechner können Sie mit Hilfe von WebLab ein oder mehrere Basisverzeichnisse erzeugen. Ein Basisverzeichnis nimmt alle Dateien auf, die WebTransactions für ein bestimmtes Anwendungsszenario konfigurieren.

Bei einer De-Installation von WebTransactions oder beim Installieren einer neuen Produktversion bleiben die individuellen Konfigurationen also erhalten.

4.1 Basisverzeichnis anlegen mit WebLab

Damit Sie ein Basisverzeichnis für eine WebTransactions-Anwendung anlegen können, muss der WebTransactions-Administrator zuvor eine Benutzerkennung für Sie konfigurieren. Anschließend muss er für diese Benutzerkennung ein oder mehrere Pools freigeben, damit Sie dort ein Basisverzeichnis anlegen können.

Bevor Sie ein Basisverzeichnis erzeugen, empfiehlt es sich außerdem, zunächst ein Projekt zu erstellen, in dem die wichtigsten Daten gespeichert werden, die WebLab beim Arbeiten mit der WebTransactions-Anwendung benötigt. Beim Anlegen des Projekts wird Ihnen dann automatisch die Option angeboten, ein Basisverzeichnis zu erstellen.

Gehen Sie folgendermaßen vor:

- ▶ Rufen Sie WebLab auf, z.B. über **Start/Programme/WebTransactions 7.5/WebLab**
 - ▶ Als Einstieg zum Anlegen eines Basisverzeichnisses gibt es folgende zwei Möglichkeiten:
 - ▶ Wählen Sie den Befehl **Projekt/Neu...** und bestätigen Sie die Abfrage, ob ein Basisverzeichnis erstellt werden soll, mit **Ja** (siehe auch [Abschnitt „Projekt anlegen“ auf Seite 35](#)).
- oder
- ▶ Wählen Sie den Befehl **Generieren/Basisverzeichnis ...** und geben Sie bei der folgenden Abfrage an, dass ein neues Projekt erstellt wird.

In beiden Fällen wird das Dialogfeld **Verbinden** geöffnet.

- ▶ Tragen Sie im Dialogfeld **Verbinden** die Verbindungsparameter ein und drücken Sie **OK**.
- ▶ Geben Sie im nachfolgenden Dialogfeld Ihre Benutzerkennung mit Passwort an und klicken Sie auf **OK**.
- ▶ Machen Sie im Dialogfeld **Basisverzeichnis** folgende Einträge:
 - Wählen Sie unter den angebotenen Pools den Pool aus, in dem das Basisverzeichnis angelegt werden soll
 - Tragen Sie den Namen des neuen Basisverzeichnisses ein
 - Aktivieren Sie im Bereich **Host-Adapter** die Option **UTM**
 - Klicken Sie **OK**.

Damit richtet WebLab das Basisverzeichnis mit allen Dateien ein, die für den Ablauf der WebTransactions-Anwendung benötigt werden. Struktur und Inhalt des Basisverzeichnisses sind im WebTransactions-Handbuch „Konzepte und Funktionen“ sowie im [Abschnitt „Struktur eines Basisverzeichnisses“ auf Seite 71](#) beschrieben.

Basisverzeichnis auf eine neue Version umstellen

- ▶ Wählen Sie **Generieren/Basisverzeichnis aktualisieren**. Das Dialogfeld **Basisverzeichnis aktualisieren** wird geöffnet.
- ▶ Wenn nur die Links aus dem Basisverzeichnis in das neue Installationsverzeichnis geändert werden sollen, wählen Sie die Option **Verweise anpassen**. Wählen Sie diese Option, wenn Sie Dateien angepasst haben, die von WebTransactions mitgeliefert oder generiert wurden.
- ▶ Wenn alle Dateien, die beim Erzeugen eines Basisverzeichnisses kopiert oder generiert werden, neu erstellt werden sollen, wählen Sie die Option **Dateien überschreiben**.

4.2 Struktur eines Basisverzeichnisses

Dieser Abschnitt beschreibt ausschließlich die für den openUTM-Host-Adapter spezifischen Besonderheiten des Basisverzeichnisses.



Informationen zur Struktur von Basisverzeichnissen, die für alle Host-Adapter gelten, finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

Konfigurationsdateien `localapps` und `upicfile`

Die Anbindung an eine openUTM-Anwendung kann wahlweise über Konfigurationsdateien oder mit Systemobjekt-Attributen realisiert werden.

Wenn Sie den Weg über Konfigurationsdateien wählen, dann stehen dazu im Basisverzeichnis die Dateien `localapps` zur Verwaltung der lokalen Anwendungen und `upicfile` zur Konfiguration der UPIC-Verbindungen zur Verfügung. Diese Dateien müssen Sie dann für Ihren konkreten Einsatzfall anpassen (siehe [Abschnitt „Datei localapps“ auf Seite 128](#) und [Abschnitt „Datei upicfile“ auf Seite 130](#)).

`upicfile` in DVS-Benutzererkennung bereitstellen (BS2000/OSD)

Die UPIC-Funktionen auf der Systemplattform OSD können nicht aus dem hierarchisch aufgebauten POSIX-Dateisystem (UFS) lesen. Wenn Sie die Datei `upicfile` verwenden, dann muss sie daher in das Datenverwaltungssystem (DVS) unter die Kennung kopiert werden, unter der der `httpd`-Dämon gestartet worden ist.

Kopieren Sie `upicfile` mit dem POSIX-Kommando `bs2cp` in die Benutzererkennung, unter der Sie den HTTP-Server starten. Dies kann `$TSOS` oder eine andere Benutzererkennung `$UserId` sein. Entsprechend müssen Sie `upicfile` dann als `$TSOS.UPICFILE` oder `$UserId.UPICFILE` ablegen.



Die Benutzererkennung, unter der Sie WebTransactions installieren, muss nicht identisch sein mit der Benutzererkennung, unter der Sie den HTTP-Server starten.

Unter OSD müssen Sie jede Zeile in der Datei `upicfile` mit einem Strichpunkt „`,`“ abschließen. Die Datei darf keine Kommentare enthalten. Außerdem sind nur Einträge vom Typ `HD` erlaubt.

Unterverzeichnis config

Für jedes konvertierte Format (Maske) enthält `config` eine Beschreibungsdatei `formatname.fld`, in der die Datenfelder des Formats beschrieben sind. Diese Dateien, auch Felddateien genannt, werden bei der Generierung der Templates automatisch erzeugt.

Für den Zeilenmodus gibt es die Datei mit dem festen Namen `wtlnmode.fld`. Der Name dieser Datei darf nicht verändert werden.

5 Templates generieren

Mit Hilfe der Entwicklungsumgebung WebLab können Sie aus bestehenden alphanumerischen Formaten (Masken) automatisch Templates erzeugen. FHS-Masken müssen Sie zuvor mit dem OSD-Tool `IFG2FLD` aufbereiten.

Bei der Generierung mit WebLab entsteht für jede Maske oder Datenstruktur ein Template und eine Felddatei.

- Templates (*.htm-Dateien)

Die Templates enthalten HTML-Tags für den Aufbau der Web-Seite und WT-Tags für die Steuerung der Kommunikation. Für jedes Format wird jeweils ein Template generiert und standardmäßig im Verzeichnis `basedir/config/forms` abgelegt. Das Template ist dem Aussehen und der Funktionalität des Terminal-Formats nachempfunden. Sie können die Gestaltung des Templates nach Ihren Wünschen anpassen, z.B. mit WebLab (siehe [Abschnitt „Templates bearbeiten“ auf Seite 52](#)).

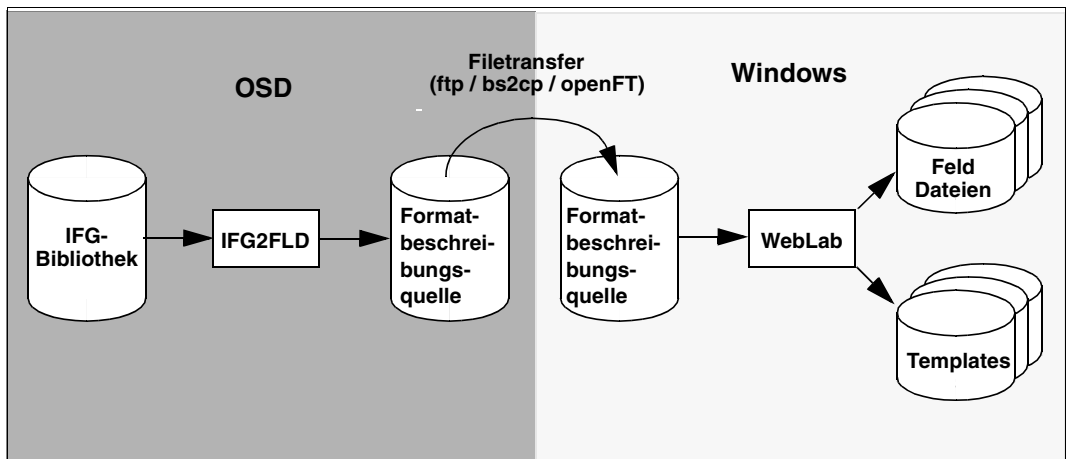
- Felddateien (*.fld-Dateien)

Die Felddateien beschreiben die Datenfelder der Formate (Attribute, Längen, Offsets). Für jedes Format wird jeweils eine Felddatei generiert und im Verzeichnis `basedir/config` abgelegt. Die darin enthaltenen Metadaten benötigt der Host-Adapter zur Interpretation der ausgetauschten Daten. Bei der Kommunikation zwischen WebTransactions und Host-Anwendung werden jeweils nur die Nettodaten über das Netz übertragen. Felddateien sollten Sie nur ändern, um Änderungen in der Host-Anwendung nachzuziehen. Meist ist es jedoch einfacher und sicherer, bei Änderungen der Host-Anwendung die Felddateien neu zu generieren.

5.1 Templates aus FHS-Formaten generieren

Templates aus FHS-Formaten generieren Sie in zwei Schritten:

1. Erzeugen Sie zunächst unter OSD mit dem Programm IFG2FLD aus der IFG-Bibliothek eine Formatbeschreibungsquelle und übertragen Sie diese auf den Entwicklungsrechner. Die Formatbeschreibungsquelle ist das Ergebnis eines IFG2FLD-Laufs, mit dem die Formatbeschreibungen aus der IFG-Bibliothek gelesen und in der Quelle abgelegt werden. Die Formatbeschreibungsquelle enthält alle Feldnamen, die der Entwickler der Host-Formate ursprünglich mit dem Formatierungssystem definiert hat.
2. Danach generieren Sie unter Windows mit WebLab aus dieser Formatbeschreibungsquelle Templates und Felddateien.



Vorgehen

Um aus den FHS-Formaten Templates und FLD-Dateien zu generieren, gehen Sie folgendermaßen vor:

- Für die Unix- und Windows-Plattformen von WebTransactions übertragen Sie eine der folgenden LMS-Bibliotheken binär auf den OSD-Rechner zu der Kennung, unter der die IFG-Bibliothek liegt und nennen diese Bibliothek auf dem OSD-Rechner WTIFG2FLD.LMS:

Bibliothek	Bedeutung
WTifg2fldFTP.lms	für die Übertragung mit FTP
WTifg2fldopenFT.lms	für die Übertragung mit openFT oder mit dem Kommando TRANSFER-FILE im BS2000

Für die Plattform OSD ist keine Übertragung notwendig, da IFG2FLD in der Installationsbibliothek von OSD vorhanden ist.

- ▶ Loggen Sie sich am OSD-Rechner unter dieser Kennung ein.
- ▶ Führen Sie den IFG2FLD-Lauf durch wie im folgenden [Abschnitt „Anwendung von IFG2FLD“](#) beschrieben.
- ▶ Übertragen Sie die Formatbeschreibungquelle im Textmodus auf den Rechner, auf dem WebLab läuft.
- ▶ Geben Sie bei der Template-Generierung im Dialogfeld **FLD und Template generieren** die Formatbeschreibungquelle an, siehe [Abschnitt „Template und FLD-Datei mit WebLab aus der Formatbeschreibungquelle generieren“ auf Seite 77.](#)

5.1.1 Anwendung von IFG2FLD

IFG2FLD ist ein OSD-Programm, deshalb müssen Sie die entsprechende Bibliothek zu einer OSD-Benutzerkennung übertragen. Anschließend können Sie IFG2FLD aus der übertragenen Bibliothek mit folgendem Kommando starten:

```
/START-PROGRAM FROM-FILE=*PHASE(LIBRARY=WTIFG2FLD.LMS,ELEMENT=IFG2FLD)
```

IFG2FLD starten Sie mit dem Kommando `START-PROGRAM`, die Ausgaben werden nach `SYSLST` geschrieben. Folgende Kommandos werden von IFG2FLD akzeptiert:

Kommando	Beschreibung
<code>MAPFILE</code> <i>IFG-Bibliothek</i>	weist die zu bearbeitende IFG-Bibliothek zu.
<code>PROFILE</code> <i>Benutzerprofil</i>	weist ein Benutzerprofil für die Konvertierung zu. Jede IFG-Bibliothek enthält zumindest ein Benutzerprofil um ein Format verarbeiten zu können. Das Benutzerprofil ist eine Menge von Standardwerten für die Handhabung des IFG, für die Eigenschaften der Formate, für die Eigenschaften der Felder des Formats und für die Eigenschaften, die die Programmierung betreffen.
<code>CONVERT</code> <i>Formatname</i> [<i>/version</i>] <i>/ALL</i>	nimmt das angegebene IFG-Format in die Formatbeschreibungquelle auf. Optional können Sie die Version des Formats mit angeben. Wenn Sie die Version nicht angeben, wird das Format mit der höchsten Version aufgenommen. Wenn Sie statt einer Version den Wert <i>/ALL</i> angeben, werden alle Versionen des Formats in die Formatbeschreibungquelle aufgenommen.

Kommando	Beschreibung
CONVERT *ALL [/version /ALL]	nimmt alle IFG-Formate in der IFG-Bibliothek in die Formatbeschreibungsource auf. Optional können Sie die Version der Formate mit angeben. Es werden dann nur die Formate dieses Versionsstandes aufgenommen. Wenn Sie die Version nicht angeben, werden die Formate der höchsten Version in die Formatbeschreibungsource aufgenommen. Wenn Sie statt einer Version den Wert /ALL angeben, werden alle Formate aller Versionen aufgenommen.
END	erzeugt die Formatbeschreibungsource und beendet IFG2FLD.

Beispiel

Für eine Konvertierung mit IFG2FLD sind in OSD z.B. folgende Schritte auszuführen:

```

/ASSIGN-SYSLST ausgabedatei
/START-PROGRAM FROM-FILE=*PHASE(LIBRARY=WTIFG2FLD.LMS,ELEMENT=IFG2FLD)
MAPFILE IFG-Bibliothek
CONVERT *ALL
END
/ASSIGN-SYSLST *P

```

5.1.2 Template und FLD-Datei mit WebLab aus der Formatbeschreibungsource generieren

- ▶ Stellen Sie ggf. mit dem Befehl **Datei/Verbindung öffnen** eine Verbindung zum Basisverzeichnis her.
- ▶ Wählen Sie den Befehl **Generieren/Templates/aus IFG-Bibliothek**, um das Dialogfeld **Optionen für FLD Generierung** zu öffnen.

Optionen für FLD Generierung

Eingabe

Formatbeschreibungsquelle:

Formate:

<input checked="" type="checkbox"/> TRAV0	<input checked="" type="checkbox"/> TRAV6
<input checked="" type="checkbox"/> TRAV1	<input checked="" type="checkbox"/> TRAV7
<input checked="" type="checkbox"/> TRAV2	<input checked="" type="checkbox"/> TRAV8
<input checked="" type="checkbox"/> TRAV3	<input checked="" type="checkbox"/> TRSGNOF
<input checked="" type="checkbox"/> TRAV4	<input checked="" type="checkbox"/> TRSGNON
<input checked="" type="checkbox"/> TRAV5	

Ausgabe

FLD-Verzeichnis:

nur für Teilformatel

Verarbeitung von Teilformaten

- ▶ Geben Sie in der Gruppe **Eingabe** die Formatbeschreibungsource ein. Die dort beschriebenen Formate werden in der Liste **Formate** angezeigt.
- ▶ Wählen Sie aus der Liste durch Anklicken die Formate, für die Templates und FLD-Dateien generiert werden sollen.
- ▶ Wählen Sie dann in der Gruppe **Ausgabe** das Verzeichnis, in das die generierten FLD-Dateien geschrieben werden sollen. Wenn eine Sitzung mit dieser WebTransactions-Anwendung gestartet ist, wird der Pfadname automatisch eingeblendet.

- Bestätigen Sie mit **Weiter**. Das Dialogfeld **FLD und Template generieren** wird eingeblendet.

In diesem Dialogfeld können Sie mit den Parametern steuern, wie die formatspezifischen Templates generiert werden. Die ausführliche Beschreibung der Generierungsoptionen finden Sie in der Online-Hilfe.

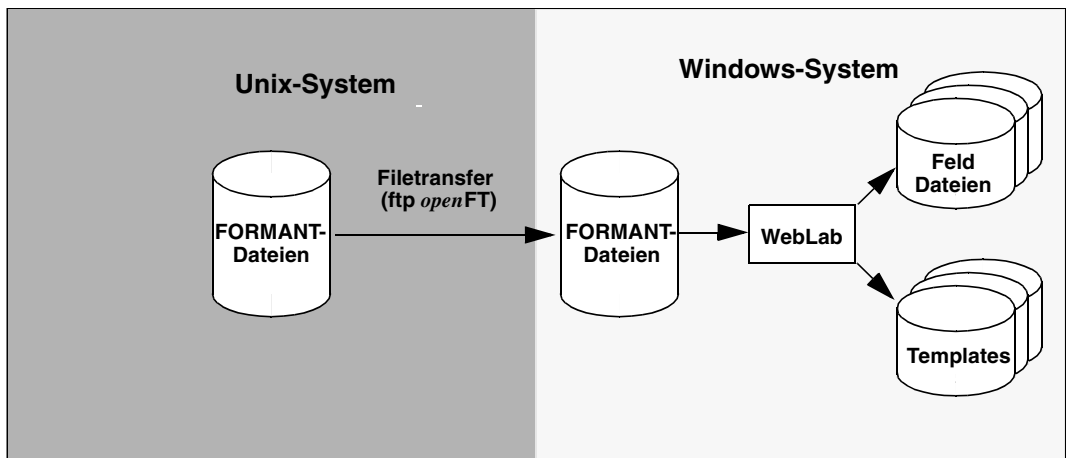
- Geben Sie im Feld **Master-Template** das zu verwendende Master-Template an. Dieses ist bei Vollformaten und bei Teilformaten unterschiedlich:
 - Bei Vollformaten geben Sie das Master-Template `UTM.wmt` an, siehe hierzu auch [Abschnitt „Master-Template UTM.wmt“ auf Seite 96](#).
 - Bei Teilformaten müssen Sie das Master-Template `UTMpartial.wmt` verwenden, siehe [Abschnitt „Aufbau des Master-Templates UTMpartial.wmt“ auf Seite 102](#).
- Geben Sie in der Gruppe **Ausgabe** das Verzeichnis an, in das die generierten Templates geschrieben werden sollen.
 Voreingestellter Pfad für Templates (HTML-Ausgabe): `basedir/config/forms`

- ▶ Wenn Sie mehrere Host-Anwendungen mit möglicherweise gleichen Formatnamen integrieren wollen, markieren Sie die Option **Anwendungspräfix verwenden**. WebLab setzt dann vor den eigentlichen Dateinamen für FLD-Datei und Template ein Präfix. Der Dateiname setzt sich dann zusammen aus:
`[comobj@]formatname.{fld|htm}`.
- ▶ Bestätigen Sie die Generierungsoptionen mit **Fertig stellen**.
- ▶ Schließen Sie das Dialogfeld **FLD und Template generieren** mit **OK**. WebLab erzeugt dann für die ausgewählten Formate je ein HTML-Template und eine FLD-Datei:
 - HTML-Templates werden von WebTransactions zur Laufzeit interpretiert und bestimmen die im Browser dargestellte Oberfläche.
 - FLD-Dateien sind spezielle Beschreibungsdateien. Sie werden von WebTransactions zur Laufzeit benötigt und von der WebTransactions-Entwicklungsumgebung WebLab verwendet, um die Funktion „grafische Objektauswahl“ zu realisieren. Damit WebTransactions während der Laufzeit auf diese Dateien zugreifen kann, müssen sie unter `basedir\config` stehen.

5.2 Templates aus FORMANT-Formaten generieren

Templates aus FORMANT-Formaten generieren Sie in drei Schritten:

1. Erstellen Sie das Konverter-Format `<format_name>.txt` mittels `formantconv`.
2. Übertragen Sie die FORMANT-Dateien (Konverter-Format und Adressierungshilfe im passenden Format (*, +, #) für die Sprache C `<formatname>a.h`) auf den Entwicklungsrechner.
Bei Bedarf – beispielsweise, wenn eine Adressierungshilfe nur für COBOL vorliegt – erstellen Sie diese Adressierungshilfe durch Speichern aus `formantgen` heraus.
3. Generieren Sie unter Windows mit WebLab aus diesen FORMANT-Dateien Templates und Felddateien.



5.2.1 Template und FLD-Datei mit WebLab generieren

- ▶ Stellen Sie ggf. mit dem Befehl **Datei/Verbindung öffnen** eine Verbindung zum Basisverzeichnis her.
- ▶ Wählen Sie den Befehl **Generieren/Templates/aus FORMANT-Datei (für openUTM)**, um das Dialogfeld **Optionen für FLD Generierung** einzublenden.

Konvertierungsweise

Jedes FORMANT-Format wird durch eine Formatbeschreibungsdatei und eine Adressierungshilfe definiert. WebLab kann sowohl für einzelne als auch für mehrere Formate Templates und FLD-Dateien generieren. Bei mehreren Formaten müssen deren Beschreibungsdateien in Verzeichnissen zusammengefasst sein.

Format-Typ

Bestimmt den Typ des Formats.

Formant-Datei(en)

Formant-Adressierungshilfe(n)

Je nach ausgewählter Konvertierungsweise geben Sie hier eine einzelne Datei oder ein Verzeichnis an, aus der WebLab die FLD-Dateien und Templates generiert.

- ▶ Im Feld **FLD-Verzeichnis** müssen Sie das Verzeichnis `config` angeben. Die neu generierten FLD-Dateien werden im Verzeichnis `basedir/config` abgelegt.
- ▶ Bestätigen Sie mit **Weiter**. Das Dialogfeld **FLD und Template generieren** wird eingeblendet (siehe [Seite 78](#)).

5.3 Aufbau der Felddateien (FLD-Dateien)

WebLab legt die Struktur eines FHS- oder FORMANT-Formats in der Felddatei als Datensatz ab. Dies sind die Metadaten über den strukturellen Aufbau der Daten, die zwischen der Host-Anwendung und WebTransactions ausgetauscht werden. Diese statischen Metadaten werden also nicht mit den Nachrichten geschickt, sondern in einer Datei abgelegt. Diese wird beim ersten Bearbeiten eines Formats durch WebTransactions gelesen. Ihre Informationen stehen dann der Host-Adapter für die Interpretation der Nettodaten (Host-Datenobjekte) von der Host-Anwendung zur Verfügung.

Das Format der Felddatei entspricht dem einer *.ini-Datei unter Windows:

- Die Datei ist in verschiedene Abschnitte (Sections) unterteilt, die die einzelnen Bereiche eines Formats beschreiben (siehe Tabelle auf [Seite 83](#)).
- Abschnitte können auch völlig entfallen, wenn keine diesen Abschnitten zugeordneten Schlüsselwörter in der Felddatei enthalten sind.
- Boolesche Attribute haben die Werte 0 oder 1, alle anderen Attribute können nur die Werte annehmen, die bei ihrer Beschreibung angegeben sind.
- Falls nicht anders angegeben, sind alle Zahlenwerte ganzzahlig und > 0 .
- Kommentare müssen in Spalte eins mit Strichpunkt (;) beginnen und dürfen maximal eine Zeile (256 Zeichen) lang sein. Leerzeilen sind erlaubt.

Folgende Sections werden unterstützt:

Sectionname	Inhalt
FormProperties	Die Version und der Formattyp sowie die Globalattribute des Formats
AttributOffsets	Die relativen Offsets der grundlegenden Attribute im Feldattributblock
<Feldname>	Hier sind die einzelnen Felder der Reihe nach beschrieben. Zu der Beschreibung gehören Daten wie Name, Länge und Offset. Der Feldname darf nur aus Buchstaben, Ziffern, Unterstrich (_) und Dollarzeichen (\$) bestehen und darf nicht mit einer Ziffer beginnen. Er setzt sich zusammen aus dem originalen Namen mit folgenden Änderungen: <ul style="list-style-type: none"> – Bindestrich ('-') wird zu Unterstrich – beginnenden Ziffern wird ein Dollarzeichen ('\$') voran gestellt – alle nicht erlaubten Zeichen werden durch Dollarzeichen ersetzt – nicht eindeutige Namen erhalten ab dem zweiten Auftreten ein Suffix in Form eines Dollarzeichens und einer Laufnummer
FieldMatching	Stellt die Verbindung her zwischen den IFG- oder FORMAT-Feldnamen und den generischen Feldnamen (E_yy_xxx_III)

Folgende Schlüsselwörter werden unterstützt:

Die Tabelle enthält alle Schlüsselwörter. Welche Schlüsselwörter in einem konkreten Fall jeweils enthalten sind, kann je nach Typ des Formats (*,+,#) und IOTYPE des Feldes variieren.

Schlüsselwort	Section	Beschreibung
AttributeLength	FormProperties	Länge des Globalattribut-Blocks bei #-Formaten
Background Colour	FormProperties	Hintergrundfarbe des Formats: WHITE (normale Helligkeit) oder GRAY (halbhell)
BaseVariant	FormProperties	Basisvariante des Zeichensatzes (nur bei 8-Bit-Formaten)
CursorControl	FormProperties	Offset des Global-Attributs CURSOR CONTROL
CursorPosition	FormProperties	Offset des Global-Attributs CURSOR POSITION
DateFormat	FormProperties	Darstellungsformat für Felder mit DataType=DATE
DecimalSeparator	FormProperties	Dezimaltrennzeichen für numerische Felder (DataType=NUMBER)
DigitSeparator	FormProperties	Zifferntrennzeichen für numerische Felder (DataType=NUMBER)
FieldsDetect	FormProperties	Offset des Global-Attributs FIELDS DETECTION
FieldsMod	FormProperties	Offset des Global-Attributs FIELDS_MOD

Schlüsselwort	Section	Beschreibung
FieldsValid	FormProperties	Offset des Global-Attributs FIELDS VALIDATION
FormatLength	FormProperties	Länge des Formatinhalts
FormatName	FormProperties	Formatname
FormattingSystem	FormProperties	Formatierungssystem: FHS, FORMANT
FormatType	FormProperties	Typ des Formats: #, + (ohne Ausrichtung), *
InputKeyClass	FormProperties	Offset des Global-Attributs INPUT KEY CLASS
InputKeyNumber	FormProperties	Offset des Global-Attributs INPUT KEY NUMBER
PopUp	FormProperties	Gibt an, ob es sich um eine Popup-Box handelt. Mögliche Werte: 0, 1
Protocol	FormProperties	Typ des Anwendungsprotokolls, hat immer den Wert UTM
ScreenDimensions	FormProperties	Bildschirmdimension in der Form <i>zeilenxspalten</i> , z.B. 24x080
TimeFormat	FormProperties	Darstellungsformat für Felder mit DataType=TIME
UndefinedValues	FormProperties	Offset des Global-Attributs UNDEFINED VALUES
UserexitRc	FormProperties	Offset des Global-Attributs USER EXITROUTINE RC
Version	FormProperties	Version von WebTransactions
AttributeCombination	AttributOffsets	Das Offset innerhalb eines Attributblocks, ab dem das Attribut AttributeCombination beginnt.
AttributeLength	AttributOffsets	Länge des Feldattribut-Blocks bei #-Formaten
Color	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Color beginnt
Cursor	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Cursor beginnt
EditRC	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut EditRC beginnt
EditState	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut EditState beginnt
FieldLength	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut FieldLength beginnt
InputControl	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut FieldInput beginnt (Eingabepflicht)
InputState	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut InputState beginnt (Eingabeart des Feldwerts seit der letzten Neuausgabe des Formats)
InputStateAct	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut InputStateAct beginnt (Kennzeichnung der aktuell eingegebenen Felder)
Intensity	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Intensity beginnt (Darstellungsweise des Felds, z.B. fett oder normal)

Schlüsselwort	Section	Beschreibung
Inverse	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Inverse beginnt (Feld invers dargestellt)
NumAttributes	AttributOffsets	Anzahl der Feldattribute bei #-Formaten
OutputControl	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut OutputControl beginnt
Protection	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Protection beginnt
Underline	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Underline beginnt (Feld unterstrichen)
Visibility	AttributOffsets	Offset innerhalb eines Attributblocks, ab dem das Attribut Visibility beginnt (Feld sichtbar)
Align	<Feldname>	Ausrichtung des Feldes: LEFT, RIGHT
AttributOffset	<Feldname>	Offset, ab dem die Attribute des Feldes beginnen. Innerhalb der Attribute kann dann mit den oben aufgeführten relativen Offsets navigiert werden.
AutoInput	<Feldname>	automatische Eingabe des Feldes. Mögliche Werte: Y,N.
Blink	<Feldname>	Feld blinkend dargestellt: 1, 0
Case	<Feldname>	Konvertierung der Buchstaben: UPPER, LOWER, BOTH; (in FHS UPPERCASE)
Color	<Feldname>	Vordergrundfarbe des Feldes, mögliche Werte:, 1 (Rot), 2 (Grün), 3 (Gelb), 4 (Blau), 5 (Magenta), 6 (Cyan), Voreinstellung (schwarz)
Column	<Feldname>	Spaltennummer
DataOffset	<Feldname>	Offset, ab dem die Daten des Feldes beginnen.
DataType	<Feldname>	Datentyp des Feldes: NUMERIC, ALPHANUMERIC, ALPHA, DATE, TIME.
DefaultCursor	<Feldname>	Gibt an, ob in diesem Feld der Cursor stehen soll, wenn vom Programm nicht anders gesteuert. Mögliche Werte: 1,0
Detectable	<Feldname>	Gibt an, ob das Feld markierbar ist. Mögliche Werte: 1,0.
DisplayLength	<Feldname>	Feldlänge auf dem Bildschirm
FillCharInput	<Feldname>	Feldfüllzeichen für die Dateneingabe in Richtung openUTM-Teilprogramm. Mögliche Werte wie im IFG-Report; Standard: ' '.
FillCharOutput	<Feldname>	Feldfüllzeichen für die Datenausgabe des openUTM-Teilprogramms. Mögliche Werte wie im IFG-Report; Standard: ' '.
GroupDigit	<Feldname>	Gibt an, ob Zifferngruppierung erlaubt ist (für Felder mit DataType=NUMERIC). Mögliche Werte:Y/N
Intensity	<Feldname>	Intensität des Feldes: BRIGHT, NORMAL

Schlüsselwort	Section	Beschreibung
Invers	<Feldname>	Gibt an, ob das Feld Invers dargestellt werden soll. Mögliche Werte: 1, 0
IOType	<Feldname>	Art des Feldes: INPUT, OUTPUT, TEXT, FIXTEXT. Felder vom Typ FIXTEXT sind Felder, auf die nicht über das Programm zugegriffen werden kann, sondern nur über die FLD-Datei. Der Wert eines derartigen Feldes wird im Attribut Text ausgegeben, siehe unten.
Length	<Feldname>	Länge des Feldes in der Adressierungshilfe. Der Wert muss > 0 sein. Der Feldlänge in der Adressierungshilfe kann sich von der Feldlänge auf dem Bildschirm (DisplayLength) unterscheiden, z.B. bei Datumsfeldern (DataType=DATE) oder bei numerischen Feldern (DataType=NUMERIC)
Line	<Feldname>	Zeilennummer
Mandatory	<Feldname>	Gibt an, ob ein Feld Pflichtfeld ist. Mögliche Werte: 1, 0
NumDecimal	<Feldname>	Anzahl der Zeichen nach dem Dezimaltrennzeichen (für Felder mit DataType=NUMERIC)
Protection	<Feldname>	Schreibschutz des Feldes: 1, 0
Sign	<Feldname>	Gibt an, ob ein Vorzeichen erlaubt ist (für Felder mit DataType=NUMERIC). Mögliche Werte:Y/N
SignFloat	<Feldname>	Gibt an, ob das Vorzeichen rechts der Zahl stehen darf (für Felder mit DataType=NUMERIC). Mögliche Werte:Y/N
SuppressZero	<Feldname>	Gibt an, ob Nullenunterdrückung erlaubt ist (für Felder mit DataType=NUMERIC). Mögliche Werte:Y/N
Text	<Feldname>	Inhalt für IOType=FIXTEXT, eingeschlossen in einfache Hochkommas (' <i>text</i> ').
Underline	<Feldname>	Feld unterstrichen dargestellt: 1, 0
UTMControl	<Feldname>	Gibt an, ob das Feld ein openUTM-Steuerfeld ist: 1,0
Visibility	<Feldname>	Gibt an, ob das Datenfeld sichtbar ist. Mögliche Werte: 1,0
<IFG-Name> oder <FORMANT-Name>	FieldMatching	Beschreibt die Abbildung zwischen IFG- bzw. FORMANT-Feldnamen und dem generischen Namen der Form E_yy_xxx_lll, z.B. BERUF=E10_005_36

5.4 Aufbau der generierten Templates

Dieser Abschnitt beschreibt Templates, die aus FHS- /FORMANT-Vollformaten generiert wurden. Über Templates, die aus FHS-/FORMANT-Teilformaten generiert wurden, informiert der [Abschnitt „Aufbau des Master-Templates UTMpartial.wmt“ auf Seite 102](#).

Jedes generierte Template basiert auf dem Master-Template, das Sie beim Generieren angeben. Das mit ausgelieferte Standard-Master-Template UTM.wmt ist im [Abschnitt „Master-Template UTM.wmt“ auf Seite 96](#) beschrieben.

Im Folgenden wird der Aufbau eines generierten Templates anhand eines Beispiels erläutert.

Die Kommentare sind die Expansion der Anweisung %%GenerationInfo%.

```
<html>

<wtrem>*****</wtrem>
<wtrem>**   WTML document: TRAV3                               **</wtrem>
<wtrem>*****</wtrem>
<wtrem>**
<wtrem>** Document generation based on Master Template :     **</wtrem>
<wtrem>**   C:\Programme\WebTransactionsV75\web\lab\UTM.wmt   **</wtrem>
<wtrem>**
<wtrem>** Generated at Wed Jun 09 13:54:57 2010              **</wtrem>
<wtrem>**
<wtrem>** Options used by the generator :                       **</wtrem>
<wtrem>**   - %OPTIONS:                                           **</wtrem>
<wtrem>**     CommObj = UTM_0                                     **</wtrem>
<wtrem>**     NationalVariant = International - PartialFormatMode = No **</wtrem>
<wtrem>**   - %LINES:                                             **</wtrem>
<wtrem>**     TaggedInput = Enabled - TaggedOutput = Enabled      **</wtrem>
<wtrem>**     DisplayAttributes = No - CursorInProtectedField = No **</wtrem>
<wtrem>**     Generate = Inline                                     **</wtrem>
<wtrem>**     CellsDelimiter = "-"                                 **</wtrem>
<wtrem>**     CellsDelimiterReplace = </PRE></TD></TR><TR><TD><PRE>\   **</wtrem>
<wtrem>**   - %RECEIVES:                                          **</wtrem>
<wtrem>**     Parameters not specified                             **</wtrem>
<wtrem>*****</wtrem>
<wtrem>** WebTransactions V7.5                               Fujitsu Technology Solutions 2010 **</wtrem>
<wtrem>*****</wtrem>
```

Es werden Referenzen auf das Kommunikations-Objekt und das dazu spezifische System-Objekt-Attribut angelegt, um einheitlich auf die Verbindungsparameter und Host-Objekte zugreifen zu können. UTM_0 ist der Name, den Sie beim Generieren im Feld **Kommunikationsobjekt** ausgewählt haben.

```

<wtoncreatescript>
<!--
  {{{WebLab(assignCommunicationObject)
  UTM_0 = WT_HOST.active || WT_HOST.UTM_0;
  if (UTM_0.WT_SYSTEM != null)
    UTM_0_system = UTM_0.WT_SYSTEM;  // communication specific system object
  else
    UTM_0_system = WT_SYSTEM;        // global system object
  }}}
  // propagate communication object to included WTML documents ////////////////
  wtCurrentComm = UTM_0;
  wtCurrentComm_system = UTM_0_system;
-->
</wtoncreatescript>

```

Mit dem <head>-Tag wird der Kopf der HTML-Seite definiert. Dieser enthält den Titel sowie globale Werte.

```

<head>
<title>WebTransactions V7.5 - application ##UTM_0_system.SYM_DEST#</title>
##WT_SYSTEM.CGI.HTTP_USER_AGENT.indexOf( 'MSIE' ) >= 0 ?
  '<meta http-equiv="Pragma" content="no-cache"/>' :
  '<meta http-equiv="Cache-Control" content="no-cache"/>'#
<wtif (WT_BROWSER.acceptClass)>
  <style type="text/css">
    input {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
    }
    input.box {
      border:       0 solid;
      padding:      1px 0 1px 0;
      margin-left:  -1px;
      margin-top:   ##WT_BROWSER.marginTop#px;
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      color:        #000000;
      background-color: #FFFFFF;
    }
    input.button {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      border-width: 1pt;
      margin-left: -1pt;
    }
    select {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
    }
    pre {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
    }
  </style>
</wtif>

```



```

        margin:      0;
    }
</style>
</wtif>
</head>

```

Im BODY-Abschnitt werden ein Formular mit dem aktuellen Format (<form>) sowie eine Tabelle geöffnet. Diese Tabelle umrahmt das ganze Format.

