Benutzerhandbuch - Deutsch

FUJITSU

WebTransactions V7.5

Anschluss an MVS-Anwendungen

Ausgabe August 2010

Kritik… Anregungen… Korrekturen…

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

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

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

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

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

Copyright und Handelsmarken

Inhalt

1 Einleitung

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 Tablett 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 MVS

Folgende Abbildung zeigt die Architektur von WebTransactions for MVS:

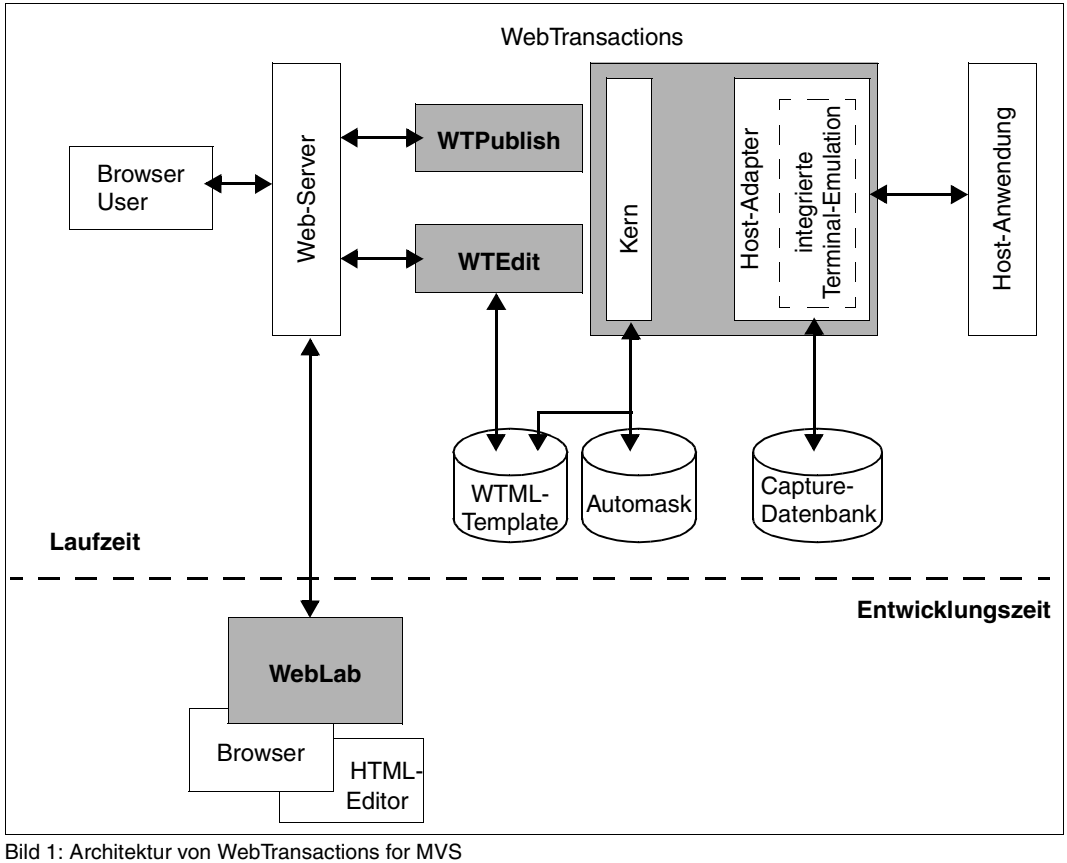


Bild 1: Architektur von WebTransactions for MVS

Integrierte Terminal-Emulation

WebTransactions for MVS benutzt zur Lauf- und Entwicklungszeit eine 3270-Emulation, die in den Host-Adapter integriert ist und die 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 formatspezifischen Templates bis zum Test der Anwendung ausführen können.

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.

Für eine 1:1 Darstellung oder eine globale Aufbereitung können Sie mit WebLab Varianten des Standard-Automask-Templates erstellen, das zur Laufzeit alle Ausgangsformate dynamisch umsetzt.

Für die individuelle Aufbereitung erzeugen Sie mit dem Capture-Verfahren Erkennungskriterien, die in der Capture-Datenbank gespeichert werden, und generieren aus dem Format ein sogenanntes formatspezifisches Template und eine Formatbeschreibungsdatei (FLD-Datei). Die formatspezifischen Templates bearbeiten Sie mit WebLab nach.

Ablauf

Zur Laufzeit sucht WebTransactions in der Capture-Datenbank nach einem Erkennungskriterium, das zu dem von der Host-Anwendung geschickten Bildschirmformat passt. Ist ein solches vorhanden, verwendet WebTransactions das entsprechende formatspezifische Template. Wird kein passendes Kriterium gefunden, setzt WebTransactions das Bildschirmformat dynamisch um: mit dem Standard-Automask-Template.

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 WTScript-Anweisungen, die Sie in den WTScript-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 instanziieren 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 WTScript-Datenstrukturen in XML-Dokumente.

- Jeweils ein Benutzerhandbuch für jeden Host-Adapter mit speziellen Informationen, zugeschnitten auf den Typ der Partneranwendung:

  Anschluss an openUTM-Anwendungen über UPIC

  Anschluss an OSD-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 alle, die mit WebTransactions MVS- bzw. z/OS-Dialoganwendungen an das Web anschließen wollen. Die einzelnen Kapitel beschreiben die hierfür notwendigen Schritte. Wenn Sie noch nicht mit WebTransactions for MVS 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 MVS-spezifischen Informationen.

Gültigkeit der Beschreibung

WebTransactions for MVS ist auf den Systemplattformen Solaris, Linux und 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

Einen Überblick über alle Neuerungen in WebTransactions V7.5 finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

1.6 Darstellungsmittel

Diese Dokumentation verwendet die folgenden Darstellungsmittel:

`dicktengleiche Schrift`	feste Teile, die genau in dieser Form ein- oder ausgegeben werden, wie z.B. Schlüsselwörter, URLs, Dateinamen
kursive Schrift	variable Teile, für die Sie konkrete Angaben einsetzen müssen, sowie Menübefehle
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.
{*alternative1* | *alternative2* }	alternative Angaben. Einen der Ausdrücke innerhalb der geschweiften Klammern müssen Sie auswählen. Die einzelnen Ausdrücke sind durch senkrechte Striche voneinander getrennt. Die geschweiften Klammern und senkrechten Striche selbst dürfen Sie nicht angeben.
...	optionale ein- oder mehrmalige Wiederholung des vorhergehenden Elements
	wichtige Hinweise und weiterführende Informationen
►	Aufforderungszeichen, wenn Sie etwas tun sollen.

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

WebTransactions for MVS 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 17).

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) `WebTransactionsMVS75.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 und die Java-Umgebung festlegen