Für die Bedienung des Dialogs werden mit <wtInclude> die Templates wtKeysUTM.htm und wtBrowserFunctions.htm aufgerufen, um eine möglichst genaue 1:1 Darstellung zu unterstützen. Deren Bedienelemente werden somit Teil des Formulars.

```

<body bgcolor="#C0C0C0">
<form WebTransactions name="TRAV3">
  <table frame="border" rules="all">
    <tr>
      <td>
        <wtinclude name="wtBrowserFunctions">
        <wtinclude name="wtKeysUTM">
        <wtif (UTM_0_system.FORMTPL)>
          <wtinclude Name="##UTM_0_system.FORMTPL#">
        </wtif>
      </td>
    </tr>
    <tr>
      <td>

```

In diesem wtOnCreate-Skript werden die Darstellungsattribute für die Host-Objekte festgelegt, wobei die Funktion taggedOutput() die Ausgabefelder bearbeitet, die Funktion taggedInput() die Eingabefelder.

```

<wtoncreatescript>
  <!--
    colors = new Array('RED','GREEN','YELLOW','BLUE','MAGENTA', 'CYAN', 'WHITE');
    space80 = " ";

    function taggedInput( hostObject )
    {
      if ( hostObject.Protected == 'Y' )
      {
        taggedOutput( hostObject );
        return;
      }
      currentLength = hostObject.Length;
      input = '<input type='+ (hostObject.Visible== 'N' ? '"password"' : '"text"' ) ;
      if ( WT_BROWSER && (WT_BROWSER.is_ie || WT_BROWSER.is_ns6lup) )
      {
        input += ' class="box" style="width:' + (currentLength *
          WT_BROWSER.charWidth + 1) + 'px';
        input += (hostObject.Blinking == 'Y' ? ' ; background-color:#FFC0C0' : '');

```

```

        input += (hostObject.Underline == 'Y' ? ( hostObject.Intensity == 'N' ?
            '; color:#A0A0FF' : '; color:#0000A0' ) :
            ( hostObject.Intensity == 'N' ? ';' : color:#A0A0A0' : '' )) + ''';
    }
    input += ' name="' + hostObject.Name + '" size="' + currentLength
        + '" maxlength="' + currentLength
        + '" value="' + hostObject.Value
        + (hostObject.DataType == 'Numeric'? " numeric="1':"')
        + (hostObject.Detectable == 'Y'? " markable="1':"')
        + '" />';
    document.write( input );
}

function taggedOutput( hostObject )
{
    if ( hostObject.Protected == 'N' )
    {
        taggedInput( hostObject );
        return;
    }
    if ( hostObject.Visible == 'Y' )
    {
        output = hostObject.HTMLValue;
        if ( hostObject.Inverse == 'Y' )
        {
            if ( hostObject.Color && hostObject.Color.toUpperCase() != 'N' )
                output = '<font color=#000000 style=\"background-color:' +
                    colors[hostObject.Color-1] + '\">' + output + '</font>';
        }
        else if ( hostObject.Color && hostObject.Color.toUpperCase() != 'N' )
            output = '<font color=' + colors[hostObject.Color-1] + '>' +
                output + '</font>';
        if (hostObject.Intensity == 'H')
            output = '<b>' + output + '</b>';
        if (hostObject.Blinking == 'Y')
            output = '<i>' + output + '</i>';
        if (hostObject.Underlined == 'Y')
            output = '<u>' + output + '</u>';
        document.write( output );
    }
}

```

```

else
{
    document.write( space80.substr(0,hostObject.Length));
}
}
//-->
</wtoncreatescript>

```

In diesem Teil wird der Inhalt des Formats auf HTML-Tags abgebildet und die Tabelle anschließend geschlossen.

```

<!-- ----- -->
<!-- begin of host screen section -->
<!-- ----- -->
<div style="color:#000000"><pre>\
*TRAV3          U T M D E M O   T r a v e l   A g e n c y   \
##UTM_0.ACT_DATE.HTMLValue#
INFO3          INFORMATION   \
##UTM_0.ACT_TIME.HTMLValue#
</PRE></TD></TR><TR><TD><PRE>\

    Golf course \
##UTM_0.GOLF_COURSE.HTMLValue# in \
##UTM_0.GOLF_COUNTRY.HTMLValue#
    From \
##UTM_0.FIRST_DAY_YEAR.HTMLValue#-\
##UTM_0.FIRST_DAY_MONTH.HTMLValue#-\
##UTM_0.FIRST_DAY_DAY.HTMLValue# to \
##UTM_0.LAST_DAY_YEAR.HTMLValue#-\
##UTM_0.LAST_DAY_MONTH.HTMLValue#-\
##UTM_0.LAST_DAY_DAY.HTMLValue# for \
##UTM_0.PERSONS_NBR.HTMLValue# persons

                Hotels                Category   EURO/day

\
<input type="##(UTM_0.HOTEL_SELECT.Visible == 'N') ? 'password' : 'text'#" ##(
WT_BROWSER.acceptClass ) ? 'class="box" style="width:9px"' : ''# name="HOTEL_SELECT"
size="1" maxlength="1" value="##UTM_0.HOTEL_SELECT.Value#"/> 1. \
##UTM_0.HOTEL_NAME$000.HTMLValue# \
##UTM_0.HOTEL_CATEGORY$000.HTMLValue# \
##UTM_0.HOTEL_PRICE$000.HTMLValue#
\
##UTM_0.HOTEL_ADDRESS$000.HTMLValue#

                2. \
##UTM_0.HOTEL_NAME$001.HTMLValue# \
##UTM_0.HOTEL_CATEGORY$001.HTMLValue# \
##UTM_0.HOTEL_PRICE$001.HTMLValue#
\
##UTM_0.HOTEL_ADDRESS$001.HTMLValue#

```

```

        3. \
        ##UTM_0.HOTEL_NAME$002.HTMLValue# \
        ##UTM_0.HOTEL_CATEGORY$002.HTMLValue# \
        ##UTM_0.HOTEL_PRICE$002.HTMLValue#
        \
        ##UTM_0.HOTEL_ADDRESS$002.HTMLValue#

</PRE></TD></TR><TR><TD><PRE>\

    Command ==&#62; \
    <input type="##(UTM_0.COMMAND.Visible == 'N') ? 'password' : 'text'#" ##(
    WT_BROWSER.acceptClass ) ? 'class="box" style="width:65px"' : '# name="COMMAND"
    size="8" maxLength="8" value="##UTM_0.COMMAND.Value#"/> ( CONFIRM / QUIT=mainmenu
    ) </PRE></TD></TR><TR><TD><PRE>\

##UTM_0.WARNING_AREA.HTMLValue#
##UTM_0.INFO_AREA.HTMLValue#
<wtoncreatescript>
<!--
    wtInputFields = {HOTEL_SELECT:UTM_0.HOTEL_SELECT,COMMAND:UTM_0.COMMAND};
//-->
</wtoncreatescript></pre></div>
    <!-- ----- -->
    <!-- end of host screen section -->
    <!-- ----- -->
</td>
</tr>
</table>

```

In einer Schleife über die Eingabefelder wird für alle Browser, die `maxLength` als Attribut beim `<input>`-Tag ignorieren, das client-seitige Attribut `maxLength` am entsprechenden Skript-Objekt angehängt, da dieses nicht automatisch vorhanden ist. Für alle Browser, die nicht beliebige Attribute beim `<input>`-Tag akzeptieren, wird das Attribut `markable` gesetzt, sofern das entsprechende Hostobjekt markierbar ist.

```

<script type="text/javascript">
<!--
<wtoncreatescript>
<!--
    for( element in wtInputFields )
    {
        if (!WT_BROWSER.acceptMaxLength)
            document.writeln('wtSetMaxLength(\'', element, '\',', wtInputFields[ element
].Length, ' ');');
        if (!WT_BROWSER.acceptInputAttributes)
        {
            //if (wtInputFields[ element ].DataType == 'NUMERIC')
            // document.writeln('wtSetInputAttribute("numeric", "', element, '");');
            if (wtInputFields[ element ].Detectable == 'Y')

```

```

        document.writeln('wtSetInputAttribute("markable", "'", element, "'");');
    }
}
//-->
</wtoncreatescript>
//-->
</script>
</form>

```

Danach wird der Fokus in dem Fenster des Web-Browsers auf das Feld gesetzt, in dessen korrespondierendem Bildschirmfeld der Cursor positioniert war (sofern das Feld ein Eingabefeld ist).

```

<wtrem** initial focus selection *****>
<script type="text/javascript">
<!--
<wtif( UTM_0[UTM_0.WT_HOST_MESSAGE.CursorField] &&
      UTM_0[UTM_0.WT_HOST_MESSAGE.CursorField].IOMType == 'INPUT') >
    wtSetFocus('##UTM_0.WT_HOST_MESSAGE.CursorField#');
<wtelse>
    wtBuildFieldList();
</wtif>
//-->
</script>

```

In einem `wtOnReceiveScript` werden zuerst die geposteten Werte für die einzelnen Felder auf die entsprechenden Host-Objekte zurückgeschrieben.

```

<wtrem** Script executed after post of HTML page *****>
<wtonreceivescript>
<!--
    //{{WebLab(processPostedData)
    UTM_0.HOTEL_SELECT.Value = WT_POSTED.HOTEL_SELECT;
    UTM_0.COMMAND.Value = WT_POSTED.COMMAND;

```

Die markierten Felder werden ermittelt.

```

wtMarkedFields = WT_POSTED.wt_markedFields.split(',');
for( i=1; i<wtMarkedFields.length; i++)
{
    UTM_0[wtMarkedFields[i]].InputState = 'D';
    UTM_0[wtMarkedFields[i]].InputStateAct = 'D';
}

```

Falls die Taste Suspend nicht gedrückt wurde, werden je ein `send()`- und `receive()`-Aufruf abgesetzt, um die WebTransactions Dialogzyklen und die Host-Dialogschritte zu synchronisieren. Zum Schluss wird mit der Funktion `setNextPage()` das nächste zu verarbeitende Template bestimmt.

```

//}}
//{{WebLab(processHostCommunication)
if ( WT_POSTED.wt_special_key == 'Suspend' || UTM_0_system.SUSPEND )
{
    UTM_0_system.SUSPEND = false;
}
else
{
    try {
        UTM_0.send();
        UTM_0.receive();
        if( UTM_0_system.APPLICATION_PREFIX )
            setNextPage( UTM_0_system.APPLICATION_PREFIX + '@' + UTM_0_system.FLD );
        else
            setNextPage( UTM_0_system.FLD );
    }
    catch (e) {
        if ( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT )
            setNextPage( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT );
    }
}
//}}
//-->
</wtonreceivescript>
</body>
<wtif (UTM_0_system.EPILOG)>
    <wtinclude Name="##UTM_0_system.EPILOG#">
</wtif>
</html>

```

6 Templates bearbeiten

Die Oberfläche der automatisch generierten Templates ist dem Aussehen und der Funktionalität der Terminal-Formate nachempfunden. Diese Oberfläche können Sie sowohl generell über das Master-Template als auch individuell über die einzelnen Templates aufbereiten:

- Das openUTM-spezifische Master-Template `UTM.wmt` legt ein generelles Layout fester Bereiche fest. Es wird bei der Generierung der formatspezifischen Templates verwendet und kann anwendungsspezifisch angepasst werden, siehe [Abschnitt „Master-Template UTM.wmt“ auf Seite 96](#).
- Sie können jedes generierte Template grafisch aufbereiten und z.B. ein Auswahlménú in eine Drop-Down-Liste umwandeln, in dem Sie die Templates nachbearbeiten. Für die Nachbearbeitung steht Ihnen die Template-Sprache zur Verfügung, die Sie im WebTransactions-Handbuch „Template-Sprache“ ausführlich beschrieben finden.

Besonders komfortabel gestaltet sich die Bearbeitung mit der Entwicklungsumgebung WebLab. Grundlegende Informationen zu WebLab finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“. Eine detaillierte Beschreibung ist in der ausführlichen Online-Hilfe von WebLab enthalten.

Im Folgenden werden die Gestaltungsmöglichkeiten - die für alle WebTransactions-Produktvarianten weitgehend gleich sind - nur kurz vorgestellt. Im Mittelpunkt dieses Kapitels stehen die openUTM-Spezifika: Templates für FHS-/FORMANT-Teilformate und openUTM-Zeilenmodus.

6.1 Master-Template UTM.wmt

Master-Templates werden von WebTransactions bei der Generierung der format-spezifischen Templates als Schablone verwendet und sorgen für ein einheitliches Layout. Master-Templates können wie jedes andere Template feste HTML-Bereiche sowie beliebige WTML-Tags und WTScripTs enthalten. Zusätzlich stehen in Master-Templates spezielle Master-Template-Tags zur Verfügung - kurz MT-Tags genannt, die im WebTransactions-Handbuch „Template-Sprache“ beschrieben sind.

Das Master-Template-Konzept zeigt seine Stärke unter anderem bei Host-Anwendungen, bei denen viele Formate einen ähnlichen Aufbau haben: z.B. eine feste Einteilung in Kopfzeile, Arbeitsbereich und Fußzeile. In solchen Fällen genügt es, den Aufbau einmal im Master-Template festzulegen und dieses Master-Template bei der Generierung der formatspezifischen Templates zuzuweisen. Alle generierten Templates erhalten dann automatisch den gewünschten Aufbau.

Für WebTransactions for openUTM werden die Standard-Master-Templates `UTM.wmt` und `UTMpartial.wmt` für Teilformate mit ausgeliefert, die Sie individuell anpassen aber auch unverändert einsetzen können. `UTMpartial.wmt` ist im [Abschnitt „Aufbau des Master-Templates UTMpartial.wmt“ auf Seite 102](#) beschrieben.

Die Standard-Master-Templates enthalten bereits alle WTML-Tags und WTScripTs, die für alle Templates der jeweiligen Liefereinheit gleich sind, z.B. eine Prüfung, ob ein verbindungs-spezifisches System-Objekt existiert.

Über die grafische Oberfläche von WebLab geben Sie an, welches Master-Template für die Generierung herangezogen werden soll. Einige Generierungsoptionen (z.B. die Generierungsmethode) können Sie sowohl im Master-Template als auch unmittelbar mit WebLab festlegen. Die Festlegungen, die Sie mit WebLab setzen, übersteuern die entsprechenden Festlegungen im Master-Template.

6.2 Templates gestalten

Die Oberfläche der formatspezifischen Templates ist dem Aussehen und der Funktionalität der Terminal-Darstellung nachempfunden. Wollen Sie das generierte Template grafisch aufbereiten und z.B. ein Auswahlmü in eine Drop-Down-Liste umwandeln, müssen Sie die Templates nachbearbeiten. Für die Nachbearbeitung steht Ihnen die Template-Sprache zur Verfügung, die Sie im WebTransactions-Handbuch „Template-Sprache“ ausführlich beschrieben finden.

Besonders komfortabel gestaltet sich die Bearbeitung mit WebLab. Grundlegende Informationen zu WebLab finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“. Eine detaillierte Beschreibung ist in der ausführlichen Online-Hilfe von WebLab enthalten.

Bei der individuellen Gestaltung der WebTransactions-Templates sind Ihnen keine Grenzen gesetzt: Animierte und anklickbare Bilder, Java-Scripts und -Applets, ActiveX-Controls, Bild- und Tonsequenzen etc. können eingefügt werden. Die Template-Sprache von WebTransactions ist offen: Sie können alle Sprachmittel verwenden, die der eingesetzte Web-Browser unterstützt.

6.2.1 Globales Layout festlegen

Für die Umsetzung eines globalen Layouts, das alle Templates gleichermaßen umfasst, haben Sie drei Möglichkeiten:

- Templates inkludieren
- Master-Templates
- Systemobjekte-Attribute `EPILOG`, `FORMTPL` und `PROLOG`

Templates inkludieren

Wollen Sie, dass sich die Gestaltungsmaßnahmen auf alle Templates auswirken, z.B. wenn Sie Firmenlogos einfügen oder generelle Informationen auf allen Ihren Seiten zur Verfügung stellen wollen, können Sie die entsprechenden Passagen in eine eigene Datei schreiben und diese mit Include-Tags in die Templates einfügen. Dadurch wird die Pflege der Templates einfacher. Wenn sich etwas ändert, brauchen Sie diese Änderung nur einmal im zentralen Include-Template vorzunehmen.

Master-Templates

Sie können das globale Layout auch über die Master-Templates festlegen. Dazu bearbeiten Sie das Master-Template vor der Generierung der formatspezifischen Templates.

Im Dialogfeld **FLD und Template Generierung** von WebLab geben Sie an, welches Master-Template für die Generierung herangezogen werden soll. Einige Generierungsoptionen (z.B. die Generierungsmethode) können Sie sowohl im Master-Template als auch unmittelbar mit WebLab festlegen.

Die Einstellungen im Master-Template werden als Vorbelegung in das Dialogfeld übernommen. Diese Vorbelegung können Sie im Dialogfeld ändern, so dass die geänderten Werte die entsprechenden Festlegungen im Master-Template übersteuern. Bei der Generierung eines Templates werden immer die Werte verwendet, die im Dialogfeld angezeigt werden.

Systemobjekte-Attribute EPILOG, FORMTPL und PROLOG

Auch mit den Attributen `EPILOG`, `FORMTPL` und `PROLOG` können Sie global die Gestaltung der Templates beeinflussen. Ein Aufruf dieser Attribute wird vom Master-Template in die Templates generiert. Ausgewertet werden die Attribute dann beim Ablauf des generierten Templates.

Die Attribute enthalten jeweils den Namen eines Templates, das entsprechend dem Attribut zu unterschiedlichen Zeitpunkten ausgeführt wird:

<code>PROLOG</code>	zu Beginn des aktuellen Templates
<code>FORMTPL</code>	vor Ausführen des DataForm-Tags im aktuellen Template
<code>EPILOG</code>	am Ende des aktuellen Templates

6.2.2 Oberfläche individuell gestalten

Standard-Verschönerungsschritte können Sie durch die Assistenten von WebLab automatisch ausführen lassen. Diese Assistenten ersetzen generierte Eingabefelder (INPUT-Tags vom Typ „Text“) durch grafische Elemente, wie Drop-Down-Listen, Radio-Knöpfe, Check-boxen oder Ausführungsknöpfe. (Ein Beispiel hierzu finden Sie in [Abschnitt „Templates bearbeiten“ auf Seite 52.](#))

Bei den wichtigsten HTML-Tags werden Sie von WebLab unterstützt (Befehl **Einfügen/HTML**). Sie können die HTML-Oberfläche aber auch mit einem HTML-Editor unabhängig von den generierten Templates entwerfen und anschließend über einen Assistenten mit dem Befehl **Gestalten/Zusammenführen** von WebLab in Templates umwandeln.

WTBeans

Mit WTBeans können Sie die Funktionalität Ihrer Templates bearbeiten und die Darstellung der Daten am Browser festlegen.

WTBeans sind wiederverwendbare Komponenten, die z.B. helfen die Oberfläche einer WebTransactions-Anwendung zu gestalten, oder Verbindung zu einer Host-Anwendung herstellen. WTBeans werden in inline- und standalone-Komponenten unterschieden. Ein inline-WTBean entspricht einem Teil eines Templates und wird in ein vorhandenes Template eingefügt. Ein standalone-WTBean entspricht einem eigenständigen Template, wie z.B. das WTBean zur Generierung eines Start-Templates.

Sie fügen die inline-WTBeans in Ihre Templates ein oder generieren mit einem standalone-WTBean ein eigenes Template. Siehe hierzu auch das WebTransactions-Handbuch „Konzepte und Funktionen“.

6.2.3 Ablauf gestalten

WebTransactions bietet Ihnen aber nicht nur die Möglichkeit zu einem „Face Lifting“ Ihrer Host-Anwendungen. Sie können auch die Dialogabläufe neu gestalten. Die starre 1:1 Zuordnung zwischen HTML-Seite und Host-Format wird aufgebrochen: Mit WebTransactions können Sie die von den Host-Anwendungen vorgesehenen Dialogstrategien aktiv steuernd verändern: Ein-/Ausgabe-Elemente können ausgefiltert oder hinzugefügt werden, Dialogschritte lassen sich zusammenfassen oder auffächern.

6.3 Besonderheiten bei FHS/FORMANT-Teilformaten

WebTransactions unterstützt openUTM-Anwendungen, die Teilformate der Formatierungssysteme FHS und FORMANT nutzen, mit dem Master-Template für Teilformate `UTMpartial.wmt`. Wenn Ihre openUTM-Anwendung mit Teilformaten arbeitet, brauchen Sie aus den Teilformaten nur unter Verwenden des Master-Templates `UTMpartial.wmt` die „Teil“-Templates zu generieren.

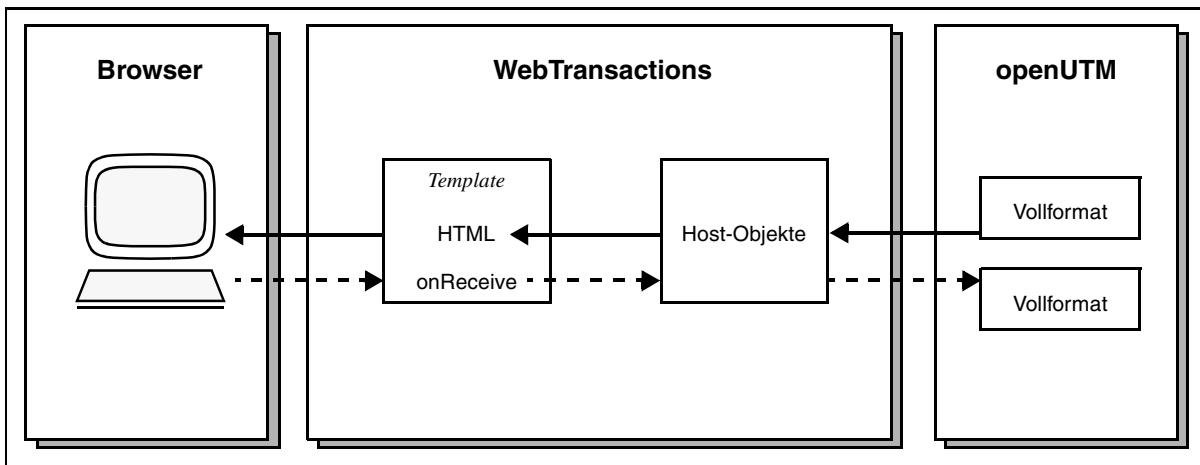
Diese Teilformat-Templates unterscheiden sich von den Vollformat-Templates. Dieser Abschnitt beschreibt

- den Ablauf der Kommunikation bei Teilformaten
- den Aufbau des Master-Templates `UTMpartial.wmt`
- den Aufbau der Teilformat-Templates

und gibt Hinweise, was bei der Bearbeitung von Teilformat-Templates zu beachten ist.

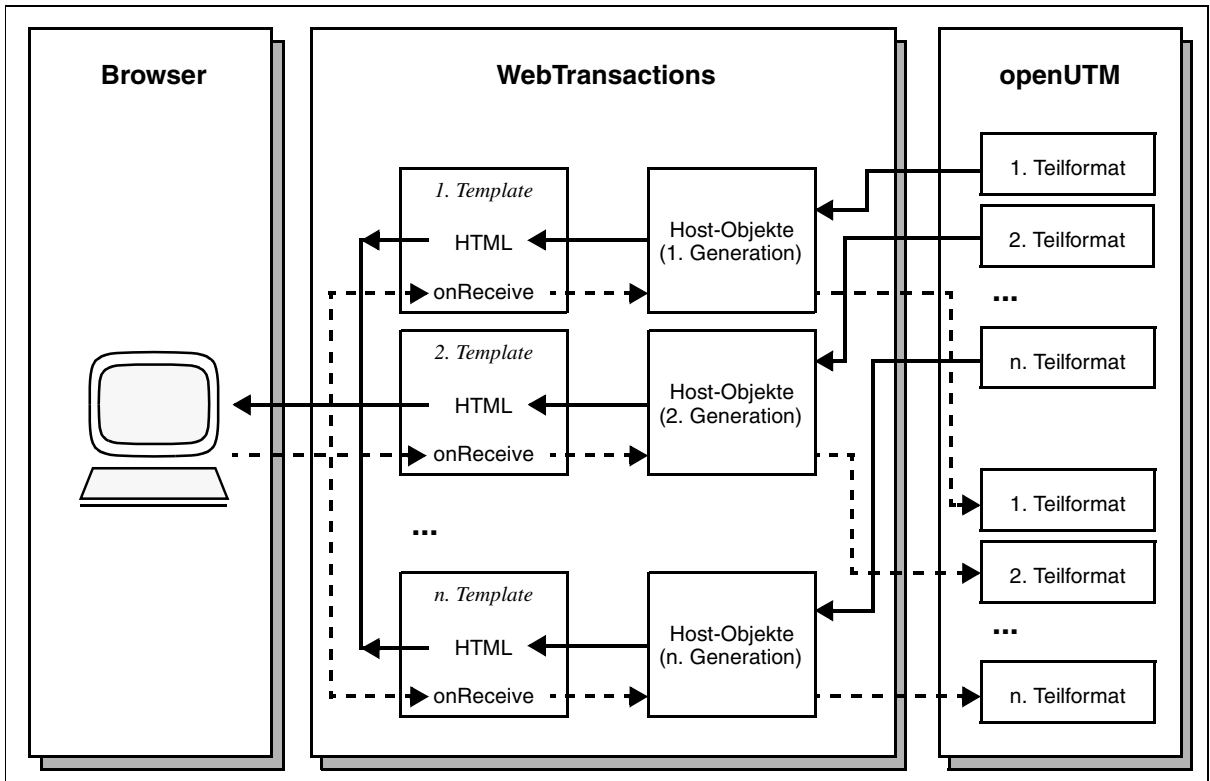
6.3.1 Kommunikationsablauf

Bei Vollformaten sendet und empfängt die openUTM-Anwendung abwechselnd Nachrichten. Die Daten haben folgenden Weg:



Die openUTM-Anwendung sendet eine Nachricht. WebTransactions speichert diese als Host-Datenobjekte. Das Template steuert die Generierung der HTML-Seite, die dann an den Browser geschickt wird. Die zurückgeschickten Daten werden entsprechend der On-Receive-Tags in die Host-Datenobjekte kopiert. Diese Host-Datenobjekte werden in einer Nachricht an die openUTM-Anwendung geschickt.

Bei Teilformaten haben die Daten folgenden Weg:



openUTM sendet das 1. Teilformat, WebTransactions speichert es in Host-Datenobjekten. Das 1.Template erzeugt das HTML-Grundgerüst. Der Inhalt der Host-Datenobjekte wird unterhalb des Kommunikationsobjekts im Array `wtPartialFormats` gesichert. Nun werden nacheinander das 2. bis n. Teilformat ebenso empfangen und in Form von Host-Datenobjekten abgelegt (2. bis n. Generation). Es wird HTML generiert, und die Host-Datenobjekte gesichert. Das gesamte Generierungsergebnis wird als HTML-Seite an den Browser geschickt.

Die geposteten Daten werden nun in mehreren Schritten an die Host-Anwendung geschickt:

Zunächst wird aus der Sicherung die 1. Generation von Host-Datenobjekten wiederhergestellt. Die zugehörigen geposteten Daten werden in die Host-Datenobjekte kopiert und diese an die openUTM-Anwendung geschickt (OnReceive-Tags des 1. Templates). Entspre-

chend wird die 2. bis n. Generation von Host-Datenobjekten wiederhergestellt, mit den entsprechenden geposteten Daten gefüllt und an die openUTM-Anwendung geschickt (On-Receive-Tags der 2. bis n. Generation).

6.3.2 Aufbau des Master-Templates UTMpartial.wmt

Wenn Teilformate eingesetzt werden, müssen Sie bei der Template-Generierung das Master-Template `UTMpartial.wmt` verwenden. `UTMpartial.wmt` ist ähnlich aufgebaut wie `UTM.wmt`. Dieser Abschnitt erläutert die teilformatspezifischen Details von `UTMpartial.wmt`.

Das Template ist zur besseren Übersicht hier in nummerierte Abschnitte gegliedert, die einzeln erläutert werden. Auf die Abschnitte wird später im [Abschnitt „Gestalten mit Teilformat-Templates“ auf Seite 112](#) Bezug genommen.

Abschnitt (1)

Das Template enthält am Anfang einen Header mit Hinweisen zur Generierung. Das erste `wtOnCreateScript` belegt die Variablen `%%CommObj%` und `%%CommObj%_system` mit einer Referenz auf das aktuelle Kommunikationsobjekt und das für die Kommunikation zu verwendende „Systemobjekt“.

Der Eingabemodus (Einfügen oder Überschreiben) wird entsprechend der Vorgabe aus `EDIT_MODE` in `isOverwrite` eingestellt.

Unterhalb des Kommunikationsobjekts wird ein Array `wtPartialFormats` angelegt, in dem das Präfix `FORMAT_SEQ` und die Inhalte der Teilformate gesichert werden. Das erste Template, das für eine Folge von Teilformaten durchlaufen wird, legt das Objekt `wtPartialFormats` an. Wird das Objekt gefunden und ist das Attribut `current < 0`, dann wurde die Seite nach einer Unterbrechung mit der Taste `Suspend` erneut aufgerufen. In diesem Fall wird das Flag `Suspend` gesetzt. Dieses steuert im Folgenden, ob die Daten beim Aufbau der Seite vom Host empfangen werden sollen oder aus der Sicherung in `wtPartialFormats` übernommen werden.

```
%%GlobalSettings Protocol="UTM"%
<!--
  {{{WebLab(assignCommunicationObject)
  %%CommObj% = WT_HOST.active || WT_HOST.%%CommObj%;
  if (%%CommObj%.WT_SYSTEM != null)
    %%CommObj%_system = %%CommObj%.WT_SYSTEM; // communication specific system object
  else
    %%CommObj%_system = WT_SYSTEM; // global system object
  }}}
  // propagate communication object to included WTML documents ///////////////
  wtCurrentComm = %%CommObj%;
  wtCurrentComm_system = %%CommObj%_system;
  if ( wtCurrentComm_system.EDIT_MODE )
  {
```

```

    if ( typeof wtCurrentComm_system.isOverwrite == 'undefined' &&
wtCurrentComm_system.EDIT_MODE.match(/OVERWRITE/) )
    wtCurrentComm_system.isOverwrite = true;
    else if (wtCurrentComm_system.EDIT_MODE == 'OVERWRITE')
    wtCurrentComm_system.isOverwrite = true;
    else if (wtCurrentComm_system.EDIT_MODE == 'INSERT')
    wtCurrentComm_system.isOverwrite = false;
} else
wtCurrentComm_system.isOverwrite = false;

// create array for contents of all partial formats //////////////////////////////////
if( ! %%CommObj%.wtPartialFormats )
{
    // first format in a sequence of partial formats
    %%CommObj%.wtPartialFormats = new Array;
    %%CommObj%.wtPartialFormats.current = 0;
    %%CommObj%.wtPartialFormats[ 0 ] = new Object;
    %%CommObj%.wtPartialFormats[ 0 ].FORMAT_SEQ = %%CommObj%_system.FORMAT_SEQ;
    %%CommObj%.wtPartialFormats[ 0 ].Contents = %%CommObj%.WT_HOST_MESSAGE.Contents;
}
else if( %%CommObj%.wtPartialFormats.current < 0 )
{
    // wtPartialFormats exits but current<0 means recovery from suspend
    %%CommObj%.wtPartialFormats.suspended = true;
    %%CommObj%.wtPartialFormats.current = 0;
}
//-->
</wtOnCreateScript>

```

Abschnitt (2)

Das Attribut `FORMAT_SEQ` wird wieder mit dem Index der Teilnachricht versorgt und der Nachrichtenpuffer des Host-Adapters (`%%CommObj%.WT_HOST_MESSAGE.Contents`) wieder hergestellt, indem die entsprechende Teilnachricht aus der Sicherung in `wtPartialFormats` genommen wird, siehe [Abschnitt \(5\)](#).

```

<wtOnReceiveScript>
<!--
// restore partial format //////////////////////////////////
if( %%CommObj%.wtPartialFormats.current < 0 )
    %%CommObj%.wtPartialFormats.current = 0;
else
{
    %%CommObj%.wtPartialFormats.current++;
    %%CommObj%_system.FORMAT_SEQ =
    %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].FORMAT_SEQ;
    %%CommObj%.WT_HOST_MESSAGE.Contents =
    %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].Contents;
}
//-->
</wtOnReceiveScript>

```

Abschnitt (3)

Dieser Abschnitt enthält Code, der pro HTML-Seite nur einmal durchlaufen werden darf. Dies muss immer beim ersten Teilformat geschehen. Die Entscheidung hierüber kann mit dem Systemobjekt-Attribut `FORMAT_SEQ` getroffen werden. Bei Vollformaten ist dieses Attribut leer, bei Teilformaten liefert der Host-Adapter darin eine fortlaufende Nummer.

Folgende Schritte werden hier durchgeführt:

- Ausführen eines ggf. vorhandenen PROLOG-Templates.
- Ausgabe des HTML-Grundgerüsts.
- Ausgabe von CSS-Definitionen, falls der Browser Style-Sheets unterstützt.
- Öffnen eines HTML-Formulars.
- Öffnen einer Tabelle, die das ganze Format umrahmt.
- Aufruf der zentral änderbaren Templates `wtBrowserFunctions.htm` (Cursor-Steuerung) und `wtKeysUTM.htm` (Funktionen zur Bedienung der Seite).
- Definieren der Funktionen `taggedInput()` und `taggedOutput()` mit denen die ungeschützten (`input`) und geschützten Felder aufbereitet werden können.

```
<wtRem** page header only generated for first partial format *****>
<wtIf ( %%CommObj%_system.FORMAT_SEQ == "1" || %%CommObj%_system.FORMAT_SEQ == " " )>
<wtIf ( %%CommObj%_system.PROLOG)>
  <wtInclude Name="###%%CommObj%_system.PROLOG#">
</wtIf>
<html>
<head>
<title>WebTransactions V7.5 - application ###%%CommObj%_system.SYM_DEST#</title>
##WT_SYSTEM.CGI.HTTP_USER_AGENT.indexOf( 'MSIE' ) >= 0 ?
  '<meta http-equiv="Pragma" content="no-cache" />' :
  '<meta http-equiv="Cache-Control" content="no-cache" />'#
<wtIf ( WT_BROWSER.acceptClass)>
  <style type="text/css">
    input {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
    }
    input.box {
      border:       0 solid;
      padding:      1px 0 1px 0;
      margin-left:  -1px;
      margin-top:   ##WT_BROWSER.marginTop#px;
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      color:        #000000;
      background-color: #FFFFFF;
    }
    input.button {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      border-width: 1pt;
      margin-left:  -1pt;

```



```

    }
    select {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
    }
    pre {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      margin:       0;
    }
  </style>
</wtIf>
</head>
<body bgcolor="#C0C0C0">
<form WebTransactions>
  <table frame="border" rules="all">
    <tr>
      <td>
        <wtInclude name="wtBrowserFunctions">
          <wtInclude name="wtKeysUTM">
            <wtIf (%%CommObj%_system.FORMTPL)>
              <wtInclude Name="##%%CommObj%_system.FORMTPL#">
            </wtIf>
          </td>
        </tr>
      <tr>
        <td>
          <wtOnCreateScript>
            <!--
              colors = new Array('RED','GREEN','YELLOW','BLUE','MAGENTA', 'CYAN', 'WHITE');
              space80 = "
";

              wtInputFields = new Object();
function taggedInput( hostObject )
{
  if ( hostObject.Protected == 'Y' )
  {
    taggedOutput( hostObject );
    return;
  }
  currentLength = hostObject.Length;
  input = '<input type=' + (hostObject.Visible == 'N' ? "password" :
"text" ) );
  if ( WT_BROWSER && ( WT_BROWSER.is_ie || WT_BROWSER.is_ns61up ) )
  {
    input += ' class="box" style="width:' + (currentLength *
WT_BROWSER.charWidth + 1) + 'px';
    input += (hostObject.Blinking == 'Y' ? ' ; background-color:#FFC0C0' :
');
    input += (hostObject.Underline == 'Y' ? ( hostObject.Intensity == 'N' ?
'; color:#A0A0FF' : ' ; color:#0000A0' ) :

```


Abschnitt (4)

Dieser Teil enthält wie bei Vollformaten alle konstanten Teile des Formats sowie HTML-Input-Tags für Eingabefelder.

```
<div style="color:#000000"><pre>
%%LINES CellsDelimiter="-" TaggedInput=Enabled TaggedOutput=Enabled%</pre></div>
<wtOnCreateScript>
<!--
    if( %%CommObj[%%%CommObj%.WT_HOST_MESSAGE.CursorField] &&
        %%CommObj[%%%CommObj%.WT_HOST_MESSAGE.CursorField].IOType == 'INPUT' )
        wtCursor = '_' + %%CommObj%_system.FORMAT_SEQ + '_' +
            %%CommObj%.WT_HOST_MESSAGE.CursorField;
    //-->
</wtOnCreateScript>
<script type="text/javascript">
<!--
<wtOnCreateScript>
<!--
    for( element in wtInputFields )
    {
        if (!WT_BROWSER.acceptMaxLength)
            document.writeln('wtSetMaxLength('_',%%CommObj%_system.FORMAT_SEQ,'_',
element, ',', wtInputFields[ element ].Length, '); ');
        if (!WT_BROWSER.acceptInputAttributes)
        {
            //if (wtInputFields[ element ].DataType == 'NUMERIC')
            // document.writeln('wtSetInputAttribute("numeric",
"_,%%CommObj%_system.FORMAT_SEQ,'_', element, '); ');
            if (wtInputFields[ element ].Detectable == 'Y')
                document.writeln('wtSetInputAttribute("markable",
"_,%%CommObj%_system.FORMAT_SEQ,'_', element, '); ');
        }
    }
    //-->
</wtOnCreateScript>
    //-->
</script>
```

Abschnitt (5)

Dieser Abschnitt enthält Code, der nur einmal zum Abschluss einer HTML-Seite durchlaufen werden darf (analog zu [Abschnitt \(3\)](#)). Dies muss immer beim letzten Teilformat geschehen. Die Entscheidung hierüber kann mit dem Systemobjekt-Attribut `FORMAT_SEQ` getroffen werden. Bei Vollformaten ist dieses Attribut leer, beim letzten Teilformat der Folge liefert der Host-Adapter darin den Wert `LAST`.

Alle im [Abschnitt \(3\)](#) geöffneten HTML-Elemente werden geschlossen (bis auf `<html>`).

Die Schreibmarke wird, wie von der Anwendung vorgegeben, in ein Eingabe-Feld positioniert.

```
<wtIf ( %%CommObj%_system.FORMAT_SEQ == "LAST" || %%CommObj%_system.FORMAT_SEQ == "" )>
<!-- ----- -->
    <!-- end of host screen section -->
    <!-- ----- -->
    </td>
</tr>
</table>
</form>
<wtRem** initial focus selection *****>
<script type="text/javascript">
<!--
    <wtIf( wtCursor ) >
        wtSetFocus('##wtCursor#');
    <wtElse>
        wtBuildFieldList();
    </wtIf>
    //-->
</script>
</body>
</wtIf>
```

Abschnitt (6)

In diesem Teil werden die vom Browser geposteten Daten auf die Hostobjekte kopiert. Falls die Taste `Suspend` gedrückt wurde, wird das Teilformat erneut in `wtPartialFormats` gesichert, andernfalls wird die Teilnachricht an den Host geschickt.

Wurde die letzte Teilnachricht an den Host geschickt, dann wird die gesamte Sicherung in `wtPartialFormats` gelöscht und die nächste Nachricht vom Host gelesen.

```
<wtRem** Script executed after post of HTML page *****>
<wtOnReceiveScript>
<!--
    if (%%CommObj%_system.EDIT_MODE &&%%CommObj%_system.EDIT_MODE.match(/USER/))
        %%CommObj%_system.isOverwrite = (WT_POSTED.wt_isOverwrite=='1');
    //{{WebLab(processPostedData)
    %%OnReceiveCopies%
    wtMarkedFields = WT_POSTED.wt_markedFields.split(',');
```

```

elementPrefix = '_' + %%CommObj%_system.FORMAT_SEQ + '_';
for( i=1; i<wtMarkedFields.length; i++)
  if ( wtMarkedFields[i].indexOf(elementPrefix) == 0 )
  {
    var elementName = wtMarkedFields[i].substr(elementPrefix.length);
    %%CommObj[elementName].InputState = 'D';
    %%CommObj[elementName].InputStateAct = 'D';
  }
//}}
//{{WebLab(processHostCommunication)
if ( WT_POSTED.wt_special_key == 'Suspend' || %%CommObj%_system.SUSPEND )
{
  %%CommObj%_system.SUSPEND = false;
  %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].Contents =
  %%CommObj%.WT_HOST_MESSAGE.Contents;
  if( %%CommObj%_system.FORMAT_SEQ == "LAST" || %%CommObj%_system.FORMAT_SEQ == "" )
  {
    %%CommObj%.wtPartialFormats.current = -1;
    %%CommObj%_system.FORMAT_SEQ = %%CommObj%.wtPartialFormats[ 0 ].FORMAT_SEQ;
    %%CommObj%.WT_HOST_MESSAGE.Contents = %%CommObj%.wtPartialFormats[ 0 ].Contents;
  }
}
else
{
  try {
    %%CommObj%.send();
    if( %%CommObj%_system.FORMAT_SEQ == "LAST" || %%CommObj%_system.FORMAT_SEQ == ""
)
    {
      delete %%CommObj%.wtPartialFormats;
      %%CommObj%.receive();
      if( %%CommObj%_system.APPLICATION_PREFIX )
        setNextPage( %%CommObj%_system.APPLICATION_PREFIX + '@' +
%%CommObj%_system.FLD );
      else
        setNextPage( %%CommObj%_system.FLD );
    }
  }
  catch (e) {
    if ( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT )
      setNextPage( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT );
  }
}
//}}
//-->
</wtOnReceiveScript>

```

Abschnitt (7)

Ist die aktuelle Teilnachricht noch nicht die letzte, so wird die nächste Teilnachricht aus der Sicherung gelesen (Wiederherstellung der Seite nach Suspend) oder vom Host empfangen.

Das WTML-Dokument zur Darstellung des folgenden Teilformats wird inkludiert.

```
<wtIf ( %%CommObj%_system.FORMAT_SEQ != "LAST" && %%CommObj%_system.FORMAT_SEQ != " " )>
<wtOnCreateScript>
<!--
  if( %%CommObj%.wtPartialFormats.suspended )
  {
    %%CommObj%.wtPartialFormats.current++;
    %%CommObj%_system.FORMAT_SEQ =
      %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].FORMAT_SEQ;
    %%CommObj%.WT_HOST_MESSAGE.Contents =
      %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].Contents;
  }
  else
  {
    try {
      %%CommObj%.receive();
    }
    catch (e) {
      if ( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT )
        setNextPage( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT );
    }
    %%CommObj%.wtPartialFormats.current++;
    %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ] = new Object;
    %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].FORMAT_SEQ =
      %%CommObj%_system.FORMAT_SEQ;
    %%CommObj%.wtPartialFormats[ %%CommObj%.wtPartialFormats.current ].Contents =
      %%CommObj%.WT_HOST_MESSAGE.Contents;
  }
  //-->
</wtOnCreateScript>
<wtInclude name=
"##( %%CommObj%_system.APPLICATION_PREFIX ?
      %%CommObj%_system.APPLICATION_PREFIX + '@' :
      ' ' ) +
  %%CommObj%_system.FLD#">
```

Abschnitt (8)

In diesem Abschnitt wird das Sicherungsobjekt `wtPartialFormats` zurückgesetzt, ggf. `EPILOG` aufgerufen und das letzte offene tag `<html>` geschlossen.

```
<wtElse>
<wtOnCreateScript>
<!--
  // current=-1 signals end of format sequence
  %%CommObj%.wtPartialFormats.current = -1;
  %%CommObj%_system.FORMAT_SEQ = %%CommObj%.wtPartialFormats[ 0 ].FORMAT_SEQ;
  %%CommObj%.WT_HOST_MESSAGE.Contents = %%CommObj%.wtPartialFormats[ 0 ].Contents;
//-->
</wtOnCreateScript>
<wtIf (%%CommObj%_system.EPILOG)>
  <wtInclude Name="##%%CommObj%_system.EPILOG#">
</wtIf>
</html>
</wtIf>
```

6.3.3 Gestalten mit Teilformat-Templates

Die Möglichkeiten der Nachbearbeitung von Teilformat-Templates hängt im Einzelfall stark davon ab, wie die Teilformate in Ihrer Host-Anwendung verwendet werden. Dieser Abschnitt kann daher in keiner Weise vollständig sein. Er skizziert nur einige Möglichkeiten, um Ihnen Anregungen für die Gestaltung zu geben.

Dabei können Sie diese Veränderung im oben beschriebenen Master-Template vornehmen (siehe [Abschnitt „Aufbau des Master-Templates UTMpartial.wmt“ auf Seite 102](#)), sie gilt dann für alle daraus generierten Templates, oder in den fertig generierten Templates.

Wollen Sie das Layout z.B. auf HTML-Knöpfe und -Auswahllisten umstellen und die vom Browser erhaltenen Daten auf die Formatschnittstelle (Felddatei) abbilden, sollten Sie sich auf [Abschnitt \(4\)](#) und [Abschnitt \(6\)](#) beschränken.

Um das generelle Layout der Seite (Kopf, Fuß, Hintergrund usw.) zu beeinflussen, können Sie auch [Abschnitt \(3\)](#) und [Abschnitt \(5\)](#) ändern.

Die übrigen Teile sollten Sie unverändert lassen, um den Ablauf der Kommunikation mit der openUTM-Anwendung nicht zu stören.

Mit einzelnen Teilformaten gestalten

Wenn Sie die Teilformat-Templates einzeln verändern, können Sie zwar komfortable HTML-Tags zur Gestaltung der Oberfläche einfügen. Sie können aber nicht die Aufteilung des Formats (der Maske) grundlegend verändern, z.B. nicht die Felder des zweiten Teilformats vor die Felder des ersten Teilformats stellen.

Neues Template für Teilformat-Folge eines Bildschirms

Dieser Abschnitt zeigt auf, wie eine Kombination von Teilformaten (ein logischer Bildschirm) frei umgestaltet werden kann. Bei solchen Änderungen müssen Sie auch die generierten Templates bearbeiten, es reicht in der Regel nicht, nur das Master-Template zu modifizieren.

Die folgenden beiden Fälle illustrieren zwei grundlegend unterschiedliche Situationen:

- Im ersten Beispiel hat die Teilformat-Folge einen eindeutigen Beginn, d.h. das 1. Teilformat kommt in dieser Host-Anwendung nur in diesem einen Bildschirm als 1. Teilformat vor.
- Im zweiten Beispiel gibt es in der Teilformat-Folge kein identifizierendes Teilformat, von dem auf den logischen Bildschirm (die Kombination der Teilformate) geschlossen werden kann. Nur die gesamte Kombination selbst lässt auf den logischen Bildschirm schließen. Erst wenn das letzte Teilformat ermittelt ist, ist also der logische Bildschirm eindeutig identifizierbar.

Eindeutiger Beginn der Teilformat-Folge

Wenn Sie wissen, dass auf ein bestimmtes Teilformat immer eine bestimmte Folge von Teilformaten folgt, so können Sie alle zugehörigen Templates hintereinander in einer Datei mit dem Namen des 1. Teilformats ablegen. Sie lösen also die Include-Tags auf und ziehen mehrere Templates zu einem großen zusammen.

Dazu kopieren Sie das Folge-Template an die Stelle des Include-Tags des Vorgänger-Templates. Die IF-Tags in [Abschnitt \(3\)](#), [Abschnitt \(5\)](#) und [Abschnitt \(7\)](#) können innerhalb der Template-Folge gestrichen werden, da die Reihenfolge der Templates festliegt. In [Abschnitt \(7\)](#) muss der ELSE-Zweig jedoch erhalten bleiben!

Der Rumpf von [Abschnitt \(3\)](#) wird einmal an den Anfang des Gesamt-Templates gestellt, der [Abschnitt \(5\)](#) an das Ende.

In einem solchen Gesamt-Template können Sie leichter Inhalte der verschiedenen Teilformate in neuer Reihenfolge auf der HTML-Seite darstellen. Allerdings müssen Sie auch hier beachten, welche Generation von Host-Datenobjekten zum Generierungszeitpunkt gerade aktiv ist. Evtl. müssen Sie eine Generation zum Generierungszeitpunkt wiederherstellen (siehe [Abschnitt \(2\)](#)) oder einige Werte im Template-Objekt zur späteren Verwendung zwischenspeichern.