Nach dem Start von `WebTransactionsMVS75.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\WebTransactionsMVS75.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\WebTransactionsMVS75.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. `webtransMVSV75.tar.gz`). Diese Datei müssen Sie zuerst dekomprimieren, mit dem Kommando:

```
gunzip -d webtransMVSV75.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 webtransMVSV75.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 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 12) oder mit `Msiexec.exe` (siehe Seite 13).

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 einfach und schnell 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 MVS auf einem Rechner installiert haben (siehe hierzu auch Kapitel „WebTransactions installieren“ auf Seite 11), 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.

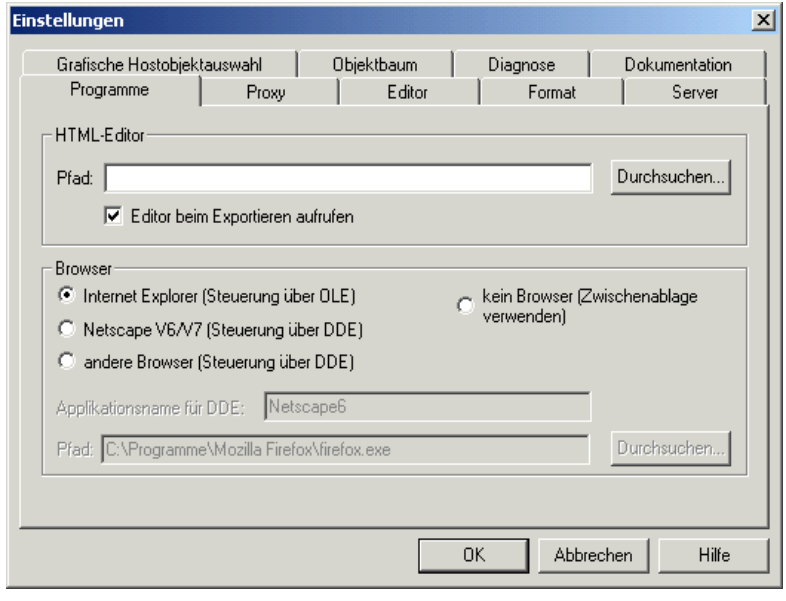
Als ersten Schritt stellen Sie in WebLab den Browser ein, den WebLab für die Bedienung der WebTransactions-Anwendung verwenden soll. Die Arbeit mit dem Administrationsprogramm gliedert sich dann in folgende Schritte:

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 in WebLab den Befehl **Optionen/Einstellungen**. Das Dialogfeld **Einstellungen** mit der ersten Registerkarte **Programme** wird am Bildschirm eingeblendet.



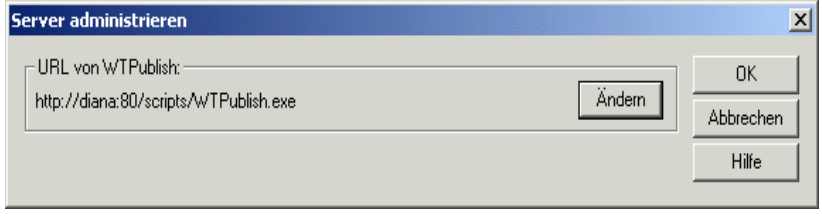
- ► Wählen Sie im unteren Bereich **Browser** den Browser aus, der auf Ihrem Rechner installiert ist, und wie er von WebLab verwendet werden soll.
- ► Bestätigen Sie Ihre Einstellung mit **OK**.

3.1.2 Administrationsprogramm starten

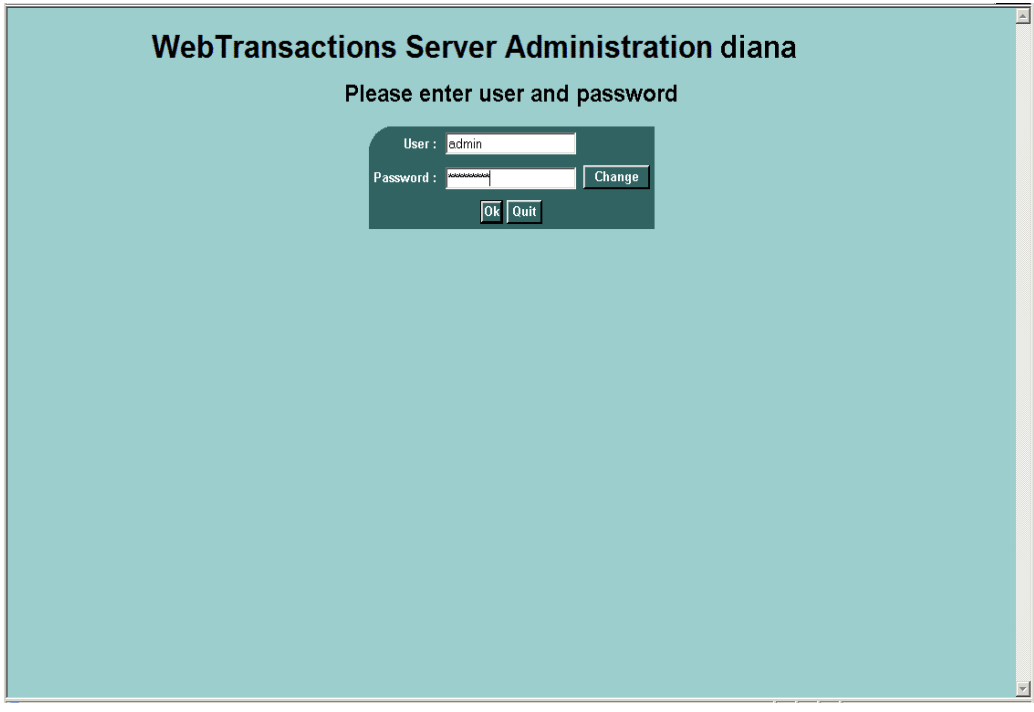
- ► Wählen Sie den Befehl **Administrieren/Server**, um zuerst das Administrationsprogramm zu starten. Das Dialogfeld **Server administrieren** wird am Bildschirm eingeblendet.
- ► Klicken Sie unter **URL von WTPublish** auf **Ändern**. Das Dialogfeld **Url von WTPublish** wird angezeigt.
- ► Wählen 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.



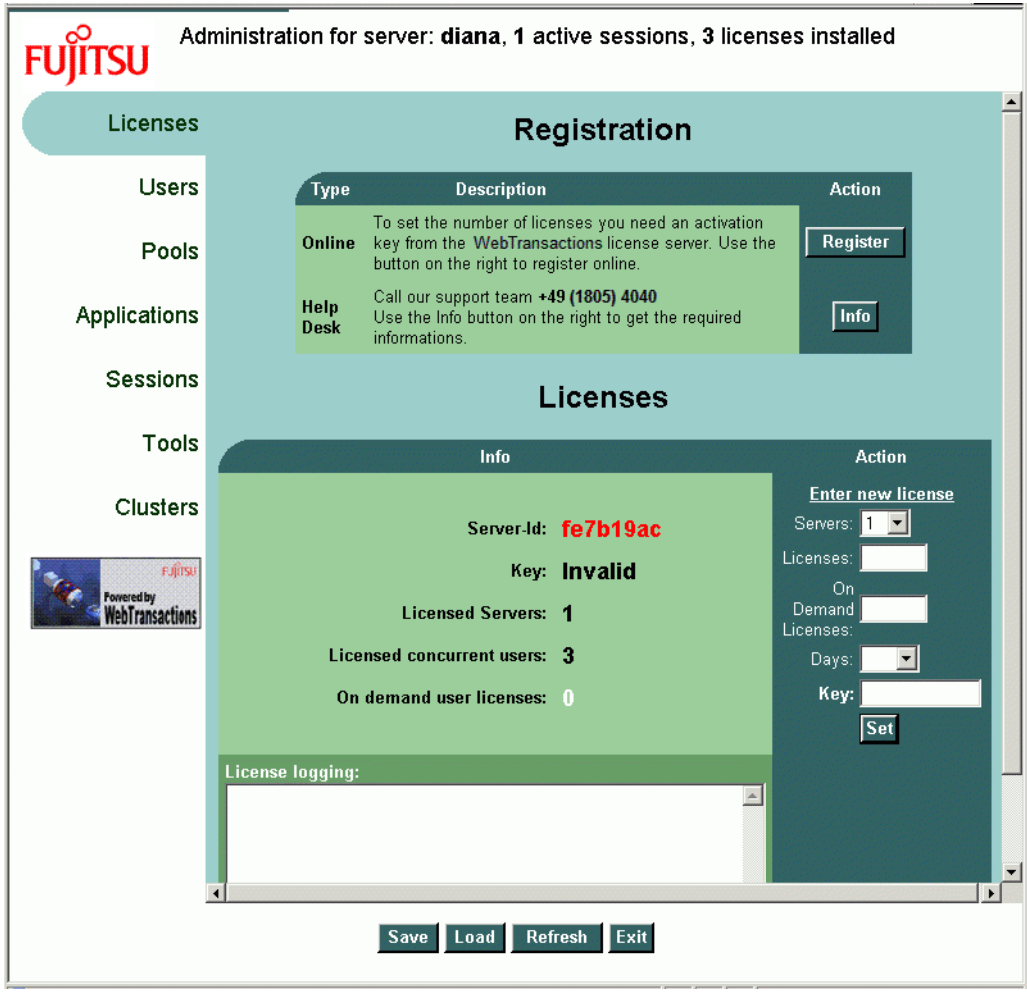
- ► Bestätigen Sie mit **OK**. Das Administrationsprogramm wird gestartet und das erste Fenster im Browser angezeigt.



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

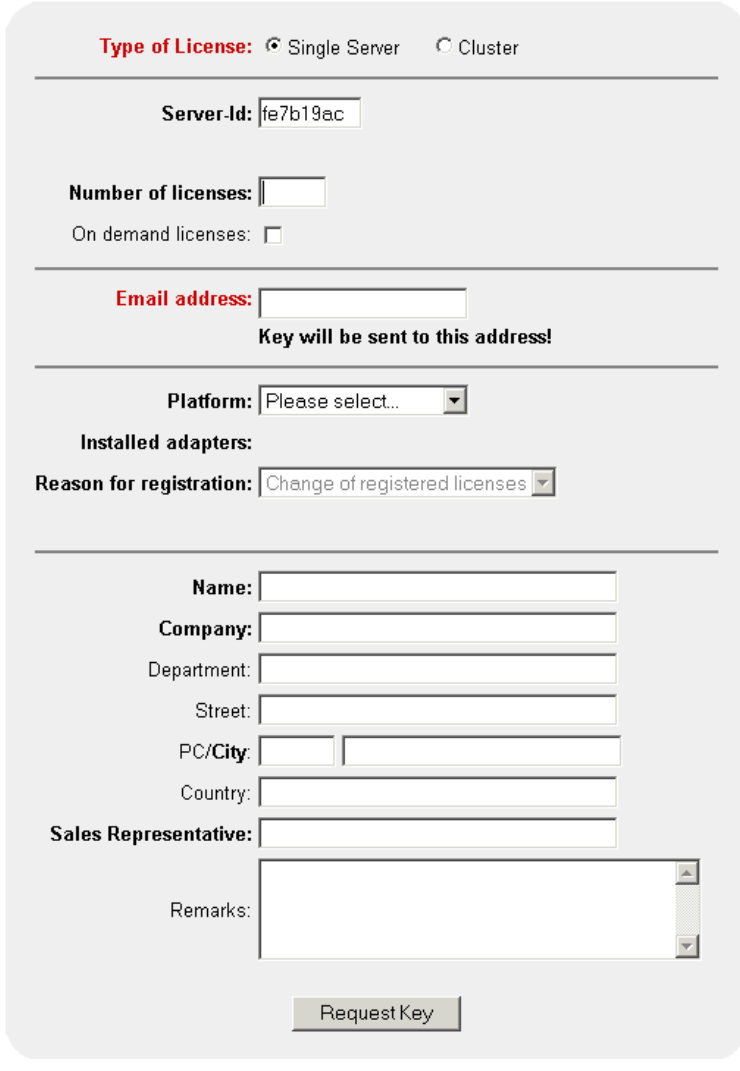
Wenn Sie zum erstenmal mit dem Administrationsprogramm arbeiten, sollten Sie zur Sicherheit nach der Anmeldung für den Benutzer `admin` ein Passwort vergeben.

3.1.3 Lizenzen eingeben



► Drücken Sie auf der Lizenzierungsseite die Schaltfläche **Register**.

Die Registrierungsseite wird geöffnet:



- 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.

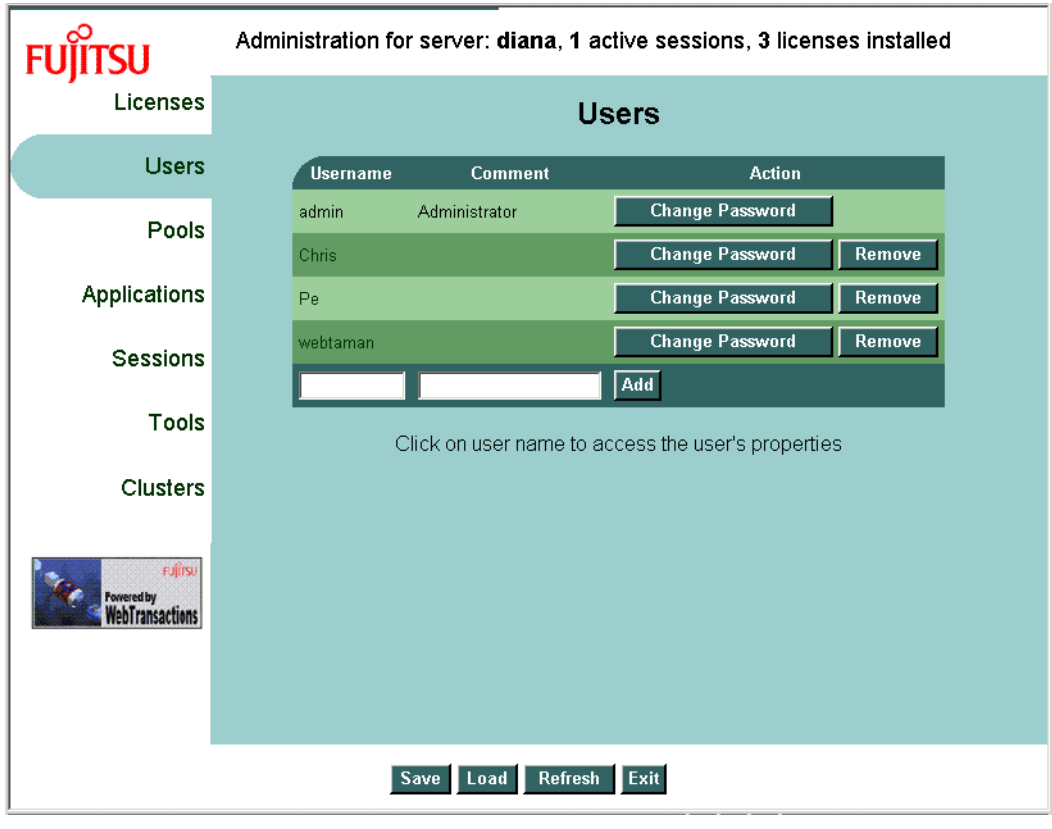
- Tragen Sie auf der Lizenzierungsseite in die Felder **Licenses** und **Key** die Anzahl der erworbenen Lizenzen bzw. den gültigen Lizenz-Schlüssel ein, der Ihnen per eMail zugesendet wurde.

- 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.



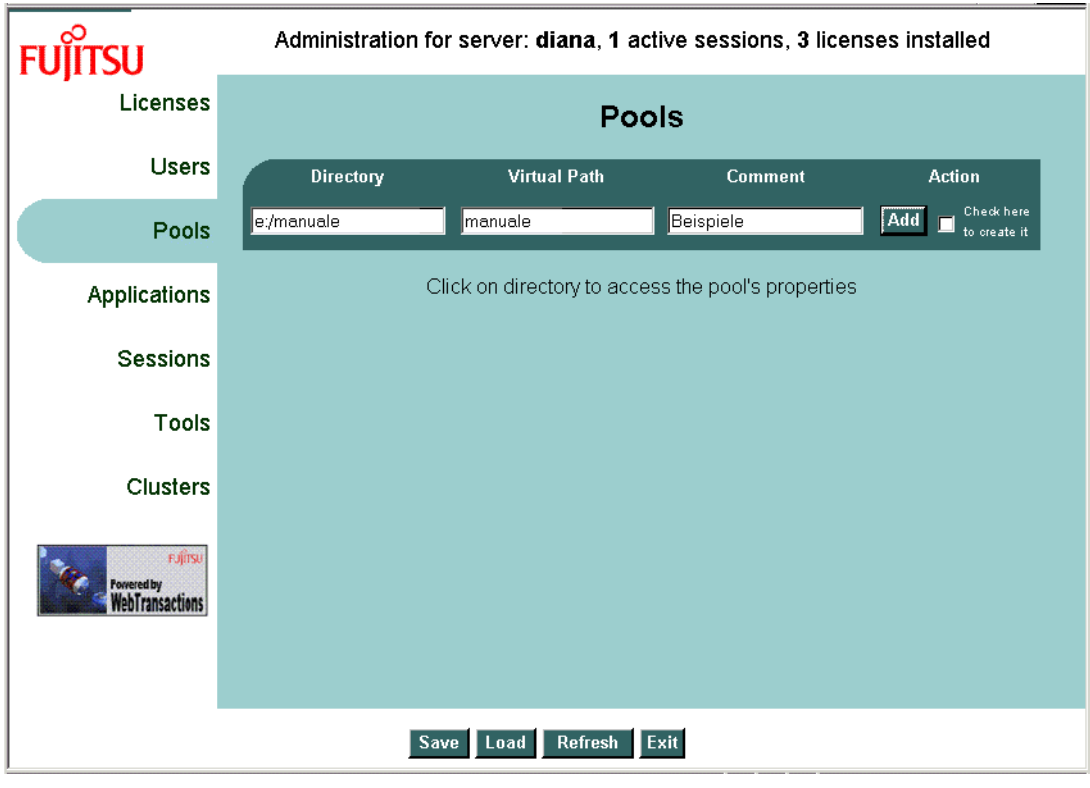
► Geben Sie im Arbeitsbereich in das Eingabefeld **Username** den Namen des neuen Benutzers ein. Genauso können Sie auch das Passwort von `admin` ändern.

► 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.

3.1.5 Pool anlegen

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



- Geben Sie im Arbeitsbereich in das Eingabefeld **Directory** den Namen des Verzeichnisses mit absolutem Pfadnamen 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 – ohne WebTransactions dafür aufrufen zu müssen – auf Dateien (z.B. Bilder, Startseite, ...) von WebTransactions-Anwendungen in diesem Verzeichis zugegriffen wird.

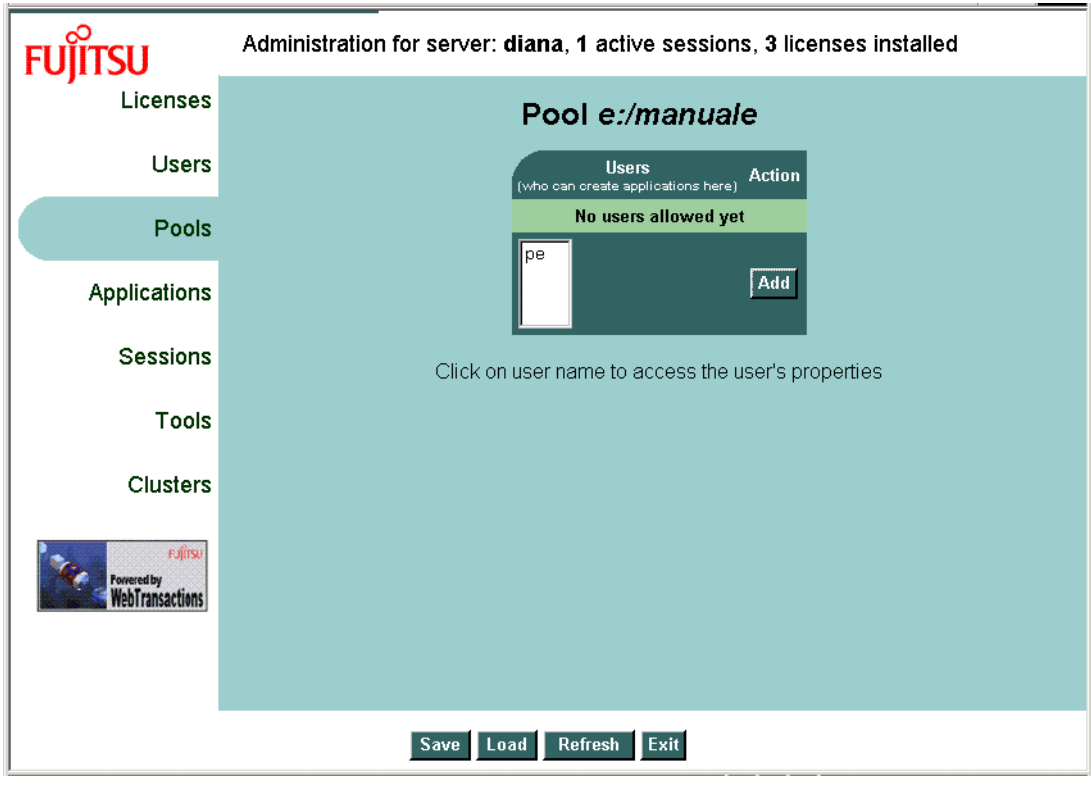
Falls Sie in unterschiedlichen Pools Basisverzeichnisse mit gleichen Namen verwenden wollen, müssen die dem Pool zugehörigen Werte bei **Virtual Path** unterschiedlich angegeben werden.

► 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.

In 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.



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.

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
   - Automask generieren
2. Projekt speichern
3. Sitzung starten

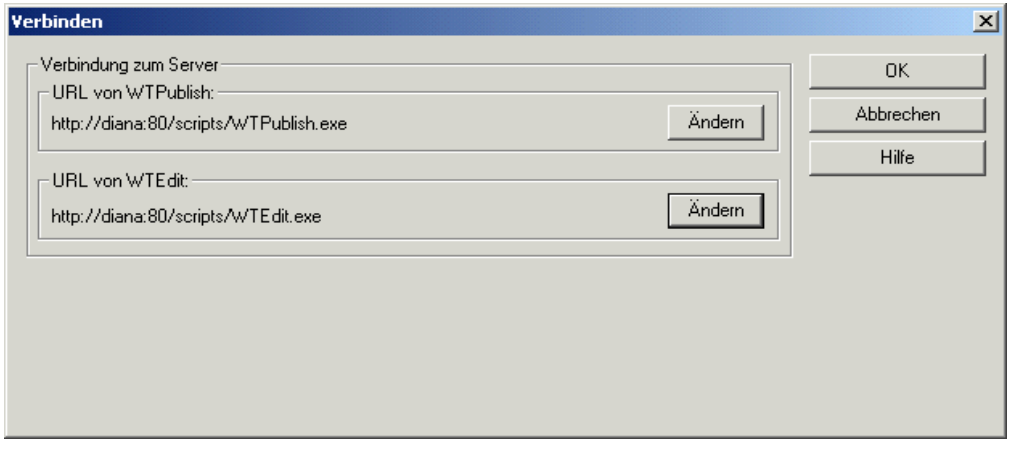
3.2.1 Projekt anlegen

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.

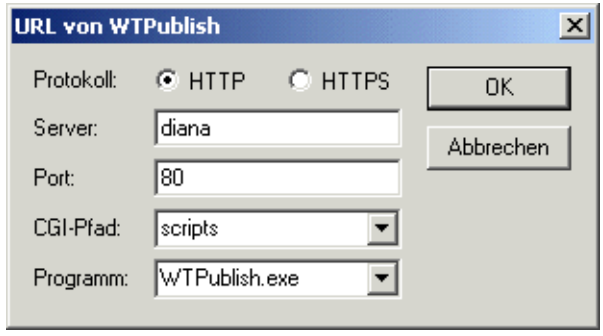


Das Basisverzeichnis muss immer auf dem Rechner angelegt werden, auf dem WebTransactions läuft. Im Dialogfeld **Verbinden** geben Sie diesen WebTransactions-Server an und die Pfade zu den CGI-Programmen WTPublish.exe und WTEdit.exe.

– `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.



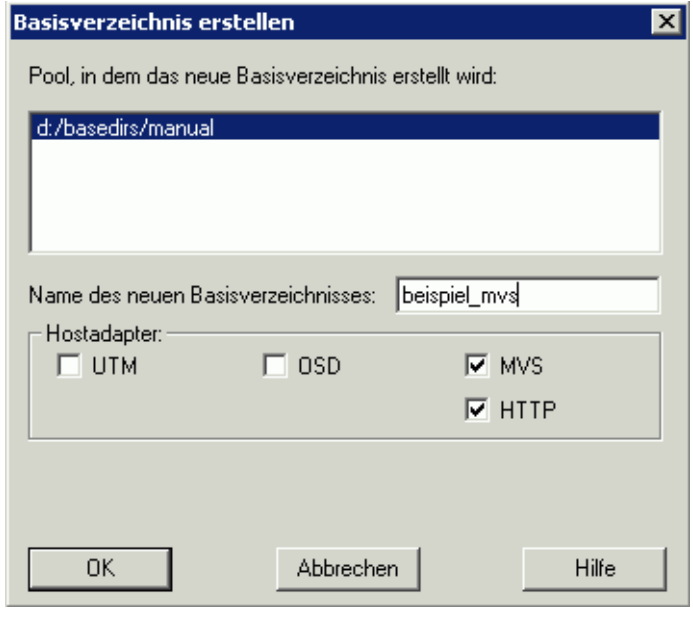
- Wählen Sie das **Protokoll**, das für die Verbindung verwendet werden soll, hier **HTTP**.
- Geben Sie im Feld **Server** den Rechner an, auf dem WebTransactions läuft, hier *diana*.
- Geben Sie im Feld **Port** die zugehörige Portnummer an, hier *80*.
- Geben Sie den Pfad für das CGI-Programm WTPublish an, hier *scripts*.
- 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	*diana*
Port	*80*
CGI-Pfad	*scripts*
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 26 angelegt haben.
- Bestätigen Sie Ihre Angaben mit **OK**. Das Dialogfeld **Basisverzeichnis erstellen** wird am Bildschirm eingeblendet.

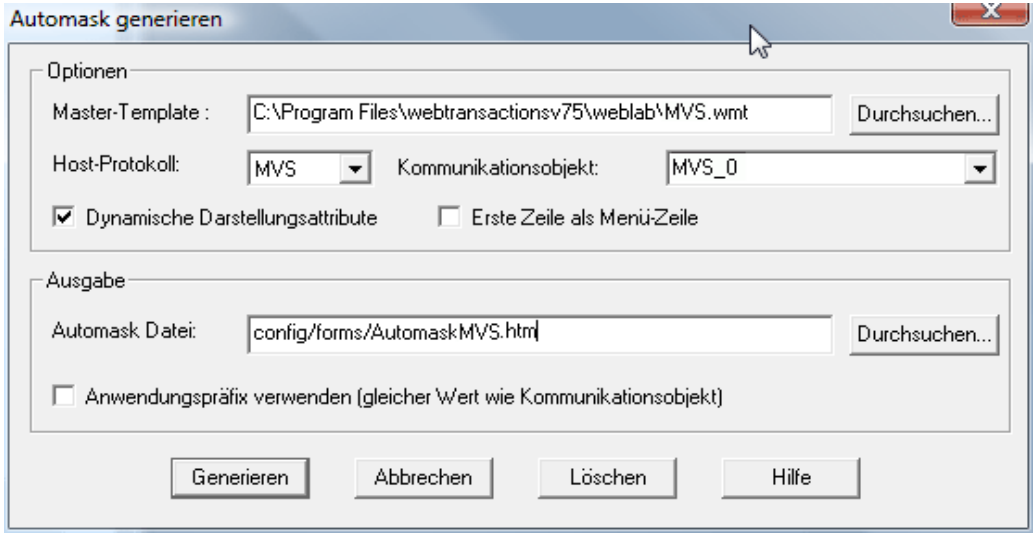


In der oberen Liste dieses Dialogfeldes werden die Pools angezeigt, unter denen der angemeldete Benutzer auf dem WebTransactions-Server Basisverzeichnisse angelegen darf.

- Wählen Sie mit einem Klick aus der Liste den Pool, den Sie in Abschnitt „Pool anlegen“ auf Seite 27 angelegt haben.
- Geben Sie dann im Eingabefeld **Name des Basisverzeichnisses** einen Namen an, hier *beispiel_mvs*.
- Wählen Sie dann den passenden Host-Adapter, über den WebTransactions mit der Host-Anwendung kommuniziert, hier *MVS* . 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 **Automask generieren** wird am Bildschirm eingeblendet, siehe nachfolgender Abschnitt.

3.2.1.2 Automask-Template generieren

Das Automask-Template ist das Template, über das die Umsetzung zwischen Host-Formaten und Browser-Darstellung automatisch abgewickelt wird.



Im Dialogfeld **Automask generieren** können Sie mit den Optionen die Generierung dieses Templates und damit die spätere Umsetzung beeinflussen. Alle Optionen sind in der Online-Hilfe beschrieben.

- Geben Sie dem **Kommunikationsobjekt** einen Namen, hier *MVS_0*.Wenn Sie hier nichts angeben, wird die Voreinstellung `MVS_0` verwendet. Wenn Sie in einer Sitzung mehrere Verbindungen öffnen wollen, ist es 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.

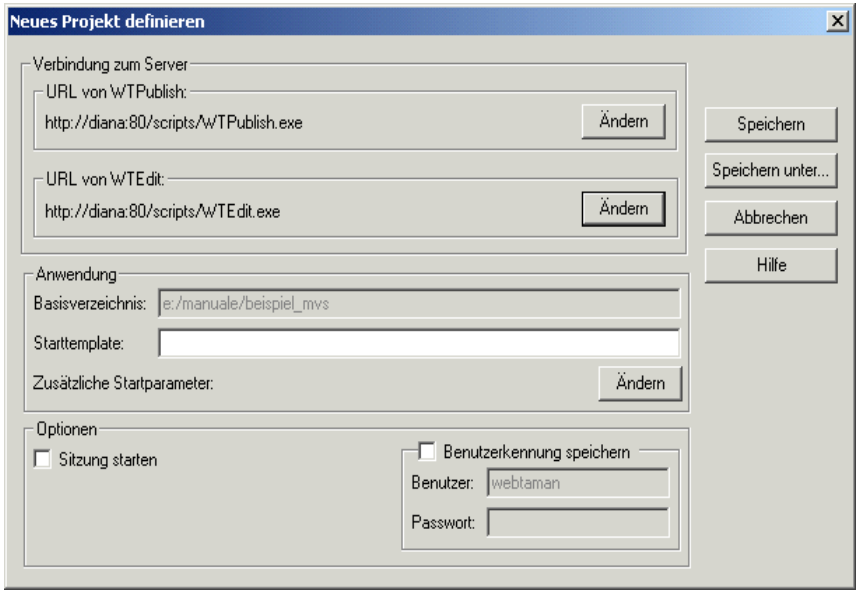
- In diesem Beispiel akzeptieren Sie alle weiteren Voreinstellungen und bestätigen das Dialogfeld mit **Generieren**. Das Dialogfeld wird geschlossen und Basisverzeichnis und Automask werden mit den angegebenen Werten am WebTransactions-Server generiert. Im WebLab-Hauptfenster wird unterhalb des Arbeitsbereichs 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 56).

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**.

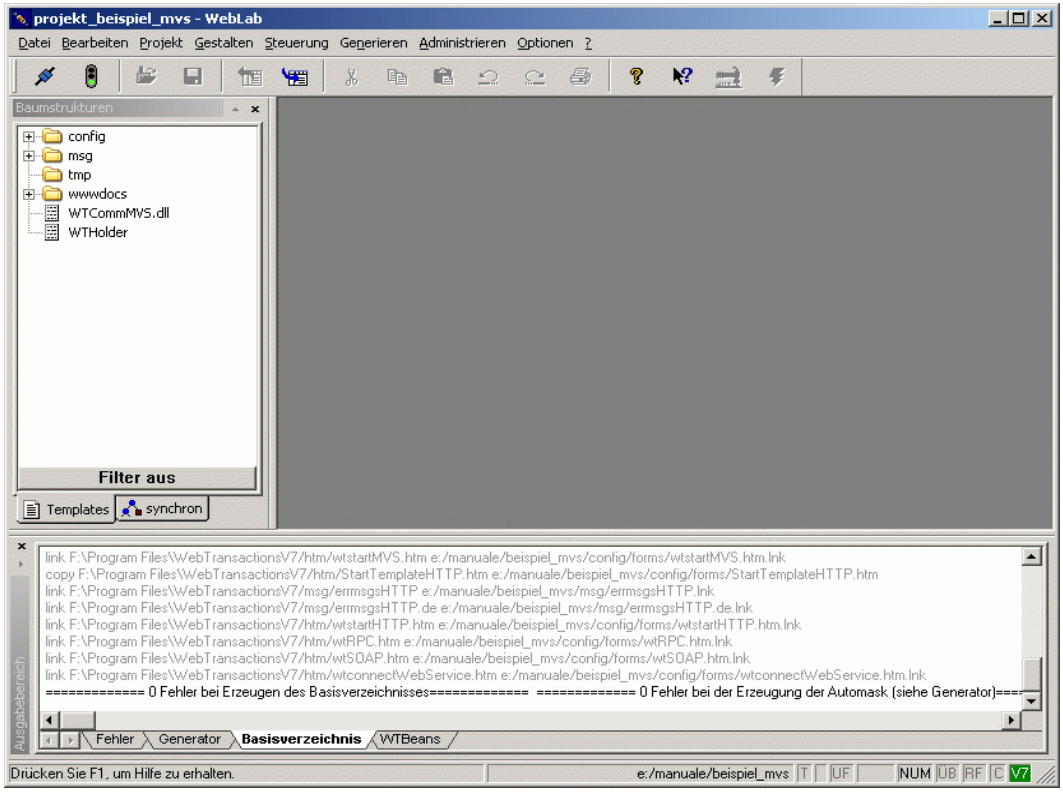
  Das Dialogfeld **Datei speichern unter** wird geöffnet.

- 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.

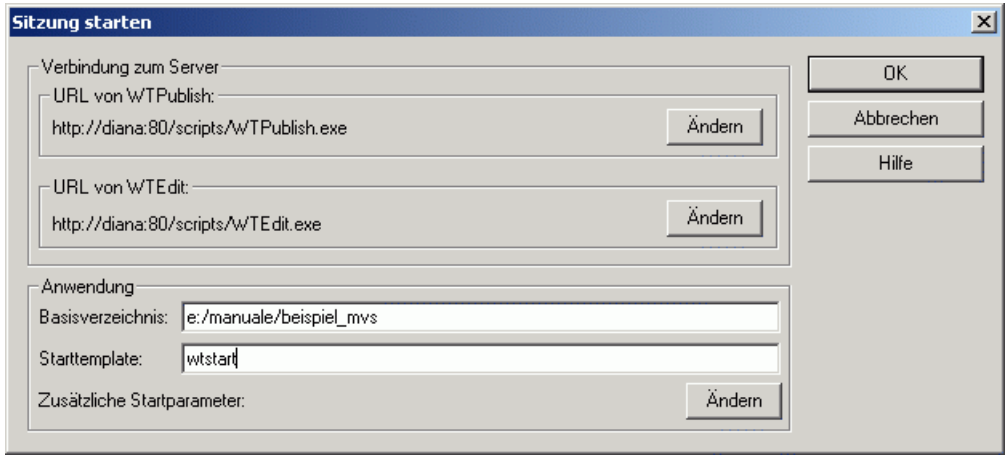
  Anschließend sind Sie mit Ihrem neuen Basisverzeichnis verbunden. Eine Übersicht über die angelegten Verzeichnisse finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.


projekt_beispiel_mvs - WebLab
Datei Bearbeiten Projekt Gestalten Steuerung Generieren Administrieren Optionen ?
Baumstrukturen
config
msg
tmp
wwwdocs
WTCommMVS.dll
WTHolder
Filter aus
Templates
synchron
link F:\Program Files\WebTransactionsV7/htm/wtstartMVS.htm e:/manuale/beispiel_mvs/config/forms/wtstartMVS.htm.lnk
copy F:\Program Files\WebTransactionsV7/htm/StartTemplateHTTP.htm e:/manuale/beispiel_mvs/config/forms/StartTemplateHTTP.htm
link F:\Program Files\WebTransactionsV7/msg/errmsgsHTTP e:/manuale/beispiel_mvs/msg/errmsgsHTTP.lnk
link F:\Program Files\WebTransactionsV7/msg/errmsgsHTTP.de e:/manuale/beispiel_mvs/msg/errmsgsHTTP.de.lnk
link F:\Program Files\WebTransactionsV7/htm/wtstartHTTP.htm e:/manuale/beispiel_mvs/config/forms/wtstartHTTP.htm.lnk
link F:\Program Files\WebTransactionsV7/htm/wtRPC.htm e:/manuale/beispiel_mvs/config/forms/wtRPC.htm.lnk
link F:\Program Files\WebTransactionsV7/htm/wtSOAP.htm e:/manuale/beispiel_mvs/config/forms/wtSOAP.htm.lnk
link F:\Program Files\WebTransactionsV7/htm/wtconnectWebService.htm e:/manuale/beispiel_mvs/config/forms/wtconnectWebService.htm.lnk
========== 0 Fehler bei Erzeugen des Basisverzeichnisses========== ========== 0 Fehler bei der Erzeugung der Automask (siehe Generator)===
Ausgabebereich
Fehler
Generator
Basisverzeichnis
WTBeans
Drücken Sie F1, um Hilfe zu erhalten.
e:/manuale/beispiel_mvs
NUM

3.2.3 Sitzung starten

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

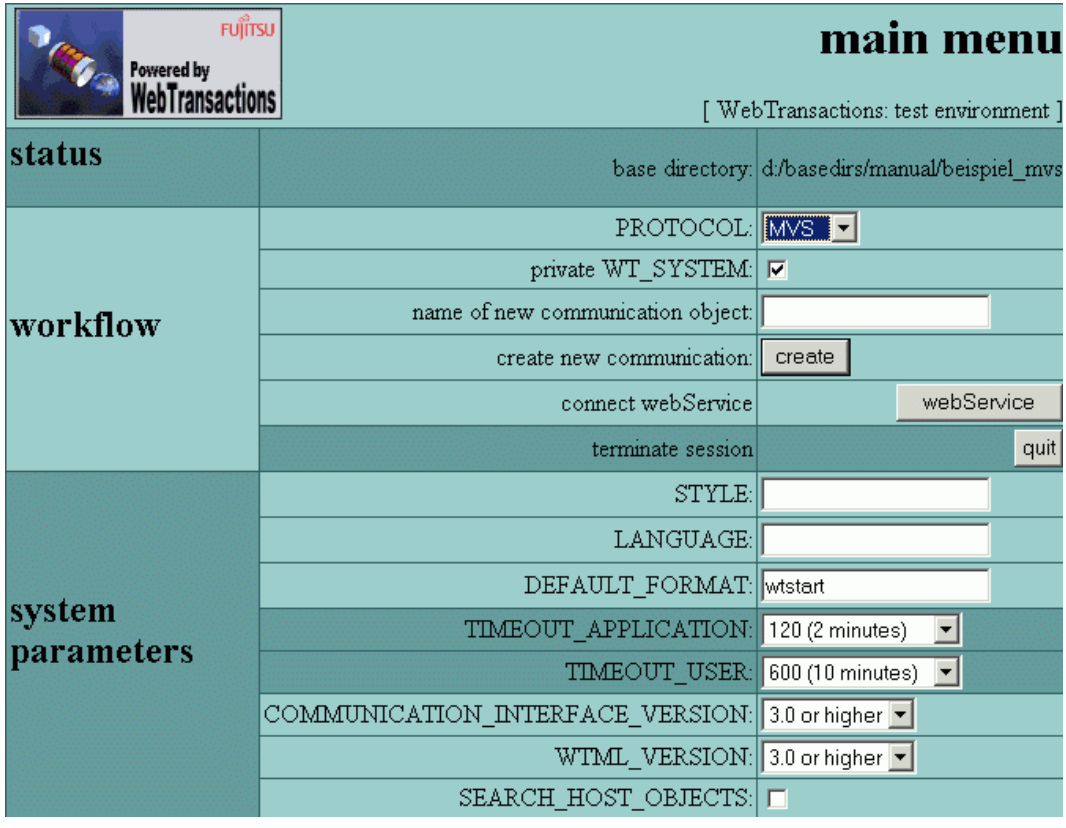
► Wählen Sie dazu in WebLab den Befehl **Datei/Sitzung starten**. Das Dialogfeld **Sitzung starten** wird am Bildschirm eingeblendet.



In diesem Dialogfeld wurden die Verbindungsdaten wie Web-Server-Name, 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 **Start-Template** 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.

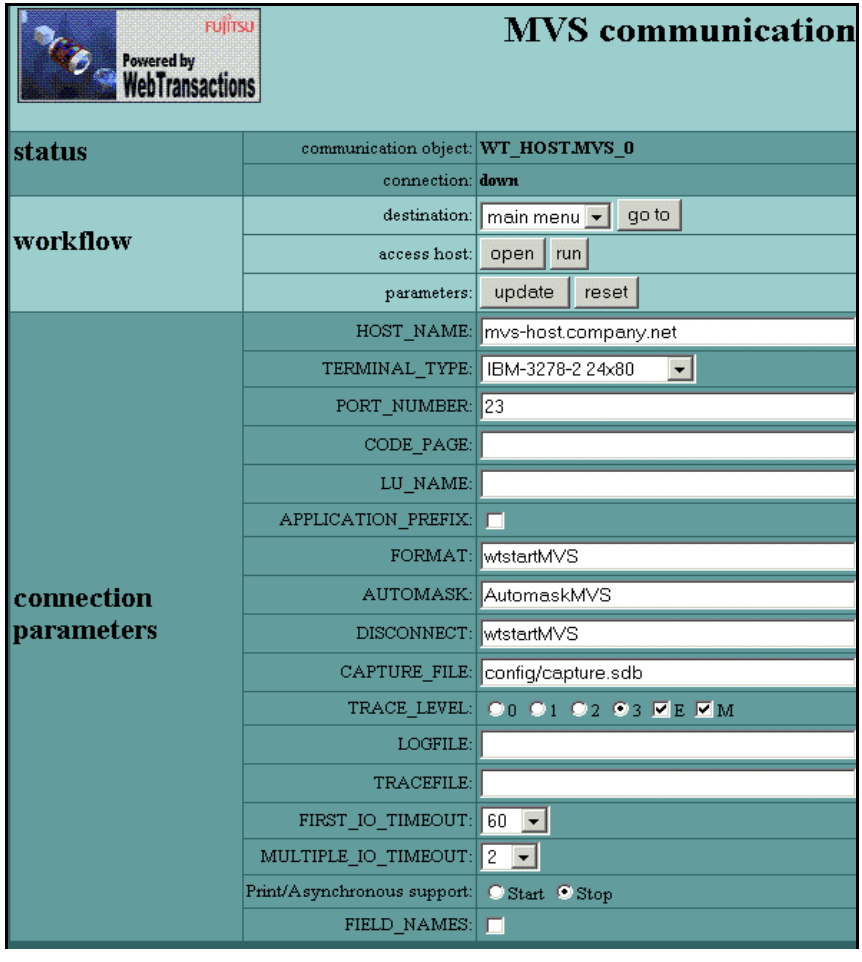
► Starten Sie die Sitzung mit **OK**. Das Dialogfeld wird geschlossen. Der eingestellte Browser wird geöffnet und veranlasst, eine neue WebTransactions-Sitzung mit dem Start-Template `wtstart` aufzurufen. `wtstart` präsentiert im Browser-Fenster ein Formular.



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

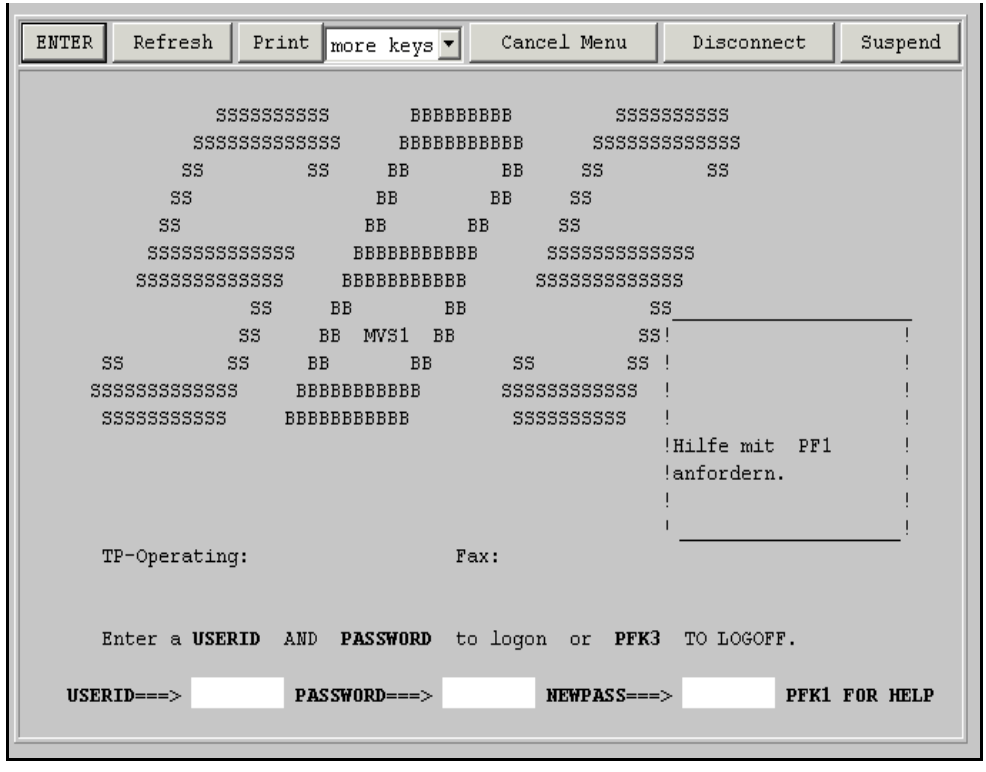
- Wählen Sie in der **PROTOCOL**-Auswahlliste den Eintrag **MVS**.
- Geben Sie den Namen des Kommunikationsobjekts an, hier *MVS_0*. Der Name des Kommunikationsobjekts muss mit dem Namen übereinstimmen, den Sie bei der Generierung des Automask-Templates verwendet haben.
- Klicken Sie nun auf den Knopf **create**, um ein neues Kommunikationsobjekt einzurichten. Ihre Angaben werden von WebTransactions verarbeitet. Das MVS-spezifische Start-Template `wtstartMVS.htm` übernimmt die Kontrolle und zeigt das nächste Formular.

Das MVS-spezifische Start-Template enthält den Parameter **HOST_NAME** für die Verbindung zur Host-Anwendung.



Mit `wtstartMVS` setzen Sie die Verbindungsparameter und öffnen die Verbindung zur Host-Anwendung, so als ob Sie sich von einem Terminal oder einer Emulation aus mit der Host-Anwendung verbinden würden.

- Tragen Sie unter **HOST_NAME** den Namen oder die IP-Adresse des Host-Rechners ein.
- Aktivieren Sie den Knopf **run**, um die Verbindung zur Host-Anwendung zu öffnen. Es wird das erste Format der MVS-Anwendung mit `AutomaskMVS.htm` ausgegeben.

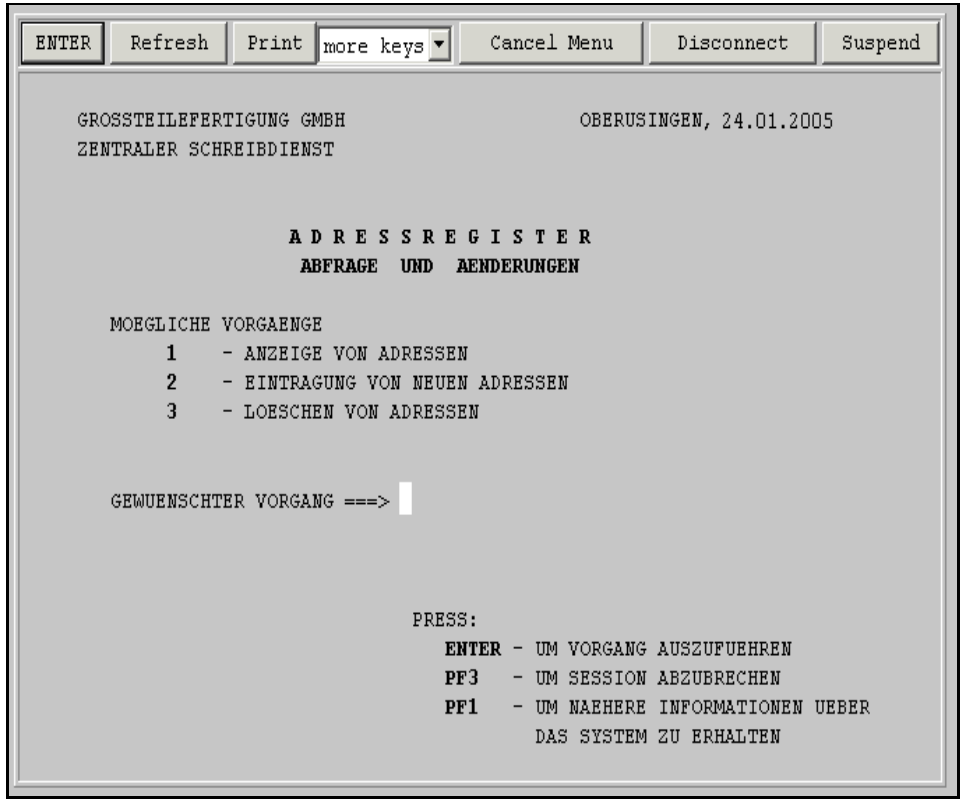


Das `AutomaskMVS.htm`-Template stellt Ihnen zur Kommunikation mit der MVS-Anwendung eine Knopfleiste zur Verfügung, die die Sondertasten des 3270-Terminals abbildet (siehe Abschnitt „Varianten von AutomaskMVS.htm erstellen“ auf Seite 68).

Wenn Sie jetzt die Verbindung zum Host beenden wollen, wählen Sie den Knopf **Disconnect**. Es wird wieder auf das Template `wtstartMVS` verzweigt (siehe Abschnitt „MVS-spezifisches Start-Template des Start-Template-Sets (wtstartMVS.htm)“ auf Seite 138). 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 der Beispielsitzung fahren Sie wie folgt fort:

- Geben Sie Ihre Kennung und Passwort ein, um sich in die Host-Anwendung *adtr* einzuloggen.
- Starten Sie die *adtr*-Host-Anwendung mit **Enter**. Das nächste Format der *adtr*-Host-Anwendung wird im Browser angezeigt.



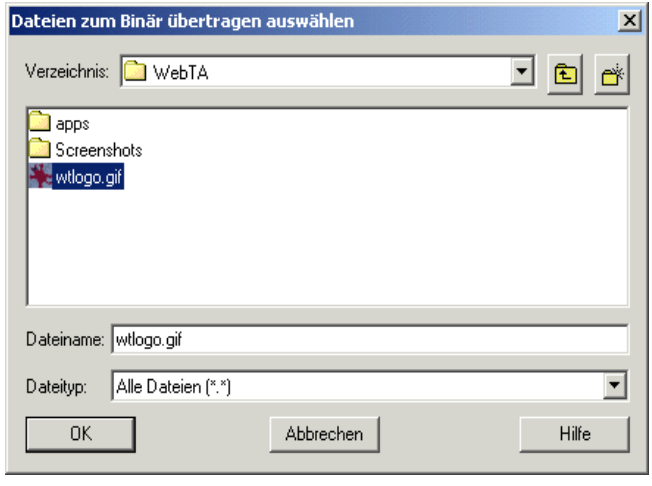
3.3 Globale Änderung der Darstellung

Als Beispiel für eine globale Änderung soll ein Firmenlogo in `AutomaskMVS.htm` integriert werden. Dazu müssen Sie vorher das Logo in ein Verzeichnis übertragen, auf das der Web-Server zugreifen kann. Sie können dazu das Verzeichnis `wwwdocs/image` im Basisverzeichnis nutzen.

Bild binär übertragen

Um das Firmenlogo mit WebLab in das Verzeichnis `wwwdocs/image` im Basisverzeichnis zu übertragen, gehen Sie folgendermaßen vor:

- Wählen Sie den Befehl **Datei/binär übertragen**. Das Dialogfeld **Dateien zum Binär übertragen auswählen** wird am Bildschirm eingeblendet.

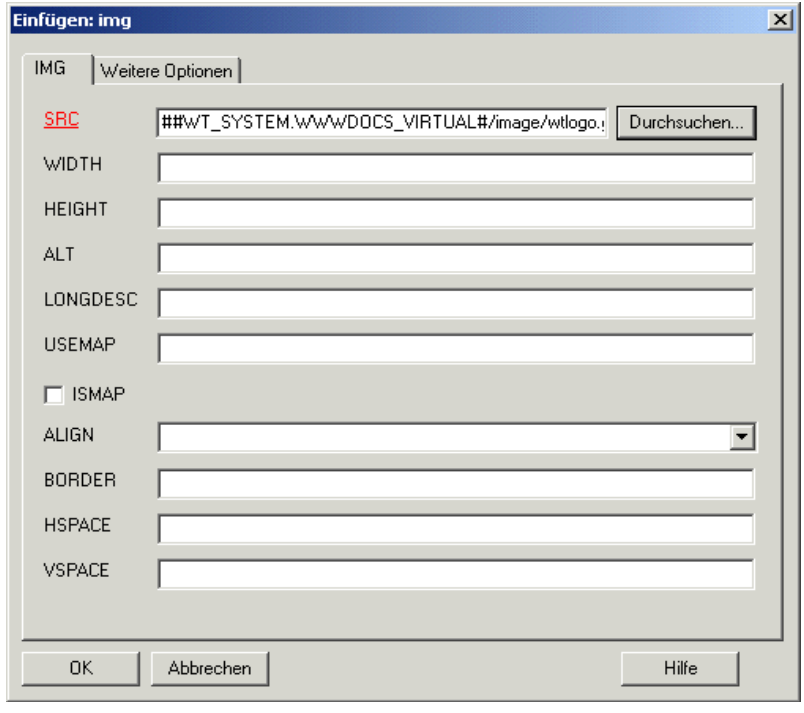


- Wählen Sie in diesem Dialogfeld das Verzeichnis, in dem Sie das Firmenlogo gespeichert haben und bestätigen Sie mit **OK**. Das Dialogfeld **Zielverzeichnis auswählen** wird am Bildschirm eingeblendet.
- Wählen Sie hier das Verzeichnis `wwwdocs/image` im Basisverzeichnis Ihrer WebTransactions-Anwendung und bestätigen Sie mit **OK**. Das Firmenlogo wird in das Basisverzeichnis übertragen. Physikalisch existiert dieses Verzeichnis unter dem Dokumentenverzeichnis des Web-Servers.

Logo in Automask-Template einfügen

Als globale Änderung soll in diesem Beispiel das WebTransactions-Logo eingefügt werden.

- Wählen Sie dazu den Befehl **Datei/Aktuelles Template öffnen.**
  - Damit laden Sie in den Arbeitsbereich von WebLab das Template `AutomaskMVS.htm`, mit dem das aktuelle Format der Host-Anwendung ausgegeben wird.
  - Außerdem aktualisieren Sie im Objektfenster von WebLab den entsprechenden Objektbaum mit allen aktuellen Variablen.
- Blättern Sie dann im Template bis zum BODY-Tag.
- Fügen Sie nach dem BODY-Tag eine leere Zeile ein und wählen Sie den Befehl **Einfügen/HTML/Bild ...**. Es wird das Dialogfeld **Einfügen:img** am Bildschirm eingeblendet.



In diesem Dialogfeld können Sie die Bilddatei und weitere Parameter für die Darstellung und Ausrichtung des Bildes angeben. Der Pfad für die Bilddatei muss relativ zum Dokumentenverzeichnis des Web-Servers liegen, da der Web-Server nur in diesem Verzeichnis nach Bildern sucht. Dies ist mit der Übertragung des Bildes in das Verzeichnis `wwwdocs/image` gewährleistet.

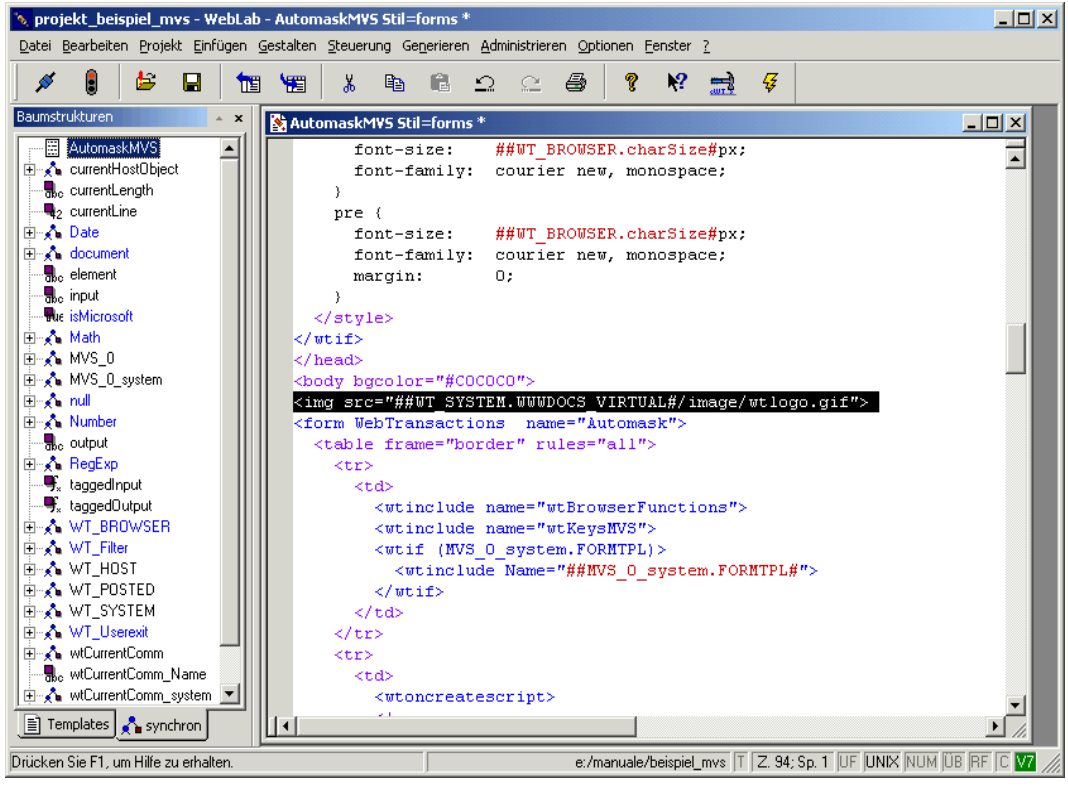
Um auf das Bild zuzugreifen, ohne einen langen Pfadnamen angeben zu müssen, können Sie das Systemobjekt-Attribut `WWWDOCS_VIRTUAL` nutzen. In `WWWDOCS_VIRTUAL` steht bereits der gesamte Pfad vom Dokumentenverzeichnis des Web-Servers bis zum Verzeichnis `wwwdocs` im Basisverzeichnis.

- Verwenden Sie den Knopf **Durchsuchen**, um die Bilddatei auszuwählen. Wenn sie sich unterhalb von `wwwdocs` befindet, verwendet WebLab automatisch `WWWDOCS_VIRTUAL` und der Name wird wie folgt erzeugt:

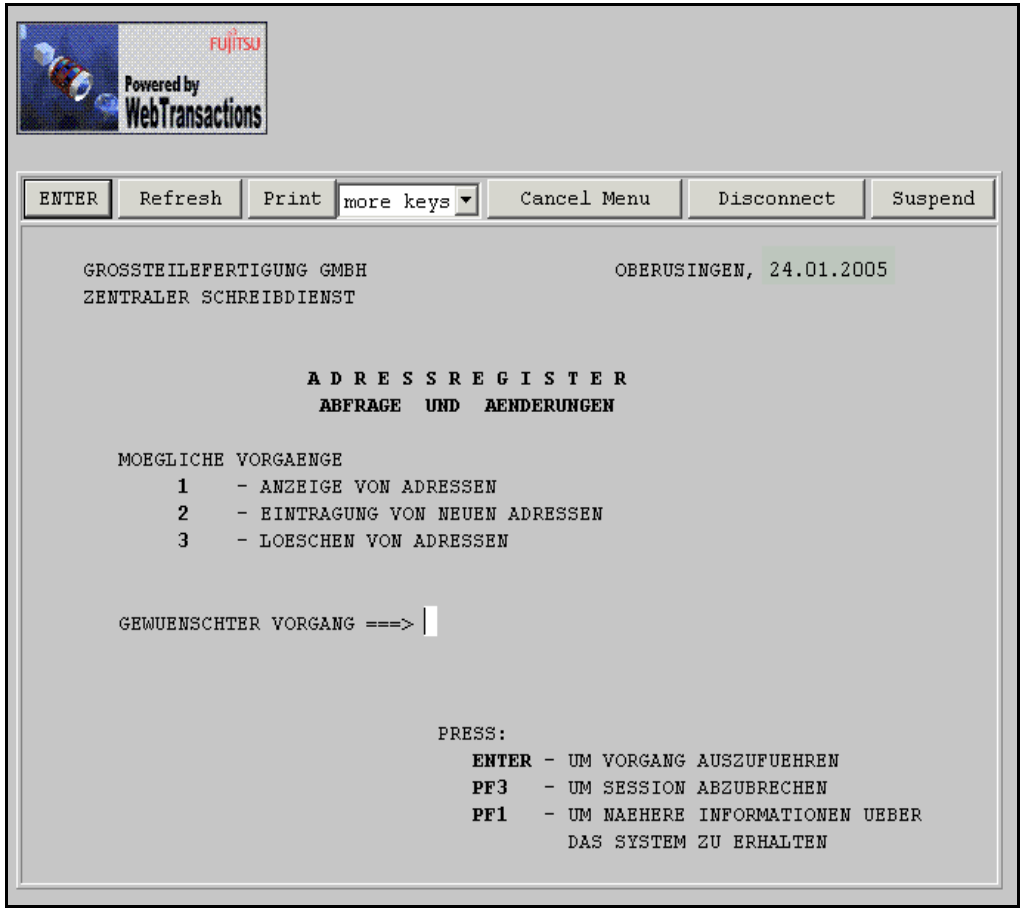
  `##WT_SYSTEM.WWWDOCS_VIRTUAL#/image/wtlogo.gif`

- Bestätigen Sie Ihre Angaben mit **OK**. WebLab fügt nach dem BODY-Tag folgende Zeile in das Automask-Template ein:

  `<img src="##WT_SYSTEM.WWWDOCS_VIRTUAL#/image/wtlogo.gif">`



► Um sich die Änderung anzusehen, wählen Sie den Befehl **Steuerung/Im Browser aktualisieren**. Die geänderte Darstellung im Automask-Template wird im Browser-Fenster ausgegeben. Wenn Sie das geänderte Automask-Template abspeichern, gilt diese Änderung für alle weiteren Formate, da das Template Automask die automatische Umsetzung für alle Formate regelt.

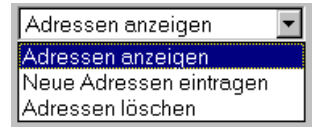


3.4 Formatspezifische Änderung der Darstellung

Wie Sie gesehen haben, wirken sich Änderungen im Automask-Template auf die Darstellung aller Formate im Browser aus. Wenn Sie nun die Änderung der Darstellung im Browser nur auf ein Format beziehen wollen, brauchen Sie ein so genanntes formatspezifisches Template. Dazu sind folgende Schritte erforderlich:

1. Formatspezifisches Template mit dem Capture-Verfahren generieren
2. Formatspezifisches Template bearbeiten

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**
MOEGLICHE VORGAENGE 1 - ANZEIGE VON ADRESSEN 2 - EINTRAGUNG VON NEUEN ADRESSEN 3 - LOESCHEN VON ADRESSEN	

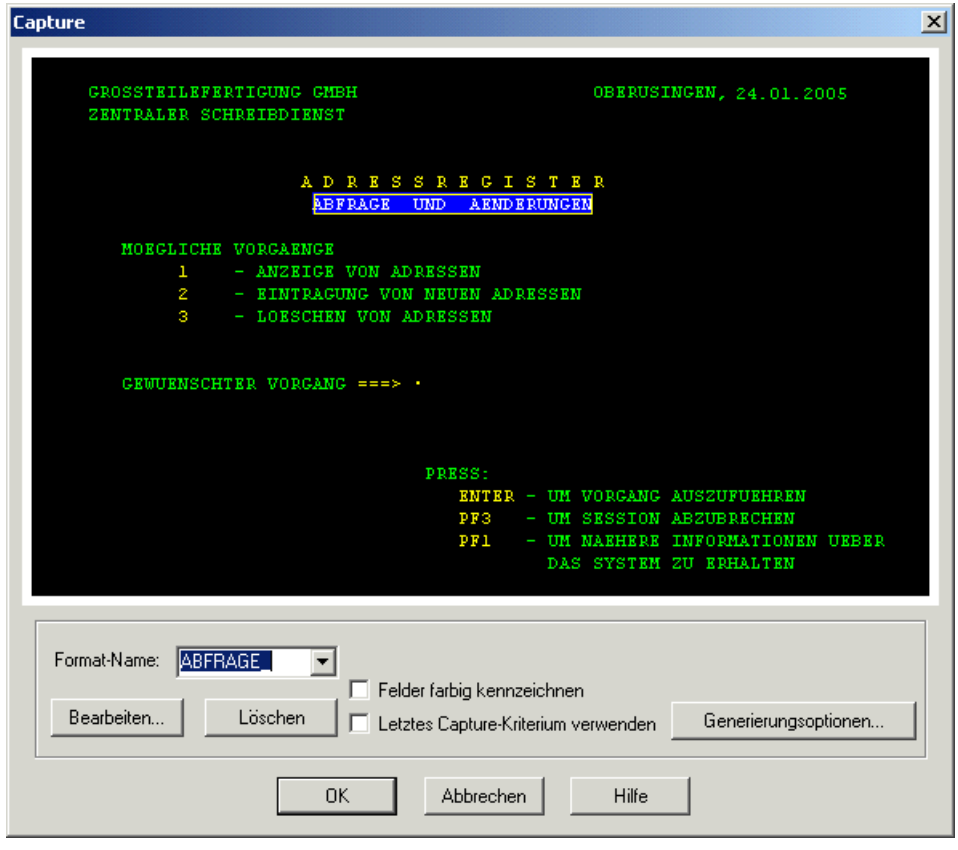
3.4.1 Formatspezifisches Template mit dem Capture-Verfahren generieren

Um individuelle Templates für die Formate der Host-Anwendung zu erstellen, benutzen Sie das Capture-Verfahren in WebLab. Hiermit erstellen Sie interaktiv Erkennungskriterien für die einzelnen Bildschirmformate, d.h. Muster zur Erkennung bekannter Formate.

► Wählen Sie dazu den Befehl **Generieren/Capture/aktuellen Schirm**.

Im Start-Template `wtstartMVS.htm` ist für das kommunikationsspezifische Systemobjekt-Attribut `CAPTURE_FILE` als Vorbelegung der Pfadname `config/capture.sdb` eingetragen (siehe Abschnitt „Sitzung starten“ auf Seite 38). Die Vorbelegung wird auch in diesem Beispiel verwendet. Die Capture-Datenbank wird beim ersten Zugriff angelegt. Hier werden alle Erkennungskriterien abgelegt, die Sie mit dem Capture-Verfahren erstellen.

Das Dialogfeld **Capture** wird mit der Darstellung des aktuellen Formats am Bildschirm eingeblendet.



- Ziehen Sie mit der Maus ein Rechteck über den unterlegten Teil *ABFRAGE UND ÄNDERUNGEN*. Damit legen Sie fest, dass dieses Format daran erkannt werden soll, dass *ABFRAGE UND ÄNDERUNGEN* an dieser Stelle im Format steht.
- Aktivieren Sie die Generierung mit **OK**. Das so ausgewählte Erkennungskriterium wird mit dem Formatnamen in der Capture-Datenbank gespeichert und das formatspezifische Template *ABFRAGE_* erstellt. Statt des Automask-Templates wird ab jetzt das formatspezifische Template zur Darstellung dieses Formats verwendet.

Generiertes Template

Der folgende Text zeigt den Abschnitt aus dem generierten Template *ABFRAGE_*.htm, der die Felder des Formats darstellt. Den Aufbau eines vollständigen Templates entnehmen Sie dem Abschnitt „Aufbau von AutomaskMVS.htm“ auf Seite 71.

Das generierte Template *ABFRAGE_*.htm wurde mit den folgenden Generierungsoptionen erzeugt: **Generierungsmethode: Inline-Script**, **Darstellungsattribute: Keine**.