Beliebige Kombinationen bis zum Ende der Teilformat-Folge

Werden die Teilformate in Ihrer Host-Anwendung in beinahe beliebigen Kombinationen zusammengestellt und wollen Sie trotzdem die einzelnen Kombinationen grundlegend neu am Browser darstellen, so empfiehlt sich das folgende Vorgehen.

Die Teilformat-Templates bleiben als einzelne Templates erhalten. Sie generieren aber nicht HTML, sondern speichern lediglich die empfangenen Werte der Host-Anwendung für die Weiterverarbeitung. Außerdem hinterlegen die einzelnen Templates ihr Auftreten in der Template-Folge in einem Spezial-Attribut. Sind alle Teilformat-Templates abgearbeitet, wird ein neues Template (Kombinations-Template) aufgerufen, das aus den zwischengespeicherten Daten der gesamten Teilformat-Folge die HTML-Seite generiert und die OnReceive-Tags enthält.

1. Löschen Sie aus den Teilformat-Templates die HTML-generierenden Teile in [Abschnitt \(3\)](#), [Abschnitt \(4\)](#) und [Abschnitt \(5\)](#). Diese Teile werden im Kombinations-Template zusammengestellt oder entsprechend abgewandelt.
2. Ersetzen Sie den [Abschnitt \(4\)](#) durch ein `wtOnCreate`-Script, welches die Inhalte der Host-Datenobjekte auf eindeutige Attribute eines Sammelobjekts kopiert. Z.B. können Sie `FORMAT_SEQ` als eine Stufe in dieser Objektstruktur verwenden:

```
collect = new Object;  
collect['_'+host_system.FORMAT_SEQ] = new Object;  
collect['_'+host_system.FORMAT_SEQ].object = %%CommObj%.object.Value;  
...
```

(In einem generierten Template steht an Stelle von `%%CommObj%` das aktuelle Kommunikationsobjekt.)

3. Löschen Sie den [Abschnitt \(6\)](#) und schreiben Sie entsprechende Anweisungen in das Kombinations-Template.
4. Ergänzen Sie eine Anweisung, die die Reihenfolge der Templates in einem Template-Objekt-Attribut festhält.

```
if ( COMBINATION_TEMPLATE )  
    COMBINATION_TEMPLATE += %%CommObj%_system.FLD;  
else  
    COMBINATION_TEMPLATE = %%CommObj%_system.FLD
```

(In einem generierten Template steht an Stelle von `%%CommObj%` das aktuelle Kommunikationsobjekt.)

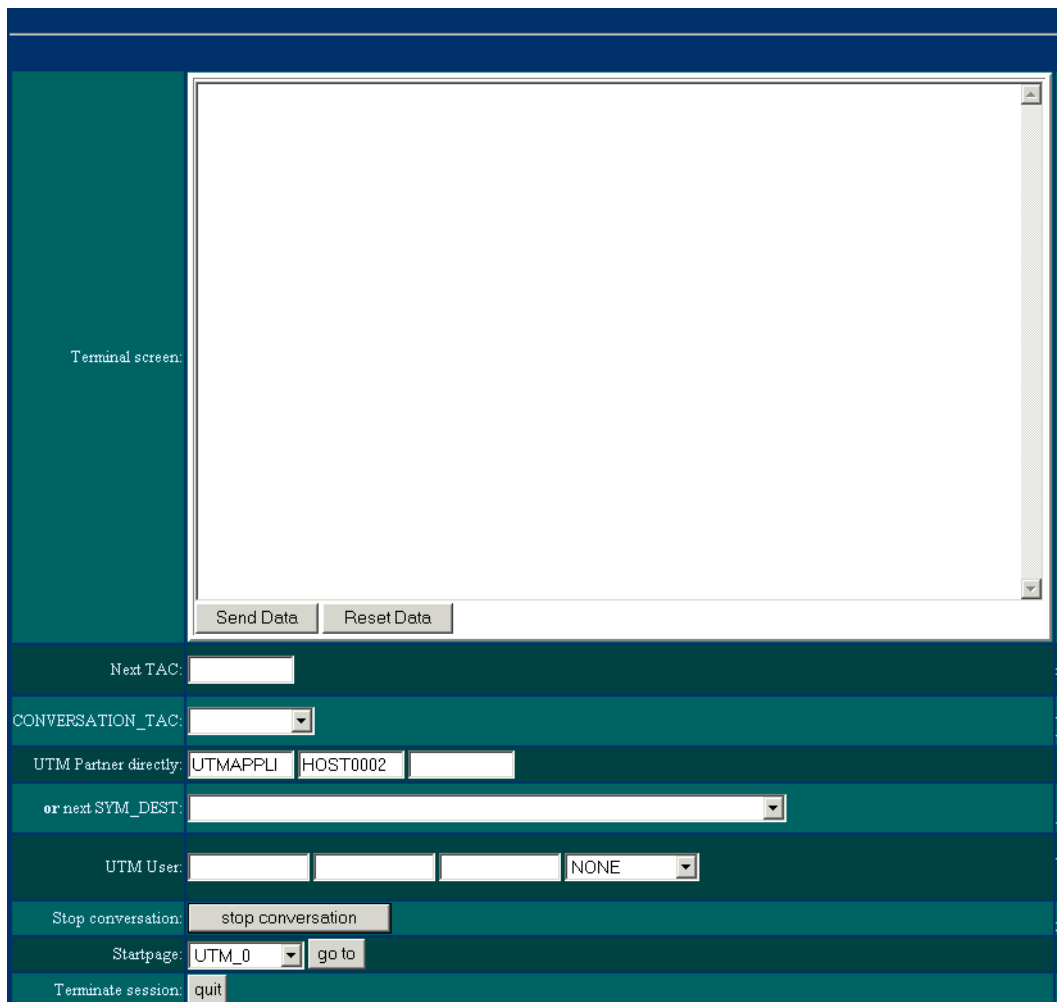
5. Ergänzen Sie ein `Include`-Tag, das ein Template für die jeweils empfangene Folge von Teilformaten aufruft.

```
<wtInclude Name="##COMBINATION_TEMPLATE#">
```

6. Das `COMBINATION_TEMPLATE` für die jeweilige Folge kann nun die vollständige HTML-Seite generieren. Dabei wird auf die im Sammelobjekt gespeicherten Daten zugegriffen. Es enthält ein `OnReceive`-Script für alle in der Folge auftretenden Teilformate, um die geposteten Daten in einzelnen Teilnachrichten an die `openUTM`-Anwendung zurück zu schicken. Den Abschluss bildet eine `receive`-Anweisung, die die nächste Nachricht von der `openUTM`-Anwendung entgegennimmt.

6.4 Unterstützung des openUTM-Zeilenmodus

WebTransactions unterstützt Nachrichten im openUTM-Zeilenmodus. Wird von der openUTM-Anwendung eine nicht formatierte Nachricht ausgegeben, so verwendet WebTransactions für die Darstellung automatisch das spezielle Zeilenmodus-Template `wtlmode.htm`. Dieses Template wird beim Erzeugen des Basisverzeichnisses als Link bzw. als Kopie in `basedir\config\forms` abgelegt. Es erscheint am Browser wie in der folgenden Abbildung dargestellt:



The screenshot displays a web-based terminal interface. On the left, a dark green vertical bar contains the text "Terminal screen:". To the right of this bar is a large white rectangular area representing the terminal output, which is currently empty. Below the terminal area are two buttons: "Send Data" and "Reset Data".

Below the terminal area, there are several input fields and buttons:

- "Next TAC:" followed by a text input field.
- "CONVERSATION_TAC:" followed by a dropdown menu.
- "UTM Partner directly:" followed by two text input fields containing "UTMAPPLI" and "HOST0002", and a third empty text input field.
- "or next SYM_DEST:" followed by a dropdown menu.
- "UTM User:" followed by three text input fields and a dropdown menu with "NONE" selected.
- "Stop conversation:" followed by a button labeled "stop conversation".
- "Startpage:" followed by a dropdown menu with "UTM_0" selected and a button labeled "go to".
- "Terminate session:" followed by a button labeled "quit".

Das Zeilenmodus-Template enthält die folgenden Dialogelemente:

Terminal screen:

Zeigt die zuletzt von der openUTM-Anwendung empfangene Nachricht an. Wie beim Terminal werden hier auch die Kommandos eingegeben und abgeschickt. Es werden die Daten ab der ersten geänderten Position bis zur nächsten Endemarke berücksichtigt.

Am Vorgangsanfang werden die maximal ersten 8 Zeichen als TAC zum Adressieren des Vorgangs herangezogen, dazu müssen Sie im Feld

CONVERSATION_TAC Leerzeichen auswählen. Die übrigen Daten werden dem Teilprogramm als Nachricht zugeschickt. Im Vorgang werden die kompletten Daten an openUTM als Nachricht geschickt. Die Endemarke wird durch das Zeichen ~ (Tilde) repräsentiert.

Das Formular kann durch die Taste „Enter“ oder die Schaltfläche **Send Data** abgeschickt werden. Zeilenumbrüche können in dem Textbereich durch die Tastenkombination „Shift+Enter“ eingegeben werden.

Next TAC

Hier können Sie den nächsten TAC eingeben. Dazu müssen Sie im Feld **CONVERSATION_TAC** die Option TAC auswählen.

CONVERSATION_TAC

Hier wählen Sie aus, wie der TAC zur Adressierung des nächsten openUTM-Vorgangs angegeben wird:

Leerzeichen Der TAC muss am Beginn der Nachricht stehen (siehe **Terminal screen**).

TAC Der TAC muss in das Feld **TAC** eingegeben werden.

SYM_DEST Der TAC wird aus dem entsprechenden Eintrag in der `upicfile` ermittelt (siehe **or next SYM_DEST**).

UTM Partner directly:

Hier können Sie die Adresse der openUTM-Anwendung direkt eingeben. In das erste Feld tragen Sie den Namen der Host-Anwendung und in das zweite Feld den Namen des Rechners ein, auf dem die Host-Anwendung läuft. Alternativ zum Namen des Rechners können Sie im dritten Feld die IP-Adresse des Rechners eingeben.

or next SYM_DEST:

Hier können Sie einen Symbolic Destination Name aus der `upicfile` angeben, um mit einem anderen Vorgang fortzufahren. Diese Vorgänge müssen mit einem Teilprogramm beginnen, das keine Eingabe erwartet, da kein `send` durchgeführt wird.

UTM User:

Jedes Kommando, das im Zeilenmodus abgesetzt wird, läuft in einem eigenen Vorgang. Für diese Vorgänge können Sie jeweils individuell die Stufe der Authentifizierung (`NONE|USER|PASSWORD`) sowie Benutzerkennung, Passwort und ggf. neues Passwort angeben.

Stop Conversation:

Hiermit können Sie einen Vorgang jederzeit abbrechen. Dieser Aktionsknopf ist als „Notausstieg“ gedacht.

Startpage:

Hier können Sie zum Start-Template `wtstart.htm` zurückkehren, falls für den neu zu startenden Vorgang zusätzliche Angaben notwendig sind. So lässt sich auch ein Vorgang starten, der mit einem Teilprogramm beginnt, das eine Eingabe erwartet (`send` beim Vorgangsstart).

Terminate session:

Aus dem Zeilenmodus heraus lässt sich mit dem Aktionsknopf **Quit** auch die gesamte Sitzung beenden.

Zeilenmodus-Template anpassen

Das Zeilenmodus-Template `wtlnmode.htm` können Sie auch kopieren und abändern. Der Name des Zeilenmodus-Templates `wtlnmode.htm` darf jedoch nicht geändert werden.

Die Struktur unformatierter Nachrichten wird durch die Datei `wtlnmode.fld` beschrieben, die sich im Verzeichnis `basedir/config` befindet. Der Inhalt der Nachricht wird in das Host-Datenobjekt `CONTENTS` geschrieben; `CONTENTS` hat die Länge von 32767 byte.

Senden unformatierter Nachrichten an openUTM

Soll eine Nachricht an ein openUTM-Teilprogramm geschickt werden, so belegt man `CONTENTS.Value` mit dem Wert und ruft die Methode `send` auf. WebTransactions schickt als Nachricht nur den Inhalt von `CONTENTS` bis zur ersten binären Null. Das Feld `CONTENTS` hat als Eingabefüllzeichen binäre Nullen. Wird `CONTENTS.Value` ein Wert zugewiesen, der kürzer als 32767 ist, so wird der Rest des Feldes mit Nullen aufgefüllt. So ist es möglich unformatierte Nachrichten unterschiedlicher Länge zu schicken.

Bis zur Version WebTransactions V3.0 war der Inhalt des Attributes `FLD` ausschlaggebend dafür, ob eine formatierte oder unformatierte Nachricht gesendet wurde. Seit der Version V4.0 von WebTransactions ist in der `FLD`-Datei, auf die das Attribut `FLD` zeigt, die Eigenschaft `FormatType=-` entscheidend.

Werden kundenspezifische Änderungen an `wtlnmode.fld` oder `wtlnmode.htm` vorgenommen, so ist darauf zu achten, dass bei einem Versionswechsel diese beiden Dateien konsistent bleiben.

Empfangen unformatierter Nachrichten von openUTM

Beim Empfang unformatierter Nachrichten legt WebTransactions den Inhalt in `CONTENTS.Value` ab. Werden mehrere unformatierte Teilnachrichten an WebTransactions geschickt, so werden diese durch den Aufruf der Methode `receive` bis zu einer Länge von 32767 im Feld `CONTENTS.value` aufgesammelt. Wurden darüber hinaus weitere unformatierte Nachrichten empfangen, so setzt WebTransactions wie bei Teilformaten das Attribut `FORMAT_SEQ` auf den Wert 1. Um die komplette Nachricht zu empfangen ist dann die Methode `receive` wiederholt aufzurufen, bis `FORMAT_SEQ` den Wert `LAST` hat.

Ist nicht von vornherein bekannt wie viele Teilnachrichten empfangen werden, so kann man die komplette Nachricht in einer Schleife aufbauen:

```
for( host.receive(), msg=host.CONTENTS.Value;
    host.WT_SYSTEM.FORMAT_SEQ!=" " && host.WT_SYSTEM.FORMAT_SEQ!="LAST";
    host.receive(), msg+=host.CONTENTS.Value );
```

Beispiel : Angepasstes Zeilenmodus-Template

```
<html> (1)
  <head>
    <title>Bulletin</title>
  </head>
  <body>
    <pre>##UTM_0.CONTENTS.Value#</pre>
    <wtDataForm>
      <input type="submit" name="GO" value="Fortsetzen">
    </wtDataForm>
  </body>
</html>
```

```
<wtOnReceiveScript> (2)
<!--
  UTM_0.CONTENTS.Value=" ";
  WT_SYSTEM.SYM_DEST="MAIN ENTER">;
  UTM_0.receive();
  setNextPage(UTM_0.WT_SYSTEM.FLD);
/-->
</wtOnReceiveScript>
```

Das obige Beispiel zeigt ein Zeilenmodus-Template, das lediglich eine Ausgabe macht und dann mit einem ganz bestimmten Vorgang fortfährt.

Im Bereich (1) wird eine HTML Seite erzeugt. Diese gibt den Inhalt der Nachricht aus dem Attribut `Value` des Host-Datenobjekts `CONTENTS` wieder. Die Seite enthält eine Schaltfläche zur Fortsetzung der Sitzung. Wird diese aktiviert, so setzt WebTransactions die Verarbeitung mit `wtOnReceiveScript` (Bereich (2)) fort.

In Bereich (2) wird zunächst `CONTENTS` gelöscht, um eine Vorgangskettung zu verhindern. Ein bestimmter Vorgang wird durch Angabe des Attributs `SYM_DEST` ausgewählt. `receive` öffnet implizit den Vorgang und empfängt die nächste Nachricht.

`receive` versorgt in dem verbindungspezifischen Systemobjekt das Attribut `FLD` mit dem Formatnamen. Wenn das empfangene Format erneut eine Linemode-Nachricht ist, wird der Wert wieder `wt1nmode` sein. Damit das Format auf jeden Fall mit dem passenden Template präsentiert werden kann, sollte aber zur Sicherheit mit `setNextPage()` der Formatname als nächstes auszuführendes Template übernommen werden.

6.5 Binärdaten-Unterstützung

Mit WebTransactions for openUTM können Sie auch beliebige Daten wie z.B. Binärdaten übertragen. Diese Daten können beim Empfangen in eine Datei geschrieben und beim Senden aus einer Datei gelesen werden.

Der Name der Datei wird im Systemattribut `COMMUNICATION_FILE_NAME` angegeben. Damit wird diese Datei im folgenden `send` als Eingabe bzw. in einem folgenden `receive` als Ausgabe verwendet. Die Angabe in `COMMUNICATION_FILE_NAME` gilt immer nur für den unmittelbar darauf folgenden Kommunikationsschritt. Danach wird das Attribut wieder zurückgesetzt, d.h. die nächste Ein-/Ausgabe wird wieder über Host-Datenobjekte abgewickelt.

Beispiel

Mit WebTransactions soll über openUTM auf eine SESAM/SQL-Datenbank zugegriffen werden, die auch Bilddateien enthält. Diese Datenbank dient zur Verwaltung von Personaldateien und enthält neben den personenbezogenen Daten (Geburtsdatum, Adresse, ...) auch Bilddateien mit den Passfotos der Personen.

Das folgende Code-Beispiel zeigt, wie man diese Fotos in der Datenbank abspeichern und sich auch wieder anzeigen lassen kann.

```
<wtOnCreateScript>
// Speichern des Fotos, evtl. über eigenen TAC
UTM_0_SYSTEM.COMMUNICATION_FILE_NAME = "hans-mustermann.gif";
    UTM_0.send();
    ...
// Bild-Retrieval und Anzeige
UTM_0_SYSTEM.COMMUNICATION_FILE_NAME = "hans-mustermann.gif";
    UTM_0.receive();
    ...
</wtOnCreateScript>
```

Verarbeitung von Binärdaten mit WTML

Binärdaten lassen sich auch in Hexadezimalform darstellen. Dazu dient das Host-Datenobjekt-Attribut `HexStringValue`. Dieses Attribut wandelt eine beliebige binäre Zeichenkette in die übliche hexadezimale Darstellung um (00 bis FF).

7 Kopplung konfigurieren

WebTransactions bildet das Bindeglied zwischen Web-Browsern und Host-Anwendung. Während WebTransactions aus Sicht der Web-Browser als Server fungiert, nimmt es gegenüber der Host-Anwendung die Rolle des Client an.

Für die Kommunikation mit der Host-Anwendung verwendet die Produktvariante „WebTransactions for openUTM“ das Client-Server-Kommunikationsprotokoll UPIC (Universal Programming Interface for Communications). UPIC ist ein komplexes Kommunikationsprotokoll, das mehr bietet als reinen Austausch von Netto-Daten. Die Kopplung über UPIC ermöglicht es WebTransactions beispielsweise, das openUTM-Benutzerkonzept und die automatischen Wiederanlauffunktionen von openUTM zu nutzen. Da über dieses Protokoll auch Informationen über Formatnamen und openUTM-Funktionstasten ausgetauscht werden können, sind für den Web-Anschluss keinerlei Änderungen an den openUTM-Teilprogrammen notwendig.

Alle Kommunikationskomponenten - Browser, HTTP-Dämon mit WebTransactions, openUTM-Anwendung und deren Datenhaltung - können beliebig über verschiedene Plattformen verteilt werden (Multi-Tier-Architektur). WebTransactions steuert den transaktionsgesicherten Informationsaustausch zwischen Client und Server und garantiert auch bei komplexen verteilten Strukturen Zuverlässigkeit/Verfügbarkeit, Datensicherheit und Performance.

UPIC-R und UPIC-L

UPIC liegt in der Variante UPIC-R (remote) und für Solaris und Linux zusätzlich in der Variante UPIC-L (local) vor.

UPIC-L können Sie verwenden, wenn WebTransactions und die openUTM-Host-Anwendung auf demselben Unix-Rechner liegen. Die UPIC-L-Bibliothek geben Sie hierfür über das Attribut `UPIC_LIB` des Systemobjekts bekannt - den Pfad, unter dem die openUTM-Anwendung installiert ist, über das Attribut `UTM_PATH`.

In allen anderen Fällen wird UPIC-R verwendet.

UPIC-Bibliotheken

Die UPIC-Bibliotheken sind Bestandteil des Produkts WebTransactions:

- Für BS2000/OSD ist die UPIC-Bibliothek (UPIC-R) fest in die WTHolder-Programme eingebunden.
- Für Solaris und Linux werden mit WebTransactions zwei UPIC-Bibliotheken ausgeliefert (jeweils eine für UPIC-R und UPIC-L). Falls Sie mit UPIC-L arbeiten wollen, müssen Sie den Pfadnamen dieser Bibliothek im Attribut `UPIC_LIB` des Systemobjekts setzen. Falls Sie dieses Attribut nicht setzen, verwendet WebTransactions standardmäßig eine Default-UPIC-R-Bibliothek, die ins WTHolder-Programm eingebunden ist.
- Für Windows wird mit WebTransactions eine UPIC-R-Bibliothek ausgeliefert (`upicws32.dll`). Unter Windows wird UPIC standardmäßig dynamisch aus dieser Bibliothek nachgeladen.

Unicode-Unterstützung

Der Host-Adapter in WebTransactions for openUTM kann Daten an der Schnittstelle UPIC auch als Unicode-Zeichen interpretieren.

Das BS2000/OSD-Programm `IFG2FLD` liest Formatbeschreibungen aus einer IFG-Bibliothek und legt sie in einer Formatbeschreibungsource ab. Die Felder in einer IFG-Bibliothek können das neue Attribut `Unicode` enthalten. `IFG2FLD` ab Version 8.3 berechnet den Offset der einzelnen Felder im UPIC-Puffer abhängig von diesem Attribut.

Aus der Formatbeschreibungsource können Sie mit WebLab Templates und Felddateien (FLD-Dateien) generieren. Bei dieser Umsetzung wird ab `IFG2FLD` Version 8.3 das Unicode-Kennzeichen automatisch berücksichtigt (siehe [Kapitel „Templates generieren“ auf Seite 73](#)).

FLD-Dateien

Für alle Formate, die Unicode-Felder enthalten, müssen die FLD-Dateien neu generiert werden.

Templates

Bereits existierende Templates können unverändert bestehen bleiben, falls die Umstellung auf Unicode-Felder die einzige Änderung in einem Format war. Allerdings muss an zentraler Stelle oder im zugehörigen Format-spezifischen Template die Zuweisung des Wertes `UTF-8` auf das globale Systemobjekt-Attribut `CHARSET` eingefügt werden (siehe [Abschnitt „Host-Steuerobjekt WT_HOST_MESSAGE“ auf Seite 162](#)).

WebTransactions for openUTM erzeugt Daten in UTF-8-Codierung, die zum Browser und zurück durchgereicht werden. Für neue Templates müssen dazu folgende Regeln eingehalten werden:

- Die Templates müssen den Browser über `charset` informieren, welche Zeichen-Codierung zum Anzeigen der Daten und zum Versenden der Antwort verwendet werden soll. Der Browser erhält diese Information
 - entweder über das Meta-Tag:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```
 - oder durch das Setzen des Attributs `charset` im HTTP-Header-Feld `Content-Type`.
- Die Templates dürfen ausschließlich ASCII-Zeichen enthalten. Enthält ein Template Sonderzeichen (z.B. Umlaute) direkt als ANSI-Zeichen in Zeichenketten, so müssen diese durch die HTML-Escape-Sequenzen ersetzt werden. Die Escape-Sequenzen bestehen ihrerseits nur aus ASCII-Zeichen und können daher in UTF-8 verwendet werden.
- Zeichenkettenoperationen können fehlerhafte Ergebnisse produzieren, wenn in der Zeichenkette Zeichen in UTF-8 enthalten sind. In den von WebTransactions generierten Templates kommen solche Operationen nicht vor.

ASCII-EBCDIC-Konvertierung

WebTransactions-Anwendungen, die bisher die in UPIC integrierte ASCII-EBCDIC-Konvertierung nutzen, müssen auf die ASCII-EBCDIC-Konvertierung von WebTransactions umgestellt werden (siehe „[HOST_CHAR_CODE](#)“ auf Seite 145).

7.1 WebTransactions und Host abstimmen

Für die Kommunikationsverbindung über UPIC-R müssen sich die Kommunikationspartner - WebTransactions und die openUTM-Anwendung - gegenseitig bekannt machen, und zwar sowohl auf der Client- als auch auf Server-Seite.

UPIC nutzt für seine Services die Dienste des plattformspezifischen Kommunikationssystems, das den Zugriff auf das Transportsystem ermöglicht und die Transportverbindungen verwaltet:

- Für BS2000/OSD ist das BCAM. Für BCAM (BS2000/OSD) sind entsprechende BCMAP-Einträge nur in Ausnahmefällen notwendig (siehe [Abschnitt „BCMAP-Einträge \(BS2000/OSD\)“](#) auf Seite 138).
- Für Solaris, Linux und Windows sind die benötigten Komponenten des Transportsystems in UPIC integriert, die Transportsysteme CMX (Solaris und Linux) bzw. PCMX (Windows) müssen daher nicht mehr verwendet werden.

Auf der Host-Seite übernimmt das plattformspezifische Kommunikationssystem die Vermittlung zwischen Transportsystem und openUTM-Host-Anwendung.

In der Praxis sind drei Kopplungsvarianten von Bedeutung:

WebTransactions-Plattform	Host-Plattform
Solaris, Linux oder Windows	BS2000/OSD
Solaris, Linux oder Windows	Solaris, Linux oder Windows
BS2000/OSD	BS2000/OSD

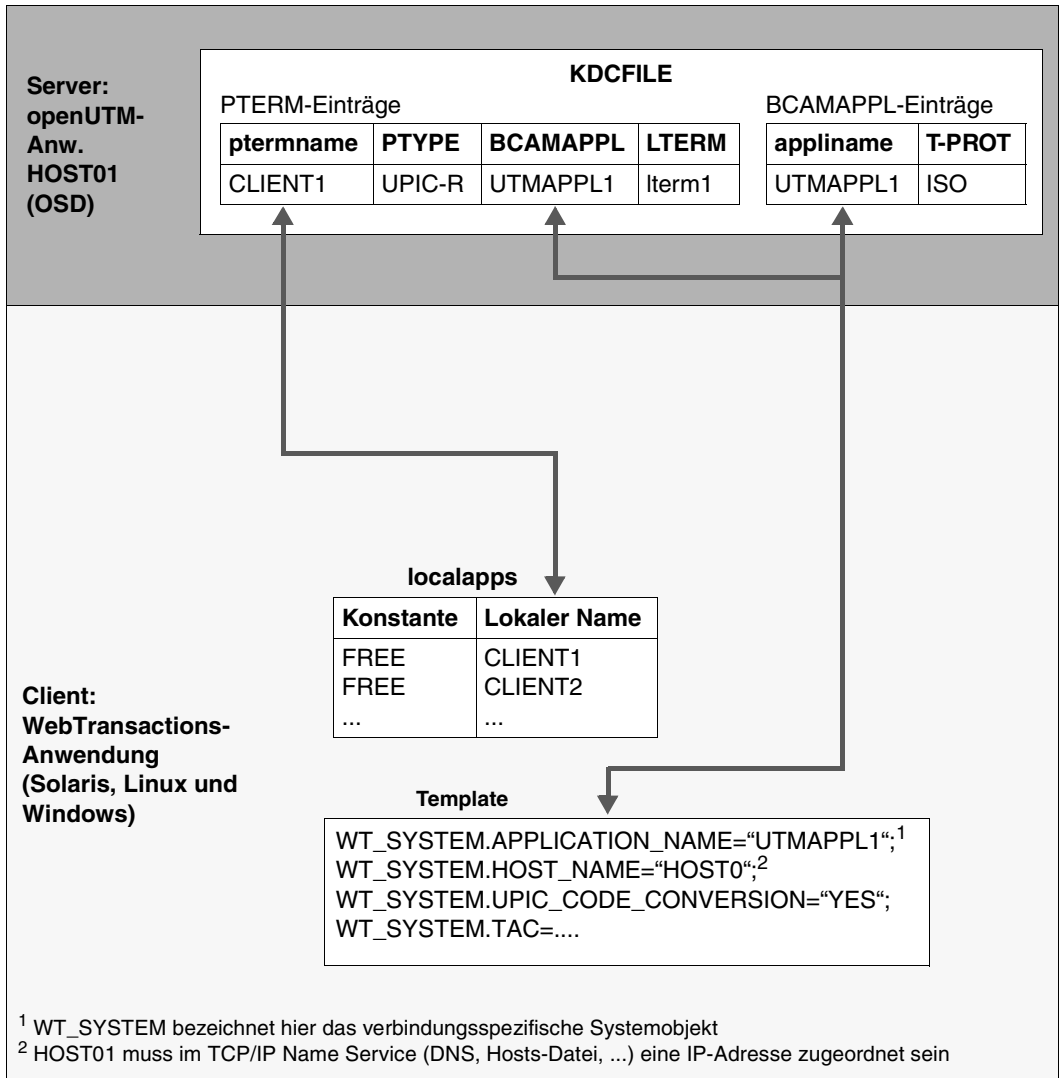
Da die Konfiguration in Solaris-, Linux- und Windows-Systemen sehr ähnlich ist, wird sie zusammen betrachtet. Theoretisch ergibt sich noch als vierte Kombinationsmöglichkeit WebTransactions im BS2000/OSD und openUTM-Host-Anwendung auf Solaris, Linux oder Windows. Eine solche Konfiguration ist aber in der Praxis unwahrscheinlich und wird deshalb auch nicht beschrieben.

Bei allen diesen Varianten ist auf WebTransactions-Seite Folgendes notwendig:

- Für die lokale WebTransactions-Anwendung:
Einträge in die Datei `localapps`, siehe [Seite 128](#)
- Für die openUTM-Anwendung:
Einträge in die Datei `upicfile` (siehe [Seite 130](#)) oder Versorgen der entsprechenden Systemattribute `APPLICATION_NAME`, `HOST_NAME`, `TAC`, ... (siehe [Seite 129](#)).

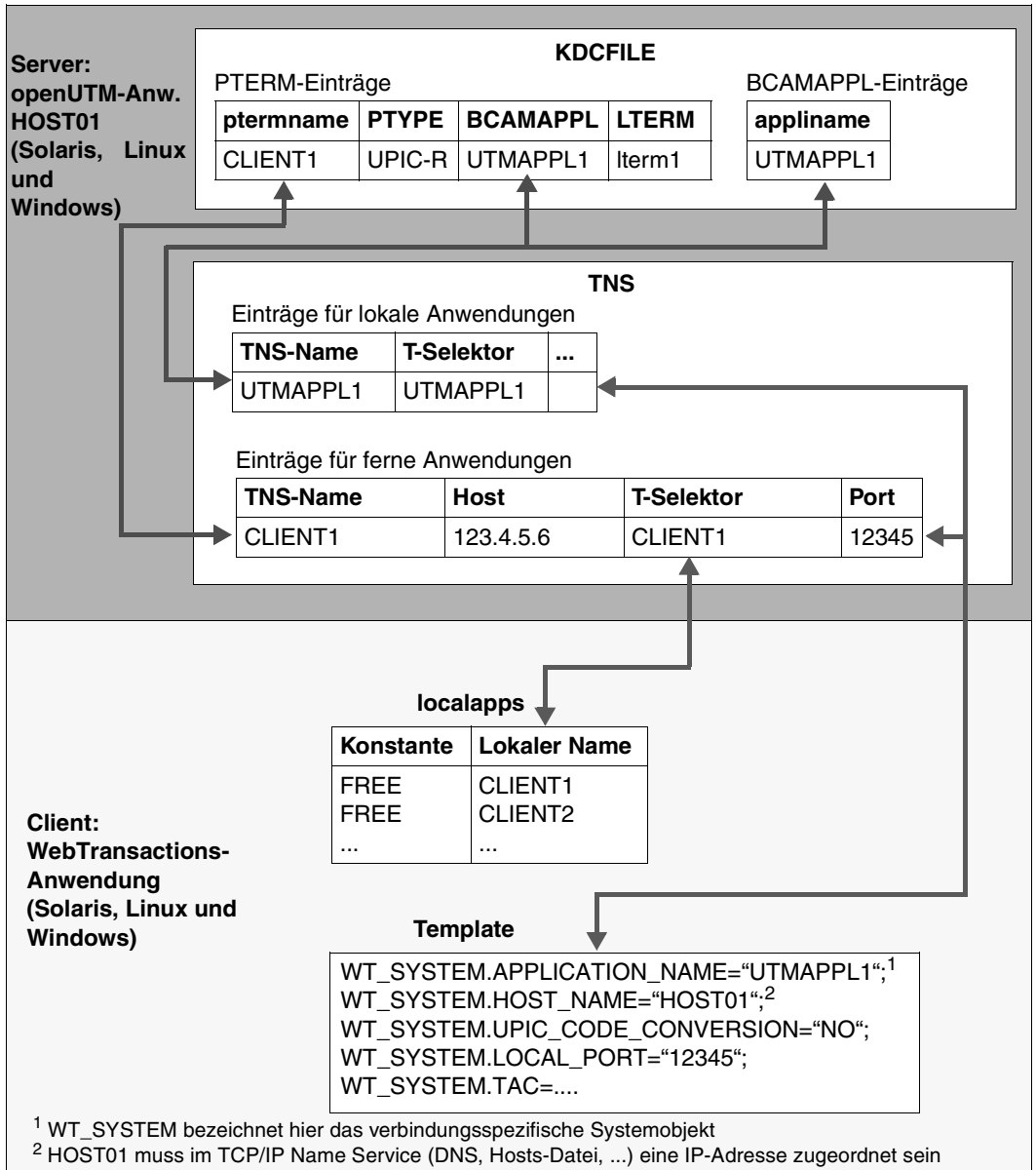
Die folgenden Abbildungen illustrieren die verschiedenen Kombinationsmöglichkeiten und die Bezüge zwischen den Einträgen. Nähere Informationen und Beispiele zu den einzelnen Einträgen enthalten die daran anschließenden Abschnitte „[WebTransactions-Seite konfigurieren](#)“ auf [Seite 128](#) und „[openUTM-Seite \(Host\) konfigurieren](#)“ auf [Seite 134](#).

WebTransactions auf Solaris, Linux oder Windows – openUTM-Host unter BS2000/OSD



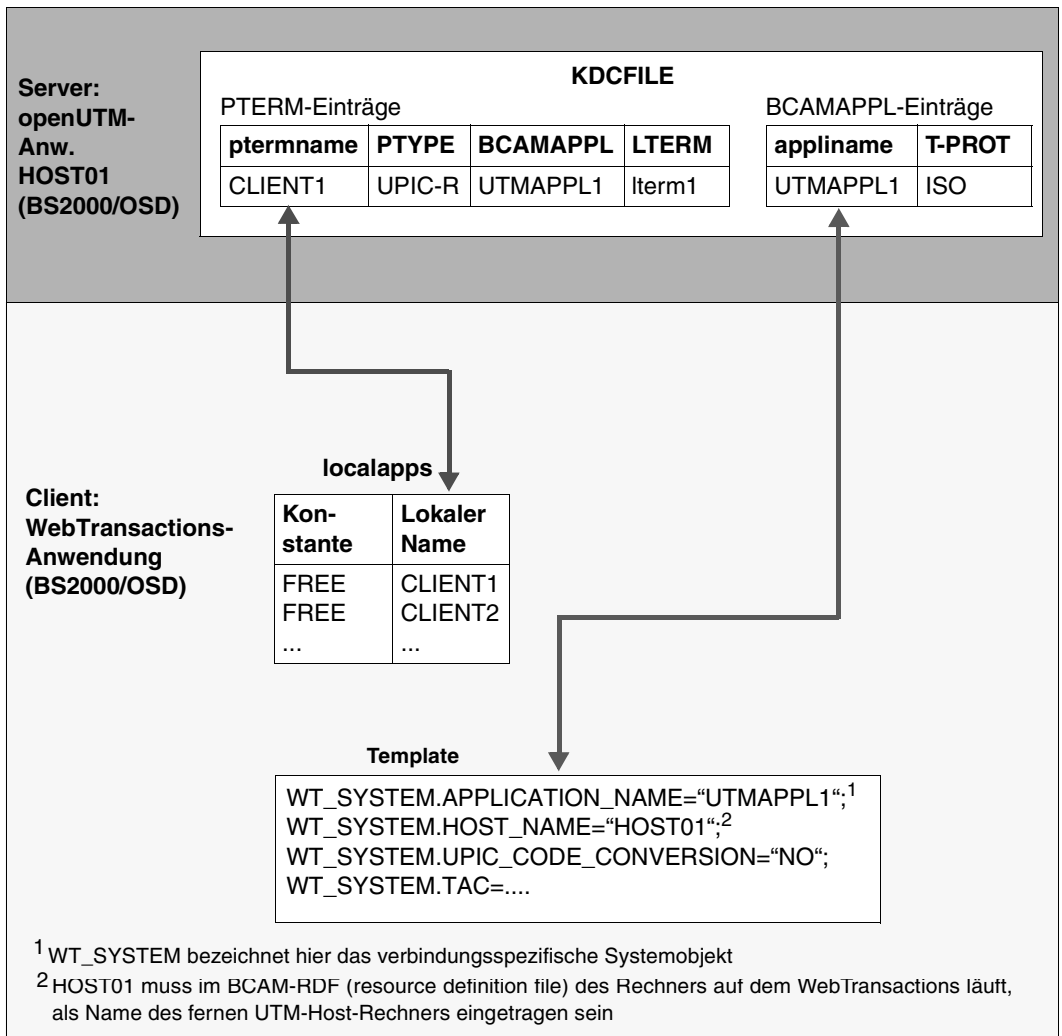
In diesem Beispiel wurden die Parameter für die openUTM-Anwendung in den Systemobjekt-Attributen angegeben. Alternativ dazu können Sie auch die `upicfile` verwenden, siehe [Seite 130](#). In diesem Fall müssen Sie das Systemobjekt-Attribut `SYM_DEST` versorgen.

**WebTransactions auf Solaris, Linux oder Windows –
openUTM-Host auf Solaris, Linux oder Windows (über UPIC-R)**



In diesem Beispiel wurden die Parameter für die openUTM-Anwendung in den Systemobjekt-Attributen angegeben. Alternativ dazu können Sie auch die `upicfile` verwenden, siehe [Seite 130](#). In diesem Fall müssen Sie das Systemobjekt-Attribut `SYM_DEST` versorgen.

WebTransactions unter BS2000/OSD – openUTM-Host unter BS2000/OSD (Standardfall: ohne BCMAP-Einträge)



BCMAP-Einträge sind nur mehr in Ausnahmefällen notwendig (siehe [Abschnitt „BCMAP-Einträge \(BS2000/OSD\)“ auf Seite 138](#)).

In diesem Beispiel wurden die Parameter für die openUTM-Anwendung in den Systemobjekt-Attributen angegeben. Alternativ dazu können Sie auch die `upicfile` verwenden, siehe [Seite 130](#). In diesem Fall müssen Sie das Systemobjekt-Attribut `SYM_DEST` versorgen.

7.2 WebTransactions-Seite konfigurieren

Sie müssen auf der WebTransactions-Seite Folgendes konfigurieren:

- Die lokale WebTransactions-Anwendung, indem Sie Einträge in die Datei `localapps` machen
- Die openUTM-Anwendung. Dazu haben Sie zwei Möglichkeiten:
 - über Systemattribute, siehe [Seite 129](#)
 - über die Datei `upicfile`, siehe [Seite 130](#)

Bei einigen Konfigurationen ist es auch notwendig, im Attribut `LOCAL_PROT` die Absender-Portnummer der lokalen Socketverbindung anzugeben, siehe auch [Seite 145](#).

7.2.1 Datei `localapps`

Das WTHolder-Programm meldet sich über einen lokalen Namen beim Kommunikationssystem an. Das Kommunikationssystem setzt diesen Namen dann in einen Namen um, der dem darunter liegenden Transportsystem bekannt ist.

Soll das WTHolder-Programm mehrfach gleichzeitig ablaufen können, ist für jede parallele Verbindung ein eigener lokaler Name notwendig, der dann vom Kommunikationssystem in einen T-Selektor umgesetzt wird.

WebTransactions steuert die Vergabe der lokalen Namen an die WTHolder-Programme über die Datei `localapps`. Diese Datei muss im WebTransactions-Basisverzeichnis stehen und folgende Einträge enthalten:

- pro gleichzeitigem Ablauf ein Eintrag
- Text „FREE“ ab Spalte 1 (während ein Eintrag benutzt wird steht hier die Session-ID)
- lokale Namen ab Spalte 16

Die Datei kann auch Kommentare enthalten. Diese beginnen mit „*“ in Spalte 1.



`localapps` wird automatisch beim Erzeugen eines Basisverzeichnisses angelegt und enthält bereits Default-Einträge.

Beispiel `localapps`

```
*<Session-ID> <Name>
FREE          LOCAL1
FREE          LOCAL2
```


WebTransactions auf BS2000/OSD: automatisches Mapping

Für BCAM sind Konfigurationseinträge für das Kommunikationssystem BCAM (BCMAP-Einträge) nur mehr in Ausnahmefällen notwendig (siehe [Abschnitt „BCMAP-Einträge \(BS2000/OSD\)“ auf Seite 138](#)).

Lokale Namen bei UPIC-L (Solaris, Linux)

Bei lokaler Kopplung über UPIC-L müssen die in `localapps` festgelegten lokalen Namen entweder mit einem PTERM-Namen der openUTM-Anwendung übereinstimmen oder es muss bei der openUTM-Generierung eine KDCDEF-Anweisung `TPOOL mit PTYPE=UPIC-L` angegeben werden.

7.2.2 Adressierung der openUTM-Anwendung über Systemattribute

Der Service der openUTM-Anwendung kann auch direkt über Systemattribute adressiert werden, ohne Verwendung eines Eintrags in der `upicfile`, auf den mit dem Systemattribut `SYM_DEST` verwiesen wird. In der `upicfile` ist dann ein Eintrag notwendig, der von aktuellen WebTransactions-Versionen vorkonfiguriert ist, aber bei Übernahme aus einer alten WebTransactions-Installation fehlen könnte.

Im BS2000/OSD lautet die Zeile:

```
HD.DEFAULT NOCONN;
```

Siehe auch Hinweise im Abschnitt [„upicfile in DVS-Benutzererkennung bereitstellen \(BS2000/OSD\)“ auf Seite 71](#).

In Windows/Linux/Solaris lautet die Zeile:

```
SD.DEFAULT NOCONN.
```

Für die direkte Adressierung stehen folgende Attribute zur Verfügung:

```
APPLICATION_NAME
```

Name der openUTM-Anwendung

```
HOST_NAME oder HOST_IP_ADDRESS
```

Name bzw. IP-Adresse des Rechners, auf dem die openUTM-Anwendung läuft.

```
HOST_PORT
```

Portnummer, unter der die openUTM-Anwendung erreichbar ist. Die Portnummer muss nur angegeben werden, wenn die openUTM-Anwendung nicht unter Port 102 erreichbar ist.

```
TAC
```

Transaktionscode, mit dem openUTM den Service startet.

```
UPIC_CODE_CONVERSION
```

Gibt an, ob eine Code-Konvertierung durchgeführt werden soll.

LOCAL_PORT

Portnummer an, die im Client-Rechner als Ausgangsport verwendet wird; nur bei bestimmten Konfigurationen notwendig.

Eine detaillierte Beschreibung dieser Attribute finden Sie ab [Seite 142](#).

7.2.3 Datei upicfile

Die Datei `upicfile` enthält die Daten, die für die Kommunikation mit der openUTM-Anwendung notwendig sind. Das Systemattribut `SYM_DEST` (siehe [Seite 147](#)) wird dazu verwendet, um den Eintrag in der Datei `upicfile` zu ermitteln.

Der `upicfile`-Eintrag konkretisiert für UPIC die symbolische Zielangabe des Systemattributs `SYM_DEST`, indem er u.a. einen Host-Anwendungsnamen, einen Host-Namen und einen Transactionscode zuordnet.

Damit UPIC auf die `upicfile` zugreifen kann, muss sie bei Solaris, Linux und Windows im Basisverzeichnis stehen, für BS2000/OSD ist sie ins DVS zu kopieren (siehe [Seite 71](#)).



Die `upicfile` wird automatisch beim Erzeugen eines Basisverzeichnisses angelegt und enthält bereits Beispiel-Einträge, darunter den Eintrag `SD.DEFAULT NOCONN`.

Die `upicfile` wird immer ausgewertet, wenn sie im Basisverzeichnis vorhanden ist, auch dann, wenn eine Host-Anwendung direkt über Systemobjekt-Attribute adressiert wird.

Inhalt der upicfile

Die Datei `upicfile` muss folgende Einträge in Großbuchstaben enthalten:

- Kennzeichen für ASCII-EBCDIC-Konvertierung (HD oder SD)
- Symbolic Destination Name, über den UPIC auf die Einträge in der Datei `upicfile` zugreift
- Name der Host-Anwendung und des Hosts, auf dem die openUTM-Anwendung läuft.
- Transaktionscode des ersten Teilprogramms eines openUTM-Vorgangs (VorgangstAC)
- ggf. die Portnummer, unter der die openUTM-Anwendung zu erreichen ist.


Unter BS2000/OSD muss jede Zeile der `upicfile` mit einem Strichpunkt abgeschlossen werden. Außerdem darf die Datei keine Kommentare enthalten.

Unter Windows muss jede Zeile (auch die letzte) der `upicfile` mit `CR LF` (Return-Taste) abgeschlossen werden. Auf Solaris, Linux und Windows sind in der `upicfile` auch Kommentarzeilen zulässig. Diese beginnen jeweils mit dem Zeichen `*` in der ersten Spalte.

Ein Kommunikationsziel wird folgendermaßen eingetragen:

```
{HD|SD}sym_dest utmappl[.utmhost] tac [PORT=portnumber] [PROTOCOL={34|40}]
```

Bedeutung

- {HD|SD} stellt für die Kommunikation mit dem Host die automatische Code-Konvertierung zwischen ASCII und EBCDIC ein.
- HD die Daten werden vom Host in EBCDIC gesendet und so beim Empfang erwartet.
- SD die Daten werden vom Host in ASCII gesendet und so beim Empfang erwartet.
-  Die Konvertierung der Daten erfolgt nur bei Bedarf, wenn also das Client-System mit einem vom Host-System abweichenden Code arbeitet.
- sym_dest* Symbolischer Name, über den UPIC auf die Einträge in der `upicfile` zugreift. UPIC verwendet jeweils den Eintrag, dessen *sym_dest*-Name dem aktuellen Wert des System-Attributs SYM_DEST entspricht. *sym_dest* muss genau 8 Zeichen lang sein und ohne Leerzeichen an das Konvertierungskennzeichen angeschlossen werden.
- utmappl.utmhost* Identifiziert die openUTM-Anwendung.
- Bei UPIC-R muss der Name zweistufig und durch einen Punkt getrennt angegeben werden. Dabei ist *utmappl* der Name der openUTM-Anwendung und *utmhost* der Name des Rechners, auf dem die openUTM-Anwendung läuft. Vor *utmappl.utmhost* muss ein Leerzeichen stehen.
- Bei UPIC-L muss der Name einstufig angegeben werden (ohne *utmhost*). *utmappl* darf dann maximal 8 Zeichen lang sein.
- tac* Transaktionscode, mit dem openUTM den Service startet. Dieser Transaktionscode muss in der openUTM-Konfiguration erstellt worden sein, entweder bei der Generierung durch die KDCDEF-Anweisung TAC oder mittels dynamischer Konfigurierung.
- PORT=*portnumber* Portnummer, unter der die openUTM-Anwendung erreicht wird. Dieser Operand muss nur angegeben werden, wenn die openUTM-Anwendung nicht unter Port 102 erreichbar ist.

```
PROTOCOL={34|40}
```

In `PROTOCOL` legen Sie fest, ob die Kommunikation über das erweiterte UPIC-Protokoll der Version V4.0 (`PROTOCOL=40`) oder über das UPIC-Protokoll der Version V3.4 (`PROTOCOL=34`) erfolgen soll. Die Angabe von `PROTOCOL` ist optional.

Geben Sie `PROTOCOL` nicht an, dann versucht UPIC zunächst, eine Conversation auf Basis des erweiterten Protokolls (40) aufzubauen. Falls dies nicht gelingt, versucht UPIC als Nächstes, die Conversation auf Basis des Protokolls der V3.4 (34) aufzubauen.

Beispiel upicfile

```
HDSERVICE4 UTMAPPLI.HOST0001 TAC4
```

Dieser Eintrag wird von UPIC verwendet, falls `SERVICE4` der aktuelle Wert des Attributs `WT_SYSTEM.SYM_DEST` ist. Den Namen `UTMAPPLI.HOST0001` benutzt UPIC zur Identifikation der Host-Anwendung. `HOST0001` ist der Hostname, der im TCP/IP Name Service eingetragen ist, siehe auch Abschnitt „[Server-Rechner bekannt machen](#)“ auf Seite 133.

`UTMAPPLI` steht für den Host-Anwendungsnamen. Wenn die openUTM-Anwendung ohne TNS (Solaris, Linux und Windows) bzw. ohne BMAP-Einträge (BS2000/OSD) arbeitet, dann ist `UTMAPPLI` der Name, der in openUTM mit `BCAMAPPL` generiert ist, siehe [Seite 134](#). Wenn die openUTM-Anwendung TNS bzw. BMAP verwendet, dann ist `UTMAPPLI` der Name im zugehörigen TNS- bzw. BMAP-Eintrag.

Als Portnummer wird hier 102 verwendet, da `PORT=` nicht angegeben wurde.

Da in diesem Beispiel die Angabe `PROTOCOL` fehlt, versucht UPIC zunächst einen Verbindungsaufbau über das Protokoll der Version 4.0, falls dieser scheitert wird die Verbindung über das Protokoll der Version 3.4 aufgebaut.

7.2.4 Server-Rechner bekannt machen

Client- und Server-Rechner müssen einander bekannt sein. Dazu gibt es zwei Möglichkeiten:

- Beide Rechner sind für den DNS (Domain Name Service) konfiguriert. In diesem Fall sind keine weiteren Einträge notwendig.
- Ansonsten muss der Server-Rechner, auf dem die openUTM-Anwendung läuft, auf dem Client-Rechner mit seiner Internet-Adresse eingetragen sein:
 - Für Solaris, Linux und Windows in der `hosts`-Datei:
internet-adresse UTM-Host
Beispiel: `123.4.5.6 HOST0001`
 - Für BS2000/OSD muss der symbolische Name statisch in der BCAM-RDF (resource-definition-file) oder dynamisch per BCIN eingetragen sein.
 - Alternativ kann das verbindungsspezifische Systemobjekt-Attribut `HOST_IP_ADDRESS` verwendet werden (siehe [Abschnitt „Adressierung der openUTM-Anwendung über Systemattribute“ auf Seite 129](#)).

7.3 openUTM-Seite (Host) konfigurieren

Auf dem Host-Rechner, auf dem die openUTM-Anwendung läuft, müssen Client-Rechner und Client-Anwendungen bekannt sein. Dies geschieht durch KDCDEF-Steueranweisungen bei der openUTM-Generierung.

Läuft die openUTM-Anwendung unter Windows oder Solaris bzw. Linux, sind eventuell auch auf Host-Seite TNS-Einträge für die Kopplung mit WebTransactions notwendig. Diese können Sie sich beim KDCDEF-Lauf automatisch erzeugen lassen (siehe openUTM-Handbuch „Anwendungen generieren“).

Läuft die openUTM-Anwendung unter BS2000/OSD, so sind auf Host-Seite entsprechende BCMAP-Einträge nur in Ausnahmefällen notwendig (siehe [Abschnitt „BCMAP-Einträge \(BS2000/OSD\)“ auf Seite 138](#)).

7.3.1 openUTM-Generierung anpassen

Die lokalen Namen der WebTransactions-Anwendungen machen Sie bei der KDCDEF-Generierung durch folgende KDCDEF-Steueranweisungen bekannt.

BCAMAPPL-Anweisung

Die BCAMAPPL-Anweisung definiert den lokalen Namen der openUTM-Host-Anwendung.

```
BCAMAPPL utmappl, T-PROT=RFC1006
```

utmappl lokaler Name der openUTM-Host-Anwendung.
Auf Solaris, Linux und Windows muss es zu *utmappl* einen TNS-Eintrag geben, der *utmappl* auf einen T-Selektor abbildet. In BS2000/OSD muss *utmappl* - falls ohne BCMAP-Einträge gearbeitet wird - unmittelbar dem in der *upicfile* bzw. dem Attribut APPLICATION_NAME vereinbarten Partner-Namen entsprechen.



Falls Sie auf einer Solaris-, Linux- oder Windows-Host-Plattform ohne TNS arbeiten, dann muss zusätzlich der Operand LISTENER=PORT=*number* angegeben werden. *number* ist die Portnummer, unter der die openUTM-Anwendung erreicht werden kann. Bei *number* ≠ 102 muss diese Portnummer auch beim Client in der *upicfile* bzw. dem Attribut HOST_PORT eingetragen werden.

Einzelne Anschlusspunkte durch PTERM-/LTERM-Anweisungen

Einen Anschlusspunkt für ein WTHolder-Programm definieren Sie jeweils als Paar aus PTERM- und LTERM-Anweisung. Für jede parallele Verbindung müssen Sie also ein eigenes PTERM-/LTERM-Paar angeben.

- PTERM-Anweisung

```
PTERM upic_client, PTYPE=UPIC-R, LTERM=lterm_name, BCAMAPPL=utmappl,
      PRONAM=client_prozessor, ...
```

upic_client Name des Clients. Auf Solaris, Linux und Windows muss es zu *upic_client* einen TNS-Eintrag geben, der *upic_client* auf einen T-Selektor abbildet. In BS2000/OSD muss *upic_client* - falls ohne BCMAP-Einträge gearbeitet wird - unmittelbar einem in der Datei `localapps` vereinbarten lokalem Namen entsprechen.

lterm_name Name eines LTERM-Partners (siehe LTERM-Anweisung).

utmappl lokaler Name der openUTM-Host-Anwendung.

client_prozessor

Symbolischer Name des Client-Rechners. Der symbolische Name wird auf die Internet-Adresse abgebildet (z.B. über den Domain Name Service DNS). Der PRONAM-Operand ist für Solaris, Linux und Windows nur dann Pflicht, falls *client_prozessor* im TNS-Eintrag für den fernen Client angegeben ist (als Namensteil 4). Für BS2000/OSD ist der PRONAM-Operand in jedem Fall Pflicht.

Auf Solaris, Linux und Windows steht UPIC auch in der Variante UPIC-L zur Verfügung (für lokale Kopplung). Diese wählen Sie, wenn Sie in der PTERM-Anweisung PTYPE=UPIC-L setzen.

PTERM-Anweisung ohne TNS (Solaris, Linux, Windows)

Falls auf Solaris, Linux oder Windows ohne TNS gearbeitet wird, dann muss die PTERM-Anweisung folgende Form haben:

```
PTERM upic_client, PTYPE=UPIC-R, LTERM=lterm_name, BCAMAPPL=utmappl,
      LISTENER-PORT=number, TPROT=RFC1006, PRONAM=client_prozessor
```

LISTENER-PORT=*number* gibt die Portnummer an, die im Client-Rechner als Ausgangsport verwendet wird. Diese Portnummer muss auch in WebTransactions im Attribut LOCAL_PORT angegeben werden. Der Operand PRONAM ist in diesem Fall Pflicht.

- LTERM-Anweisung

```
LTERM lterm_name, [operanden]
```

lterm_name

Name eines LTERM-Partners (=logischer Anschlusspunkt der openUTM-Anwendung). *lterm_name* ist frei wählbar. Diesen symbolischen Namen braucht openUTM für seine interne Verwaltung.

[*operanden*]

Sie können in der LTERM-Anweisung jedoch auch durch weitere optionale Operanden Eigenschaften für diesen Anschlusspunkt festlegen, z.B. spezielle Zugriffsrechte setzen.

Beispiel (BS2000/OSD)

```
BCAMAPPL UTMAPPL1, T-PROT=RFC1006
```

```
PTERM CLIENT1, PTYPE=UPIC-R, LTERM=lterm_name1,  
BCAMAPPL=UTMAPPL1, PRONAM=HOST002, ...
```

```
LTERM lterm_name1 [operanden]
```

```
PTERM CLIENT2, PTYPE=UPIC-R, LTERM=lterm_name2,  
BCAMAPPL=UTMAPPL1, PRONAM=HOST002, ...
```

```
LTERM lterm_name2 [operanden]
```

```
PTERM upic_client, PTYPE=UPIC-R, LTERM=lterm_name, BCAMAPPL=utmapp1,  
PRONAM=client_prozessor, ...
```

```
LTERM lterm_name, [operanden]
```

Anschlusspunkt-Pool durch TPOOL-Anweisung

Statt mit mehreren PTERM- und LTERM-Anweisungen können Sie mit einer TPOOL-Anweisung eine begrenzte Menge von Anschlusspunkten definieren. In diesem Fall gibt es keine feste Namenszuordnung zwischen den Einträgen bei der KDCDEF-Generierung und TNS-Einträgen.

```
TPOOL BCAMAPPL=utmapp1, PTYPE=UPIC-R, LTERM=prefix, NUMBER=1000, PRONAM=*ANY
```

Der Operand `LTERM=prefix` definiert ein Präfix, aus dem openUTM intern LTERM-Namen für die einzelnen Anschlusspunkte des Pools bildet, um die parallel zugreifenden WTHolder-Programme zu unterscheiden. Der intern gebildete LTERM-Name ist maximal acht Zeichen lang und besteht aus dem angegebenen Präfix gefolgt von einer Laufnummer (z.B. UPIC0001, UPIC0002,...).

Beispiel

```
TPOOL BCAMAPPL=UTMAPPL1, LTERM=CLNT, NUMBER=1000, PTYPE=UPIC-R, PRONAM=*ANY
```

- Der Operand `BCAMAPPL` ist für Solaris, Linux und Windows optional. Für BS2000/OSD ist er Pflicht. Der angegebene Name muss in einer `BCAMAPPL`-Anweisung definiert sein.
- `LTERM` und `NUMBER` erzeugen zulässige `LTERM`-Namen der Form `CLNT0001`, `CLNT0002`, ...
- `PTYPE=UPIC-R`: der physische Partner ist ein `UPIC`-Client (auf Solaris und Linux ist hier auch die Angabe `UPIC-L` möglich)
- `PRONAM=*ANY`: jeder Rechner, der den Partnernamen (`UTMAPPL1`) kennt, kann mit dieser Host-Anwendung über `UPIC` kommunizieren.

7.3.2 Client-Rechner bekannt machen

Client- und Server-Rechner müssen einander bekannt sein. Dazu gibt es zwei Möglichkeiten:

- Beide Rechner sind für den DNS (Domain Name Service) konfiguriert. In diesem Fall sind keine weiteren Einträge notwendig.
- Ansonsten muss der Client-Rechner, auf dem WebTransactions läuft, auf dem Server-Rechner mit seiner Internet-Adresse eingetragen sein:
 - Für Solaris-, Linux- und Windows-Server in der `hosts`-Datei:

internet-adresse Client-Rechner

Beispiel: 123.4.5.6 WTRECHNER

- Für BS2000/OSD-Server muss der symbolische Name des Client-Rechners statisch in der `BCAM-RDF` (resource-definition-file) oder dynamisch per `BCIN` eingetragen sein.

Beispiel für `BCIN`:

```
/BCIN HOST002, INI=ALL, ACTIVE=ALL, IPA=(123,4,5,6), PROT=(TCP,IP)
```

Die Internetadresse des Client-Rechners wird durch den Parameter `IPA` angegeben, mit Kommata als Trennzeichen.

7.4 BCPMAP-Einträge (BS2000/OSD)

Globale BCPMAP-Einträge sind nur in den Fällen notwendig, in denen kein automatisches Mapping möglich ist (z.B. weil die verwendeten Namen länger als acht Zeichen sind).

Eine Übersicht über die Beziehungen zwischen den ggf. notwendigen BCPMAP-Einträgen finden Sie auf der folgenden Seite.

- Einträge auf WebTransactions-Seite:

Nehmen Sie folgende lokale Einträge für die WebTransactions-Anwendung vor:

```
/BCMAP FU=DEFINE, SU=LOCAL, APPL=(OSI, LOCAL1), TSEL-I=(8, C'CLIENT1'),  
TSEL-N=CLIENT1
```

```
/BCMAP FU=DEFINE, SU=LOCAL, APPL=(OSI, LOCAL2), TSEL-I=(8, C'CLIENT2'),  
TSEL-N=CLIENT2
```

...

Nehmen Sie folgenden globalen Eintrag für die ferne openUTM-Anwendung vor:

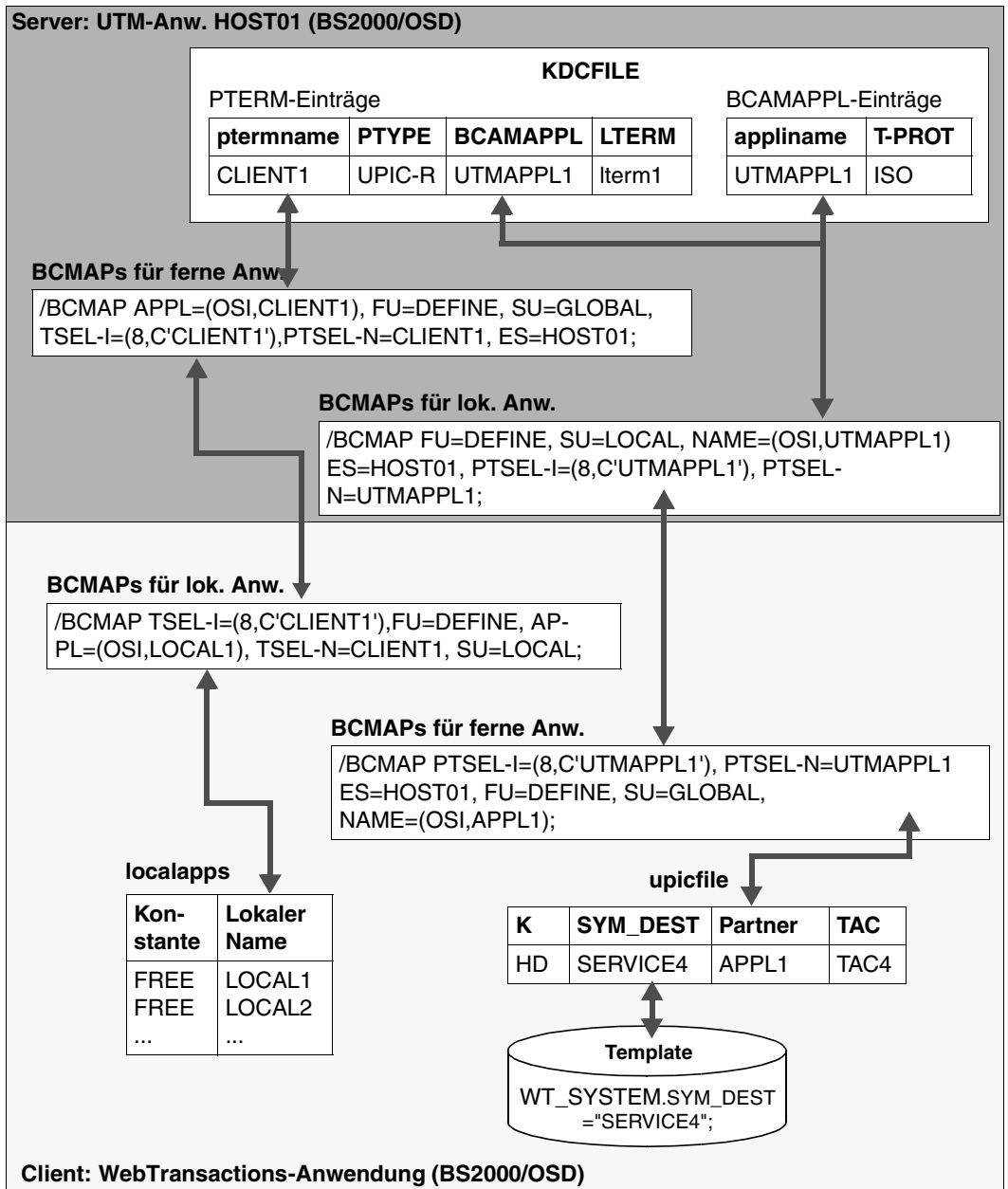
```
/BCMAP FU=DEFINE, SU=GLOBAL, NAME=(OSI, SERVER1), ES=HOST0001, PTSEL-I=(8,  
C'UTMAPPL1'), PTSEL-N=UTMAPPL1
```

Diese Abbildung (Mapping) ist nur notwendig, wenn openUTM-Anwendung und WebTransactions nicht auf demselben Rechner ablaufen.

Die genaue Ausprägung des BCPMAP-Kommandos kann je nach Transportsystem abweichen.

- Einträge auf Host-Seite:

Hier sind nur dann BCPMAP-Einträge notwendig, wenn das Standard-Mapping (d.h. Namen entsprechen sich, das Transportsystem gibt den Host-Anwendungsnamen transparent weiter usw.) die Host-Anwendung nicht findet.



8 Kommunikation steuern

8.1 openUTM-spezifische Attribute des Systemobjekts

Mit einigen Attributen des System-Objekts steuern Sie die Kommunikation zwischen WebTransactions und der openUTM-Anwendung.

Existiert unterhalb des verwendeten Kommunikationsobjekts ein Objekt `WT_SYSTEM` („verbindungspezifisches Systemobjekt“), so müssen diese Attribute dort definiert werden, anderenfalls sind sie als Attribute des globalen Systemobjekts `WT_SYSTEM` zu erklären. Grundlegende Informationen zu verbindungspezifischen und globalen Systemobjekten finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

Diese Attribute können Sie beim Start von WebTransactions im ersten Template (Start-Template) setzen und für die Sitzung beibehalten oder aktiv steuernd in der Sitzung verändern (siehe WebTransactions-Handbuch „Konzepte und Funktionen“ unter dem Stichwort „aktiver Dialog“).

Hier werden nur diejenigen Attribute beschrieben, die es speziell für openUTM gibt oder die zumindest für openUTM eine spezielle Bedeutung haben. Attribute des Systemobjektes, deren Bedeutung für alle Produktvarianten von WebTransactions gleich ist, sind im WebTransactions-Handbuch „Konzepte und Funktionen“ beschrieben.

8.1.1 Übersicht

Einen Überblick über die Attribute und ihre Wirkung gibt die folgende Tabelle.



Die Systemobjekt-Attribute können in folgende Kategorien eingeteilt werden:

- o (**open**)
Attribute, die bei open verwendet werden.
- t (**temporary**)
Attribute, die während der Kommunikation verwendet werden und die jederzeit in den Templates verändert werden können.
- r (**read only**)
Attribute, die während der Kommunikation verwendet werden und die nicht in den Templates verändert werden dürfen.
- c (**communication module**)
Attribute, die automatisch vom Host-Adapter gesetzt werden.


Die Kategorie ist jeweils in der rechten Spalte der folgenden Tabelle angegeben.

Attribut	Bedeutung	Erläuterung/Kategorie	
APPLICATION_NAME	Name der openUTM-Anwendung	Dieses Attribut wird dazu benutzt, die Kommunikation mit der openUTM-Anwendung herzustellen. Wenn dieses Attribut gesetzt ist, dann wird die upicfile ignoriert.	o
APPLICATION_PREFIX	Präfix für den Host-Anwendungsnamen	Dieses Präfix ermöglicht es, FLD- und Template-Dateien zu identifizieren, die den gleichen „Formatnamen“ besitzen, jedoch zu unterschiedlichen Host-Anwendungen gehören. Diese FLD- und Template-Dateien müssen in folgender Form abgespeichert sein: <i>application_prefix@formatname . fld</i> bzw. <i>application_prefix@formatname . htm</i> Der Host-Adapter wertet das Attribut beim Lesen der FLD-Datei aus. Die Auswertung für das Format erfolgt Template-gesteuert.	o
BADTAC	Angabe eines BADTAC-Transactions-codes	Beim Versuch, einen openUTM-Vorgang über einen ungültigen Transaktionscode zu starten, wird automatisch ein Vorgang mit dem im Attribut BADTAC angegebenen Transaktionscode gestartet (siehe Abschnitt „BADTAC - Simulation des Event-Service BADTAC“ auf Seite 197).	t

Attribut	Bedeutung	Erläuterung/Kategorie	
COMMUNICATION_FILE_NAME	Dateiname für Sende- oder Empfangsdaten	Mit diesem Attribut kann eine Datei angegeben werden, in die die beim <i>receive</i> empfangenen Daten geschrieben bzw. aus der die zu sendenden Daten gelesen werden. Nach einem Verarbeitungsschritt wird das Attribut wieder zurückgesetzt. Die Datei muss im Basisverzeichnis liegen, eine Ablage im Unterverzeichnis <i>wwwdocs</i> ist erlaubt. Voreinstellung: leer, d.h. es wird keine Datei verwendet.	t
COMMUNICATION_INTERFACE_VERSION	Schnittstellenversion, dient der Kompatibilität	Globales WT_SYSTEM-Attribut: <ul style="list-style-type: none"> – Enthält diese Variable einen Wert < "3.0", dann wird beim Empfangen einer Nachricht vom Host (<i>receive</i>) das globale System-Attribut FORMAT mit dem Namen des Host-Formats versorgt. – Enthält diese Variable einen Wert "3.0" oder höher, dann wird FORMAT <u>nicht</u> versorgt, da die Auswahl der nächstens Seite (Format) von den Templates selber getroffen wird (in der Regel durch Auswerten des Attributs <i>FLD</i>). Voreinstellung: 7.5	t
CONVERSATION_TAC	Steuerung der Vorgangsverknüpfung	Mit diesem Attribut wird eine Vorgangsverknüpfung über das openUTM-Steuerfeld oder die ersten 8 Bytes des Formats versucht (siehe Abschnitt „Automatische Vorgangsverknüpfung“ auf Seite 198).	o
CUT_TAC_FIELD	Entfernen des TAC-Feldes aus der ersten Nachricht	Bei *Formaten und +Formaten, die einem Teilprogramm bei Vorgangsbeginn geschickt werden, entfernt openUTM im Terminalbetrieb den Transaktionscode: bei *Formaten die ersten 8 Zeichen (TAC), bei +Formaten die ersten 10 Zeichen (Attributfeld plus TAC). Dieses Verhalten wird von WebTransactions standardmäßig simuliert. Soll der Transaktionscode in der ersten Nachricht eines Vorgangs nicht entfernt werden (z.B. weil die openUTM-Anwendung einen Input Exit einsetzt und der Transaktionscode gar nicht am Anfang der Nachricht steht), so ist dieses Attribut auf NO zu setzen. In allen anderen Fällen wird der Transaktionscode entfernt.	t
DISPLAY_EURO	Euro-Symbol anzeigen	Wenn dieses Attribut auf Yes gesetzt ist, dann wird das Zeichen, das dem Code X'A4' der ISO-8859-Codetabelle entspricht, als Euro-Zeichen ausgegeben. Bei DISPLAY_EURO=No (Voreinstellung) wird bei X'A4' das Währungssymbol (¤) angezeigt.	t

Attribut	Bedeutung	Erläuterung/Kategorie	
EPILOG	Nachspann	<p>Das Attribut enthält den Namen eines Templates (ohne die Endung '.htm'). Ist das Attribut definiert, so wird das entsprechende Template am Ende der generierten Templates inkludiert.</p> <p>Voreinstellung: keine Inkludierung</p> <p> Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter.</p> <p>Siehe auch PROLOG und FORMTPL</p>	t
FLD	Name der aktuellen Felddatei	<p>Dieses Attribut wird durch den Aufruf <code>receive</code> versorgt und sollte nicht geändert werden. Es verweist auf die Felddatei, welche die Struktur der gerade eingelesenen Nachricht von der Host-Anwendung enthält. Der gleiche Wert wird vom folgenden Aufruf <code>send</code> erwartet. Wenn die openUTM-Anwendung keinen Formatnamen liefert, wird FLD auf den Standard-Namen aus dem Zeilenmodus-Template <code>wtlnmode.htm</code> gesetzt.</p> <p>Siehe auch FORMAT und APPLICATION_PREFIX</p>	r
FORMAT_SEQ	Nummer eines Teilformats	<p>Dieses Attribut wird durch den Aufruf <code>receive</code> versorgt. Wurde ein Teilformat empfangen, so enthält dieses Attribut dessen Nummer: 1,2,3, Beim letzten Teilformat einer Nachricht ist der Wert dieses Attributs LAST. Wurde ein Vollformat empfangen, so wird dieses Attribut auf Leerstring gesetzt.</p>	r
FORMTPL	Formularfelder	<p>Das Attribut enthält den Namen eines Templates (ohne die Endung '.htm'). Ist das Attribut definiert, so wird das entsprechende Template am Anfang des <code>wtDataForm</code> in den generierten Templates inkludiert.</p> <p>Voreinstellung: keine Inkludierung</p> <p> Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter.</p> <p>Siehe auch PROLOG und EPILOG.</p>	t

Attribut	Bedeutung	Erläuterung/Kategorie	
HOST_CHAR_CODE	Verwendete Zeichen-codierung	Dieses Attribut gibt an, ob die Kommunikation mit der Host-Anwendung in ASCII ("A") oder EBCDIC ("E") abgewickelt werden soll. Sie können auch den Namen einer Datei angeben, die eine Umsetzungstabelle ("T[ABLE]:asciideiname") enthält. Die Tabelle wird unter dem Basisverzeichnis gesucht. Eine Vorlage wird unter <code>/install_dir/lib/ASCII.EBCDIC.G</code> ausgeliefert. Für FHS +Formate oder #Formate mit Attribut-Kombination muss die Konvertierung in UPIC ausgeschaltet werden (siehe <code>upicfile</code> und Attribut <code>UPIC_CODE_CONVERSION</code>) und <code>HOST_CHAR_CODE</code> verwendet werden. Als Voreinstellung ist in <code>HOST_CHAR_CODE</code> kein Wert gesetzt, es wird also nicht konvertiert.	t
HOST_IP_ADDRESS	IP-Adresse des Rechners, auf dem die openUTM-Anwendung läuft	Dieses Attribut wird dazu benutzt, die Kommunikation mit der openUTM-Anwendung herzustellen.	o
HOST_NAME	Name des Host-Rechners	Dieses Attribut wird dazu benutzt, die Kommunikation mit der openUTM-Anwendung herzustellen.	o
HOST_PORT	Portnummer der openUTM-Anwendung	Dieses Attribut wird dazu benutzt, die Kommunikation mit der openUTM-Anwendung herzustellen. Es muss nur angegeben werden, wenn die Partneranwendung nicht unter Port 102 erreichbar ist. Voreinstellung: 102	o
LOCAL_APPLICATION	Name, mit dem WebTransactions sich beim Transportsystem anmeldet	Soll für den Verbindungsaufbau ein fester Anwendungsname verwendet werden, an Stelle des ersten freien Eintrags aus der Datei <code>localapps</code> , ist dieser Name in <code>LOCAL_APPLICATION</code> abzulegen (siehe Abschnitt „Gezieltes Anmelden über bestimmte LTERMs“ auf Seite 201). Die Datei <code>localapps</code> wird dann nicht berücksichtigt. Voreinstellung: leer	o
LOCAL_PORT	Nummer des lokalen Ports	Ist dieses Attribut angegeben, so wird dieser Port von der Socketverbindung als Ausgangsport auf dem lokalen System verwendet.	o
PASSWORD	Passwort	WebTransactions verwendet für die Benutzeranmeldung bei openUTM das hier angegebene Passwort. Nur sinnvoll in Verbindung mit dem Attribut <code>USER</code> und <code>SECURITY_TYPE=PASSWORD</code> .	o

Attribut	Bedeutung	Erläuterung/Kategorie	
NEW_PASSWORD	Neues Passwort	<p>Mit diesem Attribut kann ein neues Passwort für die openUTM-Benutzererkennung vergeben werden. Nur sinnvoll in Verbindung mit dem Attribut USER und SECURITY_TYPE=PASSWORD.</p> <p>Das neue Passwort kann auch dann noch vergeben werden, wenn das alte Passwort abgelaufen ist, vorausgesetzt, die openUTM-Anwendung ist entsprechend generiert (Grace-Sign-On).</p> <p>Nach einem nachfolgenden receive-Aufruf sollte über das Attribut RECEIVE_ERROR geprüft werden, ob die Passwortänderung erfolgreich war, siehe auch Seite 152.</p>	o
PROLOG	Vorspann	<p>Das Attribut enthält den Namen eines Templates (ohne die Endung '.htm'). Ist das Attribut definiert, so wird das entsprechende Template am Anfang der generierten Templates inkludiert.</p> <p>Voreinstellung: keine Inkludierung</p> <p> Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter.</p> <p>Siehe auch EPILOG und FORMTPL.</p>	t
RECEIVE_ERROR	UPIC Returncode nach Receive	In diesem Attribut wird nach einem Aufruf receive der UPIC-Returncode übergeben. Dieses erlaubt eine bessere Fehlerbehandlung im Template. Mögliche Werte siehe Seite 152 .	t
RECEIVE_SECONDARY_INFORMATION	Sekundärer UPIC-Returncode nach Receive	Der sekundäre UPIC-Returncode gibt Aufschluss über die genaue Fehlerursache. Damit kann eine detaillierte Fehlerbehandlung durchgeführt werden.	t
RESTART	Vorgangswiederanlauf	Dieses Attribut spezifiziert einen Start mit oder ohne automatischen Wiederanlauf. Mögliche Werte: YES, NO. Voreingestellt ist der Wert NO. Der Wert YES ist nur sinnvoll bei SECURITY_TYPE≠NONE.	o
RETRY	Versuchswiederholung	Wird dieses Attribut beim Start gesetzt, so legt es fest, wie oft WebTransactions einen Versuch, die Methode open aufzurufen, wiederholen soll, falls dieser nicht erfolgreich war - z.B. wg. openUTM-Betriebsmittelengpass.	o

Attribut	Bedeutung	Erläuterung/Kategorie	
SECURITY_TYPE	openUTM-Sicherheitsstufe	Dieses Attribut wird beim Start gesetzt. Mögliche Werte sind: <ul style="list-style-type: none"> – NONE (voreingestellter Wert: weder Benutzerkennung noch Passwort werden verwendet) – USER (Anmeldung unter Benutzerkennung aber ohne Passwort) – PASSWORD (Anmeldung unter Benutzerkennung und mit Passwort) 	o
SPECIAL_KEY	Sondertasten	Über dieses Attribut können F- und K-Tasten an die openUTM-Anwendung übergeben werden. Mögliche Werte sind F1-F24 und K1-K14 (nur für openUTM auf BS2000/OSD)	t
SYM_DEST	Symbolic Destination	Dieses Attribut muss nur dann gesetzt werden, wenn die Parameter für die Verbindung zur openUTM-Anwendung aus der <code>upicfile</code> gelesen werden. In diesem Fall muss das Attribut vor dem Aufruf von <code>open</code> gesetzt sein. Bei aktiver Dialogsteuerung müssen Sie das Attribut vor einem erneuten Aufruf von <code>open</code> versorgen. <code>SYM_DEST</code> muss exakt 8 Zeichen lang sein. Voreinstellung: undefiniert, d.h. die Verbindungsparameter müssen einzeln über entsprechende Systemattribute gesetzt werden.	t
TAC	Transaktionscode	Über dieses Attribut wird der Service adressiert, der in der openUTM-Anwendung aufgerufen werden soll.	o
TIMEOUT	Maximale Wartezeit für <code>receive</code>	Dieses Attribut steuert, wie lange auf eine Antwort vom Host während eines <code>receive</code> maximal gewartet wird. Wenn <code>TIMEOUT</code> größer ist als das globale Systemobjekt-Attribut <code>TIMEOUT_APPLICATION</code> oder nicht gesetzt, wird ein von <code>TIMEOUT_APPLICATION</code> abgeleiteter Wert verwendet: <ul style="list-style-type: none"> – <code>TIMEOUT_APPLICATION > 10</code>: <code>TIMEOUT</code> entspricht <code>TIMEOUT_APPLICATION -5</code> – <code>TIMEOUT_APPLICATION > 1</code>: <code>TIMEOUT</code> entspricht <code>TIMEOUT_APPLICATION -1</code> – <code>TIMEOUT_APPLICATION =1</code> <code>TIMEOUT</code> entspricht <code>TIMEOUT_APPLICATION</code> Der Wert ist in ganzen Sekunden anzugeben.	t

Attribut	Bedeutung	Erläuterung/Kategorie	
UPIC_CODE_CONVERSION	Steuert die UPIC-Code-Konvertierung	Wenn dieses Attribut auf "Yes" gesetzt ist, dann wird eine Code-Konvertierung (ASCII - EBCDIC) erzwungen. In diesem Fall darf das Attribut HOST_CHAR_CODE nicht gesetzt sein. Bei "No" wird nicht konvertiert. Dies ist notwendig für FHS +Formate oder #Formate mit Attribut-Kombination. In diesem Fall muss HOST_CHAR_CODE gesetzt werden, siehe Seite 145 . Voreinstellung: "Yes"	o
UPIC_LIB	Bibliotheksname einschließlich Pfad der dynamisch einzubindenden UPIC-Bibliothek (nur relevant für Unix-Systeme bei UPIC-L)	Dieses Attribut gibt den vollständigen Dateinamen der dynamisch einzubindenden UPIC-Bibliothek an. Ausgewertet wird es bei jedem Verbindungsaufbau zur openUTM-Anwendung, wenn der Wert nicht Leerstring ist.	o
UPIC_TRACE	Schalter zum Aktivieren des UPIC-Trace (gilt nicht für Windows)	Dieses Attribut wird beim Verbindungsaufbau ausgewertet. Ist der Wert dieses Attributs nicht Leerstring, so wird der UPIC-Trace aktiviert und unter dem Verzeichnis BASEDIR/tmp/SESSION abgelegt. Auf dem BS2000 wird der Trace dort abgelegt, wo Sie auch das upicfile ablegen müssen, nämlich in der Kennung, in der der Web-Server gestartet wird (siehe auch Seite 71).	o
USER	Benutzername	WebTransactions verwendet für die Benutzeranmeldung bei openUTM die hier angegebene Benutzerkennung. Nur sinnvoll bei SECURITY_TYPE≠NONE.	o
UTM_PATH	Pfad, unter dem openUTM installiert ist; (nur relevant auf Unix-Plattform bei Verwendung von UPIC-L)	Dieses Attribut wird bei jedem Verbindungsaufbau zur openUTM-Anwendung ausgewertet, wenn in UPIC_LIB eine Bibliothek für UPIC-L angegeben ist.	o

8.1.2 Zusammenspiel: Systemobjekt-Attribute und Aktionen/Methoden

Dieser Abschnitt informiert Sie darüber, welche openUTM-spezifischen Attribute des kommunikationsspezifischen Systemobjekts bei welchen Methodenaufrufen eine Rolle spielen.

Verbindungsaufbau - open

Ein Aufruf der Methode `open` öffnet eine Verbindung zur openUTM-Anwendung. Folgende Attribute des Systemobjekts steuern den Aufruf von `open`:

Attribut	Bedeutung
APPLICATION_PREFIX	Präfix für den Host-Anwendungsnamen. Dieses Präfix ermöglicht es, FLD-Dateien zu identifizieren, die den gleichen Formatnamen besitzen, jedoch zu unterschiedlichen Host-Anwendungen gehören. Diese FLD-Dateien müssen in folgender Form abgespeichert sein: <i>application_prefix@formatname.fld</i>
APPLICATION_NAME	Name der openUTM-Anwendung. Wenn dieses Attribut gesetzt ist, dann wird die <i>upicfile</i> (Angabe bei <code>SYM_DEST</code>) ignoriert.
HOST_NAME	Name des Rechner, auf dem die openUTM-Anwendung läuft.
HOST_IP_ADDRESS	IP-Adresse des Rechners, auf dem die openUTM-Anwendung läuft. Wenn dieses Attribut gesetzt ist, dann wird der eventuell mittels <code>HOST_NAME</code> gesetzte Name ignoriert.
HOST_PORT	Portnummer der openUTM-Anwendung, muss nur angegeben werden, wenn die openUTM-Anwendung nicht unter Port 102 erreichbar ist. Voreinstellung: 102
LOCAL_PORT	Nummer des lokalen Ports
TAC	Transaktionscode des Service, der in der openUTM-Anwendung aufgerufen werden soll.
UPIC_CODE_CONVERSION	Wenn dieses Attribut auf "Yes" gesetzt ist, dann wird eine Code-Konvertierung (ASCII-EBCDIC) erzwungen. Bei "No" wird nicht konvertiert.
SYM_DEST	Angabe des Symbolic Destination Name aus der Datei <i>upicfile</i> . Dieses Attribut muss nur dann gesetzt werden, wenn der Service der openUTM-Anwendung nicht über die o.g. Attribute <code>APPLICATION_NAME</code> , <code>HOST_NAME</code> . . . , <code>TAC</code> adressiert wird. Die zur Adressierung notwendigen Parameter müssen dann in der <i>upicfile</i> hinterlegt sein. Der Symbolic Destination Name ist ein 8 Zeichen langer Name.
SECURITY_TYPE	Sicherheitsstufe: NONE, USER oder PASSWORD (siehe Seite 192)
USER	openUTM-Benutzerkennung (siehe Seite 192)
PASSWORD	openUTM-Benutzerpasswort (siehe Seite 192)
NEW_PASSWORD	Neues openUTM-Benutzerpasswort (siehe Seite 192)
RESTART	Wiederanlauf: Yes oder NO (siehe Seite 194)

Attribut	Bedeutung
ERROR	Fehlermeldung. Falls bei der Ausführung von <code>open</code> ein Fehler auftritt, wird im Systemobjekt-Attribut <code>ERROR</code> eine entsprechende Meldung hinterlegt. Wird der Aufruf erfolgreich ausgeführt, ist dieses Attribut leer.
RECEIVE_ERROR	Falls <code>SECURITY_TYPE</code> \neq <code>NONE</code> ist, das Attribut <code>USER</code> jedoch leer ist, wird in diesem Attribut bereits beim Verbindungsaufbau eine Fehlermeldung abgelegt (<code>CM_SECURITY_NOT_VALID</code>).
RECEIVE_SECONDARY_INFORMATION	Sekundärer UPIC-Returncode, der Aufschluss über die genaue Fehlerursache gibt.
LOCAL_APPLICATION	Anmeldung über festen Host-Anwendungsnamen.