```
       <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
       <!-- begin of host screen section                                               -->
       <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
<div style="color:##MVS_0.WT_Color.Default = "\#000000"#"><pre>\
```

Beim Capture-Verfahren wird jedem Feld eines Formats ein Host-Objekt zugeordnet. Die einzelnen Host-Objekte werden nacheinander auf dem Bildschirm ausgegeben. Der Auswertungsoperator *##objektname*.HTMLValue# sorgt bei Ausgabefeldern dafür, dass der Inhalt auf dem Bildschirm dargestellt wird. Zur einfacheren Orientierung im Template wird der Inhalt zum Zeitpunkt des Capture vor dem Auswertungsoperator in einem Kommentar gespeichert.

```
<span class="screenline" id="SL1"><wtrem **   **>\
##MVS_0.E_01_001_001.HTMLValue#\
<wtrem **
**>\
##MVS_0.E_01_002_079.HTMLValue#</span>
<span class="screenline" id="SL2"><wtrem **      **>\
##MVS_0.E_02_001_004.HTMLValue#\
<wtrem **   **>\
##MVS_0.E_02_005_001.HTMLValue#\
<wtrem ** GROSSTEILEFERTIGUNG GMBH                    **>\
##MVS_0.E_02_006_044.HTMLValue#\
<wtrem **   **>\
##MVS_0.E_02_050_001.HTMLValue#\
<wtrem ** OBERUSINGEN, **>\
##MVS_0.E_02_051_012.HTMLValue#\
...
<wtrem ** GEWUENSCHTER VORGANG **>\
##MVS_0.E_15_009_020.HTMLValue#\
<wtrem **   **>\
##MVS_0.E_15_029_001.HTMLValue#\
<wtrem ** ===&#62; **>\
##MVS_0.E_15_030_004.HTMLValue#\
<wtrem **   **>\
##MVS_0.E_15_034_001.HTMLValue#\
```

Eingabefelder des Formats werden durch Input-Tags vom Typ `text` dargestellt, falls das ursprüngliche Feld ungeschützt war. Falls das ursprüngliche Feld geschützt war (Eingabe unsichtbar), hat der Tag den Typ `password`. Die Angabe `value="##`*objektname*`.Value#"` sorgt dafür, dass das Eingabefeld mit dem Wert des Feldes aus dem Format versorgt wird.

```
<input type="##(MVS_0.E_15_035_001.Visible == 'No') ? 'password' : 'text'#" ##(
WT_BROWSER.acceptClass ) ? 'class="box" style="width:9px"' : ''# name="E_15_035_001"
size="1" maxlength="1" value="##MVS_0.E_15_035_001.Value#"/>\
<wtrem **   **>\
...
<wtrem ** - UM NAEHERE INFORMATIONEN UEBER     **>\
##MVS_0.E_22_045_036.HTMLValue#</span>
<span class="screenline" id="SL23"><wtrem **
**>\
##MVS_0.E_23_001_045.HTMLValue#\
<wtrem **   **>\
##MVS_0.E_23_046_001.HTMLValue#\
<wtrem ** DAS SYSTEM ZU ERHALTEN             **>\
##MVS_0.E_23_047_034.HTMLValue#</span>
<span class="screenline" id="SL24"><wtrem **
**>\
##MVS_0.E_24_001_080.HTMLValue#</span>
```

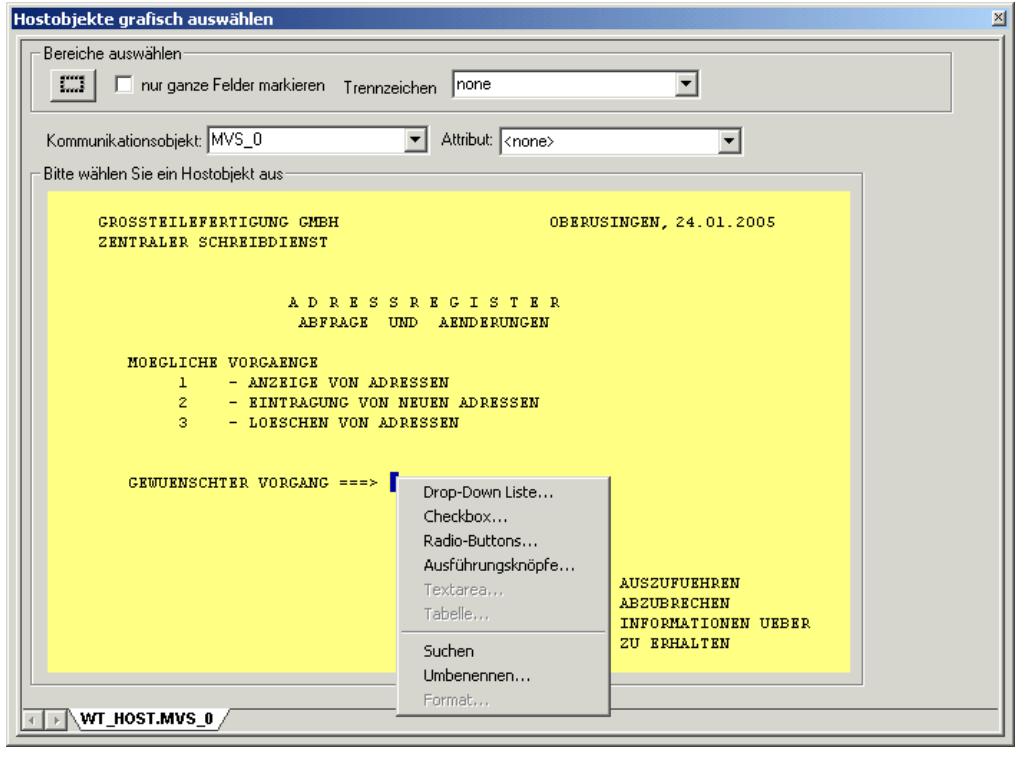
Alle Eingabefelder werden in einem Objekt verwaltet.

```
<wtoncreatescript>
<!--
  wtInputFields = {E_15_035_001:MVS_0.E_15_035_001};
//-->
</wtoncreatescript></pre></div>
        <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
        <!-- end of host screen section                                               -->
        <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  -->
</td>
```

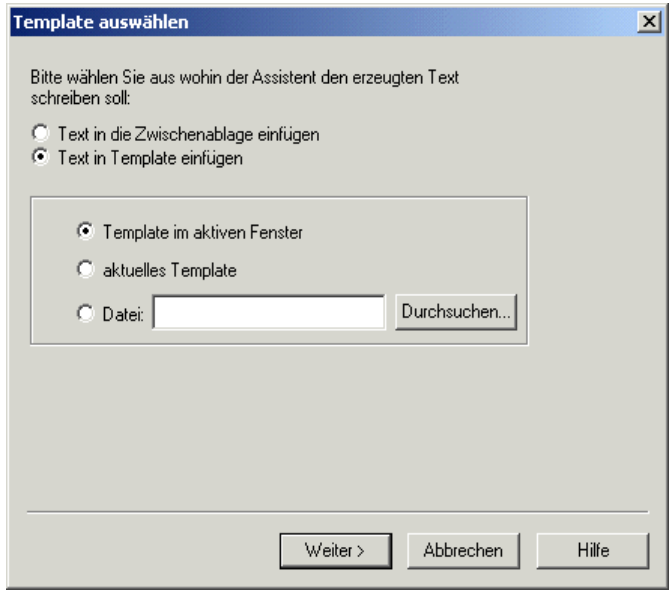
3.4.2 Formatspezifisches Template bearbeiten

- Wählen Sie den Befehl **Datei/aktuelles Template öffnen**, um das formatspezifische Template *ABFRAGE_* 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. Das einzige Eingabefeld dieses Formats ist mit der Farbe weiß hinterlegt.
- Positionieren Sie den Mauszeiger auf dieses Eingabefeld (durch die Selektion wird das Feld blau) und öffnen Sie das Kontextmenü mit der rechten Maustaste.

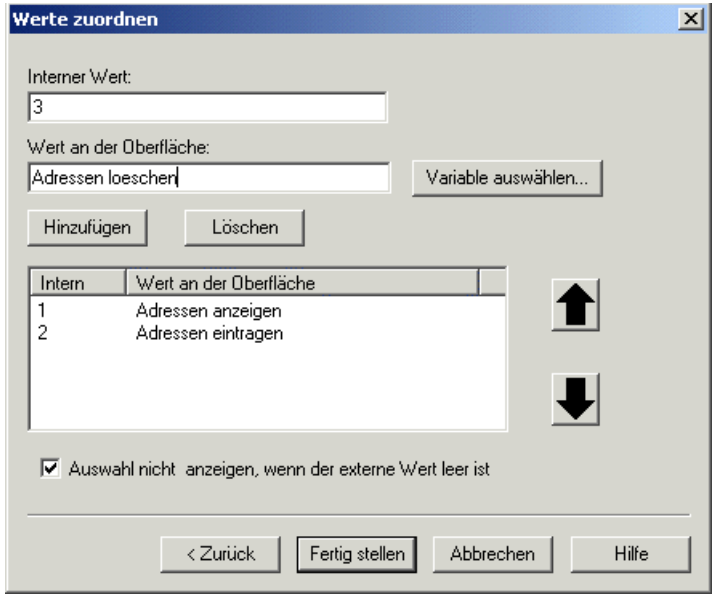


- Wählen Sie aus dem Kontextmenü den Befehl **Drop Down Liste**. Das Dialogfeld **Template auswählen** wird am Bildschirm eingeblendet. Dieses Dialogfeld ist das erste eines Assistenten, der Sie beim Erstellen einer Liste unterstützt.



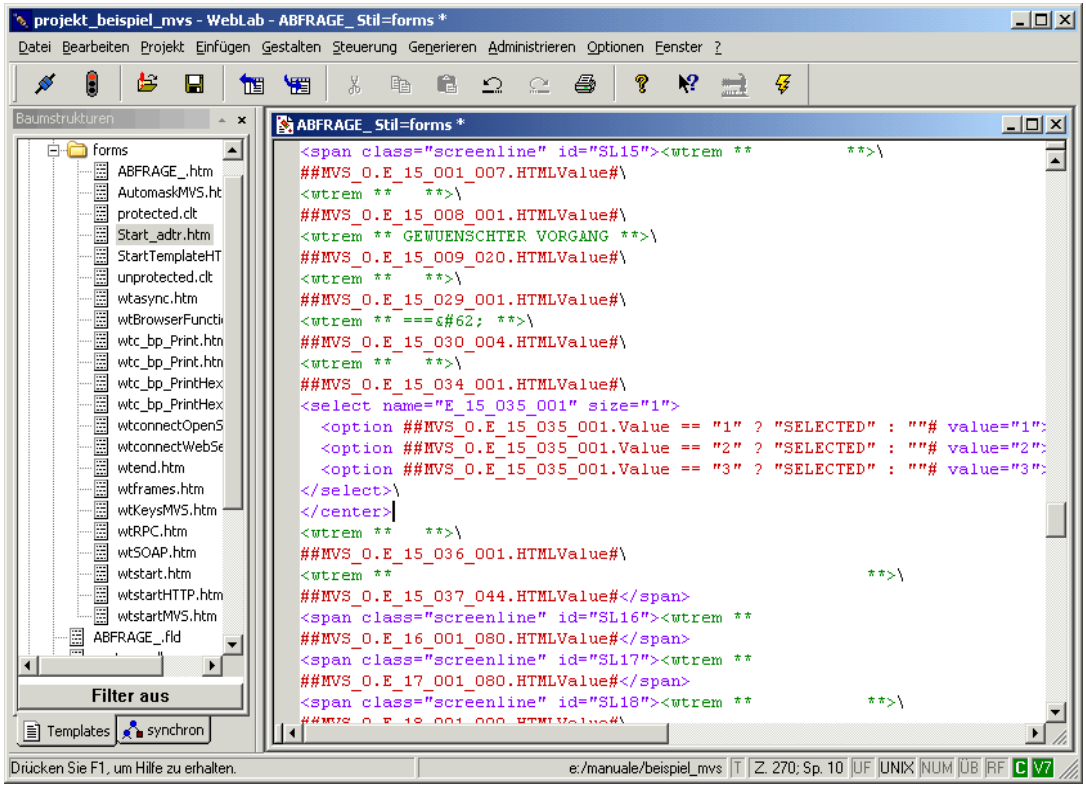
In diesem Dialogfeld legen Sie fest, in welches Template die Liste eingefügt werden soll. Die Option **Template im aktiven Fenster** ist bereits voreingestellt. Wenn Sie das aktive Template nicht vorher geöffnet haben, wählen Sie die Option **aktuelles Template**.

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



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.

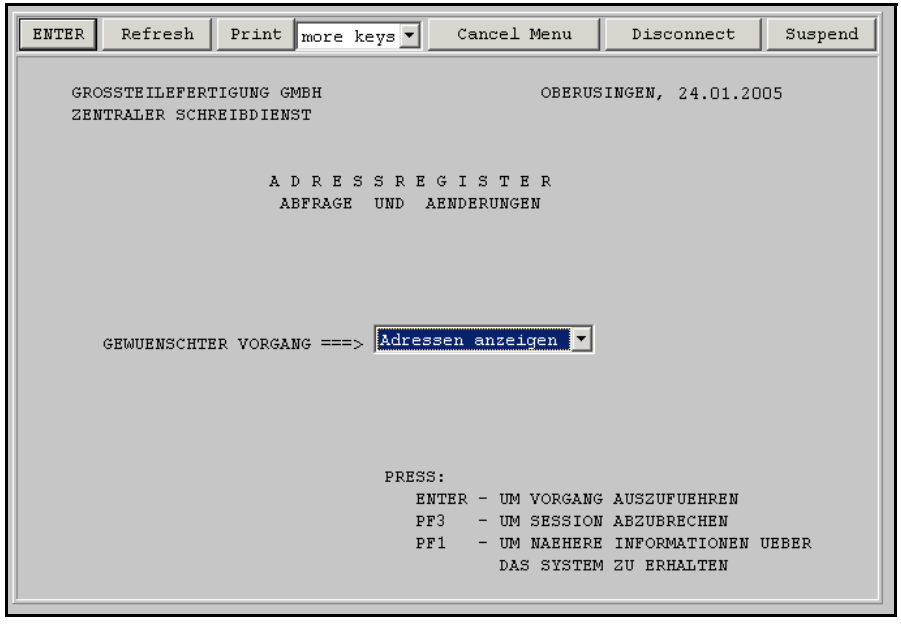
- Tragen Sie die internen Werte mit den entsprechenden Beschreibungen (siehe Abbildung auf Seite 46) in die Eingabefelder ein. Mit der Schaltfläche **Hinzufügen** übernehmen Sie ein solches Wertepaar in die Liste.
- Bestätigen Sie am Ende der Liste Ihre Eingaben mit **Fertigstellen**. Der entsprechende HTML-Code für die Umsetzung einer Liste wird mit den angegebenen Werten in das Template *ABFRAGE_*.htm eingefügt.
- Um sich diese Ersetzung anzusehen, blättern Sie im Template *ABFRAGE_*.htm, bis Sie zum Host-Abschnitt gelangen. Dieser Abschnitt beginnt mit dem Kommentar `begin of host screen section`.



Das formatspezifische Template ist analog zum Automask-Template aufgebaut. Alle Felder des Host-Formats werden im Host-Abschnitt mit Namen aufgeführt, wobei sich jeder Name aus der Zeilen- und Spaltenposition zusammensetzt. Die obersten Felder sorgen für die Darstellung des Kopfes, die neue Liste (SELECT-Tag) entspricht dem alten Eingabefeld und nimmt den ausgewählten Wert entgegen. Die weiteren Felder gestalten die restliche Darstellung, so wie Sie sie im Browser bzw. der grafischen Hostobjektauswahl sehen.

Da die Auswahl jetzt über die Liste erfolgt, sind die erläuternden Texte vor der Drop-Down-Liste nicht mehr notwendig, d.h. Sie können sie aus dem Template *ABFRAGE_*`.htm` löschen.

- Löschen Sie alle Felder des Formats von der Überschrift bis zur Liste. Dabei können Sie mit dem Befehl **Suchen** im Kontextmenü des Dialogfelds **Hostobjekte grafisch auswählen** die relevanten Felder im Template suchen lassen.
- Wählen Sie den Befehl **Steuerung/Im Browser aktualisieren**, um sich das Ergebnis Ihrer Änderung anzusehen.



Wollen Sie die Verbindung zum Host beenden, wählen Sie den Knopf **Disconnect**. Es wird zum Template `wtstartMVS.htm` verzweigt (siehe Abschnitt „MVS-spezifisches Start-Template des Start-Template-Sets (wtstartMVS.htm)“ auf Seite 138). Mit der Auswahl **main menu** und dem Knopf **go to** gelangen Sie in das allgemeine Start-Template. Hier können Sie mit **Quit** die WebTransactions-Anwendung verlassen.

3.5 WebTransactions-Anwendung starten

Eine bearbeitete WebTransactions-Anwendung starten Sie auf die gleiche Weise in WebLab wie die automatische 1:1-Umsetzung (siehe Abschnitt „Sitzung starten“ auf Seite 38). Einziger Unterschied: Sie sollten sich vergewissern, dass im Template `wtstartMVS.htm` beim Parameter **CAPTURE_FILE** der Pfadname der Formate-Datenbank (in diesem Beispiel: `config/capture.sdb`) korrekt eingetragen ist.

Sie können aber auch für die integrierte Host-Anwendung ein eigenes Start-Template erstellen, das den Benutzer direkt zum ersten Format der Host-Anwendung bringt.

3.5.1 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 zum WebTransactions-Server bestehen.

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

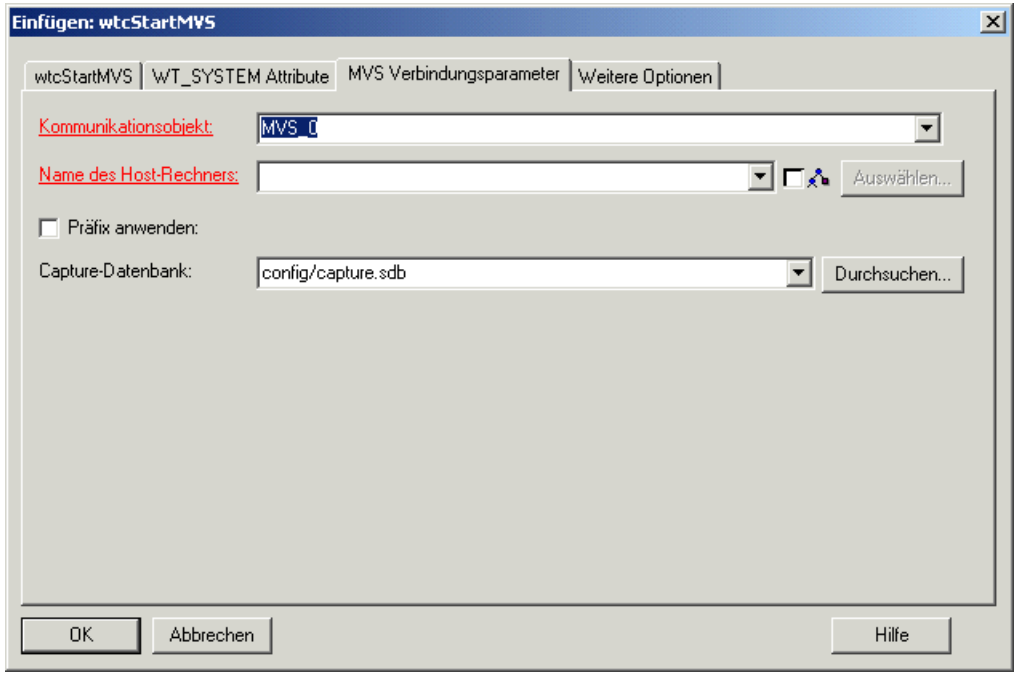
  In der Registerkarte **wtcStartMVS** legen Sie den Namen und das Verzeichnis für das Start-Template fest. Der Dateiname ist mit `config/forms/startMVS.htm` vorbelegt.

- Geben Sie unter **Dateiname** das Verzeichnis und den Namen des Start-Templates ein, hier *config/forms/Start_adtr.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 **MVS-Verbindungsparameter**.



Die wichtigsten Einstellungen sind die Namen des Kommunikationsobjekts, der Host-Anwendung und der Capture-Datenbank.

► Geben Sie den Namen für das Kommunikationsobjekt ein, hier *MVS_0*.

Beachten Sie, dass der Name des Kommunikationsobjekts mit den Namen in der Automask übereinstimmen muss.

► Geben Sie den Namen oder die IP-Adresse des Host-Rechners ein.

► Geben Sie den Namen der Capture-Datebank ein, hier `config/capture.sdb`. Mit **Durchsuchen** können Sie die Capture-Datenbank in der Dateiauswahlbox auch interaktiv suchen.

► Wählen Sie die Registerkarte **Weitere Optionen**.

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

► Setzen Sie die erforderlichen Eigenschaften für Ihre Host-Anwendung. Für die Beispielanwendung sind keine weiteren Änderungen erforderlich.

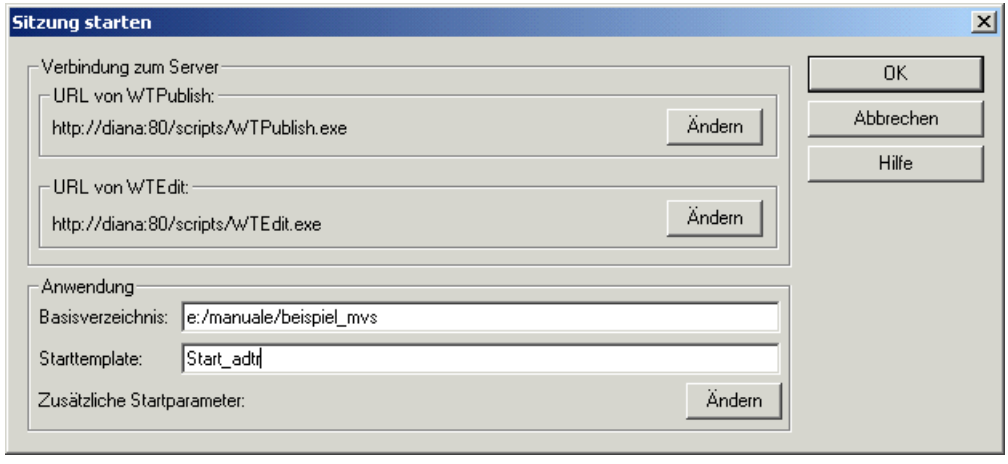
- Wählen Sie **OK**. Das Dialogfeld **Einfügen:wtcStartMVS** wird geschlossen. Das Start-Template *Start_adtr.htm* wird generiert und im Arbeitsbereich von WebLab angezeigt. Gespeichert wird das Start-Template im Basisverzeichnis unter `config/forms`.

Das Start-Template *Start_adtr.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 Abschnitt „Sitzung starten“ auf Seite 38 mit `wtstart.htm` und `wtstartMVS.htm` beschrieben wurde.

3.5.2 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 im hier im Eingabefeld **Start-Template** den Namen des anwendungs-spezifischen Start-Templates ein.

► 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.

3.5.3 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 dem 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_mvs?Start_adtr
```

4 Basisverzeichnis anlegen und WebTransactions-Anwendung starten

Nach der Installation von WebTransactions auf dem 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 30).

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 erstellen** 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 Checkbox **MVS**
- Klicken Sie **OK**.

► Tragen Sie im Dialogfeld **Automask generieren** die gewünschten Optionen ein. Diese entsprechen den Optionen für die Generierung formatspezifischer Templates.

► Drücken Sie auf **Generieren**. 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“ 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 WebTransactions-Anwendung starten

Wenn Sie keine individuellen Änderungen an einzelnen Formaten vornehmen wollen, sondern ausschließlich die im Kapitel „Host-Anwendung ohne Bearbeitung integrieren“ auf Seite 65 beschriebene automatische Umsetzung der Host-Formate verwenden, sind keine weiteren Schritte mehr notwendig. Nach Installation von WebTransactions (siehe Seite 11) und Einrichten eines Basisverzeichnisses (siehe Seite 61) ist Ihre WebTransactions-Anwendung bereits einsatzfähig. Für den Start der Anwendung steht Ihnen z.B. das fertig ausgelieferte Start-Template-Set zur Verfügung, das in Abschnitt „Start-Templates für MVS“ auf Seite 137 beschrieben ist.

Für den Produktivbetrieb sollten Sie sich mit Hilfe des WTBeans `wtcStartMVS.wtc` in WebLab ein eigenes Start-Template erzeugen.

5 Host-Anwendung ohne Bearbeitung integrieren

Für die dynamische Umsetzung der 3270-Formate auf die Oberfläche eines Web-Browsers stellt WebTransactions for MVS Umsetzungs-Templates zur Verfügung. Diese ermöglichen im Zusammenspiel die Bedienung der Host-Anwendung sehr nahe am Look & Feel der gewohnten Bedienung an einem realen Terminal oder einer Emulation:

- `MVS.wmt/MVS_pocket.wmt`
  MVS-spezifische Master-Templates für ein generelles Layout der Formulare. Diese Templates werden bei der Generierung des Automask-Templates und der formatspezifischen Templates verwendet.
- `AutomaskMVS.htm`
  Umsetzungs-Template für MVS-Anwendungen, das in der Lage ist, alle Formate einer MVS-Anwendung automatisch in eine HTML-Seite mit einem Formular umzusetzen.
- `wtKeysMVS.htm`
  Dieses Template fügt Bedienelemente ein, die Tasten des 3270-Terminals abbilden. `wtKeysMVS.htm` wird nicht nur bei der dynamischen Umsetzung verwendet. Auch die von WebLab mit dem Capture-Verfahren erstellten formatspezifischen Templates nutzen `wtKeysMVS.htm`.
- `wtBrowserFunctions.htm`
  Dieses Template sorgt für die Tastatur-Unterstützung der Browser, damit der Benutzer mit dem Browser die Formate der Host-Anwendung ebenso bearbeiten kann, als würde er mit einer Emulation oder einem Terminal arbeiten.

Bis auf das Master-Template werden die Umsetzungs-Templates beim Einrichten eines Basisverzeichnisses automatisch im Verzeichnis *basedir*`/config/forms` angelegt. Mit WebLab lassen sich auch Varianten des Templates `AutomaskMVS.htm` erstellen (siehe Seite 68).

Globale Anpassungen, z.B. für eine Corporate-Identity, können Sie im Template `AutomaskMVS.htm` durchführen. Die Anpassungen werden dann automatisch für alle Bildschirme der Host-Anwendung wirksam. Globale Anpassungen, die Sie im Master-Template durchführen, werden wirksam, wenn Sie mit Hilfe dieses Master-Templates wiederum Templates generieren (Automask- oder individuelle Templates). Das Master-Template ist im Verzeichnis `weblab` des Installationsverzeichnisses von WebLab abgelegt.