Darüber hinaus werden beim Verbindungsaufbau alle Attribute ausgewertet, die in der Tabelle in [Abschnitt „Übersicht“ auf Seite 142](#) mit „o“ oder gekennzeichnet sind.

Daten senden - send

Der Aufruf der Methode `send` sendet eine Nachricht an die openUTM-Anwendung. Folgende System-Objekt-Attribute werden ausgewertet oder gesetzt:

Attribut	Inhalt
FLD	Name der Felddatei. Anhand des Attributs <code>FLD</code> ermittelt WebTransactions die entsprechende Felddatei zur Interpretation der an die openUTM-Anwendung gesendeten Daten. Der Wert des Attributs ist i.A. durch einen vorausgehenden Aufruf von <code>receive</code> richtig belegt. Die aktuellen Host-Datenobjekte werden nun als Nachricht an die openUTM-Anwendung geschickt
ERROR	Fehlermeldung. Falls bei der Ausführung von <code>send</code> ein Fehler auftritt, wird im Systemobjekt-Attribut <code>ERROR</code> eine entsprechende Meldung hinterlegt. Wird der Aufruf erfolgreich ausgeführt, ist dieses Attribut leer.
COMMUNICATION_FILE_NAME	Name der Datei, deren Daten gesendet werden sollen. Die Datei muss im Basisverzeichnis liegen, eine Ablage im Unterverzeichnis <code>wwdocs</code> ist erlaubt.

Daten empfangen - receive

Der Aufruf der Methode `receive` empfängt eine Nachricht von der openUTM-Anwendung und belegt die folgenden Systemobjekt-Attribute:

Attribut	Inhalt
FLD	Name des empfangenen Formats/Teilformats. Die dazugehörige Felddatei erhält den gleichen Namen mit dem Suffix <code>.fld</code> . WebTransactions verwendet diese Datei, um die Struktur der empfangenen Daten zu ermitteln. Nach Aufruf von <code>receive</code> kann auf die Daten in Form von Host-Datenobjekten zugegriffen werden.
FORMAT_SEQ	Laufende Nummer des empfangenen Teilformats (beginnend mit 1). Beim letzten Teilformat erhält <code>FORMAT_SEQ</code> den Wert <code>LAST</code> . Bei Teilformaten empfängt jeder Aufruf von <code>receive</code> jeweils ein Teilformat. Sie müssen also mehrere Aufrufe verwenden, um alle Teilformate zu empfangen (siehe hierzu Abschnitt „Besonderheiten bei FHS/FORMANT-Teilformaten“ auf Seite 100). Bei Vollformaten ist dieses Attribut nicht relevant und enthält einen Leerstring.
COMMUNICATION_FILE_NAME	Name der Datei, die anstelle der Host-Datenobjekte für <code>send</code> oder <code>receive</code> verwendet werden soll. Die Datei muss im Basisverzeichnis liegen, eine Ablage im Unterverzeichnis <code>wwdocs</code> ist erlaubt.
ERROR	Fehlermeldung. Falls bei der Ausführung von <code>receive</code> ein Fehler auftritt, wird im Systemobjekt-Attribut <code>ERROR</code> eine entsprechende Meldung hinterlegt. Wird der Aufruf erfolgreich ausgeführt, ist dieses Attribut leer.
RECEIVE_ERROR	UPIC-Returncode. Nach einem Aufruf von <code>receive</code> wird hier der UPIC-Returncode übergeben. Dieses erlaubt eine bessere Fehlerbehandlung im Template (mögliche Werte siehe folgenden Abschnitt).
RECEIVE_SECONDARY_INFORMATION	Sekundärer UPIC-Returncode, der Aufschluss über die genaue Fehlerursache gibt.

UPIC-Returncodes nach receive

Im Attribut `RECEIVE_ERROR` wird nach einem Aufruf von `receive` der UPIC-Returncode übergeben. Folgende Werte sind möglich:

`CM_OK`

Der Aufruf war erfolgreich

`CM_SECURITY_NOT_VALID`

mögliche Ursachen:

- ungültige openUTM-Benutzerkennung
- ungültiges Passwort
- Die openUTM-Anwendung ist ohne USER generiert
- Der User kann sich bei der openUTM-Anwendung wegen Betriebsmittelengpass nicht anmelden

Im Attribut `RECEIVE_SECONDARY_INFORMATION` wird die genaue Fehlerursache ausgegeben, siehe [Seite 153](#).

`CM_TPN_NOT_RECOGNIZED`

mögliche Ursachen:

- ungültiger Transaktionscode (TAC) in der `upicfile`, z.B.:
 - TAC ist nicht generiert
 - Keine Berechtigung, um diesen TAC aufzurufen
 - TAC ist nur als Folge-TAC erlaubt
 - TAC ist kein Dialog-TAC
- Vorgangswiederanlauf wurde abgewiesen, da keine mit `RESTART=YES` generierte openUTM-Benutzerkennung angegeben wurde.

`CM_TP_NOT_AVAILABLE_NO_RETRY`

Vorgangswiederanlauf ist nicht möglich, da openUTM-Anwendung neu generiert wurde.

`CM_TP_NOT_AVAILABLE_RETRY`

Vorgangstart wurde abgewiesen, da openUTM-Anwendung beendet wird.

`CM_DEALLOCATED_ABEND`

mögliche Ursachen:

- Abnormale Beendigung des openUTM-Vorgangs
- openUTM-Anwendungsende
- Verbindungsabbau durch openUTM-Administration
- Verbindungsabbau durch das Transportsystem
- Verbindungsabbau durch openUTM wegen Überschreitung der maximal zulässigen Anzahl von Benutzern (`MAX-Anweisung`, `CONN-USERS=`). Die Ursache kann auch darin liegen, dass ein Administrator-USER übergeben wurde, aber der per openUTM-Generierung in der Anweisung `LTERM ...USER= zugeordnete USER` kein Administrator-USER ist.

CM_DEALLOCATED_NORMAL

Im openUTM-Vorgang wurde ein PENDING-Aufruf ausgeführt.

CM_RESOURCE_FAILURE_RETRY

Ein vorübergehender Betriebsmittelengpass führte zur Beendigung der Conversation. Möglicherweise können im openUTM-Pagepool keine Daten mehr zwischengespeichert werden.

CM_RESOURCE_FAILURE_NO_RETRY

Es ist ein Fehler aufgetreten, der zu einer vorzeitigen Beendigung der Conversation führte (z.B. ein Protokollfehler oder vorzeitiger Verlust der Netzverbindung).

CM_MAP_ROUTINE_ERROR

Tritt nur auf bei openUTM-V3.4-Anwendungen, auf die WebTransactions mit PROTOCOL=UTM_V4 zugreift.

CM_PROTOCOL_ERR

Tritt nur auf bei openUTM-V4.0-Anwendungen, auf die WebTransactions mit PROTOCOL=UTM zugreift.

CM_UNKNOWN_ERR

Programmierfehler.

CM_PROGRAM_PARAMETER_CHECK

Programmierfehler.

Sekundäre UPIC-Returncodes in RECEIVE_SECONDARY_INFORMATION

Im Attribut RECEIVE_SECONDARY_INFORMATION wird nach einem Aufruf von receive mit RECEIVE_ERROR=CM_SECURITY_NOT_VALID der sekundäre UPIC-Returncode übergeben.

Folgende Werte sind möglich:

CM_SECURITY_USER_UNKNOWN

Die angegebene Benutzerkennung ist in der openUTM-Anwendung nicht generiert.

CM_SECURITY_STA_OFF

Die angegebene Benutzerkennung ist im Augenblick gesperrt.

CM_SECURITY_USER_IS_WORKING

Die angegebene Benutzerkennung wird im Augenblick von einem anderen Benutzer verwendet.

CM_SECURITY_OLD_PASSWORD_WRONG

Das angegebene alte Passwort ist falsch (bei Änderung eines Passwortes).

CM_SECURITY_NEW_PASSWORD_WRONG

Das angegebene neue Passwort passt nicht (z.B. Fehler beim Wiederholen des Passwortes).

CM_SECURITY_PASSWORD_EXPIRED_NO_RETRY

**Das Passwort ist abgelaufen und kann vom Benutzer nicht mehr geändert werden.
Bitte den Administrator der openUTM-Anwendung verständigen.**

CM_SECURITY_PASSWORD_EXPIRED_RETRY

**Das Passwort ist abgelaufen, kann aber noch vom Benutzer geändert werden.
Nach der Änderung des Passwortes kann sich der Benutzer wieder an die
openUTM-Anwendung anmelden.**

CM_SECURITY_COMPLEXITY_ERROR

**Das neue Passwort ist nicht komplex genug. Komplexeres Passwort eingeben, z.B.
mit Sonderzeichen, Ziffern und weniger Zeichenwiederholungen.**

CM_SECURITY_PASSWORD_TOO_SHORT

Das neue Passwort ist zu kurz.

CM_SECURITY_UPD_PASSWORD_WRONG

Das geänderte Passwort ist nicht komplex genug. Komplexeres Passwort angeben.

CM_SECURITY_TA_RECOVERY

Der Benutzer muss eine Wiederanlauf der Transaktion veranlassen.

CM_SECURITY_PROTOCOL_CHANGED

Das LTERM darf die offene Transaktion nicht fortsetzen

CM_SECURITY_SHUT_WARN

**Warnung des Administrators: Die openUTM-Anwendung wird demnächst beendet.
Bitte umgehend abmelden.**

CM_SECURITY_ENC_LEVEL_TOO_HIGH

**Fehler bei der Datenverschlüsselung: Auf der Verbindung ist der für die Fortsetzung
des offenen Vorgangs notwendige Verschlüsselungsmechanismus nicht verfügbar.**

CM_SECURITY_NO_CARD_READER

Es wurde kein Kartenleser gefunden.

CM_SECURITY_CARD_INFO_WRONG

Die Daten der Karte sind falsch.

CM_SECURITY_NO_RESOURCES

Betriebsmittellengpass

CM_SECURITY_TAC_KEY_MISSING

Das LTERM darf den offenen Vorgang nicht fortsetzen

8.2 Host-Objekte und Attribute

Bei der UPIC-basierten Kommunikation zwischen WebTransactions und openUTM-Anwendung gibt es zwei Arten von Host-Objekten:

- Host-Datenobjekte

Dies sind Objekte, die den Feldern eines Formats (Maske) entsprechen:

`WT_HOST.handle.feldname` (siehe [Abschnitt „Host-Objekte für die einzelnen Format-Felder \(Host-Datenobjekte\)“ auf Seite 155](#)).

Für die Binärübertragung gibt es ein eigenes Host-Datenobjekt, mit dem sich die Daten darstellen lassen.

- Host-Steuerobjekte

Dies sind Objekte, die globale Daten verwalten und reservierte Namen haben:

`WT_HOST.handle.WT_HOST_MESSAGE` (siehe [Abschnitt „Host-Steuerobjekt WT_HOST_MESSAGE“ auf Seite 162](#))

`WT_HOST.handle.WT_HOST_GLOBALS` (siehe [Abschnitt „Host-Steuerobjekt WT_HOST_GLOBALS“ auf Seite 166](#)).

`WT_HOST.handle.$FIRST` und `WT_HOST.handle.$NEXT` (siehe [Abschnitt „Host-Steuerobjekte \\$FIRST und \\$NEXT“ auf Seite 167](#))

Ebenfalls unter der Wurzel `WT_HOST.handle` legt WebTransactions ggf. ein verbindungsspezifisches `WT_SYSTEM` an, in dem die openUTM-spezifischen Attribute des System-Objekts erzeugt werden. Diese sind im [Abschnitt „openUTM-spezifische Attribute des Systemobjekts“ auf Seite 141](#) beschrieben.

8.2.1 Host-Objekte für die einzelnen Format-Felder (Host-Datenobjekte)

Den Host-Datenobjekten entsprechen die Felder eines Formats (Maske). Ihre Namen werden aus der Felddatei ermittelt (siehe [Seite 83](#)).

Bei #-Formaten entsprechen die Attributnamen den FHS Attributnamen. Bei +-Formaten bildet WebTransactions die Attribut-Bits auf die Attributnamen ab. Falls z.B. innerhalb eines +-Formats das Bit für „Blinking“ gesetzt ist, setzt WebTransactions für das entsprechende Host-Datenobjekt `VISIBILITY=S`.

Für diese Host-Datenobjekt-Attribute sind folgende Zugriffsarten möglich: lesen (r) oder lesen und schreiben (w).

Die Namen der Attribute von Host-Objekten sind im Gegensatz zu anderen Objekten/Attributen nicht „case sensitiv“, d.h. Groß/Kleinschreibung wird nicht unterschieden.

Attribut	Beschreibung	Zugriff
<i>Attribute für alle Host-Datenobjekte</i>		
Value	Inhalt des Felds als 8-Bit-Zeichen (siehe Seite 160)	w
HTMLValue	Inhalt des Feld als 7-Bit-Zeichen in HTML-Notation (siehe Seite 160)	r
RawValue	Inhalt des Feldes als Sequenz von 8-Bit-Zeichen ohne Konvertierung; lediglich binäre Nullen werden in Leerzeichen umgewandelt (siehe Seite 160).	w
HexStringValue	Inhalt des Feldes in Form von abdruckbaren Halbbytes (siehe Seite 160).	w
<i>Statische Attribute aus der IFG-Formatdefinition, gültig für alle Host-Objekte</i>		
Align	statisches Feldattribut <code>Align</code> (Verwendung in Klassen-Templates)	r
AutoInput	statisches Feldattribut <code>AutoInput</code> (Y / N) (Verwendung in Klassen-Templates)	r
Blink	statisches Feldattribut <code>Blink</code> (Verwendung in Klassen-Templates)	r
Case	statisches Feldattribut <code>Case</code> (Verwendung in Klassen-Templates)	r
DataType	statisches Feldattribut <code>DataType</code> (Verwendung in Klassen-Templates)	r
DefaultCursor	statisches Feldattribut <code>DefaultCursor</code>	r
FloatSign	statisches Feldattribut <code>FloatSign</code>	r
GroupDigit	statisches Feldattribut <code>GroupDigit</code>	r
IOType	statisches Feldattribut <code>IOType</code> (Verwendung in Klassen-Templates)	r
Length	statisches Feldattribut <code>Length</code> (Verwendung in Klassen-Templates)	r
Name	Name des Feldes, um z.B. nach \$NEXT dasselbe Feld nochmals ansprechen zu können.	r
NumDecimals	statisches Feldattribut <code>NumDecimals</code>	r
Signed	statisches Feldattribut <code>Signed</code>	r
StartColumn	statisches Feldattribut <code>Column</code> : Spalte, in der das Feld im Format beginnt.	r
StartLine	statisches Feldattribut <code>Line</code> : Zeile, in der das Feld im Format beginnt.	r
SuppressZero	statisches Feldattribut <code>SuppressZero</code>	r

Attribut	Beschreibung	Zugriff
Unicode	<p>statisches Feldattribut Unicode (Y / N)</p> <p>Y (YES): Für alle Felder, die mit Unicode == 'Y' markiert sind, werden die Host-Daten im UPIC-Puffer für die Weiterleitung zum/vom Browser in UTF-8 und zurück umgesetzt. Es wird keine ASCII-EBCDIC-Konvertierung durchgeführt.</p> <p>N (NO): Die Host-Daten werden nicht in UTF-8 umgesetzt. Die ASCII-EBCDIC-Konvertierung wird abhängig vom Systemobjekt-Attribut HOST_CHAR_CODE ausgeführt.</p>	r
<i>Verknüpfung von statischem und dynamischem Attribut (nur bei #- und +-Formaten)</i>		
Blinking	<p>Resultat des statischen Feldattributes (siehe FLD-Datei, Section <feldname>, "Blink" und ggf. des dynamischen Feldattributs (Visibility = S). Mögliche Werte: Y (Yes), N (No)</p>	r
Detectable	<p>Resultat des statischen Feldattributes (siehe FLD-Datei, Section <feldname>, „Detectable“ auf Seite 85) und ggf. des dynamischen Feldattributs (Protection=D): Y (Yes), N (No)</p>	r
Mandatory	<p>Resultat des statischen Feldattributes (siehe FLD-Datei, Section <feldname>, „Mandatory“ auf Seite 86) und ggf. des dynamischen Feldattributs (Input Control=M): Y (Yes), N (No)</p>	r
Protected	<p>Resultat des statischen Feldattributes (siehe FLD-Datei, Section <feldname>, „Protection“ auf Seite 86) und ggf. des dynamischen Feldattributs (Protection=U/D): Y (Yes), N (No)</p>	r
Underlined	<p>Resultat des statischen Feldattributes (siehe FLD-Datei, Section <feldname>, „Underline“ auf Seite 86) und ggf. des dynamischen Feldattributs (Underline=Y/N): Y (Yes), N (No)</p>	r
Visible	<p>Resultat des statischen Feldattributes (siehe FLD-Datei, Section <feldname>, „Visibility“ auf Seite 86) und ggf. des dynamischen Feldattributs (Visibility=Y/N): Y (Yes), N (No)</p>	r

Attribut	Beschreibung	Zugriff
<i>Dynamische Attribute für #-Formate, sofern sie über das IFG-Profil ausgewählt wurden</i>		
Color	dynamisches Feld-Attribut Color, mögliche Werte: 1=rot, 2=grün, 3=gelb, 4=blau, 5=magenta, 6=cyan, 7=weiß, N=keine Farbe	w
Cursor	dynamisches Feld-Attribut Cursor, mögliche Werte: Y (Yes), N (No), H (Hold)	w
EditState	dynamisches Feld-Attribut EditState, mögliche Werte: V (Valid), I (Invalid), M (Must error), ' ' (Leerzeichen, entspricht 'not checked')	w
InputControl	dynamisches Feld-Attribut InputControl, mögliche Werte: N (Normal), M (Must), P (Potmust), A (Autoret)	w
InputState	dynamisches Feld-Attribut InputState, mögliche Werte: M (Modified), C (Cleared), D (Detected), U (Undefined), ' ' (Leerzeichen, entspricht 'not touched')	w
InputStateAct	dynamisches Feld-Attribut InputStateAct, siehe InputState	w
Intensity	dynamisches Feld-Attribut Intensity, mögliche Werte: N (Normal), H (High), D (Dark)	w
Inverse	dynamisches Feld-Attribut Inverse, mögliche Werte: Y (Yes), N (No)	w
OutputControl	dynamisches Feld-Attribut OutputControl, mögliche Werte: I (Init), D (Data), U (Undefined)	w
Protection	dynamisches Feld-Attribut Protection, mögliche Werte: A (Askip), P (Protected), U (Unprotected), D (Detectable)	w
Underline	dynamisches Feld-Attribut Underline, mögliche Werte: Y (Yes), N (No)	w
Visibility	dynamisches Feld-Attribut Visibility, mögliche Werte: V (Visible), S (Signaling), I (Invisible)	w
<i>Dynamische Attribute, die für *- und +-Formate analog der #-Formate teilweise abgebildet werden</i>		
InputState	dynamisches Feld-Attribut InputState, mögliche Werte: M (Modified), D (Detected), ' ' (Leerzeichen, entspricht 'not touched')	w

Attribut	Beschreibung	Zugriff
<i>Dynamische Attribute, die für #-Formate mit der Feldattributgruppe 'Attributkombination' und für +-Formate analog der #-Formate teilweise abgebildet werden. Schreibender Zugriff wird ohne Meldung ignoriert.</i>		
Cursor	dynamisches Feld-Attribut Cursor, mögliche Werte: Y (Yes), N (No) (abgebildet von Cursor position in this field)	r
InputControl	dynamisches Feld-Attribut InputControl, mögliche Werte: N (Normal), A (Autoret) (abgebildet von autoret)	r
Intensity	dynamisches Feld-Attribut Intensity, mögliche Werte: N (Normal), H (High), D (Dark) (abgebildet von bright, normal, dark)	r
Inverse	dynamisches Feld-Attribut Inverse, mögliche Werte: Y (Yes), N (No) (abgebildet von inverse)	r
Protection	dynamisches Feld-Attribut Protection, mögliche Werte: A (Askip), P (Protected), U (Unprotected), D (Detectable) (abgebildet von protected, unprotected, detectable (markierbar))	r
Underline	dynamisches Feld-Attribut Underline, mögliche Werte: Y (Yes), N (No) (abgebildet von underline (italic))	r
Visibility	dynamisches Feld-Attribut Visibility, mögliche Werte: V (Visible), S (Signaling), I (Invisible) (abgebildet von blinking, dark)	r

Attribute Value, HTMLValue, RawValue und HexStringValue

Sie können auf den Inhalt eines Datenfelds auf vier Arten zugreifen, wobei in allen Fällen die Nilzeichen (Binär null, \0) in Leerzeichen umgewandelt werden:

- als Attribut "Value":

Der Inhalt des Datenfelds wird um angehängte Leerzeichen gekürzt zurückgegeben. Doppelte Anführungszeichen, einfache Anführungszeichen und kaufmännisches Und (&) werden durch die entsprechenden HTML-Hex-Darstellungen (&#nn;) ersetzt. Alle übrigen Zeichen (auch Umlaute) werden als 8-Bit-Zeichen ausgegeben - also in der Form, wie sie von der Host-Anwendung ausgegeben bzw. verstanden werden. Dieses Attribut ist deshalb als Value-Vorbelegung in HTML-Tags des Typs `Input` geeignet.

Das Attribut kann auch geschrieben werden, sofern das correspondierende Feld nicht schreibgeschützt ist.

Bei Host-Objekten, deren `IOType` `INPUT` oder `OUTPUT` ist, wird der entsprechende Wert aus dem Nachrichtenpuffer gewonnen, bei `IOType` `TEXT` oder `FIXTEXT` wird der Text übernommen, der mit dem IFG definiert wurde.

- als Attribut "HTMLValue":

Der Inhalt des Datenfelds wird ungekürzt zurückgegeben. Doppelte Anführungszeichen, einfache Anführungszeichen und kaufmännisches Und (&) werden durch die entsprechenden HTML-Hex-Darstellungen (&#nn;) ersetzt. Außerdem werden einige Sonderzeichen für die folgende Ausgabe in HTML umgesetzt: <, >, ä, ö, ü, Ä, Ö, Ü, ß. Dieses Attribut ist bei konstantem HTML-Fließtext geeignet. Es kann nur gelesen werden.

- als Attribut "RawValue":

Der Inhalt des Datenfelds wird (bis auf die Umwandlung von binär null in Leerzeichen) unverändert zurückgegeben.

- als Attribut "HexStringValue":

Der Inhalt des Datenfelds wird in Form von abdruckbaren Halbbytes zurückgegeben. Damit können binäre Daten oder Steuerzeichen lesbar gemacht werden.

Beispiel

hello<CR>world (<CR> = Windows-Zeilenvorschub) wird dargestellt durch
68656B6B6F0A776F726B64 (ISO 8859-Codierung)

Unicode-Unterstützung

Operationen mit Zeichenketten

Da der WebTransactions-Kern selbst Unicode nicht unterstützt, sollten Sie Operationen mit Zeichenketten nur vorsichtig einsetzen.

Zeichenkettenoperationen können fehlerhafte Ergebnisse produzieren, wenn in der Zeichenkette Zeichen in UTF-8 enthalten sind. Das Attribut `length` liefert nicht die Anzahl der Zeichen, sondern die Anzahl der Bytes. Bei Vergleichen und Manipulationen mit solchen Zeichenketten ist also die Repräsentation der UTF-8-Zeichen zu berücksichtigen.

In den Templates, die WebTransactions generiert, liegen solche Operationen nicht vor.

Diagnose

- In Traces und in der Online-Anzeige von Host-Objekten werden Unicode-Zeichen in einer Ersatzdarstellung angezeigt.
- Host-Objekte können in WebLab nur mit 8-Bit-Zeichen überschrieben werden.

Setzen eines Dateninhalts (bei #-Formaten)

Beim Setzen eines Dateninhalts werden die Attribute `Inputstate` und `InputStateAct` sowie das Global-Attribut `FieldsMod` auf `M` (modified) gesetzt. `EditState` wird mit `V` vorbelegt.

Attribut IOType - Klassen-Templates

Die Datenfelder können Sie entsprechend ihrer `IOType`-Klasse auswerten. Host-Datenobjekte an der UPIC-Schnittstelle gehören entweder zur Klasse `INPUT` (Ein-/Ausgabefelder), zur Klasse `OUTPUT` oder zu den Klassen `TEXT` bzw. `FIXTEXT` (Textfelder). Wenn Sie z.B. über den Auswertungsoperator ein Klassen-Template mit dem Namen `INPUT.c1t` ansprechen, werden alle `INPUT`-Felder vom diesem Klassen-Template ausgewertet, ein Klassen-Template `OUTPUT.c1t` wird für die Auswertung aller `OUTPUT`-Felder herangezogen. Nähere Informationen zum Thema Klassen-Templates finden Sie im WebTransactions-Handbuch „Template-Sprache“.

8.2.2 Host-Steuerobjekt WT_HOST_MESSAGE

Außer den Host-Datenobjekten für die einzelnen Felder steht bei der UPIC-Anbindung an openUTM immer als weiteres Host-Objekt das Host-Steuerobjekt WT_HOST_MESSAGE zur Verfügung, um globale Daten zu verwalten. Auf die Attribute des Objekts kann folgendermaßen zugegriffen werden: lesen (r) oder lesen und schreiben (w):

Die Namen der Attribute von Host-Objekten sind im Gegensatz zu anderen Objekten/Attributen nicht „case sensitiv“, d.h. Groß/Kleinschreibung wird nicht unterschieden.

Attribut	Inhalt	Formattyp	Zugriff
BackgroundColor	Hintergrundfarbe	#	r
Contents	Inhalt der gesamten Host-Nachricht	alle	w
Content_<versatz>_<länge>	Inhalt des UPIC-Puffers	alle	r
CursorControl	Globalattribut CursorControl	#	w
CursorField	Enthält den Namen des Feldes, in dem der Cursor steht (Details siehe nächste Seite)	alle	r
CursorPosition	Globalattribut CursorPosition	#	w
DateFormat	statisches Feldattribut DateFormat	alle	r
DecimalSeparator	statisches Feldattribut DecimalSeparator	alle	r
DigitSeparator	statisches Feldattribut DigitSeparator	alle	r
FieldsDetect	Globalattribut FieldsDetect	#	w
FieldsMod	Globalattribut FieldsMod	#	w
FieldsValid	Globalattribut FieldsValid	#	w
FormatLength	Länge der Nachricht	alle	r
FormatName	Name des Formats	alle	r
FormattingSystem	Formatierungssystem: FHS, FORMANT	alle	r
FormatType	Typ des Formats: #, +, *	alle	r
Hex_Content_<versatz>_<länge>	Inhalt des UPIC-Puffers	alle	r
InputKeyClass	Globalattribut InputKeyClass	#	w
InputKeyNumber	Globalattribut InputKeyNumber	#	w
Level_Selection	Globalattribut Level_Selection	#	r
P_Key_Set	Globalattribut P_Key_Set	#	r
TimeFormat	statisches Feldattribut TimeFormat	alle	w
UserexitRc	Globalattribut Userexit Rc	#	w
UndefinedValues	Globalattribut Undefined Values	#	w

Attribut	Inhalt	Formattyp	Zugriff
Unicode	zeigt an, ob in der aktiven Nachricht mindestens ein Feld mit <code>Unicode == 'Y'</code> enthalten ist. Details siehe Seite 165 .	alle	r
Version	Version des Konverters	alle	r

`WT_HOST_MESSAGE` bietet lesenden Zugriff auf einige Attribute, welche die Eigenschaft des Formats beschreiben. Bei #-Formaten können Sie über das Objekt `WT_HOST_MESSAGE` auf alle globalen Attribute des Formats lesend und schreibend zugreifen.

Attribut Contents

Mit dem Attribut `Contents` können Sie die gesamte Host-Nachricht ansprechen, um diese in einem Attribut des Template- oder System-Objekts abzulegen und später wieder zurück zu kopieren. Diese Funktionalität ist z.B. bei der Verarbeitung von Teilformaten sehr nützlich (siehe [Abschnitt „Besonderheiten bei FHS/FORMANT-Teilformaten“](#) auf [Seite 100](#)).

Attribut Content_<versatz>_<länge>

Stellt beginnend ab `<versatz>` den Pufferinhalt ohne jede Aufbereitung in der Länge `<länge>` aus dem UPIC-Puffer zur Verfügung. D.h. wenn der Puffer binäre Nullen enthält, wird der Rückgabe-String beim Auftreten der ersten binären Null beendet.

Attribut CursorField

Dieses Attribut von `WT_HOST_MESSAGE` liefert den Namen des Format-Feldes, in dem der Cursor steht:

- Bei #-Formaten wird das Feldattribut `Cursor` berücksichtigt, wenn in `WT_HOST_MESSAGE.CursorControl` der Wert `F` (Field Cursor) oder `R` (Relative Cursor) steht. In diesem Fall enthält `WT_HOST_MESSAGE.CursorField` den Namen des ersten ungeschützten Feldes, dessen Feldattribut `Cursor` den Wert `Y` oder `H` hat. Der Wert `Y` des Feldattributs `Cursor` wird bei diesem Feld vor dem Abschicken des Formats implizit auf Leerzeichen gesetzt.

Hat `WT_HOST_MESSAGE.CursorControl` einen anderen Wert als `{F/R}` oder ist keines der Feldattribute `Cursor` in einem ungeschützten Feld gesetzt, so ist das Verhalten wie folgt: falls ein Feld mit dem Attribut `DefaultCursor` versehen ist, enthält `WT_HOST_MESSAGE.CursorField` den Namen dieses Feldes, ansonsten den Namen des ersten ungeschützten Eingabefeldes.

- Bei +-Formaten enthält `WT_HOST_MESSAGE.CursorField` den Namen des ersten ungeschützten Feldes, dessen Feldattribut `Cursor` gesetzt ist. Gibt es kein solches Feld und ist ein Feld mit dem Attribut `DefaultCursor` versehen, enthält `WT_HOST_MESSAGE.CursorField` den Namen dieses Feldes. Ansonsten enthält `WT_HOST_MESSAGE.CursorField` den Namen des ersten ungeschützten Eingabefeldes.
- Bei *-Formaten enthält `WT_HOST_MESSAGE.CursorField` den Namen des Feldes, das mit dem Attribut `DefaultCursor` versehen ist, ansonsten den Namen des ersten ungeschützten Eingabefeldes.
- Bei Teilformaten wird ein Leerstring zurückgeliefert, wenn sich das Feld, das den Cursor enthält, nicht im aktuellen Teilformat befindet..



Für das Setzen des Cursors stellt Ihnen WebTransactions das Template `wtBrowserFunctions.htm` zur Verfügung, das bei der Template-Generierung standardmäßig inkludiert wird.

Attribut `Hex_Content_<versatz>_<länge>`

Siehe auch Abschnitt „[Attribut `Content_<versatz>_<länge>`“ auf Seite 163.](#)

Der Inhalt des Puffers wird jedoch in einen String konvertiert, d.h. jedes Byte wird mit zwei Zeichen dargestellt.

Beispiel

Im UPIC-Puffer steht hexadezimal `000102030405...`

`Hex_Content_0_3` liefert als Zeichenkette `"000102"` zurück.

Attribut `Level_Selection`

Das Attribut `Level_Selection` hat normalerweise keine Bedeutung für openUTM-Dialoge mit WebTransactions. In Verbindung mit dem Attribut `P_Key_Set` kann der Wert „P“ als „`P_Key_Set` ist gültig“ bewertet werden, siehe FHS-Handbuch „Formatierungssystem für openUTM, TIAM, DCAM“.

Attribut `P_Key_Set`

Zeigt an, welche Definition von programmierbaren Tasten im Terminalbetrieb über FHS geladen werden soll.

- Wenn die openUTM-Anwendung über UPIC mit WebTransactions betrieben wird, hat dieses Attribut keine direkte Auswirkung, denn eine Terminal-Emulation ist nicht beteiligt.

- Wenn Sie programmierbare Tasten mit client-seitigem Java-Script abbilden, verwenden Sie das Attribut mit WebTransactions.

Siehe auch FHS-Handbuch „Formatierungssystem für openUTM, TIAM, DCAM“.

Attribut Unicode

Das Attribut `Unicode` am Host-Steuerobjekt `WT_HOST_MESSAGE` zeigt an, ob in der aktiven Nachricht mindestens ein Feld mit `Unicode == 'Y'` enthalten ist.

Abhängig von diesem Attribut kann das globale Systemobjekt-Attribut `CHARSET` auf den Wert `UTF-8` gesetzt werden. Daher wird dann das Feld `Content-Type` im HTTP-Header richtig belegt und der Browser kann die Daten korrekt interpretieren.

Alle Templates, die WebTransactions für diesen Host-Adapter generiert, enthalten diese Zuweisung automatisch. Bei bereits bestehenden Templates, die nicht neu generiert werden sollen, müssen Sie die Zuweisung gegebenenfalls manuell einfügen.

Weitere Information dazu entnehmen Sie dem Abschnitt „[Unicode-Unterstützung](#)“ auf [Seite 122](#).

8.2.3 Host-Steuerobjekt WT_HOST_GLOBALS

Ein Host-Objekt zur Steuerung der Kommunikation mit dem Host ist WT_HOST_GLOBALS. Dieses Objekt ist nur relevant für +Formate und *Formate. Seine Attribute sind alle überschreibbar. Sie sollten diese Attribute einmal im Start-Template versorgen, und zwar nach Aufruf der Methode `open`.

Die möglichen Werte sowie die Vorbelegung dieser Attribute finden Sie im FHS-Handbuch „Formatierungssystem für openUTM, TIAM, DCAM“, Kapitel „FHS-Einsatz für openUTM-Anwender“, Abschnitt „Startparameter“. Es gibt jedoch nicht für alle der dort beschriebenen openUTM-Startparameter ein entsprechendes Attribut in WT_HOST_GLOBALS.

Attribut	Beschreibung
Padding Asterisk, Padding PlusAttr, Padding PlusData	<p>geben an, mit welchem Zeichen ein Feld gefüllt werden soll, bevor Änderungen eingetragen werden. Vorbelegung ist OUTMSG (der Feldinhalt wie er von der Host-Anwendung gesendet wird). Ein abweichender Wert muss als Dezimalwert oder als Hexadezimalwert ('#xx') eingegeben werden. Dieser Wert entspricht dem Startparameter PADDING. Für Unix- und Windows-Plattformen ist zu beachten, dass der Wert ggf. beim Senden an die openUTM-Anwendung konvertiert wird (siehe Systemobjekt-Attribute UPIC_CODE_CONVERSION und HOST_CHAR_CODE oder Konvertierungskennzeichen in der upicfile).</p> <p>PaddingAsterisk betrifft die Datenfelder bei *Formaten PaddingPlusData betrifft die Datenfelder bei +Formaten PaddingPlusAttr betrifft die Attribut-Felder bei +Formaten Diese drei Attribute lösen das Attribut Padding ab. Padding wird weiterhin kompatibel unterstützt. Wenn Padding gesetzt wird, werden alle drei neuen Attribute mit dem übergebenen Wert versorgt. Die Bedeutung ist grundsätzlich gleich.</p> <p>Bei den FHS-Startparametern wird nur zwischen den Formaten + und * unterschieden: <i>Beispiel</i> .FHS PADDING=(FORM*=' ',FORM=X'00')</p> <p>Die weitergehende Unterscheidung bei WebTransactions kann genutzt werden, um z.B. solche Situationen zu konfigurieren, bei denen bei +Formaten das Attributfeld von FHS mit binär Null gefüllt wurde, während das Datenfeld von einem FHS-Format-Exit mit SPACE gefüllt wurde.</p>
Cursor	<p>gibt an, mit welcher Methode der Cursor positioniert wird. Mögliche Werte: A: Cursor wird durch Attributfelder gesetzt (ATTR), Voreinstellung N: Cursor wird durch KDCSCUR gesetzt (NATTR)</p>
Detect	<p>gibt an, mit welchem Zeichen ein Feld gefüllt werden soll, wenn es markiert ist. Voreinstellung ist 255. Der Wert kann von dem Startparameter MAPDET abgeleitet werden. Für Unix- und Windows-Plattform ist zu beachten, dass der Wert ggf. beim Senden an die openUTM-Anwendung konvertiert wird (siehe System-Objekt-Attribute UPIC_CODE_CONVERSION und HOST_CHAR_CODE oder Konvertierungskennzeichen in der upicfile).</p>

Attribut	Beschreibung
UnDetect	gibt an, mit welchem Zeichen ein Feld gefüllt werden soll, wenn es nicht markiert ist. Voreinstellung ist 0. Der Wert kann von dem Startparameter MAPDET abgeleitet werden. Für Unix- und Windows-Plattform ist zu beachten, dass der Wert ggf. beim Senden an die openUTM-Anwendung konvertiert wird (siehe System-Objekt-Attribute UPIC_CODE_CONVERSION und HOST_CHAR_CODE oder Konvertierungskennzeichen in der upicfile).
Read	gibt an, in welchem Modus die Host-Anwendung die Daten beim Einlesen erwartet. Mögliche Werte: M (Modified) oder U (Unprotected). Der Wert kann von dem Startparameter ISTD abgeleitet werden.
FieldLength	gibt an, wie das Attributfeld/Längensfeld der Felder vor dem Senden einer Nachricht an den Host versorgt werden soll. Mögliche Werte: E (Effektive Länge), D (Definierte Länge), N (Nicht modifiziert). Der Wert kann von dem Startparameter EFFLEN abgeleitet werden.
Update	<ul style="list-style-type: none"> – Hat das Attribut den Wert 0 (für ONLY), so wird für alle Felder, deren Inhalt binär Null ist, wenn möglich der Wert aus der vorangegangenen Nachricht übernommen. 0 ist die Voreinstellung. – Hat es den Wert P (für PSTN), so wird ein solches Feld mit Ausgabefüllzeichen gefüllt. Einschränkung: Der Modus 0 funktioniert nicht bei Teilformaten.

8.2.4 Host-Steuerobjekte \$FIRST und \$NEXT

Diese beiden Objekte dienen zum Bearbeiten von Host-Objekten.

Objektname	Attribut	Bedeutung des Attributs
\$FIRST	Name	Name des ersten Felds des aktuellen Formats in vollständiger Schreibweise. Existiert überhaupt kein Objekt, wird der Name \$END zurückgegeben.
		Zusätzlich alle Attribute der dynamischen Host-Datenobjekte, siehe Abschnitt „Host-Objekte für die einzelnen Format-Felder (Host-Datenobjekte)“ auf Seite 155.
\$NEXT	Name	Name des nächsten Felds des aktuellen Formats ausgehend von dem Feld, auf das zuletzt zugegriffen wurde und in vollständiger Schreibweise. Dieses Objekt können Sie verwenden, um Schritt für Schritt alle Felder des Formats durchzugehen. Existiert kein weiteres Objekt mehr, wird der Name \$END zurückgegeben.
		Zusätzlich alle Attribute der dynamischen Host-Datenobjekte, siehe Abschnitt „Host-Objekte für die einzelnen Format-Felder (Host-Datenobjekte)“ auf Seite 155.

8.3 Unterstützung von Terminal-Funktionen durch den Browser

Bei WebTransactions for openUTM können Sie die Formate der Host-Anwendung ohne Nachbearbeitung am Browser darstellen (1:1-Umsetzung). Um die Terminal-Funktionen nutzen zu können, müssen Sie das Master-Template `UTM.wmt` verwenden. Die Templates, die Sie über das Master-Template generiert haben, inkludieren die Templates `wtBrowserFunctions.htm` und `wtKeysUTM.htm`, welche die gewünschten Funktionen zur Verfügung stellen.

`wtBrowserFunctions.htm` inkludiert seinerseits die folgenden JavaScript-Dateien:

`wtCommonBrowserFunctions.js`

enthält JavaScript-Code, der für alle Browser durchlaufen wird.

`wt<browser>BrowserFunctions.js`

enthält JavaScript-Code für den jeweiligen Browser .

`wtKeysUTM.htm` enthält die Knöpfe für die openUTM-spezifischen Standard-Tasten und inkludiert die JavaScript-Dateien `wtKeysUTMFHS.js` und `wtKeysUTMFormant.js`. Diese beiden JavaScript-Dateien enthalten die FHS- bzw. Formant-spezifische Abbildung der Sondertasten für WebTransactions for openUTM. In diesen Dateien können Sie die Abbildung der Tasten entsprechend Ihren Wünschen anpassen oder erweitern (siehe [Abschnitt „Zuordnung der Tasten in wtKeysUTMFHS.js und wtKeysUTMFormant.js“ auf Seite 171](#)).

8.3.1 Unterstützte Terminal-Funktionen

Folgende Terminal-Funktionen stehen zur Verfügung:

- Pixel-genaue Ausrichtung von Text und Eingabefeldern mit Hilfe von Style-Sheets.
- Unterstützung für Terminal-Sondertasten, die an WebTransactions gesendet werden. Für diese Tasten gibt es zum Teil eine Entsprechung auf der PC-Tastatur (z.B. F-Tasten), zum Teil werden Tasten-Kombinationen benutzt, um die Terminal-Funktion auszulösen.
- Unterstützung von Terminal-Sondertasten, die direkt im Browser-Formular wirken wie z.B. Cursorpositionierung. Für diese Tasten gibt es zum Teil eine Entsprechung auf der PC-Tastatur, zum Teil werden Tasten-Kombinationen benutzt, um die Terminal-Funktion auszulösen.
- Autotab:
Beim Erreichen der maximalen Länge eines Eingabefeldes wird die Schreibmarke automatisch in das nächste Eingabefeld gesetzt.

- **Überschreiben von Feldern:**
Der Browser überschreibt analog einem Terminal die vorhandenen Zeichen in den Eingabefeldern und fügt nicht zwischen den vorhandenen Zeichen ein, wie es die Voreinstellung am Browser wäre.
- **Übermitteln der Cursor-Position vom Browser an die Host-Anwendung:**
Je nach Browserfunktionalität wird die exakte Position des Cursors oder nur das entsprechende Eingabefeld vom Browser an WebTransactions übermittelt.
- **Tabulator bleibt innerhalb des Formulars:**
Der Eingabefocus verlässt das von WebTransactions generierte Formular nicht. Ohne Eingriffe würde der Browser mit der Tabulator-Taste den Focus auch auf seine eigenen Bedienelemente setzen.

Welche der Terminal-Funktionen (F-Tasten, Cursorpositionierung,...) am Browser dargestellt werden können, hängt von der Art und der Version des eingesetzten Browsers ab. Die folgende Tabelle zeigt, welche Terminal-Funktionen mit welchem Browser unterstützt werden.

Terminal-Funktion	Unterstützung durch Browser		
	nicht speziell behandelte Browser	Netscape ab V6.0 oder Mozilla Firefox	Internet Explorer ab V4.0
Ausrichtung von Text und Eingabefeldern	nein	ja	ja
Unterstützung für Terminal-Sondertasten, die an WebTransactions gesendet werden	nur über eine Auswahl-Liste/Schaltfläche	durch individuelle konfigurierbare Abbildung über eine Taste oder Auswahl-Liste/Schaltfläche	
Unterstützung von Terminal-Sondertasten, die direkt im Browser-Formular wirken	nein	durch individuelle konfigurierbare Abbildung über eine Taste	
Autotab	nein	ja	ja
Überschreiben von Feldern	ja (wird im Browser simuliert durch automatische Auswahl des Feldinhalts)		ja
Übermitteln der Cursor-Position	Nur Position vom Beginn des zuletzt benutzten Eingabefeldes	Position vom Beginn des zuletzt benutzten Eingabefeldes und exakte Position in geschützten Feldern	exakte Position in geschützten Feldern und in Eingabefeldern (ab V5.0)

Terminal-Funktion	Unterstützung durch Browser		
	nicht speziell behandelte Browser	Netscape ab V6.0 oder Mozilla Firefox	Internet Explorer ab V4.0
Tabulator bleibt innerhalb des Formulars	nein	ja	

Tastenunterstützung durch den Internet Explorer ab V4.0 oder durch einen auf dem Gecko basierenden Browser

Wenn Sie den Internet Explorer ab V4.0 oder einen auf dem Gecko basierenden Browser (z.B. Netscape ab V6) verwenden, haben die Tasten des Browsers folgende Entsprechung:

Tasten ¹ bei Verwendung eines Browsers	entsprechende Taste am Terminal ²
ENTER	ENTER
F1 ... F12	F1 ... F12
Shift+F1 ... Shift+F12	F13 ... F24
STRG+F1 ... STRG+F12	K1 ... K12
STRG+Shift+F1	K13
STRG+Shift+F2	K14
STRG+Shift+F12	MAR
INS	INS

¹ '+' bedeutet hier, dass die angegebenen Tasten gleichzeitig gedrückt werden müssen. Die Taste STRG wird bei einigen Tastaturen mit CTRL bezeichnet.

² K-Tasten sind nur bei *openUTM*-Anwendungen auf BS2000/OSD möglich.

8.3.2 Zuordnung der Tasten in wtKeysUTMFHS.js und wtKeysUTMFormant.js

Alle Tastatureingaben werden von dem verwendeten Browser angenommen. Für die anwendungsspezifische Zuordnung von speziellen Funktionstasten stellt WebTransactions als Schnittstelle die Datei `wtKeysUTMFHS.js` (für FHS-Masken) und `wtKeysUTMFormant.js` (für Formant-Masken) zur Verfügung. Sie können diese Schnittstelle ohne tiefere Kenntnis der Browser-Templates nutzen.

Diese beiden Dateien liegen nach dem Anlegen des Basisverzeichnis im Verzeichnis `<basedir>/wwwdocs/javascript`. Die wesentliche Schnittstelle für die Anpassung der WebTransactions-Anwendung ist die Tabelle (Array) `wtKeyMappingTableInput` in der jeweiligen Datei.

In der Tabelle `wtKeyMappingTableInput` wird für jede Tasten-Abbildung ein Objekt mit mehreren Attributen angelegt. Mit diesen Attributen wird beschrieben:

- welcher Tastendruck (oder welche Tastenkombination)
- welche Aktion auslösen soll und
- ob diese Funktion auch über eine Auswahl-Liste zur Verfügung stehen soll.

Beispiel

```
wtKeyMappingTableInput = [
  { sl:'title of my select list'},
  { la:'Select', ac:doToggleMark, kc:VK_F12, mk:MK_CTRL+MK_SHIFT },
  { la:'F1', ac:'F1' },
  { la:'K1', 'ac':'K1', kc:VK_F1, mk:MK_CTRL }
];
```

Aus dieser Definition entsteht folgende Abbildung:

- CTRL+SHIFT+F12 ruft die Funktion `doToggleMark()` auf
- CTRL+F1 sendet den Funktions-Code K1 an WebTransactions

Aus der Definition entsteht eine Auswahl-Liste mit folgendem Inhalt:

title of my select list	—	ohne Funktion
Select	—	ruft die Funktion <code>doToggleMark()</code> auf
F1	—	sendet den Funktions-Code an WebTransactions
K1		(F1 bzw. K1)

In der Tabelle `wtKeyMappingTableInput` ist die Angabe folgender Attribute möglich:

Bezeichnung	Attribut	Bedeutung
la	label	Beschriftung z.B. für den Eintrag in der Auswahl-Liste. Existiert dieses Attribut nicht, wird für die Liste kein Eintrag erzeugt. Die entsprechende Taste aber trotzdem auf eine Funktionalität abgebildet.
co	comment	Kommentar, dieses Attribut wird nicht ausgewertet. Dieses Attribut ist als Alternative zum Attribut <code>la</code> gedacht; durch Ändern des Attributes <code>la</code> nach <code>co</code> kann z.B. eine Taste aus der Auswahl-Liste entfernt werden.
ac	action	<p>Auszuführende Aktion bei Auslösen der zugeordneten Taste oder Auswahl aus einer Liste.</p> <p>Ist dieses Attribut vom Typ <code>string</code>, dann wird der Inhalt nach <code>wt_special_key.value</code> übertragen und an <code>WebTransactions</code> gesendet. Es wird also das Formular an <code>WebTransactions</code> übertragen und als Sonderfunktion der Wert von <code>ac</code> mitgegeben (z.B. "@1" für die PF1-Taste).</p> <p>Ist dieses Attribut vom Typ <code>function</code>, dann wird eine Client-seitige Funktion mit diesem Namen aufgerufen. Diese Funktion muss definiert sein.</p> <p>Die JavaScript-Dateien <code>wt<browser>BrowserFunctions.js</code> stellen folgende Funktionen zur Verfügung:</p> <ul style="list-style-type: none"> - <code>doCursorHome</code> - <code>doCursorUp</code> - <code>doCursorDown</code> - <code>doCursorLeft</code> - <code>doCursorRight</code> - <code>doTab</code> - <code>doBackTab</code> - <code>doToggleMark</code> - <code>doToggleInsert</code>. <p>Die Implementierung dieser Funktionen kann abhängig von den Fähigkeiten des Browsers auch leer sein, siehe Abschnitt „Callback-Funktionen des Key mapping“ auf Seite 177.</p> <p>Ist dieses Attribut nicht definiert, kann keine Aktion durchgeführt werden. Die Tastatur-Eingabe wird dem Browser zur Bearbeitung überlassen.</p>
kc	key code	<p>Nummer, die im Tastatortreiber der gedrückten Taste zugeordnet ist. Für viele Tasten existiert in <code>wtCommonBrowserFunctions.js</code> ein Symbol, die Namen beginnen mit <code>VK_</code>.</p> <p>Für Tastenkombinationen ist zusätzlich der <code>modifier key (mk)</code> von Bedeutung.</p>

Bezeichnung	Attribut	Bedeutung
mk	modifier-key	<p>zusätzlich gedrückte Umschalttaste (siehe Definition in <code>wtCommonBrowserFunctions.js</code>):</p> <ul style="list-style-type: none">- 0 = MK_NONE (= keine Umschalttaste gedrückt)- 1 = MK_CTRL- 2 = MK_ALT- 4 = MK_SHIFT <p>bei Kombinationen werden die entsprechenden Werte addiert:</p> <ul style="list-style-type: none">- 3 = MK_CTRL + MK_ALT- 5 = MK_CTRL + MK_SHIFT- usw. <p>Ist kein <code>mk</code> angegeben, wird der Wert 0 = MK_NONE verwendet!</p>
s1	select list	<p>Am Anfang jeder zu generierenden Auswahl-Liste wird ein Element mit dem Index 0 als Überschrift erzeugt. Dieses Element hat keine Funktion. Der Text für diese 0'te Element wird im Attribut <code>s1</code> angegeben. Eine ggf. bereits vorher angelegte Auswahl-Liste wird beim Auftreten von <code>s1</code> beendet.</p> <p>Für bessere Lesbarkeit kann <code>s1</code> einziges Attribut des Tabellen-Objektes sein.</p>

Aufbau von wtKeysUTMFHS.js und wtKeysUTMFormant.js

Im Folgenden ist die Tabelle `wtKeyMappingTableInput` aus der Dateien `wtKeysUTMFHS.js` und `wtKeysUTMFormant.js` dargestellt, wie sie von `WebTransactions` ausgeliefert wird:

Das Objekt `wtKeyMappingTableInput` wird als Literal angelegt.

```
wtKeyMappingTableInput = [
```

Das Attribut `s1` kennzeichnet den Beginn einer Auswahl-Liste mit der Beschriftung `more keys`.

```
{ s1:'more keys'},
```

Das Attribut `co` kennzeichnet einen Kommentar zur besseren Lesbarkeit. Für die folgenden Einträge ist kein Attribut `la` vorhanden. Die Einträge sollen nicht in der Auswahl-Liste erscheinen. Über `kc` und `mk` wird aber die Zuordnung zu einer Taste am PC getroffen. Mit `ac` werden hier JavaScript-Funktionen definiert, die beim Drücken der entsprechenden Taste/Tastenkombination ausgeführt werden sollen.

```
{ co:'MAR', ac:doToggleMark, kc:VK_F12, mk:MK_CTRL+MK_SHIFT },
{ co:'MAR', ac:doToggleMark, kc:VK_MAR, mk:0 },
{ co:'Insert', ac:doToggleInsert, kc:VK_INS },
{ co:'CursorUP', ac:doCursorUp, kc:VK_UP },
{ co:'CursorDOWN', ac:doCursorDown, kc:VK_DOWN },
{ co:'CursorLEFT', ac:doCursorLeft, kc:VK_LEFT },
{ co:'CursorRIGHT', ac:doCursorRight, kc:VK_RIGHT },
{ co:'HOME', ac:doCursorHome, kc:VK_HOME },
{ co:'TAB', ac:doTab, kc:VK_TAB },
{ co:'BACKTAB', ac:doBackTab, kc:VK_TAB, mk:MK_SHIFT },
```

Die Funktionstasten erscheinen in der Auswahlliste. Bei FHS sind dies die Tasten K1 bis K14 und F1 bis F24 (siehe Abbildung unten), bei Formant nur die Tasten F1 bis F24.

```
{ la:'K1', ac:'K1', kc:VK_F1, mk:MK_CTRL },
{ la:'K2', ac:'K2', kc:VK_F2, mk:MK_CTRL },
{ la:'K3', ac:'K3', kc:VK_F3, mk:MK_CTRL },
{ la:'K4', ac:'K4', kc:VK_F4, mk:MK_CTRL },
{ la:'K5', ac:'K5', kc:VK_F5, mk:MK_CTRL },
{ la:'K6', ac:'K6', kc:VK_F6, mk:MK_CTRL },
{ la:'K7', ac:'K7', kc:VK_F7, mk:MK_CTRL },
{ la:'K8', ac:'K8', kc:VK_F8, mk:MK_CTRL },
{ la:'K9', ac:'K9', kc:VK_F9, mk:MK_CTRL },
{ la:'K10', ac:'K10', kc:VK_F10, mk:MK_CTRL },
{ la:'K11', ac:'K11', kc:VK_F11, mk:MK_CTRL },
{ la:'K12', ac:'K12', kc:VK_F12, mk:MK_CTRL },
```

```
{ la:'K13', ac:'K13', kc:VK_F1, mk:MK_CTRL+MK_SHIFT },
{ la:'K14', ac:'K14', kc:VK_F2, mk:MK_CTRL+MK_SHIFT },

{ la:'F1', ac:'F1', kc:VK_F1, mk:0 },
{ la:'F2', ac:'F2', kc:VK_F2 },
{ la:'F3', ac:'F3', kc:VK_F3 },
{ la:'F4', ac:'F4', kc:VK_F4 },
{ la:'F5', ac:'F5', kc:VK_F5 },
{ la:'F6', ac:'F6', kc:VK_F6 },
{ la:'F7', ac:'F7', kc:VK_F7 },
{ la:'F8', ac:'F8', kc:VK_F8 },
{ la:'F9', ac:'F9', kc:VK_F9 },
{ la:'F10', ac:'F10', kc:VK_F10 },
{ la:'F11', ac:'F11', kc:VK_F11 },
{ la:'F12', ac:'F12', kc:VK_F12 },

{ la:'F13', ac:'F13', kc:VK_F1, mk:MK_SHIFT },
{ la:'F14', ac:'F14', kc:VK_F2, mk:MK_SHIFT },
{ la:'F15', ac:'F15', kc:VK_F3, mk:MK_SHIFT },
{ la:'F16', ac:'F16', kc:VK_F4, mk:MK_SHIFT },
{ la:'F17', ac:'F17', kc:VK_F5, mk:MK_SHIFT },
{ la:'F18', ac:'F18', kc:VK_F6, mk:MK_SHIFT },
{ la:'F19', ac:'F19', kc:VK_F7, mk:MK_SHIFT },
{ la:'F20', ac:'F20', kc:VK_F8, mk:MK_SHIFT },
{ la:'F21', ac:'F21', kc:VK_F9, mk:MK_SHIFT },
{ la:'F22', ac:'F22', kc:VK_F10, mk:MK_SHIFT },
{ la:'F23', ac:'F23', kc:VK_F11, mk:MK_SHIFT },
{ la:'F24', ac:'F24', kc:VK_F12, mk:MK_SHIFT },
```

Der letzte Eintrag der Tabelle darf nicht mehr mit Komma abgeschlossen werden, da sonst vor dem Literalende (]) ein weiterer Eintrag erwartet wird.

```
{ la:'InsClip', ac:doInsertClipboard, kc:VK_V, mk:MK_CTRL+MK_SHIFT }
];
// END_OF_KEY_TABLE
```

8.3.3 Zusammenspiel von `wtCommonBrowserFunctions.js` und `wt<browser>BrowserFunctions.js`

Die Datei `wtCommonBrowserFunctions.js` enthält JavaScript-Code, der für alle Browser durchlaufen wird. Die Dateien `wt<browser>BrowserFunctions.js` enthalten JavaScript-Code, der abhängig vom jeweiligen Browser durchlaufen wird. Z.B. enthält `wtGeckoBrowserFunctions.js` JavaScript-Code für Browser, die auf dem Gecko basieren.

Die Dateien liegen nach dem Anlegen des Basisverzeichnisses im Verzeichnis `<basedir>/wwwdocs/javascript`.

Wenn Sie den JavaScript-Code in diesen Dateien anpassen wollen, sind tiefere Kenntnis des Browser-Verhaltens und des Zusammenspiels mit WebTransactions erforderlich. Im folgenden Text ist das Zusammenspiel der Funktionen und Datenstrukturen beschrieben, wie Sie es im ausgelieferten Zustand vorfinden.

Symbole

Die Datei `wtCommonBrowserFunctions.js` wird vor den Dateien `wt<browser>BrowserFunctions.js` und `wtKeysUTMFHS.js` bzw. `wtKeysUTMFormant.js` aufgerufen. Sie enthält die Definition von Variablen, damit die Tasten in den anderen Dateien `*.js` symbolisch angesprochen werden können.

```
// some symbolic keycodes //////////
VK_TAB    = 9;
VK_RETURN = 13;
VK_SHIFT  = 16;
VK_CTRL   = 17;
VK_ALT    = 18;
VK_PAUSE  = 19;
VK_ESC    = 27;
VK_PGUP   = 33;
VK_PGDN   = 34;
VK_END    = 35;
VK_HOME   = 36;
VK_LEFT   = 37;
VK_UP     = 38;
VK_RIGHT  = 39;
VK_DOWN   = 40;
VK_INS    = 45;
VK_O      = 48;
VK_1      = 49;
...
MK_NONE   = 0;
MK_CTRL   = 1;
MK_ALT    = 2;
MK_SHIFT  = 4;
```


Key mapping Funktionen

```
function wtCreateKeyMap()
```

erzeugt aus der Tabelle `wtKeyMappingTableInput` eine zur Laufzeit einfacher und schneller zugreifbare Struktur. Der Aufruf erfolgt aus `wtKeysUTM.htm`. Der Aufruf ist unbedingt erforderlich, da sonst kein Mapping möglich ist.

```
function wtCreateKeySelectList()
```

erzeugt aus der Tabelle `wtKeyMappingTableInput` eine oder mehrere Auswahl-Listen. Der Aufruf erfolgt aus `wtKeysUTM.htm`. Durch Weglassen des Aufrufs dieser Funktion in `wtKeysUTM.htm` kann die Liste unterdrückt werden, obwohl die Funktions-Tasten weiter bedient werden können.

Nach dieser Vorlage kann leicht eine weitere Funktion geschrieben werden, die z.B. für jede Funktion eine Taste oder ein Tabellen-Element generiert.

```
function wtHandleKeyboard( modifier, keyCode )
```

wird von `wt<browser>BrowserFunctions.js` beim Drücken einer Taste aufgerufen. `wtHandleKeyboard()` kann nun aufgrund der von `wtCreateKeyMap()` erzeugten Struktur ermitteln, ob für diese Taste eine Aktion zugeordnet ist, und diese ausführen, andernfalls wird das Tastatur-Ereignis dem Browser überlassen.

Callback-Funktionen des Key mapping

In der Datei `wtCommonBrowserFunctions.js` werden außerdem Funktionen angeboten, die von der Tabelle `wtKeyMappingTableInput` verwendet werden (siehe Attribut `ac` auf [Seite 172](#)).

Die meisten dieser Funktionen liefern `false` als Ergebnis, um zu signalisieren, dass keine allgemeine Behandlung für diese Tastatur-Eingabe zur Verfügung steht. Deshalb soll hier auf das Standardverhalten des entsprechenden Browsers zurückgegriffen werden. Dieses Standardverhalten wird in den Dateien `wt<browser>BrowserFunctions` gegebenenfalls durch Funktionen gleichen Namens überladen, siehe [Seite 178](#).

Ablauf

Das oben beschriebene Verhalten ist wie folgt realisiert:

1. Sobald eine Taste am PC gedrückt wird, ruft der Browser die Funktion `onKeyDown` aus der Datei `wt<browser>BrowserFunctions.js` auf.
2. Die Funktion `onKeyDown` ermittelt `modifier key` und `key code` (siehe Tabelle `wtKeyMappingTableInput` auf [Seite 172](#)) und ruft dann, sofern vorhanden, die Funktion `wtHandleKeyboard` in der Datei `wtCommonBrowserFunctions.js` auf.
3. Die Funktion `wtHandleKeyboard` erkennt, ob in der Tabelle `wtKeyMappingTableInput` unter `ac` (siehe [Seite 172](#)) eine Aktion definiert ist.

Liegt unter `ac` ein `function pointer` vor, ist der Ablauf wie folgt:

4. `wtHandleKeyboard` ruft die Funktion auf und gibt deren Rückgabewert an `onKeyDown` zurück.

Dies ist der Fall bei Aktionen wie z.B. `HOME`, `TAB` oder `CursorDown`. Die Callback-Funktionen dienen hier dazu, Aktionen sofort am Client-PC mit Hilfe des Browsers zu bearbeiten.

5. Die Funktion `onKeyDown` signalisiert dem Browser, ob die Taste bereits behandelt wurde (Rückgabewert `true`). In diesem Fall reagiert der Browser auf die Taste nicht mehr. Anderenfalls führt der Browser seine Standardreaktion für die entsprechende Tastatur-Eingabe aus.

Liegt unter `ac` dagegen eine Zeichenkette (`string`) vor, ist der Ablauf wie folgt:

4. Der Inhalt des `string` wird in das Attribut `SPECIAL_KEY` übertragen (siehe [Seite 147](#)). Das Formular wird an `WebTransactions` übergeben und der Wert von `ac` (z.B. `F1` als Funktionstaste) dabei als Sonderfunktion mitgegeben.
5. In diesem Fall ist der Rückgabewert an den Browser immer `true` (die Taste wurde bereits bearbeitet). Der Browser reagiert nicht mehr auf die Taste.

Ist das Attribut `ac` nicht definiert, existiert also für die gedrückte Taste keine Aktion, wird durch den Rückgabewert `false` signalisiert, dass der Browser selbst die Tastatur-Eingabe behandeln soll.

WebTransactions-spezifische Callback-Funktionen

Abhängig vom verwendeten Browser stellt `WebTransactions` spezielle Implementierungen der Callback-Funktionen zur Verfügung.

Einige der folgenden Funktionen werden von `wtGeckoBrowserFunctions.js` entsprechend der Möglichkeiten der Gecko-Browsers überladen. Von `wtExplorerBrowserFunctions.js` werden alle diese Funktionen überladen (die meisten Möglichkeiten sind vom Internet Explorer bekannt) und erfüllen dann die folgend beschriebene Funktionalität.



Sie können zusätzliche kundenspezifische Callback-Funktionen entwickeln, um die Oberfläche zu erweitern. In diesem Fall müssen Sie darauf achten, dass eine Funktion, die in der Tabelle `wtKeyMappingTableInput` (siehe [Seite 174](#)) angesprochen wird, auch in der Datei `wtCommonBrowserFunctions.js` und in der entsprechenden Datei `wt<browser>BrowserFunctions.js` definiert ist.

```
function doCursorUp()
```

positioniert den Cursor in ein Eingabefeld, das oberhalb des der aktuellen Cursor-Position liegt.

`function doCursorDown()`

positioniert den Cursor in ein Eingabefeld, das unterhalb des der aktuellen Cursor-Position liegt.

`function doCursorLeft()`

befindet sich der Cursor am Anfang eines Eingabefeldes, wird zum Ende des vorhergehenden Eingabefeld gesprungen. Andernfalls wird dem Browser überlassen, auf die eingegebene Taste zu reagieren (bewegen des Cursor im Feld)

`function doCursorRight()`

befindet sich der Cursor am Ende eines Eingabefeldes, wird zum Anfang des nächsten Eingabefeld gesprungen. Andernfalls wird dem Browser überlassen, auf die eingegebene Taste zu reagieren (bewegen der Schreibmarke im Feld).

`function doCursorHome()`

Positioniert den Cursor an den Anfang des ersten Eingabefeldes.

`function doTab()`

springt auf den Anfang des nächsten Eingabefeldes.

`function doBackTab()`

springt auf den Anfang des vorhergehenden Eingabefeldes.

`function doToggleMark()`

Die Markierung des Eingabefeldes, in dem der Focus sich befindet, wird umgeschaltet.

`function doToggleInsert()`

Umschalten zwischen Einfüge- und Überschreib-Modus.

8.3.4 Verwendung des Objekts WT_BROWSER

Um zu vermeiden, dass die Eigenschaften des Browsers und die Font-Größe mehrfach ermittelt werden, wird am Beginn einer Sitzung das Objekt `WT_BROWSER` erzeugt, das dann global während der gesamten Sitzung zur Verfügung steht.

Im Objekt `WT_BROWSER` werden die folgenden Attribute gesetzt:

- Identifikation des Browsers
- Version des Browsers
- Browser-Eigenschaften
- zu verwendende Font-Größe

Diese Attribute werden in den folgenden Templates verwendet:

- alle Templates, die mit den Master-Templates `UTM.wmt` generiert wurden
- `wtBrowserFunctions.htm`
`wtBrowserFunctions.htm` inkludiert `wt<browser>BrowserFunctions.js` und gibt u.a. die Font-Größe weiter.

Font-Größe im Attribut `WT_BROWSER.charSize`

Im Objekt `WT_BROWSER` wird das Attribut `WT_BROWSER.charSize` mit dem Wert 14 vorbelegt (bisheriger statischer Wert).

Existiert beim Sitzungsstart das Attribut `WT_POSTED.wtCharSize`, wird dessen Wert automatisch in `WT_BROWSER.charSize` übernommen. Diese Vorgehensweise ermöglicht es, unterschiedlichen Benutzern individuelle Schriftgrößen zu bieten (z.B. abhängig von der am Bildschirm eingestellten Auflösung).

Während einer bereits laufenden Sitzung kann der Wert von `WT_BROWSER.charSize` mit der Methode `WT_BROWSER.setCharSize` gesetzt werden.



Sie sollten das Attribut `WT_BROWSER.charSize` nicht direkt verändern, da weitere Attribute von diesem Wert abhängen.

Sie können das Objekt `WT_BROWSER` mit der Methode `WT_BROWSER.refresh` erneut initialisieren. Dabei werden die Attribute `WT_SYSTEM.CGI.HTTP_USER_AGENT` und ggf. `WT_POSTED.wtCharSize` erneut ausgewertet. Ein solches Vorgehen ist z.B. sinnvoll, wenn in einer Roaming Session eine laufende Sitzung von einem anderen Browser aus übernommen wird (zu Roaming Sessions siehe WebTransactions-Handbuch „Konzepte und Funktionen“).

Beispiel für die Verwendung von WT_BROWSER.charSize

Sie können den Benutzer auf der Aufrufseite einer WebTransactions-Anwendung auswählen lassen, mit welcher Font-Größe die Anwendung angezeigt werden soll (z.B. über eine Auswahl-Liste):

```
Font Size:
<select name="wtcharSize">
  <option value="12">12
  <option value="14" SELECTED>14
  <option value="17">17
  <option value="20">20
</select>
```

Diese Angabe wird automatisch übernommen, da beim Sitzungsstart das Attribut WT_POSTED.wtCharSize ausgewertet wird. Alle Größeneinstellungen in den generierten Templates erfolgen dann abhängig von diesem Wert.

Sie können auch mit JavaScript das Eingabefeld für wtcharSize abhängig von der Bildschirmbreite vorbelegen, z.B. beim Aufruf einer Seite über einen Submit-Knopf mit einem Eingabefeld für die Font-Größe:

```
<body onload="document.forms.wtaform.wtCharSize.value =
  Math.round(screen.width/75)">
<form method="post" name="wtaform"
  action="/scripts/WTPublish.exe/D:/webta/basedir?Start">
<input type="submit" value="Start">
Font Size:
<input type="text" name="wtCharSize">
...
</form>
</body>
```

8.4 Start-Templates für openUTM

Nach Start der WebTransactions-Anwendung (über eine Aufrufseite oder direkte Angabe der URL) müssen in einem Start-Template die Parameter für die Verbindung mit der Host-Anwendung gesetzt werden.

WebTransactions stellt Ihnen bereits fertige Start-Templates zur Verfügung, die Sie als Vorlage für eigene Start-Templates verwenden können. Dabei haben Sie verschiedene Möglichkeiten:

- **Start-Template-Set**

Dieses Start-Template-Set ist sofort ablauffähig, die benötigten Parameter werden bei jedem Start neu eingegeben, die meisten Parameter sind (sinnvoll) vorbelegt. Es eignet sich sowohl zum Start einer einzelnen Host-Anwendung als auch zum Start mehrerer, in einer WebTransactions-Anwendung integrierter Host-Anwendungen. Das Set besteht aus dem allgemeinen Start-Template `wtstart.htm`, über das Sie z.B. Kommunikationsobjekte anlegen und zwischen verschiedenen parallelen Host-Verbindungen hin- und herschalten können, sowie aus spezifischen Start-Templates für die einzelnen Host-Adapter. Speziell für „WebTransactions for openUTM“ wird das Start-Template `wtstartUTMV4.htm` ausgeliefert. Dieses openUTM-spezifische Start-Template wird in [Abschnitt „openUTM-spezifisches Start-Template des Start-Template-Sets \(wtstartUTMV4.htm\)“ auf Seite 183](#) dargestellt. Eine Darstellung des allgemeinen Start-Templates finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

- **WTBean zur Generierung eines Start-Templates**

Für den Anschluss einer einzelnen openUTM-Anwendung sollten Sie ein eigens dafür generiertes Start-Template verwenden. Bei der Generierung werden Sie vom WTBean `wtcStartUTM.wtc` unterstützt.



8.4.1 openUTM-spezifisches Start-Template des Start-Template-Sets (wtstartUTMV4.htm)

Wenn Sie im allgemeinen Start-Template `wtstart.htm` (beschrieben im WebTransactions-Handbuch „Konzepte und Funktionen“) das Protokoll `UTMV4` ausgewählt und ein neues Kommunikationsobjekt angelegt haben, wird zum Template `wtstartUTMV4.htm` verzweigt. Dieses Template ermöglichen das Setzen der openUTM-spezifischen Parameter in jeweils zwei aufeinander folgenden Schritten:

1. Im ersten Schritt können Sie Verbindungsparameter festlegen und eine Verbindung zu einer openUTM-Anwendung öffnen.
2. Nachdem diese Verbindung geöffnet ist, wird das Template erneut angezeigt. Es ist nun um zusätzliche Angabemöglichkeiten erweitert (`WT_HOST_GLOBALS`-Einstellungen) und enthält Knöpfe für die Kommunikation mit der openUTM-Anwendung.

Beide Schritte (und damit beide Seiten) werden intern über das gleiche Template realisiert, nämlich `wtstartUTMV4.htm`. Eine IF-Struktur sorgt jeweils für die entsprechende Verzweigung.

1.Schritt: Verbindungsparameter setzen und Verbindung öffnen

 		UTMV4 communication		
status	communication object:	WT_HOST.UTM_0		
	connection:	down		
workflow	destination:	main menu ▾	go to	
	access host:	open	open in linemode	
	parameters:	update	reset	
connection parameters directly	APPLICATION_NAME:	VTV10TRV		
	HOST:	NAME:	HOST001	
		or IP_ADDRESS:	<input type="text"/>	
		with HOST_PORT:	<input type="text"/>	
	TAC:	MMENUE		
UPIC_CODE_CONVERSION:	<input checked="" type="checkbox"/>			
... or via upicfile	SYM_DEST:	<input type="text"/>		
additional connection parameters	APPLICATION_PREFIX:	<input type="checkbox"/>		
	CONVERSATION_TAC:	<input type="text"/>		
	CUT_TAC_FIELD:	<input checked="" type="checkbox"/>		
	HOST_CHAR_CODE:	ASCII ▾	<input type="text"/>	
	DISPLAY_EURO:	<input type="checkbox"/>		
	FLD:	TRAV0 ▾		
	BADTAC:	<input type="text"/>		
	LOCAL_APPLICATION:	<input type="text"/>		
	UPIC_TRACE:	<input type="checkbox"/>		
	UPIC_LIB:	<input type="text"/>		
	UTM_PATH:	<input type="text"/>		
	SECURITY_TYPE:	NONE ▾		
	USER:	<input type="text"/>		
	PASSWORD:	<input type="text"/>		
NEW_PASSWORD:	<input type="text"/>			
RESTART:	<input type="checkbox"/>			

Im Abschnitt **connection parameters directly** können die folgenden Attribute definiert werden, mit denen der Service in der openUTM-Anwendung adressiert wird:

APPLICATION_NAME

Name der openUTM-Anwendung.

HOST

Name (NAME) oder IP-Adresse (IP_ADDRESS) mit Portnummer (HOST_PORT) des Rechners, auf dem die openUTM-Anwendung läuft.

TAC

Transaktionscode des Service, der in der openUTM-Anwendung aufgerufen werden soll.

UPIC_CODE_CONVERSION

Wenn diese Option ausgewählt ist, dann wird eine Code-Konvertierung (ASCII-EBCDIC) erzwungen. In diesem Fall darf die Option **HOST_CHAR_CODE** nicht gesetzt sein.

Alternativ können Sie im Abschnitt ... **or via upicfile** eintragen:

SYM_DEST

Hier kann ein Wert ausgewählt werden, der den gewünschten Service angibt und im Attribut `SYM_DEST` abgelegt wird. Die Auswahl wird automatisch aus dem Inhalt der `upicfile` ermittelt.

Im Abschnitt **additional connection parameters** tragen Sie weitere optionale Parameter ein:

APPLICATION_PREFIX

Ist diese Option ausgewählt, wird das Attribut `APPLICATION_PREFIX` mit dem Namen des Kommunikationsobjekts versorgt.

CONVERSATION_TAC, CUT_TAC_FIELD

Setzen die gleichnamigen Attribute.

HOST_CHAR_CODE

Hier ist anzugeben, ob die Nachrichten in **ASCII** oder **EBCDIC** mit dem Server ausgetauscht werden. Wird **TABLE** ausgewählt, so ist im nebenstehenden Eingabefeld der Name einer eigenen Konvertierungstabelle anzugeben. Der Inhalt dieser Auswahl wird im Attribut **HOST_CHAR_CODE** abgelegt.

Die Option darf nicht zusammen mit **UPIC_CODE_CONVERSION** verwendet werden.

FLD

Hier kann eine der installierten Felddateien ausgewählt werden. Die Auswahl wird im Attribut `FLD` abgelegt und ist später von Bedeutung, falls die Kommunikation mit dem Senden einer Nachricht begonnen wird.

BADTAC, LOCAL_APPLICATION, UPIC_TRACE, UPIC_LIB, UTM_PATH, SECURITY_TYPE, USER, PASSWORD, NEW_PASSWORD, RESTART

Setzen der gleichnamigen Attribute.

Im Abschnitt **workflow** bestimmen Sie die nächste Aktion.

destination

Hier können Sie auswählen mit welchem Template weitergemacht werden soll. Durch Klick auf **go to** wird auf die ausgewählte Seite verzweigt. Als Vorgabe wird **main menu** angeboten: Damit kann man auf die allgemeine Aufrufseite `wtstart.htm` zurückkehren. Falls mehrere Verbindungen geöffnet sind, werden sie als weitere Einträge zur Auswahl angeboten; verzweigt wird dann auf die jeweiligen Host-Adapter-spezifischen Start-Templates dieser Verbindungen.

access host

Hier werden die Aktionen angeboten, die aktuell mit der Sitzung durchgeführt werden können. Ist noch keine Verbindung geöffnet, so stehen **open** und **open in linemode** zur Verfügung. Damit kann die Verbindung geöffnet werden.

Näheres zu **open in linemode** finden Sie in [Abschnitt „Unterstützung des openUTM-Zeilenmodus“ auf Seite 115](#).

parameters

Mit **reset** werden alle Parameter in den gleichen Zustand versetzt, in dem sie vom Browser empfangen wurden.

Mit **update** können die Werte der Seite an WebTransactions geschickt werden, ohne dass eine Kommunikation mit dem Host stattfindet. Dies ist z.B. dann sinnvoll, wenn vor dem Öffnen der Verbindung das Feld `FLD` nicht belegt war und deshalb **enter dialog** nicht möglich ist. Wenn dann `FLD` belegt wird, so wird der Knopf **enter dialog** nach einem Klick auf **update** sichtbar.

2. Schritt: Kommunikation aufnehmen und Host-Attribute setzen

Sobald eine Verbindung geöffnet ist, werden im Abschnitt **workflow** unter **access host** zusätzlich Knöpfe angeboten, die eine Kommunikation mit der Host-Anwendung ermöglichen. Außerdem wird die Seite um den Abschnitt **global host attributes** erweitert. Hier können Sie die Attribute `Padding`, `Detect`, `Undetect`, `Read`, `FieldLength`, `Update` und `Cursor` des Host-Steuerobjekts `WT_HOST_GLOBALS` zu setzen.

Falls Sie in Schritt 1 **open** gewählt haben, werden die Knöpfe **send**, **receive** und **close** und ggf. **enter dialog** angeboten:

receive / send

Die Knöpfe **receive** und **send** werden alternierend angezeigt.

receive empfängt die nächste Nachricht von der Host-Anwendung und erweitert das Start-Template zum Setzen der Host-Attribute. **send** sendet eine Nachricht zur Host-Anwendung. Soll ein Schirm mit veränderten Daten an die Host-Anwendung geschickt werden, so ist die Auswahl **enter dialog** zielführend.

close

schließt die Verbindung zur Host-Anwendung und kehrt auf die im ersten Schritt angezeigte Seite zurück. Dort können Sie eine neue Verbindung auswählen und öffnen.

enter dialog

verzweigt direkt zum nächsten Schirm der Host-Anwendung. Dieser Schirm kann jetzt mit Daten gefüllt und an die Host-Anwendung geschickt werden.

Falls Sie aus einer laufenden Host-Anwendung durch Auswählen des Knopfes **suspend** in diese Seite zurückkehren, so wird an Stelle von **enter dialog** der Knopf **resume dialog** angezeigt.

Start eines Vorgangs, der weder Daten noch ein Basisformat erwartet

Benötigt das erste Teilprogramm eines Vorgangs keine Daten und erwartet kein Basisformat, sondern gibt das erste Format selbst aus, so können Sie den Vorgang folgendermaßen starten:

Erste Seite: Auswahl des Vorgangs und Festlegen der Nachrichtencodierung (über `connection parameters directly` oder `per SYM_DEST` und/oder `HOST_CHAR_CODE`), ggf. weitere Einstiegsparameter, Auswahl von `open`.

Zweite Seite: ggf. Modifikation der Attribute von `WT_HOST_GLOBALS`, Auswahl von `receive`. Anschließend wird die Seite nochmals angezeigt. Wählen Sie nun **enter dialog**.

Start eines Vorgangs, der Daten oder ein Basisformat erwartet

Benötigt das erste Teilprogramm eines Vorgangs definierte Daten oder erwartet ein Basisformat, so können Sie den Vorgang folgendermaßen starten:

- Erste Seite: Auswahl des Vorgangs und Festlegen der Nachrichtencodierung (über `connection parameters directly` oder `per SYM_DEST` und/oder `HOST_CHAR_CODE`), Festlegung der Nachricht (FLD), ggf. weitere Einstiegsparameter, Auswahl von `open`.
- Zweite Seite: ggf. Modifikation der Attribute von `WT_HOST_GLOBALS`, Auswahl von **enter dialog**.

8.4.2 WTBean `wtcStartUPIC.wtc` zur Generierung eines Start-Templates

Für den Anschluss einer einzelnen openUTM-Anwendung können Sie ein anwendungsspezifisches Start-Template generieren. Dazu steht Ihnen das WTBean `wtcStartUPIC.wtc` zur Verfügung. Es handelt sich dabei um ein standalone WTBean.

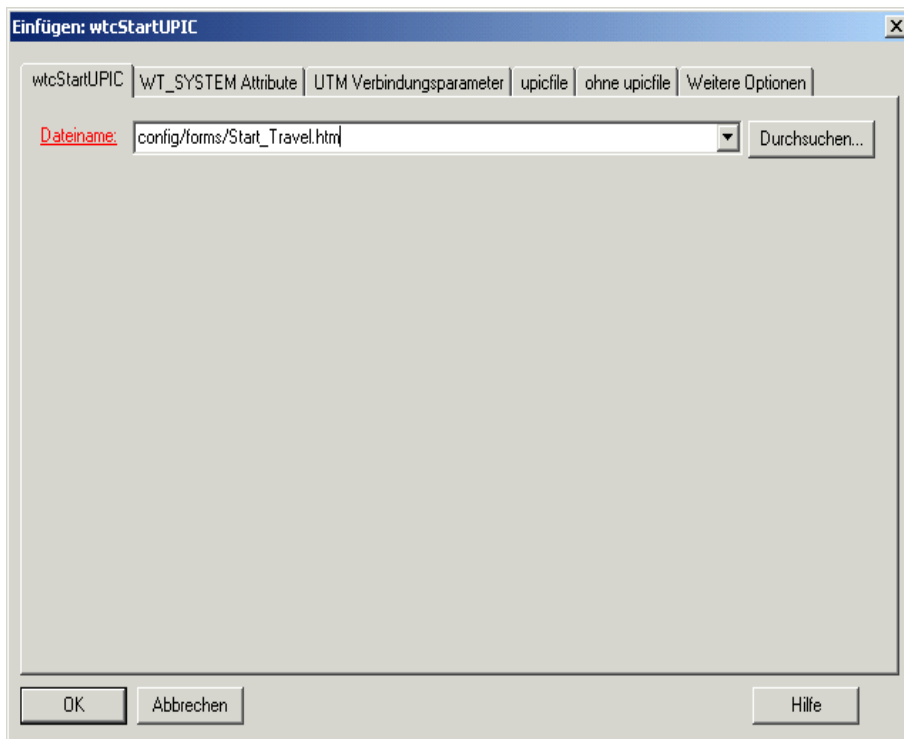
`wtcStartUPIC.wtc` enthält das inline WTBean `wtcUPIC.wtc`, mit dem Sie in einem Template ein neues openUTM-Kommunikationsobjekt anlegen, und damit eine Verbindung zu einer openUTM-Anwendung aufbauen (siehe Abschnitt „[Neues openUTM-Kommunikationsobjekt anlegen \(wtcUPIC\)](#)“ auf Seite 190).



Damit Sie auf WTBeans zugreifen können, muss eine Verbindung zur WebTransactions-Anwendung bestehen.

Mit dem Befehl **Datei/Neu/wtcStartUPIC** rufen Sie das WTBean zur Bearbeitung auf. WebLab generiert das Dialogfeld **Einfügen:wtcStartUPIC** mit sechs Registerkarten:

- In der Registerkarte **wtcStartUPIC** legen Sie den Namen des zu generierenden Start-Templates fest.
- In der Registerkarte **WT_SYSTEM-Attribute** legen Sie die wichtigsten Attribute des Systemobjekts fest.
- In den Registerkarten **UTM-Verbindungsparameter** sowie **upicile** bzw. **ohne upicfile** legen Sie die wichtigsten Verbindungsparameter fest.
- In der Registerkarte **Weitere Optionen** können Sie in einer Baumstruktur alle Parameter für die Verbindung zur openUTM-Anwendung bearbeiten.



Das generierte Start-Template selbst erzeugt keine eigene Seite im Browser. Nach Start von WebTransactions erscheint unmittelbar das Template, das dem ersten von der Host-Anwendung empfangenen Format entspricht. Dafür sorgt das `wtinclude`-Tag am Ende des Start-Templates.

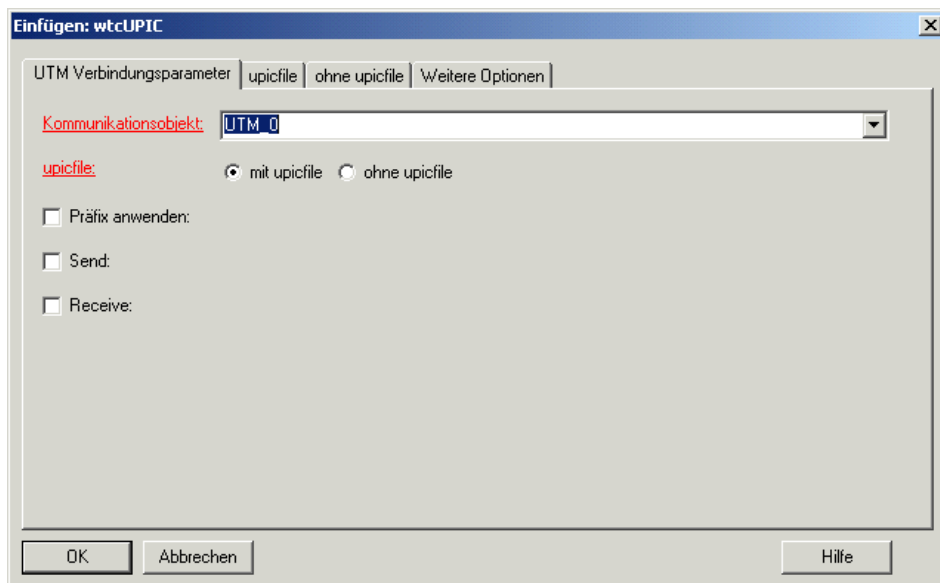
8.5 Neues openUTM-Kommunikationsobjekt anlegen (wtcUPIC)

Um in einem Template ein neues openUTM-Kommunikationsobjekt anzulegen und damit eine Verbindung zu einer openUTM-Anwendung aufzubauen, wird das WTBean `wtcUTM` mit ausgeliefert. Dieses WTBean können Sie auch dazu nutzen, um mehrere Verbindungen parallel zu öffnen. `wtcUTM` ist ein inline WTBean, siehe hierzu auch das WebTransactions-Handbuch „Konzepte und Funktionen“.