Wenn anwendungsspezifische Änderungen im Master-Template durchzuführen sind, ist es sinnvoll, das Master-Template vom WebLab-Rechner auf den WebTransactions-Server in das Basisverzeichnis zu kopieren und dort die Änderungen vorzunehmen, damit das Master-Template Bestandteil des Basisverzeichnisses wird.

5.1 Master-Templates MVS.wmt und MVS_Pocket.wmt

Master-Templates werden von WebTransactions bei der Generierung der Automask und 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 oder wenn Sie nur ausgewählte formatspezifische Templates generiert haben und die seltener verwendeten Formate durch das Automask-Template umsetzen lassen.

In solchen Fällen genügt es, den Aufbau einmal im Master-Template festzulegen und dieses Master-Template bei der Generierung sowohl der formatspezifischen Templates als auch des Automask-Templates als Schablone zuzuweisen. Alle generierten Templates erhalten dann automatisch den gewünschten Aufbau.

Für WebTransactions for MVS werden die vordefinierten Master-Templates `MVS.wmt` und `MVS_Pocket.wmt` mit ausgeliefert, die Sie individuell anpassen, aber auch unverändert einsetzen können. Die vordefinierten Master-Templates enthalten bereits alle WTML-Tags und WTScripts, die für alle Templates der jeweiligen Produktvariante gleich sind, z.B. eine Prüfung, ob ein privates Systemobjekt 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 Einstellungen aus WebLab übersteuern die entsprechenden Einstellungen im Master-Template.

Einsatz von MVS.wmt oder MVS_pocket.wmt

Das Master-Template `MVS_Pocket.wmt` wurde speziell für den Einsatz auf dem Pocket Internet Explorer entwickelt. Sie können es für die Generierung eines Automask-Template (siehe Abschnitt „Template AutomaskMVS.htm“ auf Seite 68) und für formatspezifische Templates (siehe Abschnitt „Capture-Verfahren mit WebLab“ auf Seite 82) verwenden.

`MVS_Pocket.wmt` bietet die folgenden zusätzlichen Funktionen:

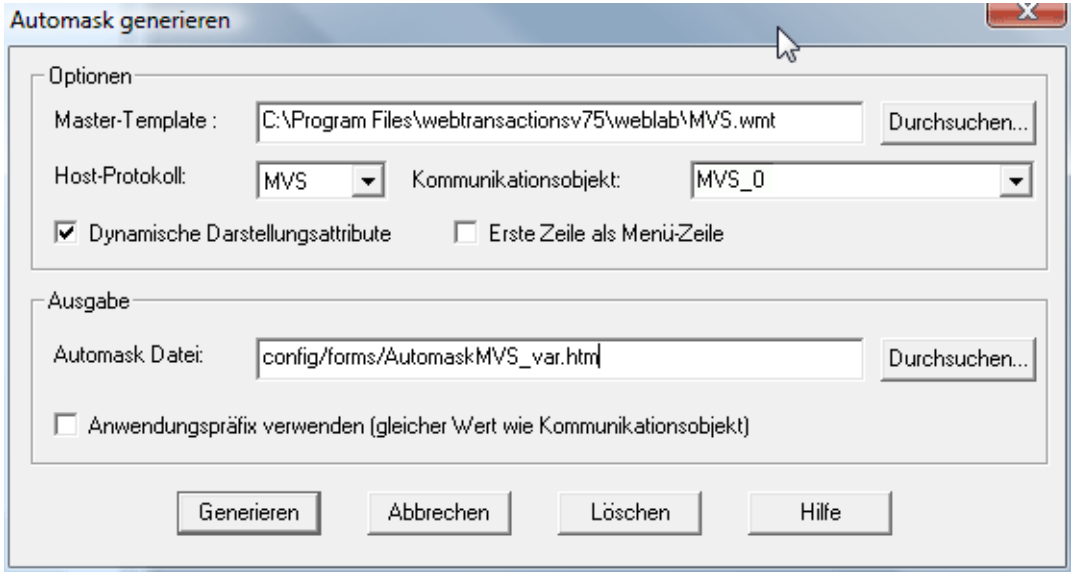
- Für den Bildschirm wird ein Frameset generiert. Es enthält zwei Frames, einen zur Kommando-Eingabe und einen für die Anzeige.
- Nur das Feld, in dem der Cursor steht, kann als Eingabefeld angesprochen werden.
- Die Zeilen auf dem Bildschirm werden automatisch umbrochen und farblich abgesetzt.
- Bei überlangen Ausgaben wird automatisch geblättert.
- Die Darstellung des Bildschirms kann zur Laufzeit geändert werden:
  - Eine 1:1-Darstellung erhalten Sie mit dem Befehl **Command**/**Screen**.
  - Die Darstellung wird auf Bildschirmbreite umbrochen mit **Command**/**Compact**. In diesem Fall wird versucht, Parameter und zugehörige Eingabefelder in einer Zeile darzustellen.

5.2 Template AutomaskMVS.htm

Das Template `AutomaskMVS.htm` erstellt dynamisch eine Abbildung des zuletzt vom Host mit `receive` empfangenen Formats. Es ist in der Lage, ohne Vorbereitung jedes beliebige 3270-Format zu verarbeiten und wird von WebTransactions immer dann verwendet, wenn kein formatspezifisches Template für das Host-Format vorhanden ist. Durch Steuerung über das Systemobjekt-Attribut `AUTOMASK` können Sie verschiedene Varianten einsetzen.

5.2.1 Varianten von AutomaskMVS.htm erstellen

Das Template `AutomaskMVS.htm` wird in der Regel beim Erzeugen eines Basisverzeichnisses automatisch angelegt. WebLab bietet Ihnen aber auch die Möglichkeit, Varianten mit unterschiedlichen Optionen zu generieren, die die unterschiedliche Gestaltung Ihrer WebTransactions-Anwendung optimal unterstützen. Wählen Sie hierfür den Befehl **Generieren/Automask**. Das Dialogfeld **Automask generieren** wird am Bildschirm eingeblendet.



In diesem Dialogfeld können Sie mit den Parametern steuern, wie das Automask-Template generiert wird. Die Parameter haben dabei folgende Bedeutung:

Master-Template

legt das Master-Template fest, das bei der Generierung des Automask-Template benutzt werden soll, siehe hierzu auch Abschnitt „Master-Templates MVS.wmt und MVS_Pocket.wmt“ auf Seite 66. Ein Master-Template wird immer benötigt. Ein Master-Template wird immer benötigt.
Voreinstellung ist das mitausgelieferte Master-Template `MVS.wmt`.

Host-Protokoll

legt das Protokoll fest, über das die Host-Anwendung mit dem Host-Adapter von WebTransactions kommuniziert

Kommunikationsobjekt

legt den Namen für das aktuelle Kommunikationsobjekt fest. Der Name des Kommunikationsobjekts ist vor allem dann wichtig, wenn Sie mehr als eine Host-Anwendung mit WebTransactions integrieren wollen. Der Name muss mit dem Namen im Start-Template übereinstimmen.
Voreinstellung: *MVS_0*.

Dynamische Darstellungsattribute

Alle Feld-Attribute werden unterstützt und zur Laufzeit aus den Hostobjekten gelesen.

Erste Zeile als Menüzeile

gibt an, ob das generierte WTML-Template die erste Zeile des Formats als Menüzeile unterstützt. Wenn Sie dieses Feld aktivieren, werden alle Text-Felder in der ersten Zeile des Bildschirms der Host-Anwendung als Knöpfe generiert.

Diese Option kann sinnvoll sein, um den Menücharakter der ersten Zeile deutlicher an der Oberfläche herauszustellen. Allerdings werden dann auch bei Formaten, die keine Menüfunktion in der ersten Zeile unterstützen, diese Knöpfe generiert.

Wenn es genügt die Schreibmarke mit der Maus (nicht mit den Cursor-Tasten) auf einen entsprechenden Menüeintrag auf dem Bildschirm zu positionieren, ist diese Option nicht nötig, um die gewünschte Funktionalität zu erreichen, und die Abbildung des Formats ist dem Original ähnlicher.
Das Positionieren der Schreibmarke mit der Maus auf geschützte Felder ist möglich, wenn in dem `%%LINES`-Tag des Master-Templates `CursorInProtectedField=Yes` gesetzt ist (Standard).

Automask Datei

legt das Verzeichnis und den Namen für das generierte Template fest.

Voreingestellter Pfad:
basedir`/config/forms/AutomaskMVS.htm`

Wenn Sie die Automask in einem anderen Verzeichnis unter `config` ablegen, können Sie Stil- und Sprachvarianten realisieren, siehe WebTransactions-Handbuch „Konzepte und Funktionen“.

Anwendungspräfix

Wenn Sie mehrere Host-Anwendungen mit möglicherweise gleichen Formatnamen integrieren wollen, kann WebLab vor den eigentlichen Dateinamen für das Template ein Präfix setzen. Der Dateiname setzt sich dann wie folgt zusammen:
commobj@formatname`.fld` bzw. *commobj@formatname*`.htm`

5.2.2 Aufbau von AutomaskMVS.htm

Im Folgenden ist das Automask-Template dargestellt, das mit dem Master-Template `MVS.wmt` generiert wurde (siehe Abschnitt „Master-Templates MVS.wmt und MVS_Pocket.wmt“ auf Seite 66).

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

```
<HTML>
<wtrem>*****************************************************************************</wtrem>
<wtrem>** WTML document: AutomaskMVS                                              **</wtrem>
<wtrem>*****************************************************************************</wtrem>
<wtrem>**                                                                         **</wtrem>
<wtrem>** Document generation based on Master Template :                          **</wtrem>
<wtrem>**  C:\Program Files\webtransactionsv75\weblab\MVS.wmt                     **</wtrem>
<wtrem>**                                                                         **</wtrem>
<wtrem>** Generated at Wed Jun 09 18:12:50 2010                                  **</wtrem>
<wtrem>**                                                                         **</wtrem>
<wtrem>** Options used by the generator :                                        **</wtrem>
<wtrem>**   -  %OPTIONS:                                                         **</wtrem>
<wtrem>**      CommObj = MVS_0                                                    **</wtrem>
<wtrem>**      NationalVariant = International - PartialFormatMode = No           **</wtrem>
<wtrem>**   -  %LINES:                                                           **</wtrem>
<wtrem>**      TaggedInput = Enabled - TaggedOutput = Enabled                     **</wtrem>
<wtrem>**      DisplayAttributes = Dynamic - CursorInProtectedField = Yes         **</wtrem>
<wtrem>**      MenuBar = No                                                       **</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.

```
<wtoncreatescript>
<!--
  //{{WebLab(assignCommunicationObject)
  MVS_0  = WT_HOST.active || WT_HOST.MVS_0;
  if (MVS_0.WT_SYSTEM != null)
    MVS_0_system = MVS_0.WT_SYSTEM;   // communication specific system object
  else
    MVS_0_system = WT_SYSTEM;        // global  system object
  //}}
  // propagate communication object to included WTML documents //////////////
  wtCurrentComm = MVS_0;
  wtCurrentComm_system = MVS_0_system;
  if ( wtCurrentComm_system.EDIT_MODE )
```

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

```
  {
    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;
//-->
</wtoncreatescript>
```

Ein ggf. vorhandenes PROLOG-Template wird ausgeführt.

```
<wtif (MVS_0_system.PROLOG)>
  <wtinclude Name="##MVS_0_system.PROLOG#">
</wtif>
```

Nach dem obligatorischen HTML-Grundgerüst wird mit dem `Style`-Tag die generelle Darstellung im Browser festgelegt. Mit Hilfe von `WT_BROWSER.charSize` kann die Zeichengröße festgelegt werden (siehe Abschnitt „Font-Größe im Attribut WT_BROWSER.charSize“ auf Seite 135).

```
<head>
<title>WebTransactions V7.5 - session ##MVS_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;
      margin:       0;
    }
  </style>
</wtif>
</head>
```

Anschließend wird ein Formular geöffnet (`<form>`), um einen Dialog mit dem Benutzer am Browser zu ermöglichen. Für die Bedienung des Dialogs werden mit `<wtInclude>` die Templates `wtKeysMVS.htm` und `wtBrowserFunctions.htm` aufgerufen zur Unterstützung einer möglichst genauen 1:1 Darstellung. Deren Bedienelemente werden somit Teil des Formulars.

```
<body bgcolor="#COCOCO">
<form WebTransactions  name="Automask">
  <table frame="border" rules="all">
    <tr>
      <td>
        <wtinclude name="wtBrowserFunctions">
        <wtinclude name="wtKeysMVS">
        <wtif (MVS_0_system.FORMTPL)>
          <wtinclude Name="##MVS_0_system.FORMTPL#">
        </wtif>
      </td>
    </tr>
```

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

```
    <tr>
      <td>
        <wtoncreatescript>
        <!--
          function taggedInput( hostObject )
          {
            if ( hostObject.Type == 'Protected' )
            {
```

```
              taggedOutput( hostObject );
              return;
            }
            currentLength = hostObject.Length;
            input = '<input type=' + (hostObject.Visible == 'No' ? '"password"' :
'"text"' );
            if (WT_BROWSER.is_ie || WT_BROWSER.is_ns61up)
            {
              input += ' class="box" style="width:' + (currentLength *
WT_BROWSER.charWidth + 1) +  'px';
              input += (hostObject.Blinking  == 'Yes' ? '; background-color:#FFC0C0' :
'');
              input += (hostObject.Underline == 'Yes' ? ( hostObject.Intensity ==
'Reduced' ? '; color:#A0A0FF' : '; color:#0000A0' ) :
                                                        ( hostObject.Intensity ==
'Reduced' ? '; color:#A0A0A0' : '' )) + '"';
            }
            input += ' name="' + hostObject.Name + '" size="' + currentLength
                              + '" maxlength="' + currentLength
                              + '" value="' + hostObject.Value
                              + (hostObject.Input == 'Numeric'?'" numeric="1':'')
                              + '"/>';
            document.write( input );
          }

          function taggedOutput( hostObject )
          {
            if ( hostObject.Type == 'Unprotected' )
            {
              taggedInput( hostObject );
              return;
            }
            output = hostObject.HTMLValue;
            if ( hostObject.Visible == 'Yes' )
            {
              if ( hostObject.Inverse == 'Yes' )
              {
                if ( hostObject.Color=='#000000' )
                  output = '<font color="#FFFFFE" style=\"background-color:#000001\">'
+ output + '</font>';
                else
                  output = '<font color="#000000" style=\"background-color:' +
hostObject.Color + '\">' + output + '</font>';
              }
              else if (hostObject.Color != MVS_0.WT_Color.Default)
                output = '<font color=\"' + hostObject.Color + '\">' + output + '</
font>';
              if (hostObject.Intensity == 'Normal')
                output = '<b>' + output + '</b>';
              if (hostObject.Blinking == 'Yes')
                output = '<i>' + output + '</i>';
              if (hostObject.Underline == 'Yes')
```

```
                output = '<u>' + output + '</u>';
              document.write( output );
            }
            else
            {
          document.write( "
".substr(0,hostObject.Length));
            }
          }
        //-->
        </wtoncreatescript>
        <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
        <!-- begin of host screen section                                            -->
        <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<div style="color:##MVS_0.WT_Color.Default = "\#000000"#"><pre>\
```

Als wesentliche Funktionalität von `AutomaskMVS.htm` wird in einer Schleife jedes Formatelement auf je ein HTML-Element abgebildet. Hierfür stellt der Host-Adapter die Host-Objekte `$FIRST` und `$NEXT` zur Verfügung. Der Code des `AutomaskMVS`-Templates kann durch Optionen beim Generieren und durch das Master-Template mit WebLab beeinflusst werden. Es kann z.B. gewählt werden, ob Attribute wie z.B. `Blinking` abgebildet werden sollen.

```
<wtoncreatescript>
<!--
  if ( typeof wtInputFields == 'undefined' )
    wtInputFields = new Object;
  currentLine   = 1;
  document.write('<span class="screenline" id="SL1">');
  for (element = MVS_0.$FIRST.Name; MVS_0 && element != '$END'; element =
MVS_0.$NEXT.Name)
  {
    currentHostObject = MVS_0[element];
    if ( currentHostObject.StartLine != currentLine )
    {
      document.write  ('</span>
<span class="screenline" id="SL',++currentLine,'">');
    }
    if ( currentHostObject.Type == 'Protected' )
    {
      taggedOutput( currentHostObject );
    }
    else
    {
      wtInputFields[ element ] = currentHostObject;
      taggedInput( currentHostObject );
    }
  }
  document.write('</span>');
//-->
</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 DOM-Objekt angehängt, da dieses nicht automatisch vorhanden ist.

```
  <script type="text/javascript">
  <!--
  <wtoncreatescript>
  <!--
    for( element in wtInputFields )
    {
      if (!WT_BROWSER.acceptMaxLength)
        document.writeln('wtSetMaxLength(\'', element, '\',', wtInputFields[ element
].Length, ' );' );
    }
  //-->
  </wtoncreatescript>
  //-->
  </script>
</form>
```

Nach der Schleife über alle Felder 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">
<!--

wtSetFocus('##MVS_0.WT_FOCUS.Field#'##MVS_0.WT_FOCUS.Offset&&wtCurrentComm_system.isOve
rwrite?',''+MVS_0.WT_FOCUS.Offset:''#);
//-->
</script>
<wtrem** Script executed after post of HTML page **************************>
```

Zum Schluss wird in einem OnReceive-Script je ein Aufruf von `send` und von `receive` abgesetzt, um die WebTransactions Dialogzyklen und die Host-Dialogschritte zu synchronisieren. Mit der Funktion `setNextPage()` wird das nächste zu bearbeitende Template bestimmt.

```
<wtonreceivescript>
<!--
```

```
  if(WT_POSTED.wt_cursorOffset && WT_POSTED.wt_cursorOffset*1>0)
  {
    wtCursorField = MVS_O[WT_POSTED.wt_cursor];
    MVS_O.WT_FOCUS.Field =
'E_'+wtCursorField.STARTLINE+'_'+(wtCursorField.STARTCOLUMN*1+WT_POSTED.wt_cursorOffset
*1)+'_1';
  }
  else
    MVS_O.WT_FOCUS.Field = WT_POSTED.wt_cursor;
  if (MVS_O_system.EDIT_MODE &&MVS_O_system.EDIT_MODE.match(/USER/))
    MVS_O_system.isOverwrite = (WT_POSTED.wt_isOverwrite=='1');
  //{{WebLab(processPostedData)
  for ( element in wtInputFields )
  {
    currentHostObject = wtInputFields[element];
    if ( currentHostObject.Type == 'Unprotected' )
      currentHostObject.Value = WT_POSTED[element];
  }

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

5.3 Template wtKeysMVS.htm

Ein reales Terminal verfügt über Sondertasten und damit verbundene Funktionen, die in einer Terminal-Emulation weitestgehend nachgebildet werden können (z.B. durch die Belegung von Ersatztasten).

Das Template `wtKeysMVS.htm` stellt Ihnen für die MVS-spezifischen Standard-Tasten Bedienelemente zur Verfügung. Die Funktionen der häufig benutzten Tasten wie Datenfreigabe werden durch `Submit`-Knöpfe dargestellt, alle weiteren durch Auswahllisten. Außerdem enthält `wtKeysMVS.htm` einige Knöpfe, die keiner Taste eines Terminals entsprechen (`Disconnect`, `Refresh`, `Cancel Menu` und `Print`).

Die einzelnen Bedienelemente sind in der Tabelle auf Seite 123 beschrieben.

`wtKeysMVS.htm` inkludiert die Datei `wtKeysMVS.js`, die die Abbildung der Sondertasten für WebTransactions for MVS enthält. In dieser Datei können Sie die Abbildung der Tasten entsprechend Ihren Wünschen anpassen oder erweitern. Die ausführliche Beschreibung dieses Verfahrens finden Sie im Abschnitt „Zuordnung der Tasten in wtKeysMVS.js“ auf Seite 125.

Das Master-Template `MVS.wmt` enthält einen Aufruf (`<wtInclude>`) für `wtKeysMVS.htm`. Alle mit diesem Master-Template (mit Automask oder mit dem Capture-Verfahren) erstellten Templates enthalten daher ebenfalls diesen Aufruf.

Aktiviert der Anwender per Mausklick oder Auswahl eines Listeneintrags eine Keys-Funktion, sendet der Browser die Formulardaten an WebTransactions. Dazu legt `wtKeysMVS.htm` ein unsichtbares Eingabefeld an mit den Paremetern `<input type="hidden"` und `name="wt_special_key" ...>`. Die gewählte Funktion wird in diesem Feld gespeichert und dadurch neben den sichtbaren Eingabefeldern an WebTransactions übertragen. Dort wird mittels `send` und `receive` durch die an den Host-Adapter angeschlossene Terminal-Emulation die gewünschte Funktion ausgeführt. Ob dabei tatsächlich auch eine Kommunikation mit der Host-Anwendung stattfindet, hängt von der gewählten Keys-Funktion ab: In einigen Fällen, z.B. bei `Refresh`, wird die Host-Kommunikation unterdrückt.

Für die Realisierung der Keys-Funktionalität verwendet der Host-Adapter das Host-Objekt `WT_KEY` (siehe die Tabelle auf Seite 123). Alle Funktionen, die `wtKeysMVS.htm` und `wtKeysMVS.js` zur Verfügung stellen, werden auf einen entsprechenden Wert in `WT_KEY.Key` abgebildet (in einem OnReceive-Script). Der Wert von `WT_KEY.Key` steuert das Verhalten der Aufrufe `send` und `receive`.

5.4 Template wtBrowserFunctions.htm

Ein reales Terminal kann auch über die Tastatur gesteuert werden. Dieses Verhalten kann von einer Terminal-Emulation weitestgehend nachgebildet werden.

Das Template `wtBrowserFunctions.htm` stellt Ihnen über JavaScripts die Tastatur-Unterstützung auch für den Browser zur Verfügung. Die Browser unterschiedlicher Versionen unterstützen auch unterschiedliche Funktionalität.

Das Master-Template `MVS.wmt` enthält einen Aufruf für `wtBrowserFunctions.htm`. Alle mit diesem Master-Template (mit Automask oder mit dem Capture-Verfahren) erstellten Templates enthalten daher ebenfalls diesen Aufruf.
`wtBrowserFunctions.htm` verwendet auch die von `wtKeysMVS.htm` angelegten `<input>`-Tags.

Wenn der Benutzer mit der Tastatur eine Browser-Funktion aktiviert, sendet der Browser die Formulardaten an WebTransactions. Dort wird mittels `send` und `receive` durch die in den Host-Adapter integrierte Terminal-Emulation die gewünschte Funktion ausgeführt.

Im Abschnitt „Unterstützung von Terminal-Funktionen durch den Browser“ auf Seite 120 finden Sie eine Aufstellung, welcher Browser welche Funktion unterstützt.

5.5 Host-Anwendung mit Semigrafik

Rechtecke, die eine Host-Anwendung mit Hilfe der Semigrafik auf dem Bildschirm ausgibt, stellt WebTransactions wie folgt dar:

```
+------+
:      :
:      :
+------+
```

Diese Art der Darstellung verwendet der Automask-Mechanismus von WebTransactions auch für die Darstellung von Popup-Boxen. Sie können Popups aber auch individuell umsetzen (siehe Abschnitt „Templates für Popups generieren“ auf Seite 87).

6 Templates bearbeiten

Wenn Sie Ihre Host-Anwendung an das WWW angebunden haben, entspricht die Darstellung der Formate im Browser der Darstellung an einem Terminal (1:1-Umsetzung). In vielen Fällen ist diese Darstellung, die über das Automask-Template abgewickelt wird, ausreichend und eine weitere Gestaltung nicht erforderlich.

Sollen dagegen die vielfältigen Möglichkeiten der Oberflächengestaltung im Web genutzt und einzelne Dialogschritte der Host-Anwendung individuell aufbereitet werden, dann reicht der Einsatz des Automask-Templates nicht mehr aus. Die Nachbearbeitung findet mit so genannten formatspezifischen Templates statt.

Diese formatspezifischen Templates können Sie mit WebLab mit dem Capture-Verfahren generieren lassen und dann Ihren Anforderungen entsprechend bearbeiten. Dieses Kapitel beschreibt, wie Sie für einzelne Formate die Darstellung im Browser individuell aufbereiten.

Dabei lassen sich folgende Schritte unterscheiden:

1. Zunächst richten Sie mit WebLab eine WebTransactions-Sitzung ein und starten eine Host-Verbindung.
2. Anschließend identifizieren Sie mit dem Capture-Verfahren in WebLab die Host-Formate, die Sie individuell aufbereiten wollen (siehe Abschnitt „Capture-Verfahren mit WebLab“ auf Seite 82).
3. Nachdem Sie für ein Host-Format, das Sie individuell anpassen wollen, ein formatspezifisches Template erstellt haben, können Sie es nach Ihren Wünschen mit WebLab bearbeiten, siehe WebTransactions-Handbuch „Konzepte und Funktionen“.

6.1 Capture-Verfahren mit WebLab

Wenn Sie mit WebLab eine Verbindung zur Host-Anwendung gestartet haben, können Sie mit dem Capture-Verfahren interaktiv Erkennungskriterien für die einzelnen Host-Formate erstellen.

WebTransactions erkennt die Formate beim Ablauf an den Erkennungskriterien, die in einer speziellen Datenbank verwaltet werden, der Capture-Datenbank. Die Capture-Datenbank benötigt WebTransactions in der Einsatzphase, um den empfangenen Host-Formaten die entsprechenden formatspezifischen Templates zuzuordnen. Jeder Eintrag in dieser Datenbank verknüpft ein oder mehrere Erkennungskriterien mit einem Format. Jedes Mal, wenn eine `Receive`-Anweisung in einem Template abgeschlossen wird, geht WebTransactions die Capture-Datenbank sequenziell durch und prüft, ob es für das übertragende Format ein zutreffendes Erkennungskriterium gibt. Bei einem Treffer wird der Formatname in das `FLD`-Attribut des privaten Systemobjekts geschrieben, ansonsten der Wert aus dem Attribut `AUTOMASK`.

Die `setNextPage`-Anweisung legt in einem Template nach der `Receive`-Anweisung fest, welches Template als nächstes ausgeführt werden soll. Dies kann z. B. das Template sein, das über das Erkennungskriterium im `FLD`-Attribut ermittelt wird. Siehe hierzu auch Abschnitt „Dialogzyklus“ im WebTransactions-Handbuch „Konzepte und Funktionen“.

6.1.1 Vorgehensweise

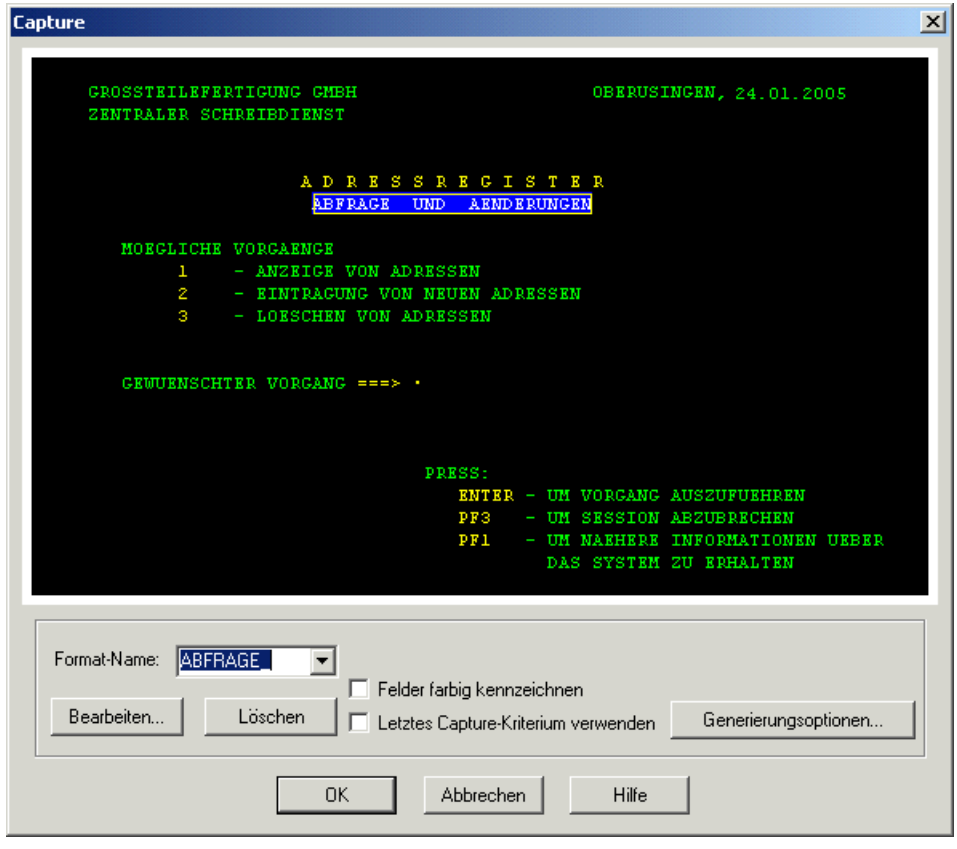
- Überprüfen Sie, ob im Start-Template für das Systemobjekt-Attribut `CAPTURE_FILE` der richtige Pfadname eingetragen ist.

  Im Start-Template `wtstartMVS.htm` ist im Parameter **CAPTURE_FILE** als Vorbelegung der Pfadname `config/capture.sdb` eingetragen. Die Capture-Datenbank, die Sie im Start-Template angeben, wird beim ersten Zugriff angelegt.

- Steuern Sie durch Eingaben an die Host-Anwendung das Format an, für das ein Erkennungskriterium erzeugt werden soll.

- Wählen Sie Befehl **Generieren/Capture/aktuellen Schirm**, um das Dialogfeld **Capture** mit dem aktuellen Format zu öffnen.

  Falls das Attribut `CAPTURE_FILE` beim Starten der Sitzung nicht gesetzt war, wird zuerst das Dialogfeld **Capture-Datenbank angeben** am Bildschirm eingeblendet, in dem Sie eine existierende Datenbank auswählen oder eine neue angeben können.



- Markieren Sie hier die Bereiche, die das Format eindeutig identifizieren sollen. Markieren Sie ein oder mehrere Rechtecke durch Ziehen mit der Maus bei gedrückter Standard-Maustaste.
- Vergeben Sie im Feld **Format-Name** einen Namen für das Erkennungskriterium. Sie können aus der Liste auch einen Namen auswählen, wenn Sie das Format schon erfasst haben und jetzt beispielsweise das Erkennungskriterium des Formats bearbeiten.
- Wenn Sie die Option **Letztes Capture-Kriterium verwenden** wählen, wird das zuletzt verwendete Erkennungskriterium voreingestellt.
- Öffnen Sie dann mit der Schaltfläche **Generierungsoptionen** das Dialogfeld **Capture: Optionen für FLD und Template Generierung**. In diesem Dialogfeld legen Sie die Generierungsoptionen fest. In der Regel sind die Voreinstellungen ausreichend.
- Geben Sie die erforderlichen Generierungsoptionen ein und bestätigen Sie sie mit **OK**. Für nicht ausgefüllte Eingabefelder werden die voreingestellten Werte verwendet.

- ► Schließen Sie das Dialogfeld **Capture** mit **OK**. WebLab erzeugt dann für dieses Format eine FLD-Datei und ein HTML-Template:
  - FLD-Dateien sind spezielle Beschreibungsdateien. Sie werden von der WebTransactions-Entwicklungsumgebung WebLab verwendet, um die Funktion „grafische Hostobjektauswahl“ zu realisieren. Damit WebTransactions während der Laufzeit auf diese Dateien zugreifen kann, müssen sie unter *basedir*`\config` stehen.
  - HTML-Templates werden von WebTransactions zur Laufzeit interpretiert und bestimmen die im Browser dargestellte Oberfläche.

6.1.2 Erkennungskriterium bearbeiten

Während des Capture-Verfahrens können Sie die einzelnen Erkennungskriterien bearbeiten. Wählen Sie dazu im Dialogfeld **Capture** die Schaltfläche **Bearbeiten**. Das Dialogfeld **Auswahl bearbeiten** wird am Bildschirm eingeblendet.

In diesem Dialogfeld werden alle Erkennungskriterien mit Position und Größe angezeigt, die Sie für das aktuelle Host-Format ausgewählt haben. Mit der Schaltfläche **Löschen** können Sie einzelne Erkennungskriterien wieder entfernen.

Über die Optionen **Zeichen** und **Attribute** können Sie bestimmen, ob der Inhalt des ausgewählten Bereichs und/oder seine Darstellung zur Erkennung des zugehörigen Formats verwendet werden soll.

6.1.3 Capture-Datenbank bearbeiten

Um die Capture-Datenbank zu bearbeiten, brauchen Sie keine Verbindung zur Host-Anwendung.

Sie bearbeiten die Capture-Datenbank mit WebLab im Dialogfeld **Capture Management**. Das Dialogfeld öffnen Sie mit dem Befehl **Generieren/Capture Management**.

Die Liste im Dialogfeld zeigt Ihnen in tabellarischer Form die Formate an, für die in der Capture-Datenbank bereits Erkennungskriterien existieren.

Das Dialogfeld **Capture Management** listet die Formate in der gleichen Reihenfolge auf, wie sie in der Capture-Datenbank angeordnet sind. Beim Capture-Verfahren werden neue Formate am Ende ergänzt. Beim Ablauf wird die Capture-Datenbank sequenziell durchsucht. Deswegen können Sie die Reihenfolge der Formate ändern, um z.B. häufig verwendete Formate weiter zum Anfang zu verschieben und so die Suchzeit zu verkürzen, oder damit bei ähnlichen Formaten das genauer spezifizierte vorrangig erkannt wird.

6.2 Individuelle Templates für Popup-Boxen

Host-Formate können Popup-Boxen enthalten.

Falls Sie ausschließlich mit dem Umsetzungs-Template AutomaskMVS.htm arbeiten, müssen Sie für Popup-Boxen keine besonderen Vorkehrungen treffen. Popup-Boxen werden vom Automask-Mechanismus als Bestandteil des Host-Formats dargestellt .

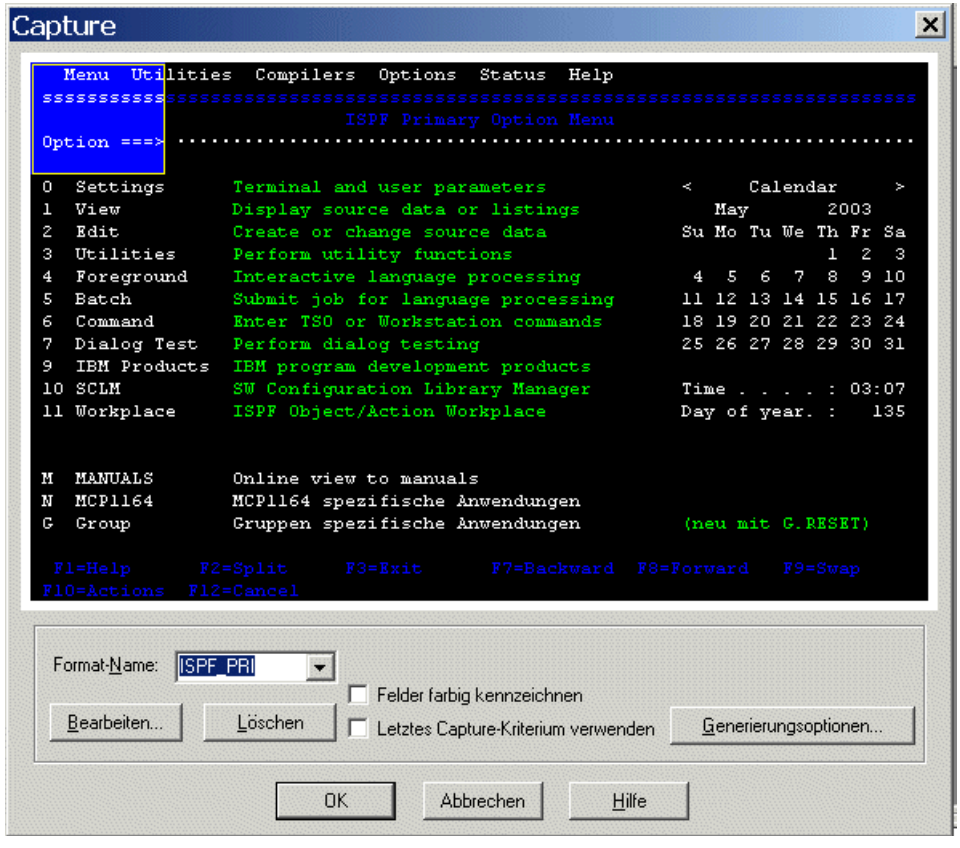
Für individuell angepasste Templates stellt WebTransactions Attribute des Systemobjekts für die Popup-Behandlung zur Verfügung. Falls zur Laufzeit der WebTransactions-Anwendung das Attribut `USE_POPUP_RECOGNITION` auf `"Yes"` gesetzt ist, werden diese Popups erkannt und als eigenständige Seiten an den Web-Browser geschickt. Wenn für die Erkennung der Popups andere Zeichen als die Voreinstellung benutzt wurden, müssen Sie diese Zeichen mit den Systemobjekt-Attributen für die Popup-Darstellung setzen, siehe hierzu auch Abschnitt „Systemobjekt-Attribute“ auf Seite 93.

Bevor im Abschnitt „Templates für Popups generieren“ auf Seite 87 die Popup-Funktionalität von WebTransactions beschrieben wird, sollen zunächst die Probleme aufgezeigt werden, die ohne diese spezielle Popup-Behandlung bei der Identifikation individueller Templates auftreten würden.

6.2.1 Ohne spezielle Popup-Behandlung: Identifikationsprobleme

Mit dem Capture-Verfahren von WebLab werden bestimmte Fortmat-Bereiche als Erkennungskriterien festgelegt. Die Auswahl dieser Bereiche kann Auswirkungen auf den korrekten Ablauf der Formaterkennung haben. Dies soll an folgendem Beispiel erläutert werden:

Für das Format (ohne Popup-Box) einer Host-Dialog-Anwendung wird ein Bereich als Erkennungskriterium definiert:



Wenn jedoch ein Popup eingeblendet wird, das das Erkennungskriterium überdeckt, wird das Format zur Laufzeit von WebTransactions nicht erkannt, da ein Teil des Erkennungskriteriums nicht im Format enthalten ist (wird vom Popup überdeckt). In einem solchen Fall würde WebTransactions das gesamte Format nicht durch ein entsprechendes formatspezifisches Template, sondern mit Hilfe des Umsetzungs-Templates `AutomaskMVS.htm` darstellen.

Wenn Sie das Erkennungskriterium so wählen, dass es nicht überdeckt wird, kann WebTransactions das Format sowohl mit als auch ohne Popup erkennen. WebTransactions würde zur Laufzeit in beiden Fällen das gleiche formatspezifische WTML-Template verwenden, was zu Problemen führen würde:

Falls beim Capture-Verfahren das Popup nicht angezeigt war, wäre das Popup im generierten Template nicht enthalten und würde im Browser nie angezeigt werden.

Falls beim Capture-Verfahren das Popup angezeigt war, wäre das Popup im generierten Template enthalten und würde im Browser immer angezeigt werden, unabhängig davon, ob es im aktuellen Format vorhanden ist oder nicht.

6.2.2 Templates für Popups generieren

Um die im vorhergehenden Abschnitt dargestellten Probleme zu vermeiden, bietet WebTransactions die Möglichkeit, für Popup-Boxen separate, formatspezifische Templates zu erstellen.

Voraussetzung

Um diese Funktion zu nutzen, müssen Sie zuerst das Attribut `USE_POPUP_RECOGNITION` des kommunikationsspezifischen Systemobjekts auf `YES` setzen.

Wenn Sie für Ihre Host-Anwendung ein eigenes Start-Template mit dem WTBean `wtcStartMVS` erzeugen (siehe Abschnitt „WTBean wtcStartMVS.wtc zur Generierung eines Start-Templates“ auf Seite 142), können Sie dieses Attribut direkt im Start-Template setzen:

- Wählen Sie im Dialogfeld **Einfügen:wtcStartMVS** die Registerkarte **Weitere Optionen**.
- Klicken Sie unter **Popup-Erkennung** auf den Eintrag **Popup-Erkennung zulassen**.
- Markieren Sie die Option **Popup-Erkennung zulassen**. Der Wert des Eintrags wird von **No** auf **Yes** geändert.

Wenn Sie sich in einer Sitzung befinden, die Sie mit WebLab gestartet haben, können Sie das Attribut für diese Sitzung auch dynamisch mit WebLab anlegen:

- Markieren Sie im Objektbaum von WebLab das Systemobjekt `MVS_0_system` und öffnen Sie das Kontextmenü.
- Wählen Sie im Kontextmenü den Befehl **Neue Variable**.
- Geben Sie dem neuen Attribut des System-Objekts den **Namen** `USE_POPUP_RECOGNITION`, wählen Sie als **Typ** `string` und geben Sie als **Wert** `Yes` ein.

> **i** Beachten Sie, dass das Attribut in diesem Fall nur für die aktuelle Sitzung festgelegt ist. Es muss vor dem nächsten Start im Start-Template erneut gesetzt werden.

Wenn die Host-Anwendung für die Darstellung der Popup-Rahmen nicht die voreingestellten Werte verwendet, müssen Sie die weiteren Systemobjekt-Attribute für die Popup-Erkennung ebenso setzen, siehe hierzu auch Abschnitt „Systemobjekt-Attribute“ auf Seite 93. Diese Systemobjekt-Attribute sind auch für den Ablauf erforderlich.

Popup-Rahmen mit Semigrafik-Zeichen

Bei einigen MVS-Anwendungen werden Popup-Rahmen durch spezielle Semigrafik-Zeichen dargestellt. Hier werden die Zeichen als normale Buchstaben dargestellt:

```
EssssssssssssssssssssssN
e                      e
e                      e
e                      e
e                      e
e                      e
DssssssssssssssssssssssM
```

Um die Standard-Darstellung solcher Popup-Rahmen zu verbessern, ersetzt WebTransactions diese Semigrafik-Rahmen implizit durch wie folgt aufgebaute Rechtecke:

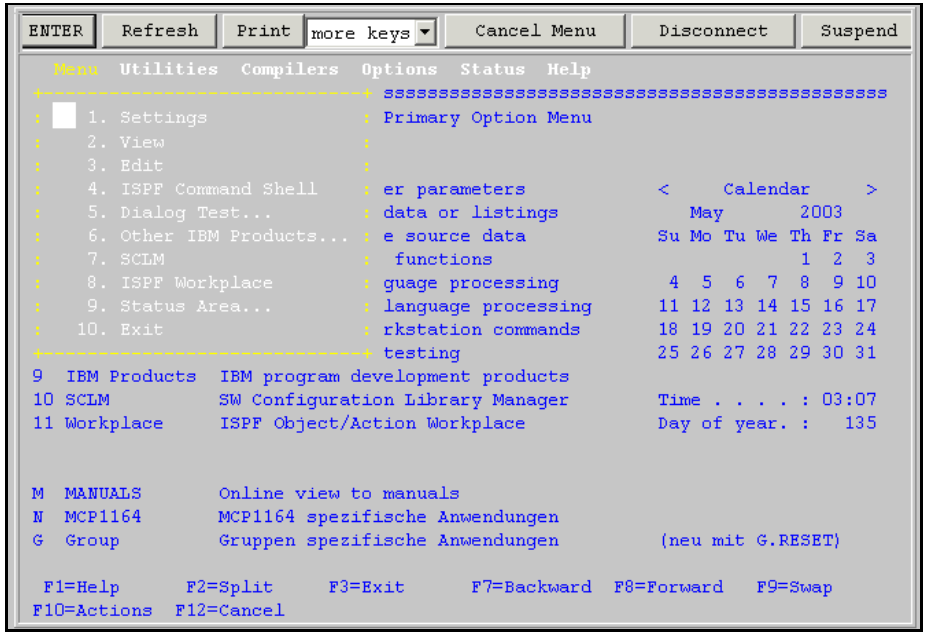
```
+----------------------+
:                      :
:                      :
:                      :
:                      :
:                      :
+----------------------+
```

Um eine automatische Erkennung und individuelle Behandlung solcher Popups zu ermöglichen, müssen Sie bei der Konfiguration der Attribute des Systemobjekts von WebTransactions die ursprünglichen MVS-Semigrafik-Zeichen angeben, **nicht** die Ersatzzeichen der WebTransactions-Standard-Darstellung. Für `POPUP.VMIDDLE` beispielsweise ist also nicht der Wert `":"`, sondern der Wert `"e"` zu setzen, siehe hierzu auch Abschnitt „Systemobjekt-Attribute“ auf Seite 93.

Vorgehen

Um die Popups mit dem Capture-Verfahren zu erfassen, gehen Sie folgendermaßen vor:

► Navigieren Sie in der Host-Anwendung bis zu dem Format mit den Popup-Boxen.



► Wählen Sie den Befehl **Generieren/Capture/aktuellen Schirm**, um die Popup-Box mit dem Capture-Verfahren zu erfassen. Das Dialogfeld **Capture** wird am Bildschirm mit der Popup-Box eingeblendet.

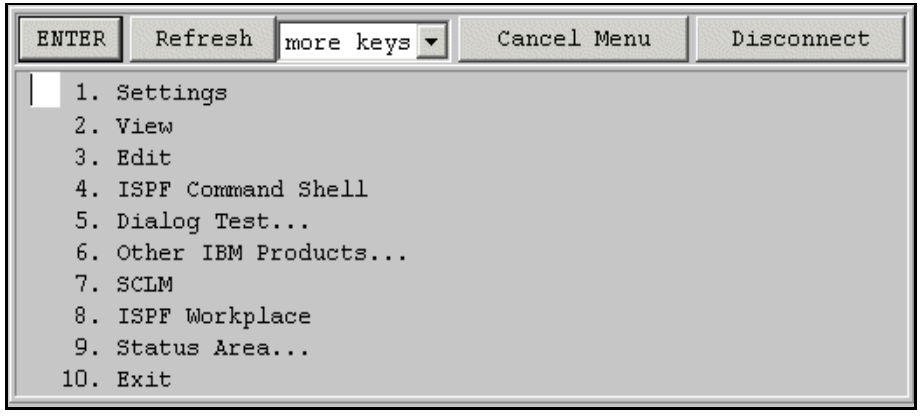
► Geben Sie der Popup-Box einen Namen und setzen Sie im Dialogfeld **Capture: Optionen für FLD und Template Generierung** die Generierungsoptionen für Template und FLD-Datei.

Popup-Erkennung zur Laufzeit

Zur Laufzeit von WebTransactions läuft die Popup-Erkennung folgendermaßen ab:

- Wenn ein Popup in einem Host-Format erkannt wird, sucht WebTransactions in der Capture-Datenbank (`config/`*anwendung*`.sdb`) ein dem Popup entsprechendes Erkennungskriterium. Wenn ein solches gefunden wird, verwendet WebTransactions das identifizierte Popup-Template. Im Browser wird also nur das Popup angezeigt, das dahinter liegende Host-Format ist nicht sichtbar.

Im Folgenden sehen Sie das auf Basis eines formatspezifischen Templates umgesetzte Popup:



- Die Erkennung von Formaten mit der Capture-Datenbank funktioniert in folgender Reihenfolge:
  - Wenn die Popup-Erkennung aktiv ist und ein Popup-Erkennungskriterium zu dem Format passt, wird das individuelle Popup-Template verwendet.
  - Wenn ein normales Erkennungskriterium zu dem Format passt, dann wird das individuelle Template für das gesamte Format verwendet.
  - Wenn für das Gesamtformat kein Erkennungskriterium vorhanden ist, verwendet WebTransactions das Umsetzungs-Template, dessen Name in dem Attribut `AUTOMASK` steht (in der Regel `AutomaskMVS`).

Auch wenn von WebTransactions ein Popup identifiziert und individuell dargestellt wird, beziehen sich die Host-Objekte `$FIRST` und `$NEXT` auf das erste bzw. nächste Feld des zu Grunde liegenden Host-Formats und nicht auf das erste oder nächste Feld des Popups.

7 Kommunikation steuern

7.1 Systemobjekt-Attribute

Mit einigen Attributen des Systemobjekts steuern Sie die Kommunikation zwischen WebTransactions und der MVS-Anwendung.

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

Existiert unterhalb des verwendeten Kommunikationsobjekts ein Objekt `WT_SYSTEM` („verbindungsspezifisches Systemobjekt“), so müssen die in diesem Abschnitt beschriebenen Attribute dort definiert werden, anderenfalls sind sie als Attribute des globalen Systemobjekts `WT_SYSTEM` zu erklären. Einzige Ausnahme sind die Attribute `COMMUNICATION_INTERFACE_VERSION` und `FORMAT`, die immer am globalen Systemobjekt zu setzen sind.

Die 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“).

Grundlegende Informationen zu verbindungsspezifischen und globalen Systemobjekten finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

7.1.1 Übersicht

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

Die Systemobjekt-Attribute können in folgende Kategorien eingeteilt werden:

o (**o**pen)
Attribute, die bei open ausgewertet werden.

t (**t**emporary)
Attribute, die während der Kommunikation verwendet werden und die jederzeit in den Templates verändert werden können.

r (**r**ead only)
Attribute, die während der Kommunikation verwendet werden und die nicht in den Templates verändert werden dürfen.

c (**c**ommunication module)
Attribute, die automatisch vom Host-Adapter gesetzt werden.

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

Attributname	**Bedeutung**	**Erläuterung/Kategorie**	
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 Templatedateien müssen in folgender Form abgespeichert sein: *application_prefix@formatname*.fld bzw. *application_prefix@formatname*.htm	o
AUTOMASK	Standard-Umsetzungs-Template	Name des Umsetzungs-Templates, das verwendet werden soll, falls es für den aktuellen Bildschirm kein Erkennungskriterium in der Capture-Datenbank gibt. Vorbelegung: AutomaskMVS. Falls kein Erkennungskriterium gefunden und der Inhalt des Attributs AUTOMASK übernommen wurde, aber das dort genannte Template nicht vorhanden ist, wird das in DEFAULT_FORMAT hinterlegte Template verwendet. Falls zwar ein Erkennungskriterium gefunden wird, aber kein entsprechendes Template, wird ebenfalls das in DEFAULT_FORMAT hinterlegte Template verwendet (siehe WebTransactions-Handbuch „Konzepte und Funktionen“).	t

Attributname	Bedeutung	Erläuterung/Kategorie	
BYPASS	Bypass-Druckdatei Flag	Dieses Attribut wird von WebTransactions auf Yes gesetzt, wenn beim Aufruf von receive Bypass-Druckdaten erkannt wurden. Vor der Auswertung von $MESSAGE.PRINTING sollte das Attribut vom Template wieder auf No gesetzt werden, wie im Template wtasync.htm, das von WebTransactions bereitgestellt wird.	c, t
CAPTURE_FILE	Capture-Datenbank	Name der Capture-Datenbank (...*anwendung*.sdb). Der Name kann als absoluter Pfadname oder als relativer Pfadname (in Bezug auf das Basisverzeichnis) angegeben werden, z.B.: – absolute Angabe: C:\mvsappli1\config\mvsappl.sdb – relative Angabe (Basisverzeichnis = mvsappli1): config2\mvsappl.sdb	t
CAPTURED_FLD	Kennzeichen für die Format-Erkennung	Dieses Attribut wird von WebTransactions beim Aufruf von receive auf Yes gesetzt, wenn das empfangene Format in der Capture-Datenbank gefunden wurde. Wurde das Format nicht gefunden, dann wird CAPTURED_FLD auf No gesetzt.	c

Attributname	Bedeutung	Erläuterung/Kategorie	
CODE_PAGE	Umsetzungs-Tabelle	Von der Emulations-Software verwendete Tabelle für die Umsetzung der Host-Meldungen von EBCDIC nach ASCII und umgekehrt. Diese Tabellen enthalten unterschiedliche Umsetzungs-Regeln für die Unterstützung von länder- oder sprach-spezifischen Zeichen. Mögliche Werte: Belgian Danish Dutch English French German Hebrew Iceland Italian Norwegn Portugue Spanish Swedish Swefin UK CP037 CP273 CP277 CP278 CP280 CP285 CP500	o
COMMUNICATION_ INTERFACE_VERSION	Schnittstellenversion	Enthält diese Variable einen Wert < „3.0“, dann wird beim Empfangen einer Nachricht vom Host (receive) das globale System-Attribut FORMAT mit dem Namen des Host-Formats versorgt. Falls kein Formatname ermittelt werden konnte, wird FORMAT auf den Wert von AUTOMASK gesetzt. Enthält diese Variable einen Wert „3.0“ oder höher, dann wird FORMAT nicht versorgt, da die Auswahl der nächsten Seite (Format) von den Templates selber getroffen wird (in der Regel durch Auswerten des Attributs FLD). Voreinstellung: 7.5	o

Attributname	Bedeutung	Erläuterung/Kategorie	
CONNECTION_INFO	Verbindungs-Information	Diese Zeichenkette kann z.B. dazu verwendet werden, um den Benutzer einer bestimmten Sitzung zu identifizieren. Die Information wird in der Session-Info-Datei abgespeichert, die mit der Administration angezeigt werden kann.	o
DISCONNECT	Template für den Verbindunsabbau	Dieses Template wird nach Beenden der Host-Verbindung aktiviert. Wenn der Anwender z.B. auf den Knopf **Disconnect** klickt, wird das Template aufgerufen, das in diesem Attribut hinterlegt ist. Voreinstellung: wtstart	t
END_WAIT_CONDITION. EXPECTED_BLOCKS	Schirmende-Erkennung über die Anzahl der Teilnachrichten, die das gesamte Formular bilden	Wenn dieses Attribut gesetzt ist, hört receive trotz MULTIPLE_IO_TIMEOUT auf zu warten, wenn die erwartete Anzahl von Teilnachrichten eingetroffen ist. Wenn bereits weitere Nachrichten im Netz anstehen, die ohne Warten vearbeitet werden können, kann RECEIVED_BLOCKS nach dem receive auch größer sein als END_WAIT_CONDITION.EXPECTED_BLOCKS. Voreinstellung: leer Siehe auch RECEIVED_BLOCKS.	t
END_WAIT_CONDITION. FLD_EXPECTED, END_WAIT_CONDITION. FLD_DIFFERENT_FROM	Schirmende-Erkennung über erkanntes Format; Voraussetzung: Arbeit mit Capture-Datenbank	Wenn ein in der Capture-Datenbank definiertes Format erkannt wurde (Attribute FLD und CAPTURED_FLD), und den im jeweiligen Attribut gesetzten Angaben entspricht, hört receive trotz MULTIPLE_IO_TIMEOUT auf zu warten. Voreinstellung: leer	t
END_WAIT_CONDITION. CURSOR_IN_LINE und END_WAIT_CONDITION. CURSOR_IN_COLUMN, END_WAIT_CONDITION. CURSOR_NOT_IN_LINE und END_WAIT_CONDITION. CURSOR_NOT_IN_COLUMN	Schirmende-Erkennung über die Position der Schreibmarke	Wenn der Cursor auf einer den gesetzten Bedingungen entsprechenden Position steht, hört receive trotz MULTIPLE_IO_TIMEOUT auf zu warten. Voreinstellung: leer	t
END_WAIT_CONDITION. MATCH_STARTLINE, END_WAIT_CONDITION. MATCH_STARTCOLUMN, END_WAIT_CONDITION. MATCH_VALUE und END_WAIT_CONDITION. MATCH_OPERATION	Schirmende-Erkennung über den Feldinhalt in einem bestimmten Bildschirmbereich	Wenn sich in dem Bildschirmpuffer beginnend an der Position, die mit MATCH_STARTLINE und MATCH_STARTCOLUMN definiert ist, eine Zeichenkette MATCH_VALUE befindet (MATCH_OPERATION ="==") oder nicht befindet (MATCH_OPERATION ="!="), hört receive trotz MULTIPLE_IO_TIMEOUT auf zu warten. Voreinstellung: leer.	t

Attributname	Bedeutung	Erläuterung/Kategorie	
EPILOG	Nachspann	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. Voreinstellung: keine Inkludierung **i** Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter. Siehe auch PROLOG und FORMTPL	t
FIELD_NAMES	Sprechende Feldnamen benutzen	Wenn dieses Attribut auf User-defined gesetzt ist, können sprechende Feldnamen benutzt werden, die in den FLD-Dateien abgelegt sind. Andernfalls werden die vom Host-Adapter erzeugten generischen Feldnamen verwendet. Mögliche Werte: User-defined, Generic Voreinstellung: User-defined **i** Beachten Sie, dass in den ausgelieferten Start-Templates und Beans Generic als Voreinstellung implementiert ist.	o

Attributname	Bedeutung	Erläuterung/Kategorie	
FIRST_IO_TIMEOUT	Timer für den receive-Aufruf.	Der Timer ist standardmäßig auf 60 Sekunden eingestellt. Wenn in dieser Zeit keine Nachricht vom Host eintrifft, dann kehrt der receive-Aufruf zurück. Damit kann man z.B. mit einem Template sowohl Host-Anwendungen abdecken, die einen Begrüßungsschirm beim Verbindungsaufbau senden als auch Host-Anwendungen, die sofort eine Eingabe erwarten. Dazu setzt man FIRST_IO_TIMEOUT auf einen niedrigen Wert und prüft nach dem ersten receive-Aufruf, ob eine Nachricht vom Host empfangen wurde (RECEIVED_BLOCKS="0"). Allerdings wird ein Fehler angezeigt, wenn binnen FIRST_IO_TIMEOUT keine Nachricht vom Host empfangen wurde. Um diese Meldung durch WebTransactions zu unterdrücken, setzen Sie DISABLE_COMMUNICATION_ERROR. Ist FIRST_IO_TIMEOUT größer als das globale Systemobjekt-Attribut TIMEOUT_APPLICATION, wird ein von TIMEOUT_APPLICATION abgeleiteter Wert verwendet: – ist TIMEOUT_APPLICATION > 10: FIRST_IO_TIMEOUT entspricht TIMEOUT_APPLICATION -5 – ist TIMEOUT_APPLICATION > 1: FIRST_IO_TIMEOUT entspricht TIMEOUT_APPLICATION -1 – ist TIMEOUT_APPLICATION =1: FIRST_IO_TIMEOUT entspricht TIMEOUT_APPLICATION Der Wert ist in Sekunden, evtl. mit einem Dezimaltrenner (Punkt oder Komma) anzugeben. Kleinste Einheit ist 500 Millisekunden.	t
FLD	Format-Name	Name des Formats, das von der Host-Anwendung empfangen wurde. Wenn WebTransactions keinen Formatnamen erkannt hat (z.B. auch wenn keine Capture-Datenbank zugewiesen ist), wird FLD auf den Wert von AUTOMASK gesetzt (wird stets vom receive gesetzt). Siehe auch FORMAT	c, r

Attributname	Bedeutung	Erläuterung/Kategorie	
FORMTPL	Formularfelder	Das Attribut enthält den Namen eines Templates (ohne die Endung '.htm'). Ist das Attribut definiert, so wird das entsprechende Template am Anfang des wtDataForm in den generierten Templates inkludiert. Voreinstellung: keine Inkludierung **i** Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter. Siehe auch PROLOG und EPILOG.	t
FTP_CODE_PAGE	Umsetzungstabelle für den File-Transfer mit $INDFILE	Mögliche Werte: 0 – 037 USA 1 – 037C USA C/370 2 – 273 Östereich, Deutschland 3 – 277 Dänemark, Norwegen 4 – 278 Schweden, Finnland 5 – 280 Italien 6 – 285 England 7 – 00 International 8 – ASCII Voreinstellung: 2	o
HARDCOPY	Hardcopy-Druckdatei Flag	Dieses Attribut wird von WebTransactions auf Yes gesetzt, wenn beim Aufruf von receive eine Hardcopy-Druckdatei aufbereitet wurde (siehe WT_KEY.key="PRINT"). Vor der Auswertung von $MESSAGE.PRINTING sollte das Attribut vom Template wieder auf No gesetzt werden, wie im Template wtasync.htm, das von WebTransactions bereitgestellt wird.	c, t
HOST_NAME	Name des Host-Rechners	Name oder Internet-Adresse des Host-Rechners. Wenn Sie einen symbolischen Namen verwenden, muss er entweder lokal oder im Domain Name Service (DNS) spezifiziert sein.	o

Attributname	Bedeutung	Erläuterung/Kategorie	
IGNORE_ASYNC	ASYNC-Bedingung ignorieren	Normalerweise (IGNORE_ASYNC='NO') wird bei dem Methodenaufruf send geprüft, ob bereits neue Daten asynchron vom Host eingegangen sind. Wenn dies der Fall ist, wird nicht an den Host gesendet sondern REFRESH_BY_ASYNC wird auf Yes gesetzt und WT_KEY.Key auf Refresh. Der folgende Methodenaufruf receive übernimmt die bereits vorliegenden Host-Daten und wartet nicht mit FIRST_IO_TIMEOUT und MULTIPLE_IO_TIMEOUT auf weitere Daten. Wenn IGNORE_ASYNC auf Yes gesetzt ist, wird unabhängig von inzwischen asynchron eingegangenen Daten gesendet. Manuelles Wiederholen des send-Aufrufes wird dadurch vermieden. WT_KEY.KEY bleibt unverändert, REFRESH_BY_ASYNC wird nicht auf Yes gesetzt, denn es wurde kein Refresh durchgeführt. IGNORE_ASYNC unterdrückt somit auch die Erkennung der asynchronen Nachricht. Voreinstellung ist NO. Siehe auch MULTIPLE_IO_TIMEOUT, REFRESH_BY_ASYNC	t
IGNORE_EMPTY_BLOCKS	Behandlung von Host-Nachrichten ohne Änderung der Host-Objekte	Wenn dieses Attribut auf Yes gesetzt ist, wird beim Warten auf Nachrichten vom Host der Übergang von FIRST_IO_TIMEOUT auf MULTIPLE_IO_TIMEOUT verhindert, falls eine Nachricht nicht den Nutzbereich des Terminal-Bildschirms – und damit nicht die Host-Objekte – betrifft. Damit können Sie Host-Anwendungen, bei denen die erste Nachricht die Tastatur entsperrt (Änderung der Statuszeile), und die zweite Nachricht den vollständigen Bildschirm (Änderung des Bildschirms) enthält, noch mit MULTIPLE_IO_TIMOUT = "0" betreiben. Wenn das Attribut auf Yes gesetzt ist, werden in dem Attribut RECEIVED_BLOCKS die „empty blocks“ trotzdem mitgezählt. EXPECTED_BLOCKS ist trotzdem einschließlich der „empty blocks“ anzusetzen. Voreinstellung: Yes	t
LU_NAME	Name der Logical Unit für die Verbindung zum MVS-Host	Der Name der Logical Unit (LU) ist eine maximal 8 byte lange Zeichenkette. LU_NAME muss nur angegeben werden, wenn die Konfiguration der Umgebung (Gateway, Host) dies erfordert.	o

Attributname	Bedeutung	Erläuterung/Kategorie	
MULTIPLE_IO_TIMEOUT	Timeout für vollständiges Bildschirmformat	Einige Host-Anwendungen senden ihre Bildschirmformate in mehreren Teilnachrichten. Da nicht eindeutig ist, welches die letzte Nachricht ist, wird mit einem Timeout gearbeitet. D.h., geht nach dem Empfang einer Nachricht vom Host innerhalb der nächsten durch MULTIPLE_IO_TIMEOUT festgelegten Sekunden keine weitere Nachricht ein, wird angenommen, dass das Bildschirmformat vollständig ist. Wenn Sie mit Druck-/Asynchron-Unterstützung arbeiten (siehe Seite 147ff), dann wird in zyklischen Abständen automatisch ein Refresh durchgeführt, sodass noch unvollständige Formate beim nächsten Refresh vollständig aufgebaut sein können. Ist sichergestellt, dass die Host-Anwendung keine Teilnachrichten benutzt, kann der Wert auf 0 gesetzt werden. Vorbelegung: 2 (Sekunden) Ziel ist es, grundsätzlich ohne MULTIPLE_IO_TIMEOUT das Schirmende zu erkennen, da dieser Timeout – bei jedem Dialogschritt die verstrichende Zeit erhöht – groß genug eingestellt werden muss, um bei starker Last nicht fälschlicherweise zu früh das Schirmende anzunehmen. Die Prüfung, ob die Emulation oder die Tastatur gesperrt ist, ist für Schirmende-Erkennung nicht verwendbar, wenn bereits mit der ersten Teilnachricht die Tastatur freigegeben wird. Mit den END_WAIT_CONDITION–Attributen können Sie MULTIPLE_IO_TIMEOUT ebenfalls abbrechen: Nur wenn alle dort eingestellten Bedingungen erfüllt sind, wird das Warten abgebrochen, obwohl MULTIPLE_IO_TIMEOUT noch nicht abgelaufen ist. Siehe auch Attribute END_WAIT_CONDITION	t
OFFLINE_COMMUNICATION	Schalter zum Abspielen eines Tracefile	Wert „Yes“ : Ein zuvor aufgezeichneter Emulations-Trace (eine Offline-Sitzung) wird „abgespielt“. Der Name der Datei, in der die Sitzung aufgezeichnet wurde, wird im Attribut OFFLINE_TRACEFILE angegeben. Default: No siehe auch RECORD_HOST_COMMUNICATION	o
OFFLINE_LOGFILE	Dateiname des zu erzeugenden Emulations-Trace	Dateiname des Emulations-Trace, der aufgezeichnet werden soll. siehe auch RECORD_HOST_COMMUNICATION	o

Attributname	Bedeutung	Erläuterung/Kategorie	
OFFLINE_TRACEFILE	Dateiname eines Emulations-Trace	Dateiname eines Emulations-Trace, der abgespielt werden soll, siehe auch OFFLINE_COMMUNICATION	o
PADDING_CHARACTER	Füllzeichen für ungeschützte Felder	Wenn dieses Attribut den Wert Nil enthält, dann werden Eingabefelder mit Nilzeichen (\0) gefüllt, andernfalls wird das Leerzeichen als Füllzeichen verwendet (Voreinstellung)	t
POPUP.COLUMN	Anfangsspalte des empfangenen Popups	Falls bei einem Aufruf von receive ein Popup empfangen wird und dieses Popup vom Capture-Mechanismus erkannt wird, enthält dieses Attribut die Spaltennummer, die die linke obere Ecke des Popups im Host-Format einnimmt. Mögliche Werte: 0 (falls kein Popup vorliegt) - *Maximale Spaltenanzahl* Voreinstellung: 0.	c, r
POPUP.HEIGHT	Höhe des empfangenen Popups	Falls bei einem Aufruf von receive ein Popup empfangen wird und dieses Popup vom Capture-Mechanismus erkannt wird, enthält dieses Attribut die Gesamthöhe des Popups (einschließlich des Rahmens). Mögliche Werte: 0 (falls kein Popup vorliegt) - *Maximale Zeilenanzahl* (inkl. Rahmen) Voreinstellung: 0.	c, r
POPUP.HSTART	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen, mit denen der horizontale Rahmen eines Popup-Fensters beginnen kann. Vorbelegung ist ein Doppelpunkt gefolgt von einem Punkt „:.“, d.h. WebTransactions erkennt den horizontalen Rahmen eines Popups, wenn er mit einem Doppelpunkt oder einem Punkt beginnt.	t
POPUP.HMIDDLE	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen, aus denen der horizontale Rahmen eines Popup-Fensters gebildet sein kann. Vorbelegung ist ein Punkt „.“.	t
POPUP.HEND	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen, mit denen der horizontale Rahmen eines Popup-Fensters endet. Vorbelegung ist ein Doppelpunkt gefolgt von einem Punkt „:.“, d.h. WebTransactions erkennt den horizontalen Rahmen eines Popups, wenn er mit einem Doppelpunkt oder einem Punkt endet.	t