Damit Sie auf inline WTBeans zugreifen können, muss eine Verbindung zur WebTransactions-Anwendung bestehen und das Template geöffnet sein, in das Sie das WTBean einfügen wollen.

Mit dem Befehl **Einfügen/WTBean/wtcUPIC** rufen Sie das WTBean zur Bearbeitung auf. WebLab generiert das Dialogfeld **Einfügen:wtcUPIC**:



In diesem Dialogfeld können Sie die Parameter für das neue Kommunikationsobjekt bearbeiten.

- ▶ Auf der Registerkarte **UTM Verbindungsparameter** müssen Sie den Namen des Kommunikationsobjekts angeben und auswählen, ob die Verbindungsparameter aus der `upicfile` gelesen werden (Option **mit upicfile**) oder direkt eingetragen werden (Option **ohne upicfile**).
- ▶ Je nach Auswahl füllen Sie entweder die Registerkarte **upicfile** oder die Registerkarte **ohne upicfile** aus.

- ▶ Alle weiteren Parameter können Sie auf der Registerkarte **Weitere Optionen** in einer Baumstruktur bearbeiten.

Wenn Sie die Parameter versorgt haben und Ihre Angaben mit **OK** bestätigen, wird aus den Parametern und der Beschreibungsdatei der Code des WTBean erzeugt und in das geöffnete Template an der Cursor-Position eingefügt.

Das WTBean besteht aus geschützten und ungeschützten Code-Bereichen. Die geschützten Bereiche sind grau hinterlegt dargestellt. Auf diese Bereiche können Sie nur über die Oberfläche des WTBean Einfluss nehmen. Sie wählen dazu im Kontextmenü der Startzeile des WTBean (pink hinterlegt) den Befehl **WTBean bearbeiten** (siehe WebTransactions-Handbuch „Konzepte und Funktionen“).

8.6 Security durch openUTM-Benutzerkonzept

Mit WebTransactions können Sie das openUTM-Benutzerkonzept nutzen. Damit ist es möglich, auch bei Zugriff über das Web die Zugangs- und Zugriffsschutzmechanismen von openUTM zu verwenden: Die skalierbare Autorisierung über das Lock-/Keycode-Konzept von openUTM kann genutzt werden.

Anmelden unter einer openUTM-Benutzererkennung

Für die Nutzung des openUTM-Benutzerkonzepts gibt es spezielle Attribute des Systemobjekts, `SECURITY_TYPE`, `USER`, `PASSWORD` und `NEW_PASSWORD`. Bei jedem Aufruf von open prüft WebTransactions das Attribut `SECURITY_TYPE` und übergibt ggf. `USER`, `PASSWORD` und `NEW_PASSWORD`.



Ein Security-Check wird also bei jedem Vorgangsstart durchgeführt.

`SECURITY_TYPE`

Angabe der Sicherheitsstufe. Mögliche Werte:

`NONE` Bei der Anmeldung übergibt WebTransactions weder den Namen einer openUTM-Benutzererkennung noch ein openUTM-Passwort an die openUTM-Anwendung. Ein Leerstring in `SECURITY_TYPE` hat die gleiche Wirkung.

`USER` Für die Anmeldung bei der openUTM-Anwendung wird der im Attribut `USER` hinterlegte Name einer Benutzererkennung verwendet. Ein Passwort wird jedoch nicht übergeben.

`PASSWORD`

Für die Anmeldung bei der openUTM-Anwendung werden der im Attribut `USER` hinterlegte Name einer openUTM-Benutzererkennung und das im Attribut `PASSWORD` hinterlegte Passwort verwendet.

`USER` Name der openUTM-Benutzererkennung, der ggf. zur Berechtigungsprüfung an die openUTM-Anwendung übergeben wird.

`PASSWORD`

Passwort der übergebenen openUTM-Benutzererkennung (Zeichenvorrat entsprechend openUTM, d.h. es sind nur abdruckbare Zeichen zulässig).

`NEW_PASSWORD`

Neues Passwort der übergebenen openUTM-Benutzererkennung (Zeichenvorrat entsprechend openUTM, d.h. es sind nur abdruckbare Zeichen zulässig).



Der Event-Service SIGNON wird ab UPIC V5.0 in Verbindung mit openUTM ab V5.0 unterstützt.

Für openUTM-Anwendungen < V5.0 hingegen kann der SIGNON-Service über WebTransactions noch nicht genutzt werden. Falls Sie solche Terminal-Host-Anwendungen, die den Event-Service SIGNON einsetzen, ans Web anschließen, müssen Sie die SIGNON-Verarbeitungsschritte in ein eigenes Teilprogramm integrieren und dieses vom Start-Template aus aufrufen.

Ungültige Benutzerkennung oder ungültiges Passwort

Falls bei der Anmeldung eine ungültige Benutzerkennung oder ein ungültiges Passwort verwendet wird, so belegt WebTransactions das Attribut `RECEIVE_ERROR` nach dem ersten `receive` mit dem Wert `CM_SECURITY_NOT_VALID`. Der Fehler kann dann im Template individuell behandelt werden, indem das Attribut `RECEIVE_SECONDARY_INFORMATION` ausgewertet wird. Die Werte dieses Attributs sind auf [Seite 153](#) beschrieben.

Falls das Attribut `USER` leer ist, `SECURITY_TYPE` jedoch auf `USER` oder `PASSWORD` gesetzt ist, so belegt WebTransactions das Attribut `RECEIVE_ERROR` bereits beim Verbindungsaufbau (`open`) mit dem Wert `CM_SECURITY_NOT_VALID`.

Passwort ändern

Das Passwort einer openUTM-Benutzerkennung lässt sich dadurch ändern, dass zusätzlich zum Attribut `PASSWORD` das Attribut `NEW_PASSWORD` versorgt wird. In `PASSWORD` müssen Sie das bisherige Passwort und in `NEW_PASSWORD` das geänderte Passwort eintragen.

Falls die openUTM-Anwendung mit Grace-Sign-On generiert ist, dann kann das Passwort auch noch nach Ablauf der Gültigkeitsdauer geändert werden.

Anmeldung bei nicht angeforderter Wiederanlauf-Information

Falls zum Zeitpunkt der Anmeldung für die verwendete Benutzerkennung eine Wiederanlaufinformation vorliegt, eine solche für diese Benutzerkennung jedoch nicht angefordert wurde, so unterscheidet WebTransactions, ob in der zuvor zurückgesetzten Transaktion eine Nachricht an die openUTM-Anwendung geschickt wurde oder nicht:

- Wurde in der zurückgesetzten Transaktion **nicht** versucht, eine Nachricht zu schicken (`open + receive`), so wiederholt WebTransactions automatisch den Verbindungsaufbau, wodurch die Wiederanlaufinformation verworfen wird.
- Wurde in der zurückgesetzten Transaktion versucht, eine Nachricht zu senden (`open + send + receive`), so hinterlegt WebTransactions im Attribut `RECEIVE_ERROR` den Wert `CM_DEALLOCATED_ABEND`. Der Fehler kann nun im Template individuell behandelt werden.

8.7 RESTART - Automatischer Wiederanlauf

Für openUTM-Anwendungen, die auf einer openUTM-Version größer gleich V3.4 basieren (für UPIC auf BS2000/OSD größer gleich V4.0), können die openUTM-Wiederanlaufaktionen auch in Client-/Serverumgebungen und somit auch von WebTransactions genutzt werden.

Wiederanlauf heißt, dass ein openUTM-Vorgang, der auf Grund eines Fehlers oder einer Störung (z.B. Verbindungsverlust durch Reboot) auf den letzten Sicherungspunkt zurückgesetzt wurde, nach erneutem Anmelden des Clients von diesem Punkt fortgesetzt werden kann.

Da openUTM den Vorgangskontext jeweils benutzerspezifisch sichert, ist ein Wiederanlauf grundsätzlich nur dann möglich, wenn mit Benutzerkennungen gearbeitet wird (SECURITY_TYPE=USER/PASSWORD). Zusätzlich muss das Attribut RESTART auf YES gesetzt sein.

Wiederanlauf beim Start

Falls im Start-Template der WebTransactions-Anwendung das Attribut WT_SYSTEM.RESTART auf YES gesetzt und eine Benutzerkennung angegeben ist, verwendet WebTransactions unabhängig vom Attribut WT_SYSTEM.TAC bzw. WT_SYSTEM.SYM_DEST intern zunächst den Wiederanlauf-Transaktionscode KDCDISP für einen automatischen Wiederanlauf. Erst wenn dieser fehlschlägt, wird mit dem Transaktionscode fortgesetzt, der durch das Attribut WT_SYSTEM.TAC (direkte Angabe) oder durch WT_SYSTEM.SYM_DEST in der `upicfile` spezifiziert ist.

Hat WT_SYSTEM.RESTART einen Wert ungleich YES oder ist keine Benutzerkennung angegeben, so beginnt WebTransactions gleich mit dem Transaktionscode, der über WT_SYSTEM.TAC bzw. WT_SYSTEM.SYM_DEST spezifiziert wird.

Reaktion, falls Wiederanlauf nicht möglich

In manchen Fällen ist kein Wiederanlauf möglich. Wie WebTransactions auf eine solche Situation reagiert, ist abhängig von der Ursache und davon, ob in der zurückgesetzten Transaktion versucht wurde, eine Nachricht an die openUTM-Anwendung zu senden.

- Falls kein Wiederanlauf möglich ist, weil die openUTM-Anwendung inzwischen neu generiert wurde oder der Benutzer zuletzt über ein Terminal (Datenstation) angemeldet war:
 - Wurde in der zurückgesetzten Transaktion **nicht** versucht, eine Nachricht zu schicken (nur `open + receive`), so wird der Transaktionscode gestartet, der im Attribut TAC bzw. durch das Attribut SYM_DEST in der `upicfile` identifiziert wird.

- Wurde in der zurückgesetzten Transaktion versucht, eine Nachricht zu senden (open + send + receive), so hinterlegt WebTransactions im Attribut `RECEIVE_ERROR` den Wert `CM_TP_NOT_AVAILABLE_NO_RETRY`. Der Fehler kann nun im Template individuell behandelt werden.
- Falls kein Wiederanlauf möglich ist, weil in der openUTM-Konfiguration für diese Benutzerkennung `RESTART=NO` gesetzt ist, reagiert WebTransactions genauso wie beim oben geschilderten Fall (einziger Unterschied: unterschiedlicher `RECEIVE_ERROR`-Wert):
 - Wurde in der zurückgesetzten Transaktion **nicht** versucht, eine Nachricht zu schicken (nur open + receive), so wird der Transaktionscode gestartet, der im Attribut `TAC` bzw. durch das Attribut `SYM_DEST` in der `upicfile` identifiziert wird.
 - Wurde in der zurückgesetzten Transaktion versucht, eine Nachricht zu senden (open + send + receive), so hinterlegt WebTransactions im Attribut `RECEIVE_ERROR` den Wert `CM_TPN_NOT_RECOGNIZED`. Der Fehler kann nun im Template individuell behandelt werden.

Reaktion, falls Wiederanlauf Vorgangsende liefert (nur bei Verbindungen über UPIC)

Liefert der Wiederanlauf ein Vorgangsende, so schreibt WebTransactions den Wert `CONVERSATION_END` in das Systemattribut `RESTART`. Erst nach dem nächsten Aufruf von `receive` wird `RESTART` auf Leerstring zurückgesetzt.

Damit ist es möglich, einen Wiederanlauf auf dem „Abmeldungsschirm“ zu erkennen und im entsprechenden Template gezielt zu behandeln.

Beispiel

Das folgende Beispiel zeigt ein typisches Template für eine Abmeldung bei Wiederanlauf bei Verwendung der `upicfile`.

```
<wtIf (host_system.RESTART == "CONVERSATION_END")>
```

```
<wtOnCreateScript>
```

(1)

```
host_system.CONVERSATION_TAC = "SYM_DEST";
host_system.SYM_DEST = "MAINENTR";
```

```
host.open();
```

```
host.receive(); (2)
host.system.CONVERSATION_TAC = "";

</wtOnCreateScript>

<wtInclude name="##wt_system.FORMAT#">

<wtElse>
<html> (3)
<head>
<title>TRSGNOF</title>
</head>
  bye bye ...
<body bgColor="#C0C0C0">
</body>
</html>
<wtOnCreateScript> (4)
  host.close();
</wtOnCreateScript>
</wtEndIf>
```

Der erste Zweig des `<wtif>` behandelt den Wiederanlauf. Wird dieses Ende-Template im Fall eines Wiederanlaufs erreicht, so soll natürlich nicht die Ausgabe einer Schlussmeldung und der Verbindungsabbau durchgeführt werden, sondern mit der Einstiegsseite der Host-Anwendung fortgefahren werden. Dazu wird in Teil 1 ein neuer Vorgang ausgewählt. Vor dem `open` wird das Attribut `CONVERSATION_TAC` auf `SYM_DEST` gesetzt, um zu verhindern, dass ein `openUTM`-Steuerfeld oder der Beginn der Formatnachricht als TAC interpretiert wird. Im Teil 2 wird dann wie bei der Anmeldung am Anfang der erste TAC angesteuert (abhängig von der Host-Anwendung kann hier ein zusätzliches `send` notwendig sein).

Im `<wtelse>`- Zweig wird das eigentliche Verhalten der Schlussseite spezifiziert. Teil 3 generiert eine Meldung an den Benutzer und Teil 4 führt die Abmeldung von der Host-Anwendung und das Beenden der WebTransactions Sitzung durch.

8.8 BADTAC - Simulation des Event-Service BADTAC

Eine openUTM-Anwendung können Sie so konfigurieren, dass immer dann, wenn im Zeilenmodus oder über das openUTM-Steuerfeld eines Formats ein ungültiger Transaktionscode angegeben wird, automatisch ein spezieller Vorgang gestartet wird: der Event-Service BADTAC.

Der Event-Service BADTAC ist ein selbst-definierter Vorgang, für den der Transaktionscode KDCBADTC generiert wurde.

Bei der Kommunikation über UPIC lässt sich der Event-Service BADTAC jedoch nicht unmittelbar nutzen, da über UPIC der Transaktionscode KDCBADTC nicht gestartet werden kann.

Mit dem Attribut BADTAC des Systemobjekts ermöglicht es Ihnen WebTransactions jedoch, diesen Event-Service zu simulieren. Hierzu generieren Sie für das Teilprogramm, das bei Angabe eines ungültigen Transaktionscodes automatisch gestartet werden soll, einen beliebigen Transaktionscode ungleich KDCBADTC und hinterlegen diesen im Attribut BADTAC. Dieses Teilprogramm darf keine Nachricht erwarten. Sie können auch dasselbe Programm verwenden wie beim Event-Service BADTAC. Sie müssen aber dann für dieses neben KDCBADTC einen weiteren Transaktionscode generieren, und für das Attribut BADTAC diesen verwenden.

Falls das Attribut BADTAC **nicht** gesetzt ist, werden ungültige Transactionscodes nicht automatisch abgefangen: Ein auf die ungültige Angabe folgender Aufruf der Methode `receive` führt zu einem Fehler.

8.9 Automatische Vorgangsverknüpfung

Die Produktvariante „WebTransactions for openUTM“ kommuniziert über UPIC mit der openUTM-Anwendung. Bei dieser Kommunikation wählt der Client (also hier WebTransactions) einen openUTM-Vorgang durch Angabe eines Transaktionscodes aus. Die Auswahl wird vom Template durch das Systemattribut `TAC` bzw. `SYM_DEST` gesteuert. Während eines Vorgangs bestimmt nun die openUTM-Anwendung die Folge der zu durchlaufenden Teilprogramme (Ausnahme: aktiver Dialog). Am Ende des Vorgangs liegt die Initiative wieder auf der Seite von WebTransactions: Wenn gewünscht, kann ein weiterer Vorgang gewählt und gestartet werden.

Beim Terminalbetrieb gibt es die Möglichkeit, einen Folgevorgang durch ein openUTM-Steuerfeld oder die ersten 8 Zeichen des Formats automatisch zu starten. Dieses Verhalten wird von WebTransactions nachvollzogen. Am Ende eines Vorgangs ermittelt WebTransactions aus dem openUTM-Steuerfeld oder den ersten 8 Zeichen des aktuellen Formats den Transaktionscode für den folgenden Vorgang. Dieser Vorgang wird dann implizit durch den nächsten Aufruf der Methode `send` gestartet.

Damit entsteht für den Template-Programmierer kein Anpassungsaufwand für die automatische Vorgangsverknüpfung.

8.10 Simulation der Funktionstasten

In einer openUTM-Anwendung können für Funktionstasten TACs und Returncodes generiert werden (F1,F2,...,F24 und in BS2000/OSD zusätzlich K1 bis K14). Jeder Funktionstaste kann so per openUTM-Generierung (KDCDEF-Anweisung `SFUNC`) eine bestimmte Funktion zugeordnet werden, die openUTM ausführt, wenn der Benutzer am Terminal diese Funktionstaste drückt.

Ab Version 4.0 ermöglicht es openUTM, das Drücken von Funktionstasten bei Kommunikation über UPIC zu simulieren. Damit lassen sich Terminal-Host-Anwendungen, die mit openUTM-Funktionstasten arbeiten, leichter in Client-/Serverumgebungen integrieren.

Diese Funktionalität können Sie auch über WebTransactions nutzen, indem Sie das Attribut `SPECIAL_KEY` des Systemobjekts mit einem der Werte `F1-F24` oder `K1-K14` belegen. Mit dem nächsten `send` sendet WebTransactions dann diese Funktionstaste an die openUTM-Anwendung. Nach dem Senden wird das Attribut von WebTransactions zurückgesetzt, um fortgesetztes Senden der Funktionstaste zu verhindern.

Für eine komfortable Auswahl einer Funktionstaste stellt WebTransactions die Dateien `wtKeysUTM.htm` und `wtBrowserFunctions.htm` zur Verfügung, die Sie mit dem Include-Tag in ein FHS- oder FORMANT-basiertes Template integrieren können.

`wtBrowserFunctions.htm` inkludiert die Dateien `wtKeysUTMFHS.js` und `wtKeysUTMFormant.js` für die Browser-Behandlung, siehe auch [Seite 171](#).

Bei der Generierung der Templates werden automatisch Include-Tags für `wtKeysUTM.htm` und `wtBrowserFunctions.htm` erzeugt. Diese Dateien definieren Knöpfe für sämtliche F- und K-Tasten und sorgen dafür, dass das Attribut `SPECIAL_KEY` entsprechend der getroffenen Wahl versorgt wird. Die Dateien sollten im konkreten Einsatzfall kopiert und auf die in der openUTM-Anwendung generierten Tasten reduziert werden.

Versorgung entsprechender Globalattribute bei #-Formaten

Unabhängig von der Übertragung einer Funktionstaste als openUTM-Returncode liefern FHS und FORMANT bei #-Formaten Informationen über eine gedrückte Funktionstaste in den Globalattributen `InputKeyClass` und `InputKeyNumber` zurück. Diese Attribute können im Template gesetzt werden (über entsprechende Attribute des Objekts `WT_HOST_MESSAGE`). Bei generierten Templates sind entsprechende Anweisungen bereits enthalten (siehe [Abschnitt „Aufbau der generierten Templates“ auf Seite 87](#)).

8.11 Unterstützung von KDCSCUR

Wenn in einem openUTM-Teilprogramm in einem Dialogschritt eine Formatausgabe vorgesehen ist und mittels des Aufrufs `KDCSCUR` der `Cursor` auf ein Feld gesetzt werden soll, dann wird diese Information an `UPIC` übergeben. `UPIC` wertet diese Information aus und gibt sie an `WebTransactions` weiter.

Damit `WebTransactions` diese Information umsetzt, muss das Attribut `WT_HOST_GLOBALS.Cursor` den Wert `N` haben. `wtBrowserFunctions.htm` wertet `WT_HOST_GLOBALS.Cursor` aus und setzt mit einem JavaScript den Fokus im Browser-Fenster. Dadurch wird der `Cursor` nach einer Ausgabe an den Browser in das Feld gestellt, das von der openUTM-Anwendung für den betreffenden Dialogschritt vorgesehen ist.

`WT_HOST_GLOBALS.Cursor` wird aber unabhängig vom Template `wtBrowserFunctions.htm` versorgt.

8.12 Gezieltes Anmelden über bestimmte LTERMs

openUTM ermöglicht es, für LTERM-Partner (logische Anschlusspunkte) spezifische Eigenschaften zu konfigurieren und sie z.B. in das Lock-/Keycode-Konzept einzubeziehen: In der openUTM-Konfiguration können Sie nicht nur für Services (openUTM-Vorgänge) sondern auch für LTERM-Partner Lockcodes definieren. Zusätzlich lassen sich für LTERM-Partner - ähnlich wie für openUTM-Benutzerkennungen - auch Keycodes festlegen:

- Die Anmeldung eines Client-Programms ist nur möglich, wenn der angegebenen Benutzerkennung ein Keycode zugeordnet ist, der mit dem Lockcode des zugeordneten LTERM-Partners übereinstimmt.
- Ein Client-Programm kann einen Service nur dann aufrufen, wenn **sowohl** das Keyset der jeweiligen Benutzerkennung **als auch** das des LTERM-Partners einen Keycode enthalten, der mit dem Lockcode des Service übereinstimmt.

Bei einem Zugriff über WebTransactions ist es vom verwendeten `localapps`-Namen abhängig, welchen LTERM-Partner das WTHolder-Programm verwendet. Da WebTransactions zur Laufzeit standardmäßig immer den ersten freien `localapps`-Namen zuteilt, ist die Zuordnung zufällig (freie Zuordnung). Falls ein WTHolder-Programm sich über einen ganz bestimmten LTERM-Partner anschließen soll, können Sie gezielt einen lokalen Host-Anwendungsname in das Systemobjekt-Attribut `LOCAL_APPLICATION` eintragen (feste Zuordnung), die Datei `localapps` bleibt dann unberücksichtigt.

Falls Sie mit fester Zuordnung arbeiten, kann WebTransactions nicht sicherstellen, dass jeder Sitzung ein noch unbenutzter Host-Anwendungsname zugeordnet wird; dies muss bei der Implementierung der WebTransactions-Anwendung anders sichergestellt werden.



Falls eine Anmeldung über das Protokoll `UPIC_V4` scheitert und in der `upicfile` der Parameter `PROTOCOL` nicht angegeben ist, versucht UPIC eine Anmeldung mit Protokoll der UPIC-Version 3.4. Deshalb kann bei Fehlermeldungen, die auf das Scheitern der Anmeldung hinweisen, eine „falsche“ Versionsangabe enthalten sein. Eine passende Versionsangabe bei diesen Meldungen erhalten Sie, wenn Sie `PROTOCOL=40` beim entsprechenden Eintrag in der `upicfile` angeben.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *->kursiver* Schrift ausgezeichnet.

aktiver Dialog

Beim aktiven Dialog greift WebTransactions aktiv in die Steuerung des Dialogablaufs ein, d.h., das nächste zu verarbeitende *->Template* wird von der Template-Programmierung bestimmt. Mit den *->WTML*-Sprachmitteln können Sie z.B. mehrere *->Host-Formate* in einer *->HTML*-Seite zusammenfassen. Dabei wird am Ende eines Host- *->Dialogschritts* keine Ausgabe an den *->Browser* geschickt, sondern unmittelbar der Folgeschritt gestartet. Ebenso sind innerhalb **eines** Host-Dialogschritts mehrere Interaktionen zwischen Web- *->Browser* und WebTransactions möglich.

Array

->Datentyp, der eine endliche Menge von Werten eines Datentyps enthalten kann. Der Datentyp kann sein

- *->skalar*
- eine *->Klasse*
- ein Array

Die Werte im Array werden durch einen numerischen Index angesprochen, der mit 0 beginnt.

Asynchrone Nachricht

Versteht WebTransactions als Nachricht, die ans Terminal geschickt wird, ohne dass sie vom Anwender ausdrücklich angefordert worden wäre - d.h. ohne dass der Anwender auf irgendeine Taste gedrückt oder auf ein Oberflächenelement geklickt hätte.

Attribut

Definiert eine Eigenschaft eines *->Objekts*.

Ein Attribut kann z.B. Farbe, Größe oder Position eines Objekts oder selbst wieder ein Objekt sein. Attribute werden auch als *->Variablen* verstanden und können abgefragt und verändert werden.

Aufrufseite

Eine ->*HTML*-Seite, die Sie benötigen, um eine ->*WebTransactions-Anwendung* zu starten. Auf dieser Seite steht der Aufruf, der *WebTransactions* mit dem ersten ->*Template* startet, dem Start-Template.

Ausdruck

Kombination aus ->*Literalen*, ->*Variablen*, Operatoren und Ausdrücken, deren Auswertung jeweils ein bestimmtes Ergebnis liefert.

Auswertungsoperator

WebTransactions versteht den Auswertungsoperator als Operator, der die angesprochenen ->*Ausdrücke* durch ihr Ergebnis ersetzt (Objekt-Attribut-Auswertung). Der Auswertungsoperator wird in der Form `##ausdruck#` angegeben.

Automask-Template

Ein *WebTransactions*- ->*Template*, das von *WebLab* implizit beim Erzeugen eines Basisverzeichnisses oder explizit mit dem Befehl **Automask erzeugen** erstellt wird. Es wird verwendet, wenn kein formatspezifisches Template identifiziert werden kann. Ein Automask-Template enthält die Anweisungen, die für die dynamischen Formatabbildungen und zur Kommunikation notwendig sind. Es können verschiedene Varianten von Automask-Templates erstellt und über das System-Objekt-Attribut `AUTOMASK` ausgewählt werden.

Basisverzeichnis

Das Basisverzeichnis liegt auf dem *WebTransactions*-Server und ist die Grundlage einer ->*WebTransactions-Anwendung*. Im Basisverzeichnis liegen die ->*Templates* und alle Dateien oder Verweise auf Programme (Links), die für den Ablauf einer *WebTransactions-Anwendung* benötigt werden.

BCAM-Applikationsname

Entspricht dem *openUTM*-Generierungsparameter `BCAMAPPL` und ist der Name der ->*openUTM-Anwendung*, über den ->*UPIC* die Verbindung aufnehmen kann.

Benutzerkennung

Bezeichner für einen Benutzer. Einer Benutzerkennung können ein ->*Passwort* (zur ->*Zugangskontrolle*) und Zugriffsrechte (->*Zugriffskontrolle*) zugeordnet werden.

Berechtigungsprüfung

siehe ->*Zugangskontrolle*.

Browser

Programm, das zum Abrufen und Darstellen von ->*HTML*-Seiten erforderlich ist. Browser sind z.B. Microsoft Internet Explorer oder Mozilla Firefox.

Browser-Plattform

Betriebssystem des Rechners, auf dem ein ->*Browser* als Client für WebTransactions läuft.

Browserdarstellungs-Druck

Beim Browserdarstellungs-Druck von WebTransactions wird die im ->*Browser* dargestellte Information ausgedruckt.

Capture-Verfahren

Damit WebTransactions in der Ablaufphase die empfangenen ->*Formate* identifizieren kann, können Sie während einer ->*Sitzung* in WebLab für jedes Format einen bestimmten Bereich markieren und das Format benennen. Der Formatname und das ->*Erkennungskriterium* werden in der ->*Capture-Datenbank* gespeichert. Für das Format wird ein ->*Template* unter gleichem Namen generiert. Das Capture-Verfahren ist die Grundlage für die Bearbeitung formatspezifischer Templates für die Liefereinheiten WebTransactions for OSD und MVS.

Capture-Datenbank

Die Capture-Datenbank von WebTransactions enthält alle Formatnamen und die zugehörigen ->*Erkennungskriterien*, die mit dem ->*Capture-Verfahren* erzeugt wurden. Reihenfolge und Erkennungskriterien der Formate können mit WebLab bearbeitet werden.

CGI

(Common Gateway Interface)

Normierte Schnittstelle für den Programmaufruf auf ->*Web-Servern*. Im Gegensatz zur statischen Ausgabe einer zuvor festgelegten ->*HTML-Seite* ermöglicht diese Schnittstelle den dynamischen Aufbau von HTML-Seiten.

Client

Anforderer und Nutzer von Diensten.

Cluster

Menge von identischen ->*WebTransactions-Anwendungen* auf verschiedenen Servern, die zu einem Lastverbund zusammengeschlossen sind.

Dämon

Bezeichnung für einen Prozesstyp in Unix-/POSIX-Systemen, der keine Ein-/Ausgaben auf Terminals durchführt und im Hintergrund abläuft.

Datentyp

Festlegung, wie der Inhalt eines Speicherplatzes zu interpretieren ist. Ein Datentyp hat einen Namen, eine Menge zulässiger Werte (Wertebereich) und eine bestimmte Anzahl von Operationen, die die Werte dieses Datentyps interpretieren und manipulieren.

Dialog

Beschreibt die gesamte Kommunikation zwischen Browser, WebTransactions und *->Host-Anwendung*. Er umfasst in der Regel mehrere *->Dialogzyklen*. Bei WebTransactions werden mehrere Dialogarten unterschieden:

- *->passiver Dialog*
- *->aktiver Dialog*
- *->synchronisierter Dialog*
- *->nicht-synchronisierter Dialog*

Dialogzyklus

Zyklus, der beim Ablauf einer *->WebTransactions-Anwendung* folgende Schritte umfasst:

- eine *->HTML-Seite* aufbauen und an den *->Browser* schicken
- auf Antwort vom Browser warten
- Antwortfelder auswerten und evtl. zur Weiterverarbeitung an die *->Host-Anwendung* schicken

Während des Ablaufs einer *->WebTransactions-Anwendung* werden mehrere Dialogzyklen durchlaufen.

Distinguished Name

Der Distinguished Name (DN) in *->LDAP* setzt sich hierarchisch aus mehreren Teilen zusammen (z.B. „Land, unterhalb von Land: Organisation, unterhalb von Organisation: Organisationseinheit, darunter: Gebräuchlicher Name“). Die Summe dieser Teile identifiziert ein Objekt innerhalb des Directory-Baums eindeutig.

Durch diese Hierarchie wird die eindeutige Benennung von Objekten selbst in einem weltweiten Directory-Baum sehr einfach:

- Der DN "Land=DE/Name=Emil Mustermann" reduziert das Eindeutigkeits-Problem auf das Land DE.
- Der DN "Organisation=FTS/Name=Emil Mustermann" reduziert es auf die Organisation FTS.
- Der DN "Land=DE/Organisation=FTS/Name=Emil Mustermann" reduziert es auf die Organisation FTS innerhalb des Landes DE.

Dokumentenverzeichnis

Verzeichnis des ->*Web-Servers*, in dem Dokumente liegen, auf die über das Netz zugegriffen werden kann. WebTransactions legt in diesem Verzeichnis Dateien zum Herunterladen ab, wie z.B. den WebLab-Client oder allgemeine Start-Seiten.

Domain Name Service (DNS)

Verfahren zur symbolischen Adressierung von Rechnern in Netzen. Bestimmte Rechner im Netz, die DNS- oder Name-Server, führen eine Datenbank mit allen bekannten Rechnernamen und IP-Nummern in ihrer Umgebung.

Dynamische Daten

Werden in WebTransactions durch das WebTransactions-Objektmodell abgebildet, z.B. als ->*Systemobjekt*, Host-Objekt oder Nutzereingaben am Browser.

Eigenschaft

Definiert die Beschaffenheit von ->*Objekten*, z.B. könnten Kundename und Kundennummer Eigenschaften eines Objekts „Kunde“ sein. Diese Eigenschaften können innerhalb des Programms gesetzt, abgefragt und verändert werden.

EJB

(Enterprise JavaBean)

Industriestandard auf Basis von Java, mit dem innerhalb einer verteilten, objektorientierten Umgebung selbstentwickelte oder auf dem Markt gekaufte Server-Komponenten zur Erstellung von verteilten Programmsystemen genutzt werden können.

EHLAPI

(Enhanced High Level Language API)

Programmschnittstelle z.B. von Terminal-Emulationen für die Kommunikation mit der SNA-Welt. Auf dieser Schnittstelle basiert die Kommunikation zwischen Transit-Client und dem SNA-Rechner, die über das Produkt TRANSIT abgewickelt wird.

Erkennungskriterium

Über Erkennungskriterien werden ->*Formate* einer ->*Terminal-Anwendung* identifiziert und Sie können auf die Daten des Formats zugreifen. Als Erkennungskriterium wählen Sie jeweils einen oder auch mehrere Bereiche des Formats, deren Inhalt das Format eindeutig identifiziert.

Felddatei (*.fld-Datei)

Enthält in WebTransactions die Struktur des Datensatzes eines ->*Formats* (Metadaten).

FHS

(Format **H**andling **S**ystem)
Formatierungssystem für BS2000/OSD-Anwendungen.

Field

Kleinster Teil eines ->*Service* und Element eines ->*Records* oder ->*Puffers*.

Filter

Programm oder Programmteil (z.B. eine Bibliothek) zur Umsetzung eines Formats in ein anderes (z.B. XML-Dokumente in ->*WTScript*-Datenstrukturen).

Format

Optische Darstellung auf alphanumerischen Bildschirmen, wird auch Maske oder Schirm genannt.

In WebTransactions wird ein Format jeweils durch eine ->*Felddatei* und ein Template repräsentiert.

Formatbeschreibungquellen

Beschreibung mehrerer ->*Formate* in einer oder mehreren Dateien, die aus einer Format-Bibliothek (FHS/IFG) erzeugt wurden oder direkt am ->*Host* vorliegen für die Nutzung „sprechender“ Namen in Formaten.

Formattyp

(nur relevant bei FHS-Anwendungen und Kommunikation über UPIC)
Spezifiziert den Typ des Formats: #Format, +Format, -Format oder *Format.

Funktion

Benutzerdefinierte Code-Teile mit einem Namen und ->*Parametern*. Durch eine Beschreibung der Funktionsschnittstelle (oder Signatur) können Funktionen in Methoden aufgerufen werden.

Holder Task

Prozess, Task oder Thread in WebTransactions, je nach Betriebssystem-Plattform. Die Anzahl der Tasks entspricht der Anzahl der Benutzer. Die Task wird beendet, wenn sich der Benutzer abmeldet oder durch Timeout. Ein Holder Task entspricht genau einer ->*WebTransactions-Sitzung*.

Host

Rechner, auf dem die ->*Host-Anwendung* läuft.

Host-Adapter

Dienen dazu, bestehende ->*Host-Anwendungen* an WebTransactions anzuschließen. Sie sorgen zur Laufzeit z.B. für den Auf- und Abbau von Verbindungen und für die Umsetzung der ausgetauschten Daten.

Host-Anwendung

Anwendung, die mit WebTransactions integriert ist.

Host-Plattform

Betriebssystem des Rechners, auf dem die ->*Host-Anwendung* läuft.

Host-Daten-Druck

Beim Host-Daten-Druck von WebTransactions werden Informationen ausgedruckt, die von der ->*Host-Anwendung* aufbereitet und gesendet wurden, z.B. Ausdruck von Host-Dateien.

Host-Datenobjekt

Bezeichnet in WebTransactions ein ->*Objekt* der Datenschnittstelle zur ->*Host-Anwendung*, das ein Feld mit all seinen Feldattributen repräsentiert. Es wird von WebTransactions nach dem Empfang von Daten der Host-Anwendung angelegt und existiert bis zum nächsten Datenempfang oder bis zum Beenden der ->*Sitzung*.

Host-Steuerobjekt

In WebTransactions enthalten Host-Steuerobjekte Informationen, die nicht nur ein einzelnes Feld betreffen, sondern das gesamte ->*Format*. Dazu gehören z.B. das Feld, in dem sich der Cursor befindet, die aktuelle Funktionstaste oder globale Formatattribute.

HTML

(Hypertext Markup Language)
Siehe ->*Hypertext Markup Language*

HTTP

(Hypertext Transfer Protocol)
Protokoll zur Übertragung von ->*HTML*-Seiten und Daten.

HTTPS

(Hypertext Transfer Protocol Secure)
Protokoll zur gesicherten Übertragung von ->*HTML*-Seiten und Daten.

Hypertext

Dokument mit Verweisen auf andere Stellen im gleichen oder in anderen Dokumenten, in die z.B. durch Anklicken mit der Maus gesprungen werden kann.

Hypertext Markup Language

Standardisierte Auszeichnungssprache für Dokumente im WWW.

JavaBean

Java-Programm (oder ->*Klasse*) mit genau festgelegten Konventionen für die Schnittstellen, die eine Wiederverwendung in mehreren Anwendungen ermöglichen.

KDCDEF

openUTM-Werkzeug für die Generierung von ->*openUTM-Anwendungen*.

Klasse

Enthält die Definition der ->*Eigenschaften* und ->*Methoden* eines ->*Objekts*. Sie ist das Modell für die Instanziierung von Objekten und definiert deren Schnittstellen.

Klassen-Template

Ein Klassen-Template in WebTransactions enthält für die gesamte Objektklasse (z. B. Eingabe- oder Ausgabefeld) gültige, immer wiederkehrende Anweisungen. Klassen-Templates werden durchlaufen, wenn auf ein ->*Host-Datenobjekt* der ->*Auswertungoperator* oder die `toString`-Methode angewendet wird.

Kommunikationsobjekt

Steuert eine Verbindung zu einer ->*Host-Anwendung* und enthält Information über den aktuellen Zustand der Verbindung, über die zuletzt empfangenen Daten etc.

Konvertierungswerkzeuge

Dienstprogramme, die mit WebTransactions ausgeliefert werden. Mit den Konvertierungswerkzeugen werden die Datenstrukturen von ->*openUTM-Anwendungen* analysiert und in Dateien abgelegt. Diese Dateien können Sie dann in WebLab als ->*Formatbeschreibungsqellen* verwenden, um WTML-Templates und ->*FLD-Dateien* zu generieren.

Die Basis für die Konvertierung können Cobol-Datenstrukturen oder IFG-Formatbibliotheken sein. Für Drive-Programme wird das Konvertierungswerkzeug mit dem Produkt Drive ausgeliefert.

LDAP

(Lightweight **D**irectory **A**ccess **P**rotocol)

Der X.500-Standard definiert als Zugriffsprotokoll DAP (Directory Access Protocol). Speziell für den Zugang zu X.500-Verzeichnisdiensten vom PC aus hat sich jedoch der Internet-Standard LDAP durchgesetzt.

Bei LDAP handelt es sich um ein vereinfachtes DAP-Protokoll, das nicht alle Möglichkeiten von DAP zulässt und mit DAP nicht kompatibel ist. Praktisch alle X.500-Verzeichnisdienste unterstützen neben DAP auch LDAP. In der Praxis kann es zu Verständigungsproblemen kommen, da es diverse Dialekte von LDAP gibt. Die Unterschiede der Dialekte sind in der Regel gering.

Literal

Zeichenfolge, die einen festen Wert repräsentiert. Literale dienen dazu, in Source-Programmen konstante Werte unmittelbar anzugeben („wörtliche“ Wertangabe).

Master-Template

WebTransactions-Template, das als Schablone für die Generierung der Automask und der formatspezifischen-Templates verwendet wird.

Message Queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete Warteschlangen (Queues) ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen, die Übermittlung der Nachricht wird garantiert, unabhängig davon, ob gerade eine Netzverbindung besteht oder nicht.

Methode

Objektorientierter Begriff für ->*Funktion*. Eine Methode wirkt auf das ->*Objekt*, in dem sie definiert ist

Modul-Template

Dient in WebTransactions dazu, ->*Klassen*, ->*Funktionen* und Konstanten global für eine komplette ->*Sitzung* zu definieren. Ein Modul-Template wird mit Hilfe der Funktion `import()` geladen.

MT-Tag

(Master-Template-Tag)

Spezielle Tags in ->*Master-Templates* für die dynamischen Teile eines Master-Templates.

Multi-Tier-Architektur

Allen Client-/Server-Architekturen liegt eine Gliederung in einzelne Software-Komponenten, auch Schichten oder Tiers genannt, zugrunde: Man spricht von 1-Tier, 2-Tier-, 3-Tier und auch von Multi-Tier-Modellen. Man kann die Aufgliederung auf der physischen oder der logischen Ebene betrachten:

- Logische Software-Tiers liegen vor, wenn die Software in modulare Komponenten mit klaren Schnittstellen gegliedert ist.
- Physische Tiers liegen dann vor, wenn die (logischen) Softwarekomponenten im Netz auf verschiedene Rechner verteilt sind.

Mit WebTransactions sind Multi-Tier-Modelle sowohl auf physischer als auch logischer Tiers-Ebene möglich.

Name/Value-Paar

In den vom ->*Browser* geschickten Daten die Kombination z.B. von einem ->*HTML*-Eingabefeldnamen mit seinem Wert.

nicht-synchronisierter Dialog

Der nicht-synchronisierte Dialog von WebTransactions erlaubt es, den Prüfmechanismus des ->*synchronisierten Dialogs* zeitweise auszuschalten. So lassen sich ->*Dialoge* zwischenschieben, die außerhalb des synchronisierten Dialogs liegen und keinen Einfluss auf den logischen Zustand der ->*Host-Anwendung* haben. Dadurch können Sie z.B. in einer ->*HTML*-Seite eine Schaltfläche anbieten, um Hilfeinformationen aus der laufenden Host-Anwendung anzufordern und in einem separaten Fenster anzuzeigen.

Objekt

Elementare Einheit innerhalb eines objektorientierten Softwaresystems. Jedes Objekt hat einen Namen, über den es angesprochen werden kann, ->*Attribute*, die seinen Zustand definieren und ->*Methoden*, die auf das Objekt angewandt werden können.

openUTM

(Universal Transaction Monitor)

Transaktionsmonitor von Fujitsu Technology Solutions, verfügbar für BS2000/OSD, verschiedenste Unix- und Windows-Plattformen.

openUTM-Anwendung

->*Host-Anwendung*, die Dienstleistungen zur Verfügung stellt, die Aufträge von Terminals, ->*Client-Programmen* oder anderen Host-Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine openUTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Einheit bildet.

Mit openUTM-Anwendungen kann sowohl über das Client/Server-Protokoll ->*UPIC* als auch über die Terminal-Schnittstelle (9750) kommuniziert werden.

openUTM-Client (UPIC)

Mit dem Produkt openUTM-Client (UPIC) können Sie Client-Programme für openUTM erstellen. openUTM-Client (UPIC) steht z.B. für Unix-, BS2000/OSD- und Windows-Plattformen zur Verfügung.

openUTM-Teilprogramm

Die Dienste einer ->*openUTM-Anwendung* werden durch ein oder mehrere openUTM-Teilprogramme realisiert. Sie sind über ->*Transaktionscodes* ansprechbar und enthalten spezielle openUTM-Funktionsaufrufe (z.B. KDCS-Aufrufe).

Parameter

Daten, die an eine ->*Funktion* oder ->*Methode* zur Verarbeitung übergeben werden (Eingabeparameter) oder Daten, die als Ergebnis von einer Funktion oder Methode zurückgeliefert werden (Ausgabeparameter).

passiver Dialog

Beim passiven Dialog von WebTransactions wird der Dialogablauf von der ->*Host-Anwendung* gesteuert, d.h., die Host-Anwendung bestimmt das nächste zu verarbeitende ->*Template*. Ein Anwender, der über WebTransactions auf die Host-Anwendung zugreift, durchläuft die gleichen Schritte wie beim Zugriff über ein Terminal. Passive Dialogsteuerung verwendet WebTransactions bei einer automatischen Umsetzung der Host-Anwendung oder wenn jedes Format der Host-Anwendung genau einem individuellen Template entspricht.