Attributname	Bedeutung	Erläuterung/Kategorie	
POPUP.LINE	Zeilennummer der linken oberen Ecke des empfangenen Popups	Falls bei einem Aufruf von `receive` ein Popup empfangen wird und dieses Popup vom Capture-Mechanismus erkannt wird, enthält dieses Attribut die Zeilennummer, die die linke obere Ecke des Popups im Host-Format einnimmt. Mögliche Werte: 0 (falls kein Popup vorliegt) - *Maximale Zeilenanzahl* Voreinstellung: 0.	c, r
POPUP.VSTART	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen, mit denen der vertikale Rahmen eines Popup-Fensters beginnen kann. Vorbelegung ist ein Punkt „.“.	t
POPUP.VMIDDLE	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen, aus denen der vertikale Rahmen eines Popup-Fensters gebildet werden kann. Vorbelegung ist ein Doppelpunkt „:“.	t
POPUP.VEND	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen, mit denen der vertikale Rahmen eines Popup-Fensters enden kann. Vorbelegung ist ein Doppelpunkt „:“.	t
POPUP.WIDTH	Breite des empfangenen Popups	Falls bei einem Aufruf von `receive` ein Popup empfangen wird und dieses Popup vom Capture-Mechanismus erkannt wird, enthält dieses Attribut die Breite des Popups (Anzahl der Spalten einschließlich des Rahmens). Mögliche Werte: 00 (falls kein Popup vorliegt) - *Maximale Spaltenanzahl* (inkl. Rahmen) Voreinstellung: 0.	c, r
PORT_NUMBER	Portnummer	Portnummer für die Kommunikation mit der MVS-Anwendung über TCP/IP. Vorbelegung: 23	o
PRINTER_APPEND_FORMFEED	LU-Drucker-Steuerung	Ergänzt (falls nötig) ein FormFeed am Ende der Druckdaten. Das Attribut wird nur ausgewertet, wenn PRINTER_LU_NAME belegt ist. Vorbelegung: `No`	o

PRINTER_CODE_PAGE	LU-Drucker-Steuerung	Zuweisen einer EBCDIC-ASCII Umsetzungsdatei. Das Attribut wird nur ausgewertet, wenn PRINTER_LU_NAME belegt ist.
		Mögliche Werte für die mitgelieferte Konfiguration:
		`German_prt`
		`GerWin_prt`
			o

Attributname	Bedeutung	Erläuterung/Kategorie	
PRINTER_CONVERT_NILS_TO_BLANKS	LU-Drucker-Steuerung	Alle NULL-Zeichen werden zu Leerzeichen konvertiert. Das Attribut wird nur ausgewertet, wenn PRINTER_LU_NAME belegt ist. Voreinstellung: Yes	o
PRINTER_INSERT_LEADING_FORMFEED	LU-Drucker-Steuerung	Ergänzt ein FormFeed am Anfang der Druckdaten. Das Attribut wird nur ausgewertet, wenn PRINTER_LU_NAME belegt ist. Voreinstellung: No	o
PRINTER_LU_NAME	LU_NAME für den Drucker	Durch Zuweisen eines LU-Namens für PRINTER_LU_NAME wird zusätzlich zu der Dialog-Verbindung (die mit dem Attribut LU_NAME gesteuert werden kann) eine Drucker-Verbindung zu dem MVS-Host (siehe HOST_NAME, PORT_NUMBER) aufgebaut. Das Attribut PRINTFILE_NAME sollte ebenfalls sinnvoll belegt werden. Unter diesem Namen werden dann die Druckdaten, die vom Host kommen, zwischengespeichert.	o
PRINTER_REMOVE_LEADING_FORMFEED	LU-Drucker-Steuerung	Löscht ein FormFeed am Anfang der Druckdaten. Das Attribut wird nur ausgewertet, wenn PRINTER_LU_NAME belegt ist. Voreinstellung: No	o
PRINTFILE_NAME	Namen der Dateien, die auf Anforderung der Anwendung gedruckt werden sollen	Vereinbart, mit welchem Namensmuster nach zu druckenden Dateien für das BYPASS-Verfahren gesucht werden soll. Der Wert ist mit absoluten Pfaden anzugeben und wird bei der Suche um „.*“ ergänzt. PRINTFILE_NAME="C:\\tmp\\printer1" erkennt also alle Dateien, die dem Muster C:\tmp\printer1.* entsprechen. Existiert mindestens eine solche Datei, wird dieses mit dem Attribut BYPASS angezeigt. PRINTFILE_NAME wird auch verwendet bei der Auswertung von $MESSAGE.PRINTING. Diese Auswertung geschieht nur dann automatisch, wenn WT_ASYNC auf Yes gesetzt ist. Falls es zu diesem Zeitpunkt eine oder mehrere Dateien gibt, die dem Wert von PRINTFILE_NAME entsprechen, wird die Älteste gedruckt. Der Ausdruck folgt dem gleichen Prinzip wie der Terminal-Hardcopy-Ausdruck (siehe auch Abschnitt „Druckunterstützung“ auf Seite 156ff). Bei Verwendung des Attributes PRINTER_LU_NAME werden die Druckdaten ebenfalls an der unter PRINTFILE_NAME bezeichneten Stelle zwischengespreichert	o, t

Attributname	Bedeutung	Erläuterung/Kategorie	
PROLOG	Vorspann	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. Voreinstellung: keine Inkludierung **i** Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter. Siehe auch EPILOG und FORMTPL.	t
RECEIVED_BLOCKS	Nachrichtenzahl	Anzahl der während des letzten receive-Aufrufs verarbeiteten Nachrichten vom Host. Mit diesem Attribut kann überprüft werden, mit wie vielen Teilnachrichten das aktuelle Format vom Host ausgegeben wurde. Wenn dieser Wert bei allen Formaten 1 ist, kann das Attribut MULTIPLE_IO_TIMEOUT auf 0 gesetzt werden, da nach Erhalt der ersten Nachricht nicht mehr länger auf Vervollständigung des Formats gewartet werden muss. Ansonsten kann dieser Wert über das Attribut END_WAIT_CONDITION.EXPECTED_BLOCKS auch verwendet werden, um die Wartezeiten wegen MULTIPLE_IO_TIMEOUT abzubrechen.	c, r
RECORD_HOST_ COMMUNICATION	Schalter für Emulations-Trace	Wert Yes: Schaltet den Emulations-Trace ein Voreinstellung: No siehe auch OFFLINE_TRACEFILE	o
REFRESH_BY_ASYNC	Automatisches Setzen von Refresh, wenn Asynchron-Nachrichten eingetroffen sind.	Falls dieses Attribut auf Yes gesetzt ist, wurde das Host-Steuerobjekt WT_KEY.Key während des letzten send automatisch auf den Wert Refresh gesetzt, weil eine Asynchron-Nachricht seit dem letzten receive eingetroffen ist. Damit wurde beim letzten Aufruf der Form send/receive das alte Format aktualisiert. D.h. es kam nicht zu einem Dialogschritt mit der Host-Anwendung, wie es der Benutzer vielleicht erwartet. Dem Benutzer wird das aktualisierte Bild präsentiert. Um zu verhindern, dass der Benutzer dann vergeblich auf eine Antwort von der Host-Anwendung wartet, sollte man - abhängig vom Wert von REFRESH_BY_ASYNC - einen Hinweis für den Benutzer generieren. No (Vorbelegung) heißt, dass der Inhalt der Eingabefelder des Formats und der Wert von WT_KEY.Key unverändert zur Host-Anwendung gesendet wurde (und damit auch der Inhalt des Bildschirmpuffers). Siehe auch IGNORE_ASYNC	c

Attributname	Bedeutung	Erläuterung/Kategorie	
SYNCHRONIZE_ON_EMPTY_BLOCK	Automatisches erneutes Senden nach Empfangen eines asynchronen Protokollelements.	Bei den Emulationen, die per Telnet mit dem Host kommunizieren, existiert folgendes Problem: Nach dem vollständigen Bildschirm kann ein weiteres asynchrones Protokoll-Element vom Host eintreffen, das den Bildschirm nicht mehr verändert, aber von der Emulation quittiert wird. Im direkten Dialog mit dem Anwender verstreicht genügend Zeit, um diese Protokoll-Elemente auszutauschen. Dies erledigt die Emulation ohne Zutun des Host-Adapters. Wenn die neue Eingabe aber ohne jede Verzögerung (z.B. script-gesteuert) an den Host gesendet wird, wird diese Eingabe im Netz verworfen und die Host-Anwendung ist nicht mehr in dem erwarteten Zustand. Symptom: Obwohl eine Eingabe an den Host gesendet wurde, bleibt als Ergebnis der alte Bildschirm erhalten. SYNCHRONIZE_ON_EMPTY_BLOCK sorgt dafür, dass bei Eintreffen eines solchen asynchronen Protokoll-Elements nach dem Senden der nächsten Nachricht die gesendete Nachricht erneut gesendet wird. Dieses Attribut darf nur gesetzt werden, wenn bei Analyse der Daten zwischen Host und WebTransactions tatsächlich eine solche Situation erkannt wurde. Andernfalls würde ein zweifaches Senden ausgelöst, ohne dass die beschriebene Situation aufgetreten ist. In vielen Fällen ist es normal, dass erst eine sogenannte leere Nachricht z.B. zur Freigabe der Tastatur vom Host vor dem eigentlichen Bildschirminhalt verschickt wird. Diese zwei Situationen kann der Host-Adapter nicht unterscheiden. Voreinstellung: No	t

Attributname	Bedeutung	Erläuterung/Kategorie	
TERMINAL_TYPE	Terminaltyp	Terminaltyp wie er von WebTransactions simuliert wird. Mögliche Werte: "IBM-3278-2" 24x80 "IBM-3278-2-E" 24x80 "IBM-3278-3" 32x80 "IBM-3278-4" 43x80 "IBM-3278-4-E" 43x80 "IBM-3278-5" 27x132 "IBM-3278-5-E" 27x132 "IBM-3279-2-E" 24x80 "IBM-3279-3-E" 32x80 "IBM-3279-4" 43x80 "IBM-3279-4-E" 43x80 "IBM-3279-5" 27x132 "IBM-3279-5-E" 27x132	o
TRACE_LEVEL	Tracelevel	Steuert den Inhalt des Tracefiles. Mögliche Werte: 0,1,2,3,3E,3M,3EM Dabei bedeuten: – 0,1,2,3 unterschiedliche Tracelevels – E Ausgabe der Aufrufe der Emulationsfunktionen – M Ausgabe aller Host-Matritzen, d.h. der „rohen“ Maskendaten Vorbelegung: 3EM (= Maximal-Trace)	t
USE_POPUP_RECOGNITION	Flag für die Popup Erkennung	Ist auf Yes zu setzen, falls Popup-Fenster erkannt werden sollen. Vorbelegung: No.	t
WT_ASYNC	Ausdruck und asynchrone Meldungen	Ist auf Yes zu setzen, falls Ausdruckfunktionen und asynchrone Meldungen unterstützt werden sollen. Vorbelegung: No.	t
WT_BROWSER_PRINT	Aktivierung des Browser-Drucks nur für Browser-Plattform Windows	Ist auf Yes zu setzen, wenn der Browser-Druck aktiviert werden soll (siehe Abschnitt „Browser-Druck“ auf Seite 166). Die Attribute unter WT_BROWSER_PRINT_OPTIONS werden nur ausgewertet, wenn hier Yes gesetzt ist. Vorbelegung: No	t
WT_BROWSER_PRINT_OPTIONS.MODE	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Gibt an, ob sofort auf dem Standard-Drucker ausgedruckt oder eine Druck-Vorschau angezeigt werden soll. Mögliche Werte: Automatic Druck auf dem Standard-Drucker Preview Druck-Vorschau	t

Attributname	Bedeutung	Erläuterung/Kategorie	
WT_BROWSER_PRINT_OPTIONS.ORIENTATION	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Gibt die Seitenorientierung an, nur bei Verwendung des Internet Explorer. Mögliche Werte: Portrait Hochformat Landscape Querformat	t
WT_BROWSER_PRINT_OPTIONS.HEADER	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Text für den Seitenkopf, nur bei Verwendung des Internet Explorer (siehe Abschnitt „Variablen in Kopf- und Fußzeile“ auf Seite 168)	t
WT_BROWSER_PRINT_OPTIONS.FOOTER	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Text für die Fußzeile, nur bei Verwendung des Internet Explorer (siehe Abschnitt „Variablen in Kopf- und Fußzeile“ auf Seite 168)	t
WT_BROWSER_PRINT_OPTIONS.LEFT	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Linker Rand in mm, nur bei Verwendung des Internet Explorer	t
WT_BROWSER_PRINT_OPTIONS.RIGHT	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Rechter Rand in mm, nur bei Verwendung des Internet Explorer	t
WT_BROWSER_PRINT_OPTIONS.TOP	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Oberer Rand in mm, nur bei Verwendung des Internet Explorer	t
WT_BROWSER_PRINT_OPTIONS.BOTTOM	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Unterer Rand in mm, nur bei Verwendung des Internet Explorer	t

7.1.2 Zusammenspiel: Systemobjekt-Attribute und Methoden

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

open - Verbindung zum Host öffnen

Ein Aufruf der Methode `open` öffnet eine Verbindung zur Host-Anwendung. Welche Verbindung aufgebaut werden soll, bestimmt das folgende Attribut des kommunikationsspezifischen Systemobjekts, das Sie z.B. im Start-Template setzen können:

Sytemobjekt-Attribut	Wert
`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: *application_prefix@formatname*`.fld`
`HOST_NAME`	Name des Host-Rechners
`LU_NAME`	Name der Logical Unit für die Verbindung zum MVS-Host
`OFFLINE_COMMUNICATION`	Abspielen eines Emulations-Trace ohne Verbindung zur Host-Anwendung
`OFFLINE_LOGFILE`	Dateiname des Emulations-Trace, der aufgezeichnet werden soll.
`OFFLINE_TRACEFILE`	Dateiname eines Emulations-Trace, der abgespielt werden soll.
`PORT_NUMBER`	Portnummer für die Kommunikation mit der MVS-Anwendung über TCP/IP
`RECORD_HOST_COMMUNICATION`	Schalter für Emulations-Trace
`TERMINAL_TYPE`	Terminaltyp, wie er von WebTransactions simuliert wird.

Ein Aufruf von `open` schließt zuerst eine evtl. noch bestehende Verbindung. Anschließend wird die neue Verbindung aufgebaut. Beachten Sie bitte, dass unter Umständen erst beim ersten `receive`-Aufruf festgestellt werden kann, ob die Host-Anwendung erreichbar ist.

close - Verbindung zum Host schließen

Ein Aufruf der Methode `close` schließt die Verbindung zur Host-Anwendung. Diese Anweisung ist am Sitzungsende auszuführen. Sie führt grundsätzlich zu keiner Fehlermeldung.

send - Nachricht zur Host-Anwendung senden

Ein Aufruf der Methode `send` sendet im Allgemeinen eine Nachricht an die Host-Anwendung. Ob wirklich eine Kommunikation mit der Host-Anwendung stattfindet oder ob der Aufruf ausschließlich vom Host-Adapter bearbeitet wird (z.B. bei Refresh), wird von dem Host-Objekt-Attribut `WT_KEY.Key` gesteuert. Die Templates versorgen `WT_KEY.Key` mit einem Wert für die Funktionen, die `wtKeysMVS.htm` zur Verfügung stellt (siehe Abschnitt „Template wtKeysMVS.htm“ auf Seite 78). Eine wichtige Rolle spielen hier auch die Systemobjekt-Attribute `REFRESH_BY_ASYNC` und `IGNORE_ASYNC`. Wenn `REFRESH_BY_ASYNC` auf `Yes` gesetzt wurde und `IGNORE_ASYNC` auf `No` steht, wurde `WT_KEY.Key` während des letzten `send` automatisch auf den Wert `Refresh` gesetzt.

receive - Empfangen einer Nachricht von der Host-Anwendung

Ein Aufruf der Methode `receive` empfängt im Allgemeinen eine Nachricht von der Host-Anwendung. Ob ein Dialogschritt mit der Host-Anwendung stattfindet, hängt vom Wert des Host-Objekt-Attributs `WT_KEY.Key` ab.

Der Host-Adapter prüft, ob die von der Host-Anwendung empfangene Nachricht einem Erkennungskriterium in der Capture-Datenbank entspricht. Ist dies der Fall, hängt das weitere Verfahren ab vom Wert des Attributs `COMMUNICATION_INTERFACE_VERSION`:

- `COMMUNICATION_INTERFACE_VERSION` >= „3.0“

  Der Host-Adapter versorgt das Attribut `FLD` am verbindungsspezifischen Systemobjekt. Das Template muss selber dafür sorgen, dass `FORMAT` richtig gesetzt ist. In den generierten Templates sorgt die Funktion `setNextPage()` dafür, dass `FORMAT` ab Version V3.0 richtig gesetzt wird.

- `COMMUNICATION_INTERFACE_VERSION` < „3.0“

  Zusätzlich zu `FLD` wird auch `FORMAT` gesetzt, damit „alte“ Templates weiterhin ablauffähig sind.

Wird in der Capture-Datenbank kein Erkennungskriterium gefunden, dann wird `FLD` bzw. `FORMAT` mit dem Wert des Systemobjekt-Attributs `AUTOMASK` versorgt.

Weitere Informationen entnehmen Sie auch den Beschreibungen der Systemobjekt-Attribute „FIRST_IO_TIMEOUT“ auf Seite 99 und „MULTIPLE_IO_TIMEOUT“ auf Seite 102 sowie der Attribute der `END_WAIT_CONDITION`-Gruppe ab Seite 97.

7.2 Host-Objekte

WebTransactions for MVS verwendet zwei Arten von Host-Objekten:

– Host-Datenobjekte, die den Daten der Host-Anwendung entsprechen
– Host-Steuerobjekte, welche die Host-Anbindung steuern

7.2.1 Host-Datenobjekte

Für den Datenaustausch zwischen WebTransactions und der Host-Anwendung gibt es Host-Datenobjekte, die WebTransactions erzeugt, wenn ein neues Bildschirmformat (event. auch mit Popup) vom Host eingetroffen ist. Dann wird jedes Feld des Bildschirmformats (und
- falls vorhanden - des Popups) einem Host-Objekt zugeordnet. Nach einem Aufruf von `receive` stehen diese Objekte als Bildschirmabbild zur Verfügung.

Maßgebend für die Benennung dieser Objekte ist ihre Position im Bildschirmforma, bestimmt nach Zeile und Spalte:

E_*yy_xxx_lll* (für die Felder eines Bildschirmformats)
F_*yy_xxx_lll* (für die Felder eines Popups)

Erläuterung

E	steht für ein Feld des Bildschirmformats
F	steht für ein Feld eines Popups
yy	steht für die 2-stellige Zeilenposition; die oberste Zeile ist 1 (bei Popups wird der Popup-Rahmen nicht mitgezählt)
xxx	steht für die 3-stellige Spaltenposition; gezählt ab 1 (bei Popups wird der Popup-Rahmen nicht mitgezählt)
lll	steht für die 3-stellige Anzahl von Bildschirmzeichen in einer Bildschirmzeile ab der Position *yy_xxx*

Mit dieser Namensgebung kann auch beliebig auf eine Folge von Bildschirmzeichen innerhalb einer Bildschirmzeile zugegriffen werden. Es kann also beispielsweise auf eine Bildschirmzeile mit 80 Host-Datenobjekten der Länge 1 oder auf 1 Host-Datenobjekt der Länge 80 zugegriffen werden. Enthält ein Host-Datenobjekt mehr als ein Feld oder ist es Teil eines Feldes, werden die Attribute *`Value` entsprechend *yy*, *xxx* und *lll* versorgt. Alle anderen Attribute (`Input`, `Modified`, etc.) werden dem Feld entsprechend gesetzt, in dem das Host-Datenobjekt beginnt. In der Regel wird man allerdings auf Host-Objekte, die den Bildschirmfeldern entsprechen, zugreifen.

Wenn sich ein Host-Datenobjekt über Feldergrenzen hinaus erstreckt, wird es bei Bildschirmformaten bei Spalte 80 abgeschnitten, bei Popups wird bei der letzten Spalte des Popups abgeschnitten.

Feldattribute belegen ein Bildschirmzeichen. Sie werden durch ein Leerzeichen dargestellt und bilden ein eigenes Host-Objekt mit den Attributen `Type=Protected` und `Visible=No` (siehe Tabelle ab Seite 114).

Verkürzte Angabe von Host-Datenobjekten

Sie können die Objektnamen verkürzt angeben, indem Sie die Längenangabe oder die führenden Nullen weglassen. Diese Möglichkeiten sollten Sie allerdings vorsichtig nutzen: So liefert z.B. `WT_FOCUS` den Elementnamen immer in der vollständigen Schreibweise. Die erste der folgenden Abfragebedingungen hätte also nie den Wert `TRUE`:

```
<If (WT_FOCUS.Field == "E_1_2_3")>          --> immer falsch
...
<If (WT_FOCUS.Field == "E_01_002_003")>     --> richtig
```

Sprechende Namen von Host-Datenobjekten

Sie können in den Templates auch mit sprechenden Namen an Stelle der generischen Namen arbeiten. Die sprechenden Namen können Sie im WebLab im Dialogfeld **Hostobjekte grafisch auswählen** vergeben. Verwenden Sie dazu den Befehl **Umbenennen** im Kontextmenü zu jedem Objekt.

Zusätzlich müssen Sie dafür sorgen, dass das kommunikationsspezifische Systemobjekt-Attribut `FIELD_NAMES` den Wert `User-defined` enthält.

Attribute von Host-Datenobjekten

Die folgende Tabelle enthält alle Attribute der dynamischen Host-Datenobjekte. Fett ausgezeichnete Attribute können überschrieben werden.

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

Objektname	Attributname	Bedeutung des Attributs
E_*yy_xxx_lll* oder F_*yy_xxx_lll*	**VALUE**	Inhalt des Bildschirmfelds, das durch den Objektnamen repräsentiert wird. Binäre Nullen (NIL) werden durch Leerzeichen ersetzt, wenn nicht das Systemobjekt-Attribut NIL_MODE auf true gesetzt ist. Leerzeichen am Schluss des Feldes werden entfernt, einfache und doppelte Hochkommata sowie kaufmännisches Und (&) werden für die Ausgabe in HTML umgesetzt. Bei Eingabefeldern (siehe Type Unprotected) kann der Inhalt des Attributes VALUE geändert werden. Das Verändern von VALUE bildet die Tastatureingabe eines Benutzers an einem Terminal ab.
	HTMLVALUE	Der Inhalt entspricht dem des Attributs Value, wird aber ungekürzt zurückgegeben. Die folgenden Sonderzeichen werden für die Ausgabe in HTML umgesetzt: <, >, ä, ö, ü, Ä, Ö, Ü, ß
	RAWVALUE	Der Inhalt des Feldes wird als Sequenz von 8-Bit-Zeichen ohne Konvertierung zurückgegeben; lediglich binäre Nullen werden in Leerzeichen umgewandelt.
	STARTLINE	Zeile, in der das repräsentierte Bildschirmfeld beginnt, mögliche Werte: 1<= n<=*maximale Bildschirmhöhe* Diese hängt vom jeweiligen Bildschirmtyp ab: 24x80: n<=24 32x80: n<=32 43x80: n<=43 27x132: n<=27
	NAME	Name des Feldes.
	STARTCOLUMN	Spalte, in der das repräsentierte Bildschirmfeld beginnt; mögliche Werte: 1<=n<=*maximale Bildschirmbreite* Siehe auch Beispiel [1]
	LENGTH	Länge des Feldes, mögliche Werte: 1<=n<=*maximale Bildschirmbreite*
	TYPE	Angabe des Feldtyps Protected nicht überschreibbares Feld Unprotected Eingabefeld
	INPUT	Datentyp der Eingabe Alpha oder Numeric
	MODIFIED	Yes oder No
	BLINKING	Yes oder No
	UNDERLINE	Yes oder No
	VISIBLE	Yes oder No

Objektname	Attributname	Bedeutung des Attributs
	INTENSITY	`Normal` oder `Reduced`
	INVERSE	`Yes` oder `No`
	COLOR	Hier wird der RGB-Farbwert des betreffenden Feldes zurückgegeben. Der Wert hängt auch ab von den Einstellungen des Host-Objekts WT_COLOR, siehe dort. Wenn WT_COLOR nicht gesetzt wurden, werden folgende Werte zurückgegeben: `#000000`: Kein Farbattribut gesetzt `#0000FF`: Blau `#FF0000`: Rot `#FFC0CB`: Margenta `#008000`: Grün `#40E0D0`: Türkis `#FFFF00`: Gelb `#FFFFFE`: Weiß
	RangeName	Name des durch das Hostobjekt spezifizierten Bereichs.
	RangeLength	Länge des durch das Hostobjekt spezifizierten Bereichs
	RangeStart Column	Spalte, in der der durch das Hostobjekt spezifizierte Bereich beginnt. Mögliche Werte: 1<=n<=*maximale Bildschirmbreite*

[1] Der Unterschied zwischen `RangeStartColumn` und `StartColumn` soll anhand eines Beispiels erläutert werden.

Beim Empfang der Daten vom Host wurde u.a. das Feld E_03_020_010 erkannt:
E_03_020_010.StartColumn liefert 20, E_03_020_010.RangeStartColumn liefert ebenfalls 20

Nun Zugriff ungleich erkanntes Feld:
E_03_022_001.StartColumn liefert 20, E_03_022_001.RangeStartColumn liefert 22

Dito bei `RangeLength`, `RangeName` ...

7.2.2 Host-Steuerobjekte

Zur Steuerung der Host-Anbindung gibt es Host-Objekte, die für die gesamte Sitzung existieren.

- die Reihenfolge der Felder eines aktuellen Bildschirms
- das Feld, in dem sich der Cursor befindet
- Angabe, welche Datenfreigabetaste benutzt wird
- den vollständigen Namen eines Felds

Die folgende Tabelle enthält alle Host-Steuerobjekte sowie deren Attribute.
Fett ausgezeichnete Attribute können überschrieben werden.

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

Objektname	**Attribut**	**Bedeutung des Attributs**
$FIRST	Name	Name des ersten Felds des aktuellen Bildschirms in vollständiger Schreibweise. Existiert überhaupt kein Objekt, wird der Name `$END` zurückgegeben.
	Zusätzlich alle Attribute der dynamischen Host-Datenobjekte (`E_`*yy_xxx_lll*`)`	
$NEXT	Name	Name des nächsten Felds des aktuellen Bildschirms ausgehend von dem Feld, auf das zuletzt zugegriffen wurde und in vollständiger Schreibweise. Dieses Objekt können Sie verwenden, um Schritt für Schritt alle Felder des Bildschirms durchzugehen. Existiert kein weiteres Objekt mehr, wird der Name `$END` zurückgegeben. Achtung: Auch Attributfelder werden von `$NEXT` berücksichtigt. Da 3270-Felder durch ein Attribut-Byte eingeleitet werden, liefert `$NEXT` zunächst das Attribut-Byte (durch einen Space dargestellt). Das folgende `$NEXT` liefert dann den Feld-Inhalt.
	Zusätzlich alle Attribute der dynamischen Host-Datenobjekte (`E_`*yy_xxx_lll*`)`	
$SCREEN	CONTENT	Alle Zeichen des gesamten Bildschirms in einem String. Diese Angabe kann dazu verwendet werden, zwei Bildschirme zu vergleichen.

Objektname	Attribut	Bedeutung des Attributs
IND$FILE	**LOCAL_FILE**	Bezeichnet die Datei, die auf Seiten des WebTransactions-Rechners am Transfer beteiligt ist. Dieser Wert wird von den Attributen PUT bzw. GET ausgewertet.
	HOST_FILE	Bezeichnet die Datei, die auf Seiten des MVS-Host am Transfer beteiligt ist. Dieser Wert wird von den Attributen PUT bzw. GET ausgewertet. Vor dem Namen sollte als Prefix TSO:, CICS: oder CMS: eingefügt werden, um der Emulation den Zustand der Dialog-Sitzung mitzuteilen. Von dieser Angabe wird die Syntax des Kommandos IND$FILE beeinflusst. Voreinstellung: TSO:
	PUT	Das Setzen dieses Attributes löst einen File-Transfer von LOCAL_FILE nach HOST_FILE aus. Der dem Attribut PUT zugewiesene Wert enthält die für den Transfer zu benutzenden Optionen. Diese Optionen werden als Teil des Kommandos IND$FILE an den Host gesendet. **i** Bitte beachten Sie, dass das Kommando IND$FILE nicht in jedem Zustand der Dialog-Sitzung akzeptiert wird und die Syntax abhängig vom Zustand ist (TSO command mode, CMS, CICS).
	GET	Das Setzen dieses Attributes löst einen File-Transfer von HOST_FILE nach LOCAL_FILE aus. Der dem Attribut GET zugewiesene Wert enthält die für den Transfer zu benutzenden Optionen. Diese Optionen werden als Teil des Kommandos IND$FILE an den Host gesendet. **i** Bitte beachten Sie, dass das Kommando IND$FILE nicht in jedem Zustand der Dialog-Sitzung akzeptiert wird und die Syntax abhängig vom Zustand ist (TSO command mode, CMS, CICS).
WT_FOCUS	**Field**	Objektname des Felds, in dem der Cursor sich gerade befindet. Der Name wird vollständig einschließlich Länge angegeben, und zwar in der Form E_*yy_xxx_lll*. Durch Schreiben des Attributs wird der Cursor positioniert. Ist das Attribut leer oder ungültig, wird der Cursor auf die erste Spalte der letzten Zeile gesetzt.
	OFFSET	Versatz des Cursors vom Feldanfang der Cursor-Position
WT_FOCUS_ SHORT	**Field**	Wie WT_FOCUS, aber der Objektname wird ohne Länge abgelegt in der Form E_*yy_xxx*.

Objektname	Attribut	Bedeutung des Attributs
WT_KEY	**Key**	Gibt an, welche Sondertaste in der Terminal-Emulation ausgeführt werden soll, wenn `send` ausgeführt wird. Die voreingestellten Werte sind in der Tabelle auf der Seite 123 beschrieben. Entsprechende Knöpfe werden durch `wtKeysMVS.htm` in die aktuellen Templates inkludiert. (`<wtInclude ...>`).
WT_COLOR	**DEFAULT**	RGB-Farbwert für Felder, für die kein Farbattribut gesetzt ist. Mögliche Werte: `#000000`: Schwarz (Voreinstellung) `#0000FF`: Blau `#FF0000`: Rot `#FF0CB`: Margenta `#008000`: Grün `#40E0D0`: Türkis `#FFFF00`: Gelb `#FFFFFE`: Weiß
	BLUE	RGB-Farbwert für Felder mit dem Farbattribut `blue` Voreinstellung: `#0000FF`
	RED	RGB-Farbwert für Felder mit dem Farbattribut `red` Voreinstellung: `#FF0000`
	PINK	RGB-Farbwert für Felder mit dem Farbattribut `pink` Voreinstellung: `#FFC0CB` Als Alias zu `PINK` wird auch `MAGENTA` akzeptiert, um gemeinsame Scripts für verschiedene Host-Verbindungen zu erlauben.
	GREEN	RGB-Farbwert für Felder mit dem Farbattribut `green` Voreinstellung: `#008000`
	TURQUOISE	RGB-Farbwert für Felder mit dem Farbattribut `turquoise` Voreinstellung: `#40E0D0` Als Alias zu `TURQUOISE` wird auch `CYAN` akzeptiert, um gemeinsame Scripts für verschiedene Host-Verbindungen zu erlauben.
	YELLOW	RGB-Farbwert für Felder mit dem Farbattribut `yellow` Voreinstellung: `#FFFF00`
	WHITE	RGB-Farbwert für Felder mit dem Farbattribut `white` Voreinstellung: `#FFFFFE`

Objektname	Attribut	Bedeutung des Attributs
$MESSAGE	PRINTING	Wenn dieses Attribut ausgewertet wird, sendet WebTransactions die älteste zum Druck anstehende Datei an den Browser (mit dem MIME-Typ `webta/hardcopy-print` oder MIME-Typ `webta/bypass-print`). **i** Beachten Sie, dass mit der Auswertung von `$MESSAGE.PRINTING` die Interpretation des Templates abgebrochen und die HTML-Generierung unterdrückt wird.
	PRINTFILE_ NAME	Dieses Attribut liefert den Namen der Datei, die als nächste zum Drucken ansteht. Wenn keine Datei zum Drucken vorliegt, wird ein Leerstring zurückgeliefert.
	WAITING	Dieses Attribut zeigt an, ob WebTransactions eine asynchrone Nachricht von der Host-Anwendung empfangen hat. Jedes Mal, wenn dieses Attribut abgefragt wird, prüft WebTransactions, ob sich eine asynchrone Nachricht im Puffer befindet. Falls ja, wird der Wert von `$MESSAGE.WAITING` intern auf „`Yes`“ gesetzt. WebTransactions liest diese asynchrone Meldung dann beim nächsten Aufruf von `receive`.

7.3 Unterstützung von Terminal-Funktionen durch den Browser

Bei WebTransactions for MVS 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 `MVS.wmt` bzw. `MVS_Pocket.wmt` verwenden (siehe Seite 66). Die Templates, die Sie über das Master-Template generiert haben, inkludieren die Templates `wtBrowserFunctions.htm` und `wtKeysMVS.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

`wtKeysMVS.htm` enthält die MVS-spezifischen Standard-Tasten und inkludiert die JavaScript-Datei `wtKeysMVS.js`, die die Abbildung der Sondertasten für WebTransactions for MVS enthält. In dieser Datei können Sie die Abbildung der Tasten entsprechend Ihren Wünschen anpassen oder erweitern (siehe Abschnitt „Zuordnung der Tasten in wtKeysMVS.js“ auf Seite 125).

7.3.1 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.

<table>
<tr><th rowspan="2">Terminal-Funktion</th><th colspan="4">Unterstützung durch Browser</th></tr>
<tr><th>nicht speziell behandelte Browser</th><th>Netscape V4.0</th><th>Netscape ab V6.0 oder Gecko</th><th>Internet Explorer ab V4.0</th></tr>
<tr><td>Ausrichtung von Text und Eingabefeldern</td><td>nein</td><td>nein</td><td>ja</td><td>ja</td></tr>
<tr><td>Unterstützung für Terminal-Sondertasten, die an WebTransactions gesendet werden</td><td>nur über eine Auswahl-Liste/Schaltfläche</td><td>nur die ENTER-Taste direkt, alle anderen über eine Auswahl-Liste/Schaltfläche</td><td colspan="2">durch individuelle konfigurierbare Abbildung über eine Taste oder Auswahl-Liste/Schaltfläche</td></tr>
<tr><td>Unterstützung von Terminal-Sondertasten, die direkt im Browser-Formular wirken</td><td>nein</td><td>nein</td><td colspan="2">durch individuelle konfigurierbare Abbildung über eine Taste</td></tr>
<tr><td>Autotab</td><td>nein</td><td>ja</td><td>ja</td><td>ja</td></tr>
<tr><td>Überschreiben von Feldern</td><td colspan="3">ja
(wird im Browser simuliert durch automatische Auswahl des Feldinhalts)</td><td>ja</td></tr>
<tr><td>Übermitteln der Cursor-Position</td><td colspan="2">Nur Position vom Beginn des zuletzt benutzten Eingabefeldes</td><td>Position vom Beginn des zuletzt benutzten Eingabefeldes und exakte Position in geschützten Feldern</td><td>exakte Position in geschützten Feldern und in Eingabefeldern (ab V5.0)</td></tr>
<tr><td>Tabulator bleibt innerhalb des Formulars</td><td colspan="2">nein</td><td colspan="2">ja</td></tr>
</table>

Tastenunterstützung durch Internet Explorer ab V4.0

Wenn Sie den Internet Explorer ab V4.0 als Browser verwenden, dann haben die Tasten des 3270-Terminals folgende Entsprechung[1]:

Tasten bei Verwendung des Internet Explorer	entsprechende Taste am 3270-Terminal
ENTER	ENTER
F1 ... F12	PF1 ... PF12
Shift+F1 ... Shift+F12	PF13 ... PF24
STRG+F1	PA1
STRG+F2	PA2
STRG+F3	PA3
Alt+F9	Attn
Alt+F10	Sysreq
Pause	Clear
ESC	Reset

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

7.3.2 Zusammenspiel von Host-Steuerobjekt WT_KEY, Template wtKeysMVS.htm und Datei wtKeysMVS.js

Das Host-Steuerobjekt `WT_KEY` gibt an, welche Sondertaste in der Terminal-Emulation ausgeführt werden soll, wenn `send` ausgeführt wird.

Die voreingestellten Werte sind in der untenstehenden Tabelle beschrieben. Entsprechende Knöpfe werden durch `wtKeysMVS.htm` in die aktuellen Templates inkludiert (`<wtInclude ...>`). `wtKeysMVS.htm` inkludiert die JavaScript-Datei `wtKeysMVS.js`, die die Abbildung der Sondertasten für WebTransactions for MVS enthält (siehe Abschnitt „Zuordnung der Tasten in wtKeysMVS.js“ auf Seite 125).

Die folgende Tabelle zeigt:

– die Bedienungselemente, die das Template `wtKeysMVS.htm` und die inkludierte Datei `wtKeysMVS.js` standardmäßig zur Verfügung stellen.

– die Werte, welche für die damit verbundenen Funktionen im Host-Steuerobjekt `WT_KEY` hinterlegt werden

Taste auf 3270-Terminal	Wert von WT_KEY.Key	Bedeutung
ENTER	@E	Datenfreigabe an den Host
RESET	@R	Zurücksetzen einer Fehlersituation; z.B. nach dem Eintragen von Daten in ein Feld, für das dies nicht erlaubt ist
ATTN	@A@Q	Kurznachricht an die Host-Anwendung; es werden keine weiteren Daten übertragen.
CLEAR	@C	Löschen des Bildschirms mit entsprechender Kurznachricht an die Host-Anwendung
PF1..PF9	@1..@9	wie ENTER Datenfreigabe an den Host, aber mit anderem Übertragungskennzeichen
PF10..PF24	@a..@o	wie PF1..PF9
SYSREQ	@A@H	System-Request: Umschalten auf ein zweites Fenster, in dem dann eine Systemnachricht gesendet werden kann
PA1..PA3	@x..@z	Kurznachricht an die Host-Anwendung
	Disconnect	Die Verbindung zum Host wird getrennt. Der folgende Aufruf `receive` versorgt daraufhin Systemobjekt-Attribut `FLD` mit dem Inhalt von Systemobjekt-Attribut `DISCONNECT`. Diese Funktion entspricht in etwa dem Ausschalten eines realen Terminals oder dem Beenden eines Emulationsprogramms, ohne sich bei der Host-Anwendung ordnungsgemäß abzumelden.

Taste auf 3270-Terminal	Wert von WT_KEY.Key	Bedeutung
	`Refresh`	Ausgabe der HTML-Seite erneuern. Diese Funktion wird im Wesentlichen benötigt, um Nachrichten sichtbar zu machen, die asynchron oder verspätet eingegangen sind, also nachdem der Bildschirminhalt als vollständig angenommen und als HTML-Seite zum Browser gesendet wurde (siehe hierzu auch Systemobjekt-Attribut `MULTIPLE_IO_TIMEOUT`, Seite 102). Die Funktion `Refresh` führt nicht zu einem Dialogschritt mit der Host-Anwendung: Bei den Aufrufen von `send` und `receive` wird die Host-Kommunikation unterdrückt.
	`Cancel Menu`	In der ersten Bildschirmzeile kann bei bestimmten Host-Anwendungen mittels markierbarer Felder ein Pull-Down-Menü angewählt werden (siehe Seite 85). Die Funktion `Cancel Menu` schließt das Menü wieder; der ursprüngliche Bildschirminhalt wird erneut ausgegeben. Diese Funktion führt zum Dialog mit der Host-Anwendung.
	`Print`	Anfordern eines Terminal-Hardcopy-Drucks (siehe Abschnitt „Terminal-Hardcopy-Druck“ auf Seite 157).

7.3.3 Zuordnung der Tasten in wtKeysMVS.js

Alle Tastatureingaben werden von dem verwendeten Browser angenommen. Für die anwendungsspezifische Zuordnung von speziellen Funktionstasten stellt WebTransactions als Schnittstelle die Datei `wtKeysMVS.js` zur Verfügung. Aus dieser Datei wird durch Aufruf der Funktion `wtCreateKeySelectList()` eine Auswahl-Liste generiert (siehe Abschnitt „Zusammenspiel von wtCommonBrowserFunctions.js und wt<browser>BrowserFunctions.js“ auf Seite 131).

Im folgenden Text ist die Abbildung der Tasten beschrieben, wie sie von WebTransactions ausgeliefert wird. Sie können diese Schnittstelle ohne tiefere Kenntnis der Browser-Templates nutzen.

Die Datei `wtKeysMVS.js` liegt nach dem Anlegen des Basisverzeichnisses im Verzeichnis <*basedir*>`/wwwdocs/javascript`. Die wesentliche Schnittstelle für die Anpassung der WebTransactions-Anwendung ist die Tabelle (Array) `wtKeyMappingTableInput` in dieser Datei.

In der Tabelle `wtKeyMappingTableInput` wird für jede Tasten-Abbildung ein Objekt mit mehreren Attributen angelegt (siehe Tabelle auf Seite 127). 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

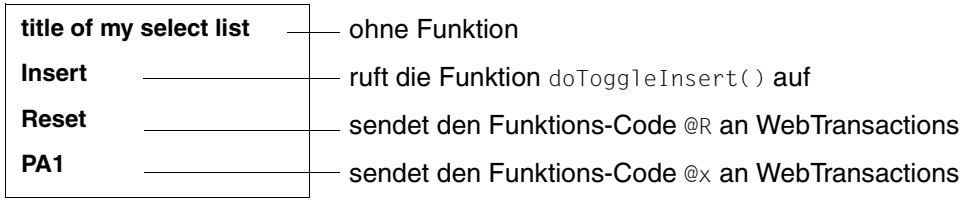
Beispiel

```
wtKeyMappingTableInput = [
{ sl:'title of my select list'},
{ la:'Insert', co:'Insert', ac:doToggleInsert, kc:VK_INS },
{ la:'Reset', co:'RESET', ac:'@R', kc:VK_ESC, mk:MK_SHIFT },
{ la:'PA1', ac:'@x', kc:VK_F1, mk:MK_CTRL }
];
```

Aus dieser Definition entsteht folgende Abbildung:

- `Insert` ruft die Funktion `doToggleInsert` auf (Einfügemodus ein/ausschalten)
- `ESC+SHIFT` sendet den Funktions-Code `@R` (entspricht `Reset`) an WebTransactions
- `F1+CRTL` sendet den Funktionscode `@x` (entspricht Taste `PA1`) an WebTransactions

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



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 `la` gedacht; durch Ändern des Attributes `la` nach `co` kann z.B. eine Taste aus der Auswahl-Liste entfernt werden.
`ac`	`action`	Auszuführende Aktion bei Auslösen der zugeordneten Taste oder Auswahl aus einer Liste. Ist dieses Attribut vom Typ `string`, dann wird der Inhalt nach `wt_special_key.value` übertragen und an WebTransactions gesendet. Es wird also das Formular an WebTransactions übertragen und als Sonderfunktion der Wert von `ac` mitgegeben (z.B. `F1` als Funktionstaste). Ist dieses Attribut vom Typ `function`, dann wird eine Client-seitige Funktion mit diesem Namen aufgerufen. Diese Funktion muss definiert sein. Die JavaScript-Dateien `wt<`*browser*`>BrowserFunctions.js` stellen folgende Funktionen zur Verfügung: - `doCursorHome` - `doCursorUp` - `doCursorDown` - `doCursorLeft` - `doCursorRight` - `doTab` - `doBackTab` - `doToggleMark` – `doToggleInsert`. Die Implementierung dieser Funktionen kann abhängig von den Fähigkeiten des Browsers auch leer sein, siehe Abschnitt „Callback-Funktionen des Key mapping“ auf Seite 132. Ist dieses Attribut nicht definiert, kann keine Aktion durchgeführt werden. Die Tastatur-Eingabe wird dem Browser zur Bearbeitung überlassen.
`kc`	`key code`	Nummer, die im Tastaturtreiber der gedrückten Taste zugeordnet ist. Für viele Tasten existiert in `wtCommonBrowserFunctions.js` ein Symbol, die Namen beginnen mit `VK_`. Für Tastenkombinationen ist zusätzlich der `modifier key (mk)` von Bedeutung.

Bezeichnung	Attribut	Bedeutung
mk	modifier key	zusätzlich gedrückte Umschalttaste (siehe Definition in wtCommonBrowserFunctions.js): – 0 = MK_NONE (= keine Umschalttaste gedrückt) – 1 = MK_CTRL – 2 = MK_ALT – 4 = MK_SHIFT bei Kombinationen werden die entsprechenden Werte addiert: – 3 = MK_CTRL + MK_ALT – 5 = MK_CTRL + MK_SHIFT – usw. Ist kein mk angegeben, wird der Wert 0 = MK_NONE verwendet!
sl	select list	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 sl angegeben. Eine ggf. bereits vorher angelegte Auswahl-Liste wird beim Auftreten von sl beendet. Für bessere Lesbarkeit kann sl einziges Attribut des Tabellen-Objektes sein.

Aufbau von wtKeysMVS.js

Im folgenden ist die Tabelle `wtKeyMappingTableInput` aus der Datei `wtKeysMVS.js` dargestellt, wie sie von WebTransactions ausgeliefert wird:

Das Objekt `wtKeyMappingTableInput` wird als Literal angelegt.

```
wtKeyMappingTableInput = [
```

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

```
{ sl:'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:'ENTER', ac:'@E', kc:13, mk:MK_NONE },
{ 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 },
```

`Reset` erscheint als Eintrag in der Auswahl-Liste (Attribut `la`). Außerdem wird `RESET` auf die Taste `ESC` und die Tastenkombination `SHIFT` + `ESC` abgebildet. Der Anwender kann also, um `RESET` auszulösen

- `RESET` aus der Liste auswählen
- `ESC` auf der Tastatur drücken
- `SHIFT` + `ESC` auf der Tastatur drücken

```
{ la:'RESET', ac:'@R', kc:VK_ESC, mk:0 },
{ co:'RESET', ac:'@R', kc:VK_ESC, mk:MK_SHIFT },
{ la:'ATTN', ac:'@A@Q', kc:VK_F9, mk:MK_ALT },
{ la:'CLEAR', ac:'@C', kc:VK_PAUSE, mk:0 },
{ co:'CLEAR', ac:'@C', kc:VK_PAUSE, mk:MK_SHIFT },

{ la:'PF1', ac:'@1', kc:VK_F1, mk:0 },
{ la:'PF2', ac:'@2', kc:VK_F2 },
{ la:'PF3', ac:'@3', kc:VK_F3 },
```

```
{ la:'PF4', ac:'@4', kc:VK_F4 },
{ la:'PF5', ac:'@5', kc:VK_F5 },
{ la:'PF6', ac:'@6', kc:VK_F6 },
{ la:'PF7', ac:'@7', kc:VK_F7 },
{ la:'PF8', ac:'@8', kc:VK_F8 },
{ la:'PF9', ac:'@9', kc:VK_F9 },
{ la:'PF10', ac:'@a', kc:VK_F10 },
{ la:'PF11', ac:'@b', kc:VK_F11 },
{ la:'PF12', ac:'@c', kc:VK_F12 },
{ la:'PF13', ac:'@d', kc:VK_F1, mk:MK_SHIFT },
{ la:'PF14', ac:'@e', kc:VK_F2, mk:MK_SHIFT },
{ la:'PF15', ac:'@f', kc:VK_F3, mk:MK_SHIFT },
{ la:'PF16', ac:'@g', kc:VK_F4, mk:MK_SHIFT },
{ la:'PF17', ac:'@h', kc:VK_F5, mk:MK_SHIFT },
{ la:'PF18', ac:'@i', kc:VK_F6, mk:MK_SHIFT },
{ la:'PF19', ac:'@j', kc:VK_F7, mk:MK_SHIFT },
{ la:'PF20', ac:'@k', kc:VK_F8, mk:MK_SHIFT },
{ la:'PF21', ac:'@l', kc:VK_F9, mk:MK_SHIFT },
{ la:'PF22', ac:'@m', kc:VK_F10, mk:MK_SHIFT },
{ la:'PF23', ac:'@n', kc:VK_F11, mk:MK_SHIFT },
{ la:'PF24', ac:'@o', kc:VK_F12, mk:MK_SHIFT },

{ la:'PA1', ac:'@x', kc:VK_F1, mk:MK_CTRL },
{ la:'PA2', ac:'@y', kc:VK_F2, mk:MK_CTRL },
{ la:'PA3', ac:'@z', kc:VK_F3, mk:MK_CTRL },
{ la:'SYSREQ', ac:'@A@H', kc:VK_F10, mk:MK_ALT },
```

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 }
];
```

7.3.4 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 `wtKeysMVS.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_0     = 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 `wtKeysMVS.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 `wtKeysMVS.htm`. Durch Weglassen des Aufrufs dieser Funktion in `wtKeysMVS.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 127).

Die meisten dieser Funktionen liefern `false` als Ergebnis, um zu signalisieren, dass keine allgemeine Behandung 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.

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 127) 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 127) eine Aktion definiert ist.

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

4. Der Inhalt des `string` wird in das Attribut `wt_special_key.value` übertragen (siehe Abschnitt „Template wtKeysMVS.htm“ auf Seite 78). Das Formular wird an WebTransactions übergeben und der Wert von `ac` (z.B. „`@1`“ für die PF1-Taste) 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 127) 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 des Cursor 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.

7.3.5 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 `MVS.wmt` oder `MVS_Pocket.wmt` generiert wurden (z.B. `AutomaskMVS.htm`)
- `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 Schriftgößen 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 `AutomaskMVS.htm` und 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>
```

7.4 Start-Templates für MVS

Nach dem 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 (sofort ablauffähig)**

  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 MVS wird das Start-Template `wtstartMVS.htm` ausgeliefert. Dieses MVS-spezifische Start-Template wird ab Seite 138 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 MVS-Anwendung sollten Sie ein eigens dafür generiertes Start-Template verwenden. Bei der Generierung werden Sie vom WTBean `wtcStartMVS.wtc` unterstützt.

7.4.1 MVS-spezifisches Start-Template des Start-Template-Sets (wtstartMVS.htm)

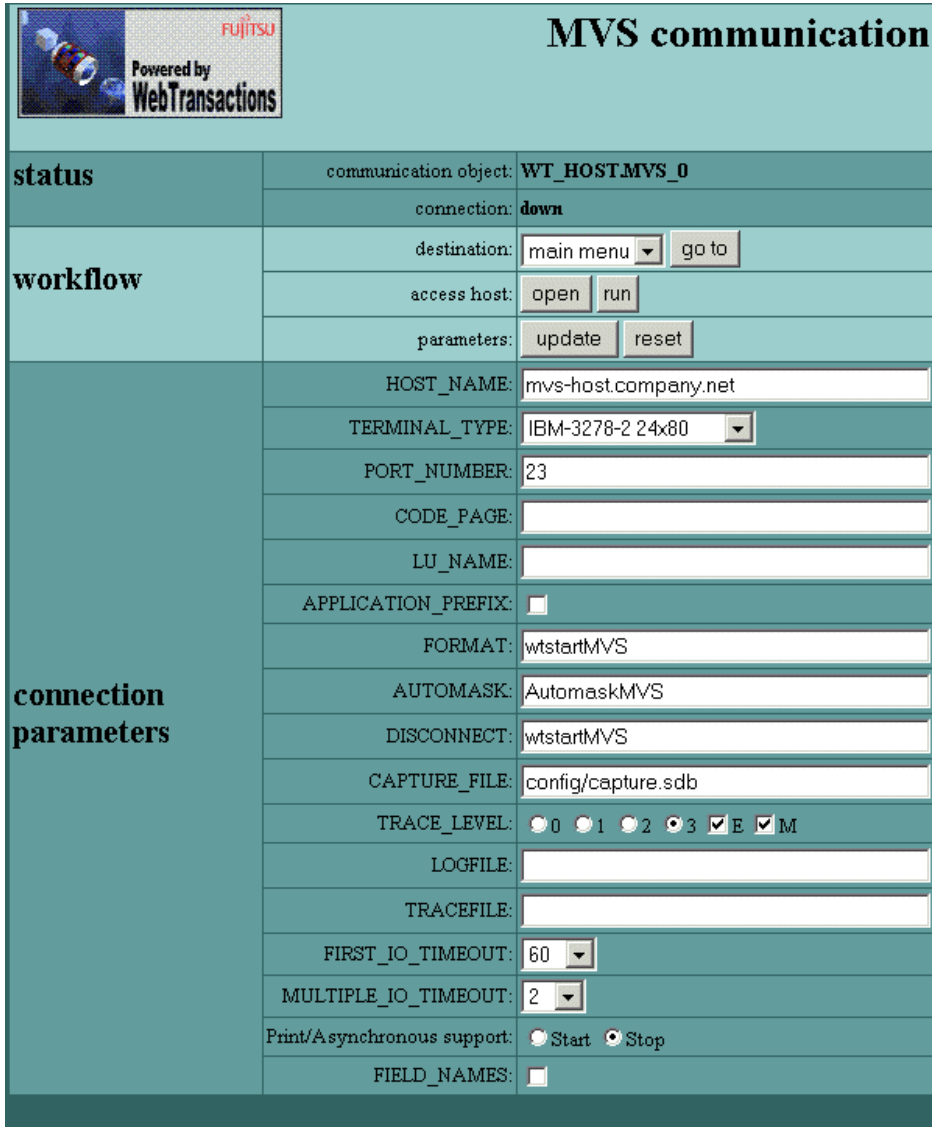
Wenn Sie im allgemeinen Start-Template `wtstart.htm` (beschrieben im WebTransactions-Handbuch „Konzepte und Funktionen“) das Protokoll `MVS` ausgewählt und ein neues Kommunikationsobjekt angelegt haben, wird zum Template `wtstartMVS.htm` verzweigt. Dieses Template ermöglicht das Setzen der MVS-spezifischen Parameter:

– Sie können Verbindungsparameter festlegen und eine Verbindung zu einer MVS-Anwendung öffnen. Wenn Sie hier die Option **run** wählen, wird die Verbindung zur Hostanwendung aufgebaut (**open**) und der erste Schirm der Anwendung geholt (**receive**).

– Wenn Sie die Option **open** wählen, wird die Verbindung zur Host-Anwendung aufgebaut und das Template `wtstartMVS.htm` erneut angezeigt. Es enthält nun - falls die Verbindung erfolgreich geöffnet werden konnte - zusätzliche Knöpfe für die Kommunikation mit der MVS-Anwendung. Wenn Sie hier die Option **close** wählen, wird die Verbindung zur Host-Anwendung beendet.

– Wenn Sie die Option **receive** wählen, wird das Template `wtstartMVS.htm` erneut angezeigt. Es enthält jetzt einen neuen Abschnitt **host attributes**, in dem Sie Host-Attribute z.B. für die Popup-Erkennung setzen können. Mit dem Knopf **enter dialog** wird der erste Schirm der Host-Anwendung eingeblendet.

Verbindungsparameter setzen und Verbindung öffnen



Im Abschnitt **connection parameters** können Sie die gleichnamigen Attribute des Systemobjekts auf die gewünschten Werte setzen (siehe Abschnitt „Systemobjekt-Attribute“ auf Seite 93).

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 ausgewählt Seite verzweigt. Als Vorgabe wird `main menu` angeboten: Damit kann man auf die allgemeine Startseite `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 hier **open** und **run** zur Verfügung:

open
Mit diesem Knopf wird eine Verbindung zum Host erzeugt. Das Start-Template zeigt nun zusätzliche Knöpfe für die Kommunikation.

run
Auch mit diesem Knopf wird - wie mit **open** - eine Verbindung zum Host erzeugt. Zusätzlich wird jedoch auch die erste Nachricht vom Host empfangen und diese Nachricht im Web-Browser angezeigt.

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.

Kommunikation aufnehmen
(nur möglich, während eine Verbindung zum Host mit open geöffnet wurde)

Sobald eine Verbindung geöffnet ist, werden im Abschnitt **workflow** unter **access host** folgende Knöpfe angeboten, die eine Kommunikation mit der Host-Anwendung ermöglichen. Zu jedem Zeitpunkt der Kommunikation werden nur die zu diesem Zeitpunkt möglichen Knöpfe angeboten. Falls Sie **open** gewählt haben, sind das die Knöpfe **receive** und **close**:

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. **send** sendet den aktuellen Datenpuffer, ohne Daten zu verändern.

close
schließt die Verbindung zur Host-Anwendung und kehrt auf die erste Seite zurück. Dort können Sie eine neue Verbindung auswählen und öffnen.

Host-Attribute setzen
(nur möglich, während eine Verbindung zum Host offen ist und über send/receive mindestens ein Dialog mit dem Host stattgefunden hat)

Es wird ein neuer Abschnitt **host attributes** angeboten, in dem Sie die Host-Attribute für die Popup-Erkennung und Tastenumsetzung angeben können. Folgende Knöpfe werden im weiteren Verlauf der Kommunikation angezeigt, falls eine Nachricht von der Host-Anwendung empfangen wurde.

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. **send** sendet den aktuellen Datenpuffer, ohne Daten zu verändern. Soll z.B. im ersten Schritt schon ein Schirm mit veränderten Daten an die Host-Anwendung geschickt werden, so ist die Auswahl **enter dialog** zielführend.

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.

close
schließt die Verbindung zur Host-Anwendung und kehrt auf die erste Seite zurück. Dort können Sie eine neue Verbindung auswählen und öffnen.

7.4.2 WTBean wtcStartMVS.wtc zur Generierung eines Start-Templates

Für den Anschluss einer einzelnen MVS-Anwendung können Sie ein anwendungs-spezifisches Start-Template generieren. Dazu steht Ihnen das WTBean `wtcStartMVS.wtc` zur Verfügung. Es handelt sich dabei um ein standalone WTBean.

`wtcStartMVS.wtc` enthält das inline WTBean `wtcMVS.wtc`, mit dem Sie in einem Template ein neues MVS-Kommunikationsobjekt anlegen, und damit eine Verbindung zu einer MVS-Anwendung aufbauen (siehe Abschnitt
„Neues MVS-Kommunikationsobjekt anlegen (wtcMVS)“ auf Seite 144).

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

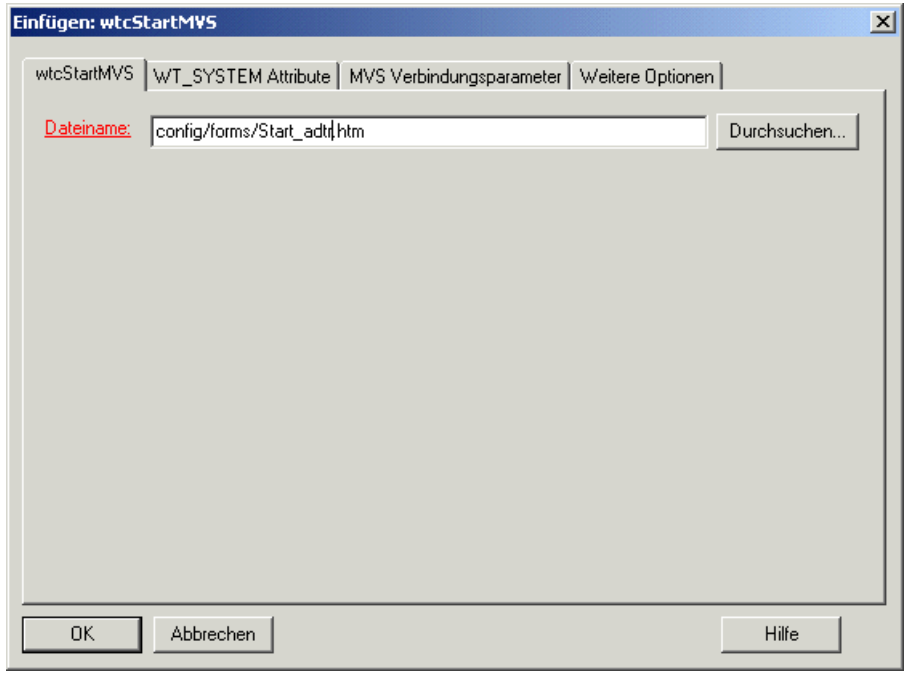
Mit dem Befehl **Datei/Neu/wtcStartMVS** rufen Sie das WTBean zur Bearbeitung auf. WebLab generiert das Dialogfeld **Einfügen:wtcStartMVS** mit vier Registerkarten:

– In der Registerkarte **wtcStartMVS** 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 der Registerkarte **MVS-Verbindungsparameter** legen Sie die wichtigsten Verbindungsparameter fest.

– In der Registerkarte **Weitere Optionen** können Sie in einer Baumstruktur alle Parameter für die Verbindung zur MVS-Anwendung bearbeiten.



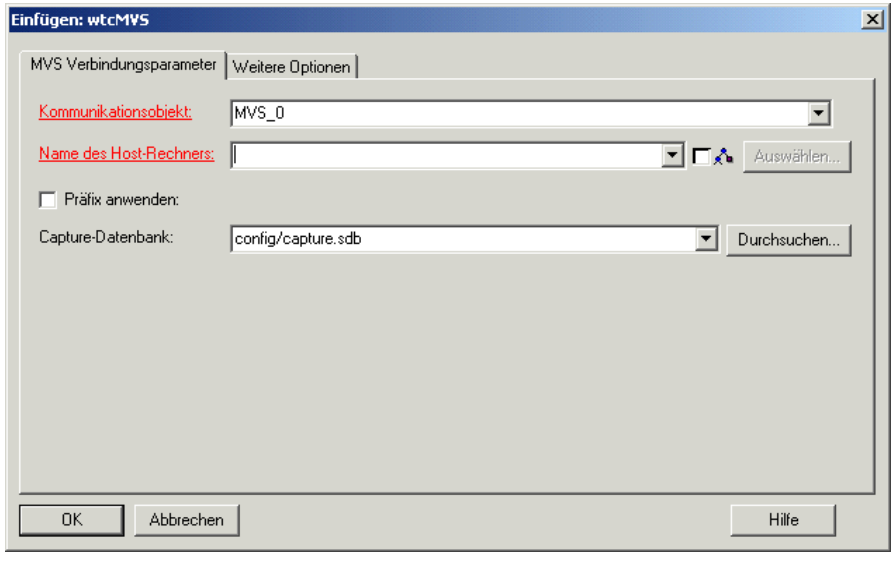
Das generierte Start-Template selbst erzeugt keine eigene Seite im Browser. Nach Start von WebTransactions erscheint unmittelbar das Template, das dem ersten von der Host-Anwendung empfangenen Format entspricht. Dafür sorgt das `wtinclude`-Tag am Ende des Start-Templates.

7.5 Neues MVS-Kommunikationsobjekt anlegen (wtcMVS)

Um in einem Template ein neues MVS-Kommunikationsobjekt anzulegen und damit eine Verbindung zu einer MVS-Anwendung aufzubauen, wird das WTBean `wtcMVS` mit ausgeliefert. Dieses WTBean können Sie auch dazu nutzen, um mehrere Verbindungen parallel zu öffnen. `wtcMVS` ist ein inline WTBean, siehe hierzu auch das WebTransactions-Handbuch „Konzepte und Funktionen“.

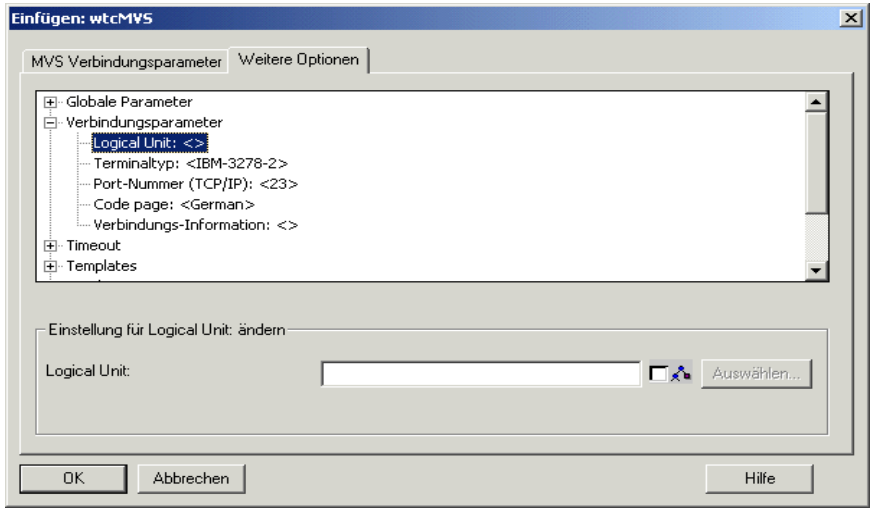
Damit Sie auf inline WTBeans zugreifen können, muß eine Verbindung zur WebTransactions-Anwendung bestehen und das Template geöffnet sein, in das Sie das WTBean einfügen wollen.

Mit dem Befehl **Einfügen/WTBean/wtcMVS** rufen Sie das WTBean zur Bearbeitung auf. WebLab generiert das Dialogfeld **Einfügen:wtcMVS**:



In diesem Dialogfeld können Sie die Parameter für das neue Kommunikationsobjekt bearbeiten. Die wichtigsten Parameter sind auf der ersten Registerkarte **MVS Verbindungsparameter** zusammengefasst. Die Pflichtparameter sind rot dargestellt.

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 geschützen 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 Druck-/Asynchron-Unterstützung nutzen

Wenn Sie die Druck-/Asynchron-Unterstützung einschalten, prüft WebTransactions automatisch in festlegbaren Abständen, ob asynchrone Nachrichten oder Druckinformationen vorliegen.

Falls asynchrone Nachrichten vorliegen, wird der im Browser dargestellte Host-Anwendungsschirm aktualisiert (automatisches Refresh) - falls Druckinformationen vorliegen, wird der Ausdruck veranlasst.

WebTransactions realisiert die Druck-/Asynchron-Unterstützung über das in Abschnitt „Funktionsweise der Druck-/Asynchron-Unterstützung“ auf Seite 148ff beschriebene Frameset. In der Konfiguration der eingesetzten Web-Browser muss JavaScript zugelassen sein.

8.1 Druck-/Asynchron-Unterstützung einschalten

Die Druck-/Asynchron-Unterstützung wird eingeschaltet, wenn das Attribut `WT_ASYNC` des verbindungsspezifischen Systemobjekts auf `"Yes"` gesetzt wird (Standardwert `"No"`).

Wenn Sie für den Start der WebTransactions-Sitzung das ausgelieferte Start-Template `wtstartMVS.htm` benutzen, klicken Sie bei der Option **Print/Asynchrone support** auf den Radio-Knopf **Start**.

Wenn Sie ein eigenes Start-Template verwenden, müssen Sie zum Einschalten der Druck-/Asynchron-Unterstützung lediglich eine Zuweisung einfügen, in der `WT_ASYNC` auf `"Yes"` gesetzt wird. Dies erfolgt in der Regel über das WTBean `wtcMVS`, indem Sie den Eintrag **Asynchrone Meldungen zulassen** auf **Yes** setzen.

Bei den heute üblichen Browsern, die das HTML-Tag `<iframe>` unterstützen, ist eine eingeschränkte Druckunterstützung auch ohne Einschalten des Attributes `WT_ASYNC` möglich: Nur bei jedem Dialogschritt, der vom Benutzer ausgelöst wurde, wird auf vorliegende Druckdaten geprüft. Asynchrone Druckdaten durch BYPASS-Printing können daher verzögert gedruckt werden.

8.2 Funktionsweise der Druck-/Asynchron-Unterstützung

Druck- und Asynchron-Unterstützung funktionieren nach dem gleichen Prinzip.

- Unterstützt der Browser das HTML-Tag `<iframe>` nicht und wird die Druck-/Asynchron-Unterstützung eingeschaltet - d.h. `WT_ASYNC` wird auf `"Yes"` gesetzt - so startet ein Skript im Template `wtBrowserFunctions.htm` das Frameset `wtframes.htm`.

  `wtframes` umfasst zwei Frames:

  – Ein Anwendungs-Frame, in dem wie gewohnt die Bildschirme der Host-Anwendung dargestellt werden.