Passwort

In einer Anwendung für eine ->*Benutzererkennung* eingetragene Zeichenkette zur Authentisierung (->*Zugangsschutz*).

polling

Zyklische Abfrage auf Zustandsänderungen.

Pool

WebTransactions bezeichnet hiermit ein freigegebenes Verzeichnis, in dem WebLab ->*Basisverzeichnisse* anlegen und pflegen kann. Den Zugriff auf dieses Verzeichnis steuern Sie mit der Administration.

Posted-Objekt (wt_Posted)

Enthält in WebTransactions eine Liste der vom ->*Browser* zurückgeschickten Daten. Dieses ->*Objekt* wird von WebTransactions angelegt und lebt nur für die Dauer eines ->*Dialogzyklus*.

posten

Daten versenden

Projekt

Enthält in der WebTransactions-Entwicklungsumgebung verschiedene Einstellungen einer ->*WebTransactions-Anwendung*, die in einer Projektdatei (Endung .wtp) gespeichert werden. Sie sollten für jede WebTransactions-Anwendung, die Sie entwickeln, ein Projekt anlegen und zum Bearbeiten immer dieses Projekt öffnen.

Protokoll

Vereinbarungen über Verhaltensregeln und Formate bei der Kommunikation unter entfernten Partnern gleichen logischen Niveaus.

Protokolldatei

- openUTM-Client: Datei, in die bei abnormalem Beenden einer Conversation openUTM-Fehlermeldungen geschrieben werden.
- In WebTransactions werden Protokolldateien als Trace-Dateien bezeichnet.

Prozess

Der Begriff „Prozess“ wird als Oberbegriff für Prozess (Solaris, Linux und Windows) und Task (BS2000/OSD) verwendet.

Puffer

Definition eines Datensatzes, der von einem ->*Service* übertragen wird. Der Puffer dient zum Senden und zum Empfangen von Nachrichten. Zusätzlich gibt es einen speziellen Puffer für die Ablage der ->*Erkennungskriterien* und für Daten zur Darstellung am Bildschirm.

Roaming Session

->*WebTransactions-Sitzung*, die nacheinander oder gleichzeitig von verschiedenen ->*Clients* aus angesprochen werden kann.

Record

Definition eines Datensatzes, der in einem ->*Puffer* übertragen wird. Er beschreibt einen Teil des Puffers, der ein- oder mehrfach vorkommen kann.

Service-Anwendung

->*WebTransactions-Sitzung*, die abwechselnd von verschiedenen Benutzern aufgerufen werden kann.

Service-Knoten

Instanz eines ->*Service*. Beim Entwickeln und beim Ablauf einer ->*Methode* kann ein Service mehrfach instanziiert werden. Beim Modellieren und Code bearbeiten werden diese Instanzen als Service-Knoten bezeichnet.

Sichtbarkeit von Variablen

->*Objekte* und ->*Variablen* unterschiedlicher Dialogarten werden von WebTransactions in unterschiedlichen Adressräumen verwaltet. Das bedeutet, dass Variablen eines ->*synchronen Dialogs* im ->*asynchronen Dialog* oder im Dialog mit einer entfernten Anwendung nicht sichtbar und damit auch nicht zugreifbar sind.

Sitzung

Beginnt ein Endanwender mit einer ->*WebTransactions-Anwendung* zu arbeiten, so wird für ihn auf dem WebTransactions-Server eine WebTransactions-Sitzung eingerichtet. Diese Sitzung enthält alle für diesen Benutzer geöffneten Verbindungen zum ->*Browser*, zu speziellen ->*Clients* und ->*Hosts*.

Eine Sitzung kann gestartet werden

- durch Eingabe eines URL von WebTransactions im Browser.
- durch die Methode `START_SESSION` der Client/Server-Schnittstelle `WT_REMOTE`.

Eine Sitzung endet

- mit einer entsprechenden Eingabe des Benutzers im Ausgabebereich dieser ->*WebTransactions-Anwendung* (nicht über Standard-Buttons des Browsers).
- durch Überschreiten der konfigurierten Zeit, die WebTransactions auf eine Antwort von der ->*Host-Anwendung* oder vom ->*Browser* wartet.
- durch Terminierung mit Hilfe der WebTransactions-Administration.
- durch die Methode `EXIT_SESSION` der Client/Server-Schnittstelle `WT_REMOTE`.

Eine WebTransactions-Sitzung ist eindeutig durch eine ->*WebTransactions-Anwendung* und eine Session Id bestimmt. Während ihrer Lebensdauer existiert zu jeder WebTransactions-Sitzung auf dem WebTransactions-Server genau ein ->*Holder Task*.

Skalar

->*Variable*, die nur aus einem einzelnen Wert besteht - im Gegensatz zu einer ->*Klasse*, einem ->*Array* oder einer anderen komplexen Datenstruktur.

SOAP

(ursprünglich **S**imple **O**bject **A**ccess **P**rotocol)

Das ->*XML*-basierte SOAP-Protocol realisiert einen einfachen und transparenten Mechanismus, mit dem strukturierte und typisierte Informationen zwischen Rechnern in einer dezentralisierten, verteilten Umgebung ausgetauscht werden können.

SOAP stellt ein modulares Paketmodell sowie Mechanismen zum Verschlüsseln von Daten innerhalb von Modulen zur Verfügung. Dies ermöglicht die unkomplizierte Beschreibung der externen Schnittstellen eines ->*Web-Service*.

Stil

Realisiert in WebTransactions ein anderes Layout für ein ->*Template*, z.B. mit mehr oder weniger Grafikelementen für unterschiedliche ->*Browser*. Der Stil kann während einer ->*Sitzung* jederzeit geändert werden.

synchronisierter Dialog

Beim synchronisierten Dialog (Standardfall) überprüft WebTransactions automatisch, ob die Daten, die vom Web-Browser eingehen, auch wirklich die Antwort auf die letzte an den ->*Browser* geschickte ->*HTML*-Seite sind. Wenn z.B. der Anwender am Web-Browser über die Schaltfläche **Zurück** oder die History-Funktion zu einer „alten“ HTML-Seite der aktuellen ->*Sitzung* wechselt und diese zurückschickt, erkennt WebTransactions, dass die Daten nicht zum aktuellen ->*Dialogzyklus* passen und reagiert mit einer Fehlermeldung. Die zuletzt an den Browser gesendete Seite wird automatisch erneut an den Browser geschickt.

Systemobjekt (wt_System)

Das Systemobjekt von WebTransactions enthält ->*Variablen*, die während einer gesamten ->*Sitzung* existieren und erst am Ende einer Sitzung oder durch explizites Löschen wieder entfernt werden. Es ist immer sichtbar und identisch für alle Namensräume.

TAC

Siehe ->*Transaktionscode*

Tag

->*HTML*-, ->*XML*- und ->*WTML*-Dokumente bestehen aus Tags und dem eigentlichen Inhalt. Mit den Tags werden Auszeichnungen im Dokument durchgeführt z.B. Überschriften, Texthervorhebungen (fett, kursiv) oder Quellangaben für Grafikdateien.

TCP/IP

(Transport **C**ontrol **P**rotocol/**I**nternet **P**rotocol)

Sammelname für eine Protokollfamilie in Rechnernetzen, die unter anderem im Internet verwendet wird.

Template

Vorlage für die Generierung von spezifischem Code. Ein Template enthält feste Teile, die bei der Generierung unverändert übernommen werden und variable Teile, die bei der Generierung durch die jeweils aktuellen Werte ersetzt werden. Ein Template ist eine ->WTML-Datei mit speziellen Tags zur Steuerung der dynamischen Generierung einer ->HTML-Seite und zur Verarbeitung der am ->Browser eingegebenen Werte. Es können mehrere Sätze von Templates parallel gehalten werden. Diese repräsentieren unterschiedliche Stile (z.B. viel/wenig Grafik, Java-Benutzung etc.).

WebTransactions nutzt verschiedene Arten von Templates:

- ->Automask-Templates für die automatische Umsetzung der ->Formate von MVS- und OSD-Anwendungen
- eigene Templates, die vom Programmierer selbst geschrieben werden, z.B. zur Steuerung eines ->aktiven Dialogs
- formatspezifische Templates, die für eine spätere Nachbearbeitung generiert werden
- Include-Templates, die in andere Templates eingefügt werden
- ->Klassen-Templates
- ->Master-Templates für ein einheitliches Layout fester Bereiche bei der Generierung der Automask und formatspezifischer Templates
- Start-Template, das als erstes Template einer WebTransactions-Anwendung durchlaufen wird

Template-Objekte

->Variablen zur Zwischenspeicherung von Werten für einen ->Dialogzyklus in WebTransactions.

Terminal-Anwendung

Anwendung auf einem ->Host-Rechner, auf die über die 9750- oder 3270-Schnittstelle zugegriffen wird.

Terminal-Hardcopy-Druck

Beim Terminal-Hardcopy-Druck von WebTransactions wird die alphanumerische Darstellung des ->Formats gedruckt, wie es von einem Terminal oder einer Terminal-Emulation dargestellt würde.

Transaktion

Verarbeitungsschritt zwischen zwei Sicherungspunkten (innerhalb eines Vorgangs), der durch die ACID-Bedingungen gekennzeichnet ist (**A**tomicity, **C**onsistency, **I**solation und **D**urability). Die in einer Transaktion beabsichtigten Änderungen an der Anwenderinformation werden entweder alle oder gar nicht durchgeführt (Alles-oder-Nichts-Regel).

Transaktionscode/TAC

Name, über den ein openUTM-Vorgang oder ein ->*openUTM-Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem openUTM-Teilprogramm bei der openUTM-Konfigurierung zugeordnet. Einem Teilprogramm können auch mehrere TACs zugeordnet sein.

UDDI

(**U**niversal **D**escription, **D**iscovery and **I**ntegration)

Umfasst Verzeichnisse, die Beschreibungen von ->*Web-Services* enthalten. Diese Informationen stehen Web-Usern allgemein zur Verfügung.

Unicode

Von der International Standardisation Organisation (ISO) und dem Unicode-Konsortium genormter alphanumerischer Zeichensatz zur Codierung von Zeichen – Buchstaben, Ziffern, Satzzeichen, Silbenzeichen, Sonderzeichen sowie Ideogrammen. Unicode fasst alle weltweit bekannten Textzeichen in einem einzigen Zeichensatz zusammen.

Unicode ist hersteller- und systemunabhängig. Es verwendet Zeichensätze der Länge zwei oder vier Bytes für die Codierung jedes Textzeichens. Diese Zeichensätze werden bei ISO als UCS-2 (Universal Character Set 2) beziehungsweise UCS-4 bezeichnet. Statt der durch ISO definierten Bezeichnung UCS-2 wird häufig die Bezeichnung UTF-16 (Unicode Transformation Format 16 Bit) verwendet, ein vom Unicode-Konsortium definierter Standard.

Neben der Nutzung von UTF-16 ist auch der Einsatz von UTF-8 (Unicode Transformation Format 8 Bit) weit verbreitet. UTF-8 ist inzwischen die globale Zeichen-Codierung im Internet.

UPIC

(**U**niversal **P**rogramming **I**nterface for **C**ommunication)

Trägersystem für openUTM-Clients, das über die X/Open-Schnittstelle CPI-C die Client-Server-Kommunikation zwischen CPI-C-Client-Anwendung und der openUTM-Anwendung ermöglicht.

URI

(**U**niform **R**esource **I**dentifier)

Oberbegriff für alle Namen und Adressen die im Internet Objekte referenzieren. Die allgemein gebräuchlichen URIs sind ->*URLs*.

URL

(Uniform Resource Locator)

Beschreibung von Ort und Zugriffsart einer Ressource im Internet.

Userexit

In C/C++ implementierte Funktion, die der Programmierer aus einem ->*Template* aufruft.

Variable

Speicherplatz für variable Werte, der einen Namen und einen ->*Datentyp* benötigt.

Vorgang

In ->*openUTM* Bearbeitung eines Auftrags durch eine ->*openUTM-Anwendung*. Es gibt Dialog-Vorgänge und Asynchronvorgänge. Dem Vorgang werden von openUTM eigene Speicherbereiche zugeordnet. Ein Vorgang setzt sich aus einer oder mehreren ->*Transaktionen* zusammen.

Web-Server

Rechner und Software zum Bereitstellen von ->*HTML*-Seiten und dynamischen Daten über ->*HTTP*.

Web-Service

Dienst, der im Internet bereitgestellt wird, z.B. ein Währungsumrechnungs-Programm, und über das SOAP-Protokoll angesprochen werden kann. Die Schnittstelle eines Web-Service ist in ->*WSDL* beschrieben.

WebTransactions-Anwendung

Anwendung, die ->*Host-Anwendungen* für den Internet-/Intranet-Zugriff integriert. Eine WebTransactions-Anwendung besteht aus

- einem ->*Basisverzeichnis*
- einem Start-Template
- den ->*Templates*, die die Umsetzung zwischen ->*Host* und ->*Browser* steuern
- protokollspezifischen Konfigurationsdateien

WebTransactions-Plattform

Betriebssystem des Rechners, auf dem WebTransactions läuft.

WebTransactions-Server

Rechner, auf dem WebTransactions läuft.

WebTransactions-Sitzung

Siehe ->*Sitzung*.

WSDL

(Web Services Description Language)

Bietet ->*XML*-Sprachregeln für die Beschreibung von ->*Web-Services*. Ein Web-Service wird dabei durch eine Auswahl von Ports definiert.

WTBean

In WebTransactions werden ->*WTML*-Komponenten mit selbstbeschreibender Schnittstelle als WTBeans bezeichnet. Es wird zwischen inline und standalone WTBeans unterschieden:

- ein inline WTBean entspricht einem Teil eines WTML-Dokuments
- ein standalone WTBean ist ein eigenständiges WTML-Dokument

Verschiedene WTBeans gehören zum Produktumfang von WebTransactions, weitere WTBeans stehen Ihnen auf der WebTransactions-Homepage zum Download zur Verfügung:

ts.fujitsu.com/products/software/openseas/webtransactions.html

WTML

(WebTransactions Markup Language)

Auszeichnungs- und Programmiersprache für WebTransactions ->*Templates*. WTML besteht aus ->*WTML-Tags*, die ->*HTML* erweitern, und der server-seitigen Programmiersprache ->*WTScript*, die z.B. den Datenaustausch mit ->*Host-Anwendungen* ermöglicht. WTML wird von WebTransactions und nicht vom ->*Browser* ausgeführt (serverside scripting).

WTML-Tag

(WebTransactions Markup Language-Tag)

Spezielle Tags von WebTransactions zur Generierung der dynamischen Teile einer ->*HTML*-Seite mit Daten aus der ->*Host-Anwendung*.

WTScript

Server-seitige Programmiersprache von WebTransactions. WTScripts stehen ähnlich wie client-seitige JavaScripts in Bereichen, die mit speziellen Tags eingeleitet und beendet werden. Statt ->*HTML-SCRIPT*-Tags verwenden Sie hierfür jedoch ->*WTML-Tags*: `wtOnCreateScript` und `wtOnReceiveScript`. Damit zeigen Sie an, dass diese Scripts von WebTransactions und nicht vom ->*Browser* ausgeführt werden sollen und signalisieren zusätzlich den gewünschten Ausführungszeitpunkt. `OnCreate`-Scripts werden ausgeführt, bevor die Seite an den Browser geschickt wird. `OnReceive`-Scripts werden erst ausgeführt, nachdem die Antwort vom Browser empfangen wurde.

XML

(e**X**tensible **M**arkup **L**anguage)

Definiert eine Sprache zur logischen Strukturierung von Dokumenten mit dem Ziel, diese einfach zwischen verschiedenen Anwendungen auszutauschen.

XML-Schema

Ein XML-Schema im allgemeinen Sinn definiert die zulässigen Elemente und Attribute einer XML-Beschreibung. XML-Schemata können verschiedene Formate haben, z.B. DTD (**D**ocument **T**ype **D**efinition), XML Schema (**W**3**C**-Standard) oder XDR (**X**ML **D**ata **R**educed).

Zugangskontrolle

Prüfung, ob ein Benutzer berechtigt ist, unter einer bestimmten Benutzerkennung mit der Anwendung zu arbeiten.

Zugriffskontrolle

Überwachung der Zugriffe auf die Daten und ->*Objekte* einer Anwendung.

Abkürzungen

BO	B usiness O bject
CGI	C ommon G ateway I nterface
DN	D istinguished N ame
DNS	D omain N ame S ervice
EJB	E nterprise J ava B ean
FHS	F ormat H andling S ystem
HTML	H ypertext M arkup L anguage
HTTP	H ypertext T ransfer P rotocol
HTTPS	H ypertext T ransfer P rotocol S ecure
IFG	I nteraktiver F ormat G enerator
ISAPI	I nternet S erver A pplication P rogramming I nterface
LDAP	L ightweight D irectory A ccess P rotocol
LPD	L ine P rinter D aemon
MT-Tag	M aster- T emplate- T ag
MVS	M ultiple V irtual S torage
OSD	O pen S ystems D irection
SGML	S tandard G eneralized M arkup L anguage
SOAP	S imple O bject A ccess P rotocol

Abkürzungen

SSL	S ecure S ocket L ayer
TCP/IP	T ransport C ontrol P rotocol/ I nternet P rotocol
Upic	U niversal P rogramming I nterface for C ommunication
URL	U niform R esource L ocator
WSDL	W eb S ervices D escription L anguage
wtc	W eb T ransactions C omponent
WTML	W eb T ransactions M arkup L anguage
XML	e Xtensible M arkup L anguage

Literatur

WebTransactions-Handbücher

Unter der Web-Adresse <http://manuals.ts.fujitsu.com> stehen Ihnen sämtliche Handbücher zum Download zur Verfügung.

WebTransactions
Konzepte und Funktionen
Einführung

WebTransactions
Template-Sprache
Referenzhandbuch

WebTransactions
Client-APIs für WebTransactions
Benutzerhandbuch

WebTransactions
Anschluss an OSD-Anwendungen
Benutzerhandbuch

WebTransactions
Anschluss an MVS-Anwendungen
Benutzerhandbuch

WebTransactions
Zugriff auf dynamische Web-Inhalte
Benutzerhandbuch

WebTransactions
Web-Frontend für Web-Services
Benutzerhandbuch

Sonstige Handbücher

Die Handbücher sind online unter <http://manuals.ts.fujitsu.com> zu finden oder in gedruckter Form gegen gesondertes Entgelt unter <http://manualshop.ts.fujitsu.com> zu bestellen.

openUTM

Konzepte und Funktionen

Benutzerhandbuch

openUTM

Anwendungen programmieren mit KDCS für COBOL, C und C++

Basishandbuch

openUTM

Anwendungen generieren

Benutzerhandbuch

openUTM

Anwendungen administrieren

Benutzerhandbuch

FHS (BS2000/OSD)

Formatierungssystem für openUTM, TIAM, DCAM

Benutzerhandbuch

Stichwörter

\$FIRST (Host-Steuerobjekt) [167](#)

\$NEXT (Host-Steuerobjekt) [167](#)

*.fld-Datei siehe Felddatei

*.htm-Datei siehe Template

A

Abspeichern

 Bilddatei [120](#)

Aktiver Dialog [203, 206](#)

Aktualisieren

 Basisverzeichnis [70](#)

Align (Felddatei) [85](#)

Align (Host-Datenobjekt-Attribut) [156](#)

Ändern

 Passwort [193](#)

Anlegen

 Projekt [35](#)

Anmeldung (bei openUTM) [192](#)

 über bestimmten LTERM-Partner [201](#)

Anschlusspunkt (LTERM) [201](#)

Anzeigen

 Bilddatei [120](#)

APPLICATION_NAME (Systemobjekt-Attribut) [129, 142, 149](#)

APPLICATION_PREFIX (Systemobjekt-Attribut) [142, 149](#)

Architektur

 WebTransactions [9](#)

Array [203](#)

ASCII-EBCDIC-Konvertierung [123, 157](#)

Assistent [99](#)

Asynchrone Nachricht [203](#)

Attribut [203](#)

 WT_HOST_GLOBALS [166](#)

 WT_HOST_MESSAGE [162](#)

AttributeCombination [84](#)

AttributeLength (Feldattribut-Block) [84](#)

AttributeLength (Globalattribut-Block) [83](#)

AttributOffset (Felddatei) [85](#)

AttributOffsets Section (Felddatei) [83](#)

Aufrufen

 Start-Template [67](#)

Aufrufseite [204](#)

Ausdruck [204](#)

Auswertungsoperator [204](#)

AutoInput (Felddatei) [85](#)

AutoInput (Host-Datenobjekt-Attribut) [156](#)

Automask-Template [204](#)

Automatischer Wiederanlauf [146](#)

B

BackgroundColor (Felddatei) [83](#)

BackgroundColor (WT_HOST_MESSAGE-Attribut) [162](#)

BADTAC (Systemobjekt-Attribut) [142, 197](#)

BADTAC Event-Service [197](#)

BaseVariant (Felddatei) [83](#)

Basisdatentyp [203](#)

Basisformat [187](#)

Basisverzeichnis [204](#)

 auf eine neue Version umstellen [70](#)

 Beispiel [36](#)

 localapps [71](#)

 upicfile [71](#)

BCAM-Applikationsname [204](#)

BCAMAPPL [204](#)

Beispiel

 Travel Agency [25](#)

Benutzerkennung [192, 204](#)

 ungültig [152, 193](#)

Berechtigungsprüfung siehe Zugangskontrolle

Bilddatei anzeigen [120](#)

Binärdaten [120](#)

Blink (Felddatei) [85](#)

Blink (Host-Datenobjekt-Attribut) [156](#)

Blinking (Host-Datenobjekt-Attribut) [157](#)

Browser [204](#)

 Terminal-Funktionen [168](#)

Browser-Plattform [205](#)

Browserdarstellungs-Druck 205
Button einfügen 59

C

Capture-Datenbank 205
Capture-Verfahren 205
Case (Felddatei) 85
Case (Host-Datenobjekt-Attribut) 156
CGI (Common Gateway Interface) 205
CHARSET (Systemobjekt-Attribut) 10, 122, 165
clickable image 61
Client 205
Cluster 205
Color (Felddatei) 84, 85
Color (Host-Datenobjekt-Attribut) 158
Column (Felddatei) 85
COMMUNICATION_FILE_NAME (Systemobjekt-Attribut) 143, 150, 151
COMMUNICATION_INTERFACE_VERSION (Systemobjekt-Attribut) 143
Content-Type (HTTP-Header-Feld) 165
Content_ (WT_HOST_MESSAGE-Attribut) 162, 163
Contents (WT_HOST_MESSAGE-Attribut) 162
CONVERSATION_TAC (Systemobjekt-Attribut) 143
Cursor (Felddatei) 84
Cursor (Host-Datenobjekt-Attribut) 158, 159
Cursor (WT_HOST_GLOBALS-Attribut) 166
CursorControl (Felddatei) 83
CursorControl (WT_HOST_MESSAGE-Attribut) 162
CursorField (WT_HOST_MESSAGE-Attribut) 162, 163
CursorPosition (Felddatei) 83
CursorPosition (WT_HOST_MESSAGE-Attribut) 162
CUT_TAC_FIELD (Systemobjekt-Attribut) 143

D

Dämon 205
DataOffset (Felddatei) 85
DataType (Felddatei) 85
DataType (Host-Datenobjekt-Attribut) 156

DateFormat (Felddatei) 83
DateFormat (WT_HOST_MESSAGE-Attribut) 162
Daten
 dynamisch 207
Datensatzstruktur 207
Datentyp 206
DecimalSeparator (Felddatei) 83
DecimalSeparator (WT_HOST_MESSAGE-Attribut) 162
DefaultCursor (Felddatei) 85
DefaultCursor (Host-Datenobjekt-Attribut) 156
Detect (WT_HOST_GLOBALS-Attribut) 166
Detectable (Felddatei) 85
Detectable (Host-Datenobjekt-Attribut) 157
Diagnose
 Unicode 161
Dialog 206
 aktiv 203, 206
 Arten 206
 nicht synchron 206
 passiv 206
 synchron 206
Dialogzyklus 206
DigitSeparator (Felddatei) 83
DigitSeparator (WT_HOST_MESSAGE-Attribut) 162
DISPLAY_EURO (Systemobjekt-Attribut) 143
DisplayLength (Felddatei) 85
Distinguished Name 206
Dokumentenverzeichnis 207
Domain Name Service (DNS) 207
Drop-Down-Liste 53

E

EditRC (Felddatei) 84
EditState (Felddatei) 84
EditState (Host-Datenobjekt-Attribut) 158, 161
EHLLAPI 207
Eigenschaft 207
Einfügen
 inline WtBean 190
 standalone WtBean 188
EJB 207

- EPILOG (Systemobjekt-Attribut) 144
- Erkennungskriterium 207
- ERROR (Systemobjekt-Attribut) 150
 - bei receive 151
- erstes Template siehe Start-Template
- Erzeugen
 - Start-Template 63
- Event-Service
 - BADTAC 197
 - SIGNON 193
- F**
- Felddatei 73, 207
 - Schlüsselwörter 83
 - Sections 83
 - spezifizieren 144
- Felddateien 10, 122
- Feldname Section (Felddatei) 83
- Festlegen
 - Formularfelder 144
 - Nachspann 144
 - Vorspann 146
- FHS 208
 - Format in Template umwandeln 74
- FHS-Teilformate 100
- Field 208
- FieldLength (Felddatei, Offset) 84
- FieldLength (WT_HOST_GLOBALS-Attribut) 167
- FieldsDetect (Felddatei) 83
- FieldsDetect (WT_HOST_MESSAGE-Attribut) 162
- FieldsMod (Felddatei) 83
- FieldsMod (WT_HOST_MESSAGE-Attribut) 162
- FieldsMod Attribut (Host-Datenobjekt) 161
- FieldsValid (Felddatei) 84
- FieldsValid Attribut (WT_HOST_MESSAGE-Attribut) 162
- FillCharInput (Felddatei) 85
- FillCharOutput (Felddatei) 85
- Filter 208
- FLD (Systemobjekt-Attribut) 144, 150
 - bei receive 151
- fld-Datei 207
- FLD-Dateien 122
- FloatSign (Host-Datenobjekt-Attribut) 156
- Font-Größe 180
- FORMANT 80
 - Teilformate 100
- Format 208
 - #Format 208
 - *Format 208
 - +Format 208
 - Format 208
- FORMAT_SEQ (Systemobjekt-Attribut) 144
 - bei receive 151
- Formatbeschreibungsquelle 10, 122, 208
- FormatLength (Felddatei) 84
- FormatLength (WT_HOST_MESSAGE-Attribut) 162
- FormatName (Felddatei) 84
- FormatName (WT_HOST_MESSAGE-Attribut) 162
- FormattingSystem (Felddatei) 84
- FormattingSystem (WT_HOST_MESSAGE-Attribut) 162
- Formattyp 208
- FormatType (Felddatei) 84
- FormatType (WT_HOST_MESSAGE-Attribut) 162
- FormProperties Section (Felddatei) 83
- FORMTPL (Systemobjekt-Attribut) 144
- Formularfelder festlegen 144
- function
 - doBackTab() 179
 - doCursorDown() 179
 - doCursorHome() 179
 - doCursorLeft() 179
 - doCursorRight() 179
 - doCursorUp() 178
 - doTab() 179
 - doToggleInsert() 179
 - doToggleMark() 179
 - wtCreateKeyMap() 177
 - wtCreateKeySelectList() 177
 - wtHandleKeyboard() 177
- Funktion 208
- Funktionstasten 147, 199

G

Grace-Sign-On 146, 193
Grafik, verweissensitiv 61
GroupDigit (Felddatei) 85
GroupDigit (Host-Datenobjekt-Attribut) 156

H

Hex_Content__ (WT_HOST_MESSAGE-
Attribut) 162, 164
Hexadezimaldarstellung 120
HexStringValue (Host-Datenobjekt-Attribut) 120,
156, 160
Holder Task 208
Host 208
Host-Adapter 209
Host-Anwendung 209
Host-Daten-Druck 209
Host-Datenobjekt 155, 209
Host-Plattform 209
Host-Steuerobjekt 209
 WT_HOST_GLOBALS 166
 WT_HOST_MESSAGE 162
HOST_CHAR_CODE (Systemobjekt-
Attribut) 145
HOST_IP_ADDRESS (Systemobjekt-
Attribut) 129, 145, 149
HOST_NAME (Systemobjekt-Attribut) 129, 145,
149
HOST_PORT (Systemobjekt-Attribut) 145, 149
HTML 210
HTMLValue (Host-Datenobjekt-Attribut) 156, 160
HTTP 209
HTTPS 209
Hypertext 209
Hypertext Markup Language (HTML) 210

I

IFG-Bibliothek 10, 122
IFG2FLD 10, 122
 Anwendung 75
 Beispiel 41
Include-Tag 97
Inline WTBean 220
 einfügen 190

INPUT-Feld 161
INPUT.clt 161
InputControl (Felddatei) 84
InputControl (Host-Datenobjekt-Attribut) 158,
159
InputKeyClass (Felddatei) 84
InputKeyClass (WT_HOST_MESSAGE-
Attribut) 162
 versorgen 199
InputKeyNumber (Felddatei) 84
InputKeyNumber (WT_HOST_MESSAGE-
Attribut) 162
 versorgen 199
InputState (Felddatei) 84
InputState (Host-Datenobjekt-Attribut) 158, 161
InputStateAct (Felddatei) 84
InputStateAct (Host-Datenobjekt-Attribut) 158,
161
Installation
 bedienerlos 19
 BS2000/OSD 23
 Host-Adapter 17
 Linux 22
 Solaris 21
 über die Bedienoberfläche 18
 UNIX-Systeme 21
 WebLab 23
 WebTransactions 17
 Windows 18
Integrierte Terminal-Emulation 9
Intensity (Felddatei) 84, 85
Intensity (Host-Datenobjekt-Attribut) 158, 159
Invers (Felddatei) 86
Inverse (Felddatei) 85
Inverse (Host-Datenobjekt-Attribut) 158, 159
IOType (Felddatei) 86
IOType (Host-Datenobjekt-Attribut) 156

J

JavaBean 210

K

KDCBADTC 197
KDCDEF 210

- Klasse 210
 Klassen-Template 161, 210
 Kommando im Zeilenmodus 116
 Kommunikationsobjekt 210
 anlegen 190
 Verbindungsparameter 190
 Konvertierungswerkzeuge 210
 Kopplung konfigurieren 121
 Beispiel 45
- L**
 Layout
 "verschönern" 95, 97
 LDAP 211
 Length (Felddatei) 86
 Length (Host-Datenobjekt-Attribut) 156
 Level_Selection (WT_HOST_MESSAGE-
 Attribut) 162, 164
 Line (Felddatei) 86
 line mode siehe Zeilenmodus
 Literal 211
 Lizenzen
 eingeben (Beispiel) 28
 Lizenzierung 24
 LOCAL_APPLICATION (Systemobjekt-
 Attribut) 145, 150, 201
 LOCAL_PORT (Systemobjekt-Attribut) 130, 145,
 149
 localapps (Basisverzeichnis) 71
 Lock-/Keycode-Konzept 192, 201
 LTERM-Partner 201
- M**
 Mandatory (Felddatei) 86
 Mandatory (Host-Datenobjekt-Attribut) 157
 Master-Template 211, 217
 Tags 211
 UTM.wmt 96
 UTMpartial.wmt 102
 Message Queuing 211
 Metadaten 82
 Methode 211
 Modul-Template 211
 MT-Tag 211
- Multi-Tier-Architektur 212
- N**
 Nachspann festlegen 144
 Name (Host-Datenobjekt-Attribut) 156
 Name/Value-Paar 212
 NEW_PASSWORD (Systemobjekt-Attribut) 146,
 149, 192
 Next SYM_DEST (Zeilenmodus) 116
 Nicht synchronisierter Dialog 206, 212
 NumAttributes (Felddatei) 85
 NumDecimal (Felddatei) 86
 NumDecimal (Host-Datenobjekt-Attribut) 156
- O**
 Objekt 212
 openUTM 212
 Vorgang 219
 openUTM-Anwendung 213
 openUTM-Benutzererkennung 192
 ungültig 152, 193
 openUTM-Benutzerkonzept 192
 openUTM-Client (UPIC) 213
 openUTM-Pagepool 153
 openUTM-Partner (Zeilenmodus) 116
 openUTM-Passwort 192
 openUTM-Teilprogramm 213
 openUTM-Version
 kleiner V4.0, UPIC-Protokoll für 132
 openUTM-Zeilenmodus 115
 Operationen 206
 OUTPUT-Feld 161
 OUTPUT.clt 161
 OutputControl (Felddatei) 85
 OutputControl (Host-Datenobjekt-Attribut) 158
- P**
 P_Key_Set (WT_HOST_MESSAGE-
 Attribut) 162, 164
 Padding (WT_HOST_GLOBALS-Attribut) 166
 PaddingAsterix (WT_HOST_GLOBALS-
 Attribut) 166
 PaddingPlusAttr (WT_HOST_GLOBALS-
 Attribut) 166

PaddingPlusData (WT_HOST_GLOBALS-Attribut) [166](#)
Parameter [213](#)
Passiver Dialog [206](#), [213](#)
PASSWORD (Systemobjekt-Attribut) [145](#), [149](#), [192](#)
Passwort [192](#), [213](#)
 abgelaufen [146](#)
 ändern [193](#)
 neu [192](#)
 ungültig [152](#)
Passwort-Schutz (openUTM) [145](#)
polling [213](#)
Pool [214](#)
PopUp (Felddatei) [84](#)
Posted-Objekt [214](#)
Posten [214](#)
Projekt [214](#)
 anlegen [35](#)
 Beispiel [35](#), [39](#)
 speichern [39](#)
PROLOG (Systemobjekt-Attribut) [146](#)
Protected (Host-Datenobjekt-Attribut) [157](#)
Protection (Felddatei) [85](#), [86](#)
Protection (Host-Datenobjekt-Attribut) [158](#), [159](#)
Protocol (Felddatei) [84](#)
Protokoll [214](#)
Protokolldatei [214](#)
Prozess [214](#)
Puffer [214](#)

R

RawValue (Host-Datenobjekt-Attribut) [156](#), [160](#)
Read (WT_HOST_GLOBALS-Attribut) [167](#)
RECEIVE_ERROR (Systemobjekt-Attribut) [146](#), [150](#)
 bei receive [151](#)
RECEIVE_SECONDARY_INFORMATION (Systemobjekt-Attribut) [146](#), [150](#), [151](#)
 mögliche Werte [153](#)
Record [215](#)
RESTART (Systemobjekt-Attribut) [146](#), [149](#), [194](#)
RETRY (Systemobjekt-Attribut) [146](#)

S

Schlüsselwörter
 Felddatei [83](#)
ScreenDimensions (Felddatei) [84](#)
SD.DEFAULT [130](#)
Sections, Felddatei [83](#)
Security-Funktionen, openUTM-
 Benutzerkonzept [192](#)
SECURITY_TYPE (Systemobjekt-Attribut) [147](#), [149](#), [192](#)
Sekundärer UPIC-Returncode [153](#)
Service-Anwendung [215](#)
Service-Knoten [215](#)
Services einlesen (Beispiel) [25](#)
SFUNC (KDCDEF-Anweisung) [199](#)
Sichtbarkeit [215](#)
Sign (Felddatei) [86](#)
Sign (Host-Datenobjekt-Attribut) [156](#)
SignFloat (Felddatei) [86](#)
SIGNON Event-Service [193](#)
Sitzung [215](#)
 Start-Templates [182](#)
 starten [46](#)
 starten (WebLab) [67](#)
 WebTransactions [215](#)
Skalar [216](#)
SOAP [216](#)
SPECIAL_KEY (Systemobjekt-Attribut) [147](#)
Speichern
 Projekt [39](#)
Standalone WTBean [220](#)
 einfügen [188](#)
Start-Template [182](#), [217](#)
 aufrufen [67](#)
 eigenes erzeugen [63](#)
 Systemobjekt-Attribute setzen [184](#)
 WT_HOST_GLOBALS-Attribute setzen [187](#)
StartColumn (Host-Datenobjekt-Attribut) [156](#)
Starten
 Sitzung [46](#)
 Start-Template [182](#)
 WebTransactions [63](#)
StartLine (Host-Datenobjekt-Attribut) [156](#)
Startpage-Button (Zeilenmodus) [117](#)

- Stil [216](#)
- SuppressZero (Felddatei) [86](#)
- SuppressZero (Host-Datenobjekt-Attribut) [156](#)
- SYM_DEST (Systemobjekt-Attribut) [147](#)
 - bei open [149](#)
- Symbolic Destination Name [147](#)
- Symbolic Destination Name, upicfile [130](#)
- Synchronisierter Dialog [206, 216](#)
- Systemobjekt [216](#)
 - Attribute im Start-Template setzen [184](#)
 - openUTM-spezifische Attribute [141](#)
 - Zusammenspiel Attribute u. Aufrufe [149](#)
- Systemobjekt-Attribut
 - CHARSET [10, 122, 165](#)
- T**
- TAC [218](#)
- TAC (Systemobjekt-Attribut) [129, 147, 149](#)
- TAC im Zeilenmodus [116](#)
- Tag [216](#)
- Tasten-Zuordnung
 - definieren [171](#)
 - wtKeysUTMFHS.js und wtKeysUTMFormant.js [171](#)
- Tastenunterstützung
 - durch Browser [170](#)
- TCP/IP [217](#)
- Teilformat
 - FHS/FORMANT [100](#)
 - spezifizieren [144](#)
- Teilformat-Template
 - Aufbau [102](#)
 - Verschönerung [112](#)
- Template [73, 217](#)
 - aus FHS-Format generieren [41, 74](#)
 - für openUTM-Zeilenmodus [116](#)
 - Klasse [210](#)
 - Master [217](#)
 - nachbearbeiten [95, 97](#)
 - nachbearbeiten (Beispiel) [52](#)
 - Start [217](#)
- Template-Objekt [217](#)
- Templates [10, 122](#)
 - Unicode [122](#)
- Terminal screen (Zeilenmodus) [116](#)
- Terminal-Anwendung [217](#)
- Terminal-Funktionen [168](#)
 - Unterstützung durch Browser [168](#)
- Terminal-Hardcopy-Druck [217](#)
- Terminate session (Zeilenmodus) [117](#)
- Text (Felddatei) [86](#)
- Thread [208](#)
- TimeFormat (Felddatei) [84](#)
- TimeFormat (WT_HOST_MESSAGE-Attribut) [162](#)
- Traces [161](#)
- Transaktion [218](#)
- Transaktionscode [218](#)
 - entfernen [143](#)
 - ungültig [152](#)
- Travel Agency [25](#)
- U**
- UDDI [218](#)
- Umlaute [123](#)
- UndefinedValues (Felddatei) [84](#)
- UndefinedValues (WT_HOST_MESSAGE-Attribut) [162](#)
- Underline (Felddatei) [85, 86](#)
- Underline (Host-Datenobjekt-Attribut) [158, 159](#)
- Underlined (Host-Datenobjekt-Attribut) [157](#)
- UnDetect (WT_HOST_GLOBALS-Attribut) [167](#)
- Unicode [218](#)
 - bei Zeichenkettenoperationen [161](#)
 - FLD-Dateien [122](#)
 - in der Diagnose [161](#)
 - Regeln für Templates [122](#)
- Unicode (Host-Datenobjekt-Attribut) [157](#)
- Unicode (WT_HOST_MESSAGE-Attribut) [163, 165](#)
- Unicode-Unterstützung [122, 161](#)
- Update (WT_HOST_GLOBALS-Attribut) [167](#)
- UPIC [218](#)
- UPIC-Kommunikation, Systemobjekt [141](#)
- UPIC-R Verteilung über Rechner [121](#)
- UPIC-Returncode [152](#)
 - sekundärer [153](#)
- UPIC-Schnittstelle [10, 122](#)

- UPIC_CODE_CONVERSION (Systemobjekt-Attribut) [148](#), [149](#)
 - im Start-Template [185](#)
- UPIC_LIB (Systemobjekt-Attribut) [148](#)
- UPIC_TRACE (Systemobjekt-Attribut) [148](#)
- upicfile [149](#)
 - Basisverzeichnis [71](#)
 - Standardeintrag [130](#)
- URI [218](#)
- URL [219](#)
- USER (Systemobjekt-Attribut) [148](#), [149](#), [192](#)
- Userexit [219](#)
- UserexitRc (Felddatei) [84](#)
- UserexitRc (WT_HOST_MESSAGE-Attribut) [162](#)
- UTM siehe openUTM
- UTM User (Zeilenmodus) [117](#)
- UTM.wmt [96](#)
- UTM_PATH (Systemobjekt-Attribut) [148](#)
- UTMControl (Felddatei) [86](#)
- UTMpartial.wmt [102](#)

- V**
- Value (Host-Datenobjekt-Attribut) [156](#), [160](#)
- Variable [219](#)
- Verbindung
 - mehrere öffnen [190](#)
- Verbindungsabbau [152](#)
- Version (Felddatei) [84](#)
- Version (WT_HOST_MESSAGE-Attribut) [163](#)
- Verweissensitive Grafik [61](#)
- Visibility (Felddatei) [85](#), [86](#)
- Visibility (Host-Datenobjekt-Attribut) [158](#), [159](#)
- Visible (Host-Datenobjekt-Attribut) [157](#)
- Vorgang
 - starten [187](#)
- Vorgang (openUTM) [219](#)
- Vorgangskettung siehe Vorgangsverknüpfung
- Vorgangsverknüpfung [198](#)
- Vorspann festlegen [146](#)

- W**
- web server [219](#)
- Web-Service [219](#)
- WebLab [10](#), [161](#)
 - installieren [23](#)
- WebLab-Editor [95](#), [97](#)
- WebTransactions
 - Architektur [9](#)
 - starten [63](#)
 - verteilen auf Rechner [121](#)
- WebTransactions-Anwendung [219](#)
- WebTransactions-Plattform [219](#)
- WebTransactions-Server [219](#)
- WebTransactions-Sitzung [215](#)
- Wertebereich eines Datentyps [206](#)
- Wiederanlauf [146](#)
- WSDL [220](#)
- WT_BROWSER [180](#)
 - Font-Größe [180](#)
- WT_HOST_GLOBALS [166](#)
- WT_HOST_MESSAGE [162](#)
- WTBean [220](#)
 - wtcUTM [190](#)
- wtBrowserFunctions.htm [168](#)
- wtCommonBrowserFunctions.js [176](#)
- wtcStartUPIC [188](#)
- wtcUTM [190](#)
- wtKeyMappingTableInput [171](#)
- wtKeysUTM.htm [168](#)
- wtKeysUTMFHS.js [171](#)
 - Aufbau [174](#)
- wtKeysUTMFormant.js [171](#)
 - Aufbau [174](#)
- wtInmode.htm [115](#)
- WTML [220](#)
- WTML-Tag [220](#)
- WTScript [220](#)
- wtstartUTMV4.htm [182](#)
- WWW-Browser [204](#)
- WWW-Server [219](#)

- X**
- XML [221](#)
- XML-Schema [221](#)

- Z**
- Zeichencodierung (openUTM) [145](#)

Zeichenkettenoperationen [123](#), [161](#)
 Unicode [161](#)
Zeilenmodus [115](#)
Zeilenmodus-Template [115](#)
Zugangskontrolle [221](#)
Zugriffskontrolle [221](#)