  – Ein „unsichtbares“ Kontroll-Frame, das lediglich die Aufgabe hat, abzufragen, ob asynchrone Meldungen oder Druckinformationen vorliegen, und ggf. entsprechende Aktionen einzuleiten. In diesem Kontroll-Frame übernimmt das von WebTransactions bereitgestellte Template `wtasync.htm` die Kontrolle. Es ist der gleichen Sitzung zugeordnet wie das Anwendungs-Frame. In diesem Template wird WebTransactions im sogenannten asynchronen Modus verwendet, der den Dialogfluss der Host-Anwendung nicht beeinflusst.

  Da das Anwendungs-Frame das gesamte Browser-Fenster ausfüllt, das Kontroll-Frame an der Oberfläche jedoch nicht sichtbar ist, bleibt für die Anwender am Web-Browser verborgen, dass nach Einschalten der Druck-/Asynchron-Unterstützung die Sitzung mit einem Frameset fortgesetzt wird. Wenn Sie die Verbindung schließen und zum Start-Template zurückkehren, wird das Frameset automatisch entladen.

  Aufbau des Templates wtframes.htm

```
<html>
  <head>
    <title>WebTransactions</title>
  </head>
  <frameset rows="100%,*" border="0">
    <frame name="application" src="##WT_System.HREF_ASYNC#" />
    <frame name="control"
src="##WT_System.HREF_ASYNC#&WT_ASYNC_PAGE=wtasync" />
  </frameset>
</html>
```

- Unterstützt der Browser dagegen das HTML-Tag `<iframe>` und wird die Druck-/Asynchron-Unterstützung eingeschaltet (d.h. `WT_ASYNC` wird auf `"Yes"` gesetzt), wird nur inline, also mit `<iframe>`, das „unsichtbare“ Kontrollframe aufgerufen, in dem das von WebTransactions bereitgestellte Template `wtasync.htm` die Kontrolle übernimmt.

Kontroll-Frame: wtasync.htm

Im Kontroll-Frame sind die Verarbeitungsschritte für die Behandlung asynchroner Nachrichten und die Druckfunktionalität definiert. Das Kontroll-Frame muss ein `wtDataform`-Tag mit dem Attribut `asyncPage="wtasync"` enthalten. Dadurch wird angezeigt, dass dieses Template außerhalb der sequenziellen Dialogschrittfolge steht (siehe hierzu auch das WebTransactions-Handbuch „Konzepte und Funktionen“, Stichwort „nicht synchronisierter Dialog“).

Grundgerüst des Templates `wtasync.htm` :

```
<html>
  <body>
    <wtDataform name="async" asyncPage="wtasync">
    </wtDataform>
    <wtOnCreateScript>
    <!--
      host = WT_HOST.active;
      if ( host.WT_SYSTEM != null )
        host_system = host.WT_SYSTEM;  // Private System Objects
      else
        host_system = WT_SYSTEM;       // Public  System Objects
```

Verarbeitungsschritte für Druckanforderungen
... siehe Seite 158

Behandlung asynchroner Nachrichten
... siehe Seite 154

```
    //-->
    </wtOnCreateScript>
  </body>
</html>
```

Template wtBrowserFunctions.htm

Das Template `wtBrowserFunctions.htm` spielt auch eine wichtige Rolle bei der Druck-/Asynchron-Unterstützung: In `wtBrowserFunctions.htm` ist ein Skript integriert, das bei eingeschalteter Druck-/Asynchron-Unterstützung folgende Aufgaben übernimmt :

– Start des Framesets `wtframes`, falls dieses notwendig, aber noch nicht geladen ist.

– Zyklisches Abschicken (`Submit`) des Kontroll-Frames

Aufbau des Skripts in `wtBrowserFunctions.htm`:

```
<wtIf ( wtCurrentComm_system.WT_ASYNC == 'Yes')>
  <wtrem>support of asynchronous message and printing with browsers not
supporting <iframe>//////</wtrem>
  <script type="text/javascript">
  <!--
  function AutomaticRefresh()
  {
    if( parent.control && parent.control.document.forms[0] ) {
      parent.control.document.forms[0].submit();
    }
    setTimeout('AutomaticRefresh()', 5000);
  }
  if( !parent.control )
  {
    self.location.href =
'##WT_SYSTEM.HREF_ASYNC#&DUMMY=##WT_SYSTEM.FORMAT_STATE#' +
      '&WT_ASYNC_PAGE=wtframes';
  }
  else
  {
    ##(wtCurrentComm_system.HARDCOPY == 'Yes' || wtCurrentComm_system.BYPASS
== 'Yes') ?
       'AutomaticRefresh();' :
       'setTimeout("AutomaticRefresh()", 5000);'#

  }
  //-->
</script>
  <wtElse><wtIf ( (wtCurrentComm_system.HARDCOPY == 'Yes' ||
wtCurrentComm_system.BYPASS == 'Yes') && WT_BROWSER.acceptIframe )>
  <wtrem>just synchronous support of printing with browsers supporting
<iframe>//////////////////////////</wtrem>
```

```
  <iframe id="asyncFrame"
style="visibility:hidden;height:0;position:absolute;"
src="##WT_SYSTEM.HREF_ASYNC#&WT_ASYNC_PAGE=wtasync"></iframe>
  </wtIf></wtIf></wtIf>
```

Sie können die Zeit für das zyklische Abschicken des Kontroll-Frames verändern (Voreinstellung: 5000 ms).

8.3 Behandlung asynchroner Nachrichten

Asynchrone Nachrichten sind Nachrichten, die ans Terminal geschickt werden, ohne dass sie vom Anwender ausdrücklich angefordert worden wären - d.h. ohne dass der Anwender auf irgendeine Taste gedrückt oder auf ein Oberflächenelement geklickt hätte.

Wird auf eine Host-Anwendung, die mit asynchronen Nachrichten arbeitet, über das Web zugegriffen, verhält sich WebTransactions wie ein Terminal und nimmt asynchrone Nachrichten entgegen. Zu dem Zeitpunkt, an dem eine asynchrone Nachricht von WebTransactions empfangen wird, ist die aktuelle Seite aber bereits an den Browser geschickt worden. Die Modifikation durch die asynchrone Nachricht wird im Browser nur dann sichtbar, wenn auf Browserseite ein Refresh ausgelöst wird.

Ohne Druck-/Asynchron-Unterstützung ergibt sich folgendes Problem: Der Anwender am Web-Browser kann nicht feststellen, ob von WebTransactions eine asynchrone Nachricht empfangen wurde, und kann deshalb nur „auf Verdacht“ einen Refresh auslösen.

WebTransactions löst dieses Problem, indem automatisch ein Refresh ausgelöst wird, falls eine asynchrone Nachricht vorliegt - vorausgesetzt die Druck-/Asynchronunterstützung ist eingeschaltet (`WT_SYSTEM.WT_ASYNC="Yes"`). Für diese Funktionalität verwendet WebTransactions das in Abschnitt „Funktionsweise der Druck-/Asynchron-Unterstützung“ auf Seite 148 vorgestellte Frameset `wtframes` oder das HTML-Tag `<iframe>`.

Unabhängig vom Inhalt des `WT_ASYNC` wird `REFRESH_BY_ASYNC` auf `Yes` gesetzt, wenn eine asynchrone Nachricht empfangen wurde. Das Template kann diesen Wert auswerten und den Benutzer warnen, dass seine letzte Sendeaufforderung wegen einer asynchronen Nachricht ignoriert wurde, und ihn gleichzeitig auffordern, die Nachricht noch einmal zu senden.

Konzept

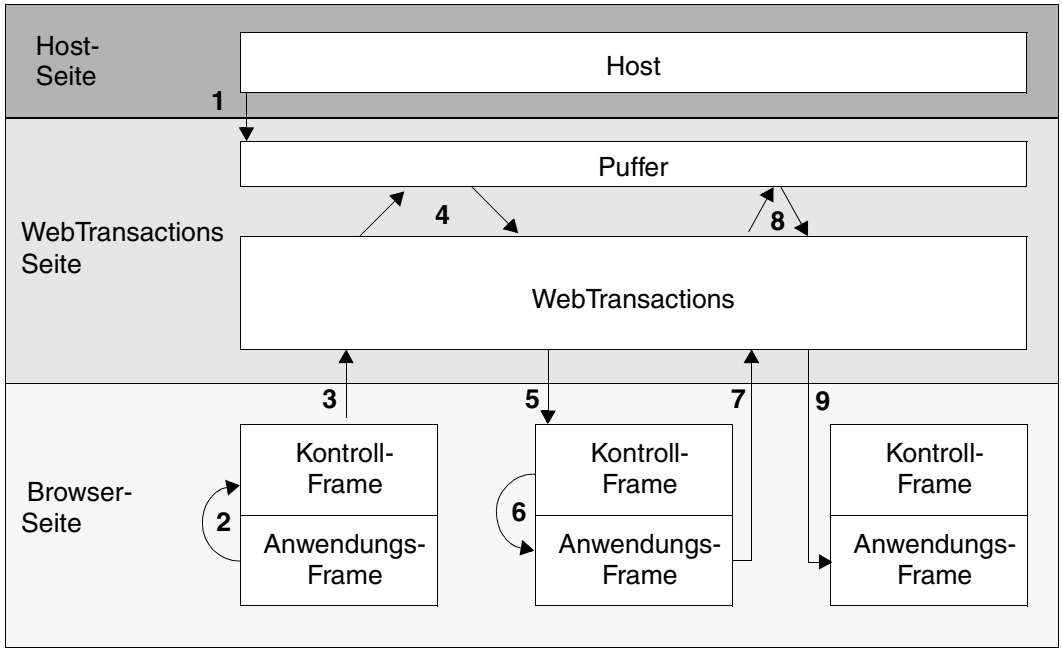


Bild 2: Behandlung asynchroner Nachrichten

1. Die Host-Anwendung sendet eine asynchrone Nachricht. Diese wird von WebTransactions noch nicht verarbeitet.
2. `wtBrowserFunctions.htm` startet, falls notwendig (wenn der Browser das HTML-Tag `<iframe>` nicht unterstützt), über das Template `wtframes.htm` ein zusätzliches unsichtbares Kontroll-Frame.
3. `wtBrowserFunctions.htm` startet zyklisch in dem unsichtbaren Bereich (`frame` oder `<iframe>`) einen asynchronen Aufruf an WebTransactions mit dem Template `wtasync.htm`.
4. `wtasync.htm` prüft, ob eine unverarbeitete Meldung ansteht (WebTransactions liefert bei Auswertung von `$MESSAGE:WAITING` dann `"Yes"`). Falls Ja, wird JavaScript-Code in die Antwort generiert, der einen Refresh auslöst (siehe Schritt 6).
5. Der Kontroll-Frame erhält die Antwort auf den asynchronen Aufruf.
6. Falls noch unverarbeitete Nachrichten anstehen, enthält der Kontroll-Frame nun eine JavaScript-Anweisung, die automatisch ein Refresh des Anwendungs-Frames auslöst (siehe Seite 154).
7. Der Anwendungs-Frame wird daraufhin an WebTransactions geschickt und von WebTransactions ausgewertet.

8. Das für den aktuellen Bildschirm zuständige Template führt, wie gewohnt, `send` und `receive` aus. Da aber `WT_KEY.KEY` den Wert "Refresh" enthält, wird keine Nachricht an den Host gesendet, sondern es werden nur alle noch ausstehenden Nachrichten vom Host verarbeitet.
9. Der durch die asynchrone Meldung veränderte Schirm wird an den Browser geschickt und im Anwendungs-Frame angezeigt.

Falls Sie individuelle Templates für einzelne Formate erstellen, müssen Sie den Bereich, in dem die asynchrone Nachricht dargestellt wird, berücksichtigen. Dies wird von WebLab nicht automatisch gemacht, falls zum Capture-Zeitpunkt der Bereich für asynchrone Nachrichten leer ist und die Option **statisch** gesetzt war.

In wtasync.htm definierte Behandlung asynchroner Nachrichten

In `wtasync.htm` sind für die Behandlung asynchroner Nachrichten folgende Verarbeitungsschritte definiert:

```
// Support of Asynchronous message ////////////////////////////////////
else if ( host.$MESSAGE.WAITING == "Yes" )
        document.write (
          '<script>' +
          '  if( parent.application && parent.application.document.forms[0]
)' +
          '    parent.application.wtSubmitKey( \'Refresh\' );' +
          '  else if( parent.document.forms[0] && parent.wtSubmitKey )' +
          '    parent.wtSubmitKey( \'Refresh\' );' +
          '</script>' );
      else if (host.$CONNECTION.ALIVE == 'No' )
        document.write (
          '<script>' +
          '  if( parent.application && parent.application.document.forms[0]
)' +
          '    parent.application.wtSubmitKey( \'Disconnect\' );' +
          '  else if( parent.document.forms[0] && parent.wtSubmitKey )' +
          '    parent.wtSubmitKey( \'Disconnect\' );' +
          '</script>' );
```

Zunächst wird über das Attribut `WAITING` des Host-Steuerobjekts `$MESSAGE` abgefragt, ob asynchrone Nachrichten vorliegen. Jedes Mal, wenn dieses Attribut abgefragt wird, prüft WebTransactions, ob sich eine asynchrone Nachricht im Puffer befindet. Falls ja, wird der Wert von `$MESSAGE.WAITING` intern auf `"Yes"` gesetzt.

Im Template `wtasync.htm` wird, falls die Auswertung von `$MESSAGE.WAITING` den Wert `"Yes"` ergibt, ein Refresh des Anwendungs-Frames ausgelöst. Dabei wird zunächst geprüft, ob der Anwendungs-Frame aktiv und die Methode `wtSubmitKey` vorhanden ist (die Kommunikation mit dem Host könnte ja durch ein Disconnect geschlossen worden sein).

Individuelle Anpassung des Templates wtasync.htm

Sie können das Template `wtasync.htm` leicht modifizieren und die Behandlung asynchroner Nachrichten verändern. Sie können z.B. den Anwender auch durch eine Meldungsbox darauf hinweisen, dass asynchrone Meldungen vorliegen, anstatt einen automatischen Refresh des Anwendungs-Frames auszulösen.

```
if (host.$MESSAGE.WAITING == "Yes")
 document.write ("<script> top.alert ('You received an asynchronous
    message. Please Refresh')</script>");
```

8.4 Druckunterstützung

In diesem Abschnitt wird erläutert, wie Sie mit WebTransactions Daten ausdrucken können.

Die Druckfunktionen, die WebTransactions unterstützt, unterscheiden sich, je nachdem was gedruckt werden soll:

WebTransactions ermöglicht das Ausdrucken folgender Informationen:

- Terminal-Hardcopy-Druck
  Beim Terminal-Hardcopy-Druck wird die alphanumerische Darstellung des Host-Anwendungsschirms gedruckt, wie er von einem Terminal oder einer Terminal-Emulation dargestellt würde.
- Ausdruck von Host-Daten
  Dabei werden Informationen ausgedruckt, die von der Host-Anwendung aufbereitet und gesendet wurden.
- Browserdarstellungs-Druck
  Dabei wird die im Web-Browser dargestellte Information ausgedruckt.

Für Terminal-Hardcopy-Druck und Browserdarstellungs-Druck kann man wiederum unterscheiden, ob der Ausdruck vom Anwender am Web-Browser angestoßen wird (anwendergetrieben), oder durch die Host-Anwendung ausgelöst wird (software-getrieben). Der Ausdruck von Host-Daten ist immer software-getrieben.

In Abhängigkeit von der gewählten Druckfunktion müssen Sie ggf. Ihren Web-Browser und/oder einen Druck-Server konfigurieren. Wie Sie den Web-Browser konfigurieren, ist in Abschnitt Abschnitt „Druck-Plugin WTAPrint“ auf Seite 171 beschrieben. Hinweise zur Konfiguration der Druck-Server finden Sie in der Dokumentation der Druck-Server-Produkte.

Normalerweise wird der Ausdruck unmittelbar an einen Drucker geschickt. Sie können bei der Konfiguration jedoch auch festlegen, dass er in eine Datei gelenkt wird. Diese Möglichkeit besteht für alle in diesem Abschnitt beschriebenen Druckfunktionen.

8.4.1 Terminal-Hardcopy-Druck

Beim Terminal-Hardcopy-Druck wird die alphanumerische Darstellung des Host-Anwendungsschirms gedruckt, wie er von einem Terminal oder einer Terminal-Emulation dargestellt würde. WebTransactions druckt dabei den gesamten Host-Anwendungsschirm aus.

Konzept

Grundlage für die Behandlung des Terminal-Hardcopy-Drucks ist das in Abschnitt „Funktionsweise der Druck-/Asynchron-Unterstützung“ auf Seite 148ff beschriebene Frameset `wtframes` aus Anwendungs-Frame und Kontroll-Frame.

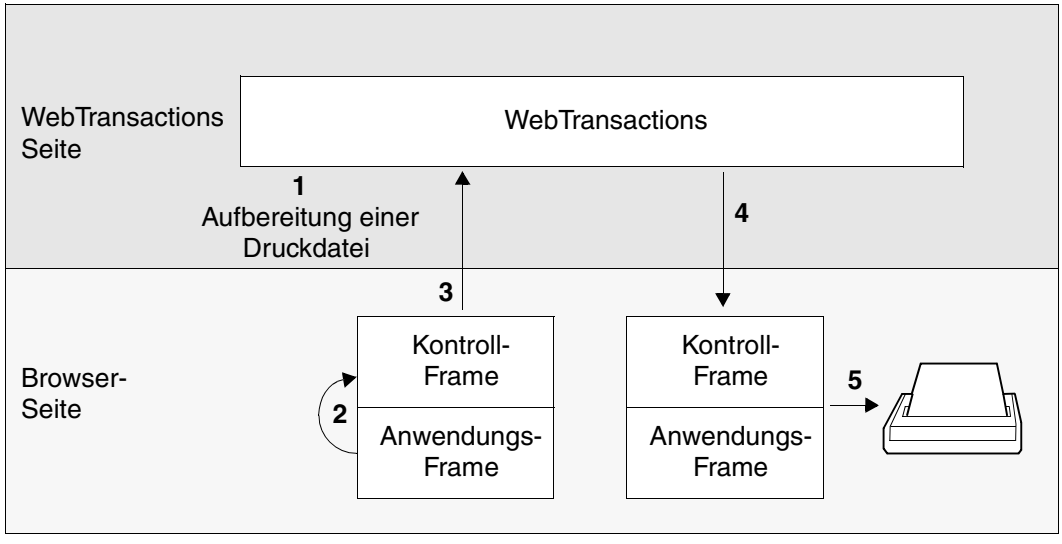


Bild 3: Terminal-Hardcopy-Druck

Wenn ein Terminal-Hardcopy-Ausdruck angefordert ist, stellt WebTransactions eine entsprechende Druckdatei zur Verfügung (Schritt 1). Getriggert durch das Skript in `wtBrowserFunctions.htm` wird der Kontroll-Frame automatisch abgeschickt (Schritt 2) wodurch bei WebTransactions angefragt wird, ob eine Druckdatei vorliegt (Schritt 3). Falls ja, wird die aufbereitete Datei an den Browser geschickt (Schritt 4). Wie Sie ein entsprechendes Druckprogramm konfigurieren und starten, ist in Abschnitt „Ausgelieferte Druckfunktionen (Browser-Plattform Windows)“ auf Seite 166 beschrieben.

Die von WebTransactions erzeugte Druckdatei ist eine reine Textdatei, in der die Zeilen durch die Kontroll-Zeichen <CR><LF> getrennt sind.

In wtasync.htm für Terminal-Hardcopy-Druck definierte Verarbeitungsschritte

```
// Support of hardcopy print ////////////////////////////////////////
if ( host_system.HARDCOPY == "Yes" )
      {
        host_system.HARDCOPY = "No";   // Reset flag before
        if (!host_system.WT_BROWSER_PRINT ||
host_system.WT_BROWSER_PRINT.toLowerCase() != 'yes')
          host.$MESSAGE.PRINTING;        // calling host.$MESSAGE.PRINTING
        else
        {
            WT_SYSTEM._printFile = WT_SYSTEM.BASEDIR + "/tmp/" +
WT_SYSTEM.SESSION + "/" + host.$MESSAGE.PRINTFILE_NAME;
          if (host_system.WT_BROWSER_PRINT_OPTIONS.MODE != "Automatic")
             document.writeln("<script>window.open
('"+WT_SYSTEM.HREF_ASYNC+"&WT_ASYNC_PAGE=wtc_bp_Print','_blank','toolbar=no,s
crollbars=yes,width=800,height=600')</script>");
          else
          {
            document.clear();
            forward ('wtc_bp_Print');
          }
        }
      }
```

Wenn eine Terminal-Hardcopy-Druckdatei zum Ausdruck ansteht, setzt WebTransactions das Attribut `HARDCOPY` des verbindungsspezifischen Systemobjekts auf `"Yes"`. Wird auf diese Weise eine Druckanforderung angezeigt, wird in `wtasync.htm` zunächst der Wert von `HARDCOPY` auf `"No"` zurückgesetzt.

Soll der Ausdruck mit der Browser-Druckfunktion durchgeführt werden, wird über `$MESSAGE.PRINTFILE_NAME` der Name der Druckdatei ermittelt und ein separates Template (`wtc_bp_print.htm`) für den Druckauftrag gestartet.

Andernfalls wird das Attribut `PRINTING` des Host-Steuerobjekts `$MESSAGE` ausgewertet, was WebTransactions dazu veranlasst, die zu druckende Datei an den Browser zu schicken. Dieser kann aufgrund des MIME-Typs die Daten als Druckdaten erkennen und an eine entsprechend konfigurierte Anwendung weitergeben (z.B. `WTAPrint.exe`).

Beachten Sie, dass der Wert von `HARDCOPY` **vor** Auswertung des Host-Steuerobjekt-Attributs `$MESSAGE.PRINTING` auf `"No"` zurückgesetzt werden muss, da mit der Auswertung von `$MESSAGE.PRINTING` die Interpretation des Templates abgebrochen und die HTML-Generierung unterdrückt wird.

Vom Anwender ausgelöster Terminal-Hardcopy-Druck

Der Anwender löst einen Terminal-Hardcopy-Druck aus, indem er auf den Knopf `Print` klickt. Dieser Knopf ist im Template `wtKeysMVS.htm` definiert, das vom Umsetzungstemplate `AutomaskMVS.htm` und den mit WebLab erstellten individuellen Templates inkludiert wird.

Bei einem Klick auf diesen Knopf wird die aktuelle Seite vom Browser an WebTransactions geschickt. WebTransactions interpretiert das Template wie gewohnt - jedoch mit einer Ausnahme: Bei der Ausführung des Aufrufs `send` erkennt WebTransactions, dass der `Print`-Knopf ausgelöst wurde und sendet keine Nachricht an den Host. Stattdessen wird eine Druckdatei generiert, die eine alphanumerische Darstellung des aktuellen Schirms enthält.

8.4.2 Ausdruck von Host-Daten

Beim Ausdruck von Host-Daten werden Informationen ausgedruckt, die von der Host-Anwendung aufbereitet und gesendet wurden, z.B. Host-Dateien.

8.4.2.1 Konzept

Beim Ausdruck von Host-Daten können die Druckdaten nicht aus der bestehenden Dialogverbindung zwischen WebTransactions und dem Host-Rechner gewonnen werden. In diesem Fall muss eine zusätzliche Netzverbindung zwischen jeder Drucker-Station auf dem Host-Rechner und einem Drucker oder einem Druck-Server aufgebaut werden. Die Drucker-Station auf dem Host-Rechner ist dabei als entfernter Drucker zu konfigurieren.

Der Druck-Server läuft auf dem WebTransactions-Server neben den WebTransactions-Anwendungen. Er nimmt die Druckdaten vom Host-Rechner entgegen und kann sie mit Hilfe einer geeigneten Konfiguration den WebTransactions-Sitzungen zur Verfügung stellen.

WebTransactions for MVS unterstützt das Protokoll LPD als Schnittstelle zwischen der Drucker-Station auf dem Host-Rechner und dem Druck-Server.

Beim Ausdruck von Host-Daten sind zwei Szenarien möglich:

- Der Druck wird auf einem zentralen Drucker ohne logischen Zusammenhang zum auslösenden Benutzer ausgegeben.

  Diese Variante wird durch den Einsatz von WebTransactions nicht verändert.

- Der Druck wird auf dezentrale Drucker geleitet. Die Auswahl eines Druckers steht im logischen Zusammenhang mit dem auslösenden Benutzer oder dem verwendeten Terminal.

  Um Druckaufträge auf einen bestimmten Drucker im Netz zu schicken, müssen Sie einen Druck-Server auf dem PC installieren, an dem der Drucker angeschlossen ist. Auf diese Weise sind Netzwerk-Drucker für Host-kontrollierte Druckfunktionen zugänglich.

  In diesem Fall soll auch beim Einsatz von WebTransactions der Druck am Client-Rechner, also über den Browser am Terminal ausgegeben werden. Da auf der Seite des Host-Rechners kein physikalischer Zusammenhang zwischen dem Terminal und dem ausgewählten Drucker besteht, wird der logische Zusammenhang durch eine entsprechende Konfiguration hergestellt.

  Dieses Verfahren ist im folgenden Text beschrieben.

8.4.2.2 Zuordnung eines Druckers zu einem Web-Browser-Client

Die Druckfunktion von WebTransactions ermöglicht es einem Benutzer, im Browser-Fenster auf seinem PC einen Druck anzustoßen und diesen Druck auf einem Drucker auszugeben, der an seinem PC angeschlossen ist.

Die Konfiguration muss daher:
– den Druckauftrag entgegennehmen,
– erkennen, zu welcher Dialog-Sitzung er gehört und
– genau diesem System die Druckdaten schicken.

Der Weg der Druckdaten, der sich bei einer solchen Konfiguration ergibt, ist in der folgenden Abbildung dargestellt:

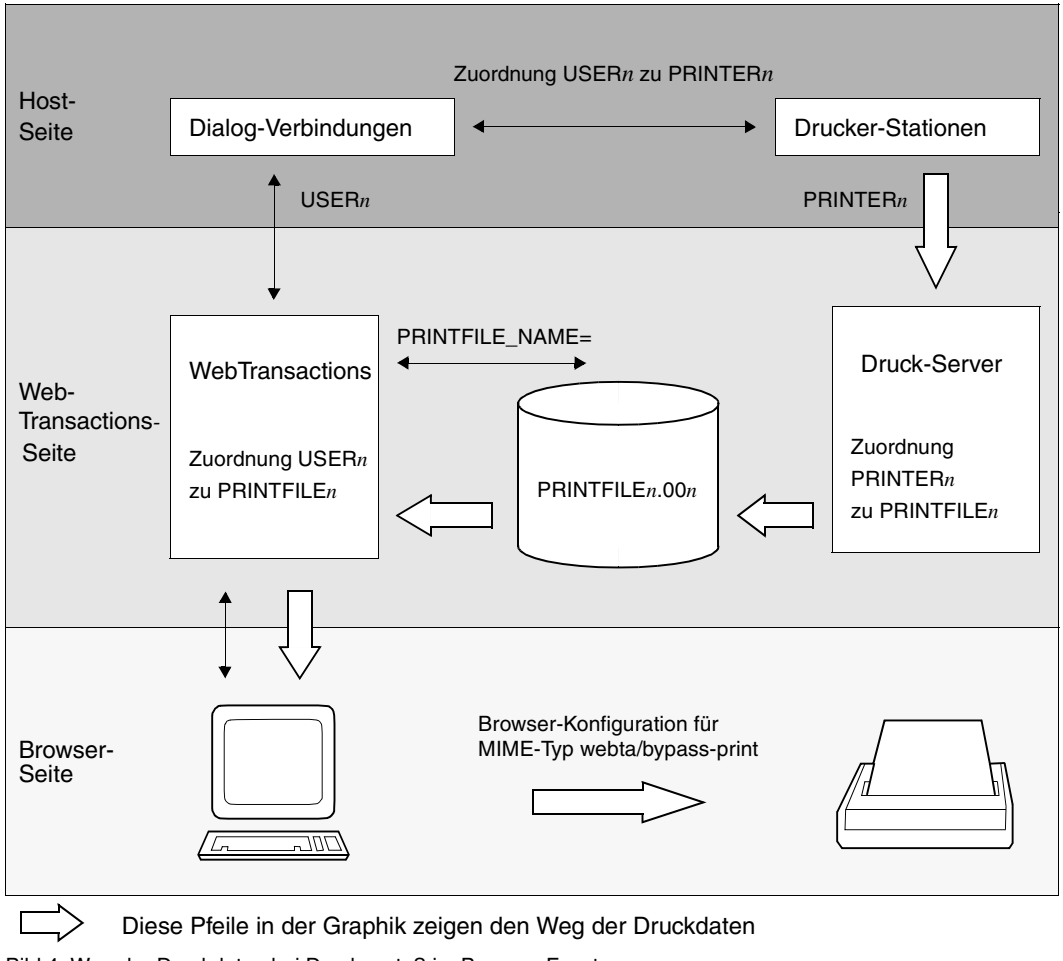


Bild 4: Weg der Druckdaten bei Druckanstoß im Browser-Fenster

Auf dem Host-Rechner werden die Druckaufträge über den Druckernamen adressiert. Hier muss sichergestellt sein, dass der „Adressraum“ ausreicht, um hinreichende Unterscheidungskriterien für die Zuordnung des Ausdrucks zu haben. Ggf. müssen Sie die Zuordnung in der Host-Anwendung und in der WebTransactions-Anwendung erweitern.

Um die Druckinformation auf dem Drucker auszugeben, der von Ihrem PC aus erreichbar ist, müssen die Druckdaten über WebTransactions an den Browser geleitet werden. Dazu muss die Konfiguration der Anwendung auf dem Host eine Namenszuordnung verwenden, die diese Korrelation widerspiegelt.

Zuordnung USER*n* zu PRINTER*n*

Diese Zuordnung auf dem Host kann auf folgende Weise erfolgt sein:

– statisch, z.B. durch eine Tabellenzuordnung
– generisch, z.B. auf Grund von Namenskonventionen
– dynamisch durch entsprechende Dialog-Eingaben an die Host-Anwendung

Zuordnung USER*n* zu PRINTFILE*n*

Diese Zuordnung in der WebTransactions-Sitzung kann auf folgende Weise erfolgen:

– statisch, z.B. durch eine Tabellenzuordnung
– generisch, z.B. auf Grund von Namenskonventionen

Beides kann mit Hilfe von normalen WTScript-Anweisungen oder mit Unterstützung des Userexits `ReplaceByConfigFile` konfiguriert werden.

Zuordnung PRINTER*n* zu PRINTFILE*n*

Diese Zuordnung auf dem Druck-Server muss statisch konfiguriert werden, sie kann nicht dynamisch durch die WebTransactions-Sitzung verändert werden. Bei Unix-Systemen muss die Zuordnung `PRINTER` zu `PRINTFILE_NAME` mit einer geeigneten Konfiguration des Spool-Systems getroffen werden.

8.4.3 Ausdruck von Host-Daten auf der WebTransactions-Plattform Windows

MVS-Anwendungen sind in der Lage, Druckinformationen gezielt zu spezifischen Druckern zu schicken. Das Konzept „LUNAME“, das auch für Terminals verwendet wird, erlaubt den MVS-Anwendungen, einzelne Drucker über eine logische Adresse anzusprechen.

Soll eine ähnliche Funktionalität auch für den Web-Zugriff auf MVS-Anwendungen realisiert werden, ergibt sich folgendes Problem: In MVS-Dialoganwendungen sind in vielen Fällen die Drucker mit den Terminals korreliert. Beim Web-Zugriff übernimmt WebTransactions die Rolle eines Terminals. Um zu gewährleisten, dass auch beim Web-Zugriff die Druckinformationen zu dem vom Anwender benutzten Drucker geleitet werden können, setzen Sie den Microsoft Host Integration Server ein (geeignet für Intranet und Internet, jedoch nur für Textdateien einsetzbar), siehe folgender Abschnitt.

Einsatz des Microsoft Host Integration Servers als Gateway

Diese Methode, Host-Daten auszudrucken, kann sowohl im Intranet als auch im Internet eingesetzt werden. Bei dieser Methode wird der Microsoft Host Integration Servers (kurz MHIS) als Gateway verwendet. Voraussetzung ist, dass es sich bei den Druckdateien um reine Textdateien handelt, die nur einfache Kontroll-Zeichen wie <CR> oder <LF> enthalten. Die Methode basiert auf einem reglementierten Einsatz von LUs (Logical Units): ein spezieller End-Anwender nutzt eine spezielle LU für die Anzeige und eine spezielle LU für den Ausdruck.

Wenn der MVS-Host einen Ausdruck an eine spezielle Drucker-LU sendet, kann vom MHIS die Druckinformation in einer Datei gespeichert werden. Die Dateinamen müssen dabei für die verschiedenen Drucker-LUs unterschiedlich sein. Werden mehrere Dateien für eine Drucker-LU erzeugt, verwendet MHIS ein Inkrement-Suffix (.001, .002, .003, ...).

Der WebTransactions-Rechner muss vom MHIS-Rechner aus über das Netz erreichbar sein. MHIS und WebTransactions können auch auf der gleichen Maschine laufen.

Das Attribut `PRINTFILE_NAME` des Systemobjekts müssen Sie mit dem Namen der Druckdateien versorgen, die dieser Sitzung zugeordnet sein sollen. Der Name ist mit absoluter Pfadangabe ohne Inkrement-Suffix anzugeben.

Wird im Template das Attribut `$MESSAGE.PRINTING` ausgewertet, werden die zum Druck anstehenden Host-Dateien, die dem in `PRINTFILE_NAME` gesetzten Namen entsprechen, an den Browser geschickt. Das Ausdruck-Verfahren ist das Gleiche wie beim Terminal-Hardcopy-Ausdruck (siehe Seite 157): Falls WebTransactions eine Druck-Datei mit dem in `PRINTFILE_NAME` gesetzten Namen findet, wird intern das Systemobjekt-Attribut `HARDCOPY` auf `Yes` gesezt.

Abbildung eines speziellen Benutzers auf eine spezielle MVS Anzeige-LU

Information	Erläuterung
sym. Name/IP-Adresse/Kennung	Zu Beginn der WebTransactions-Sitzung muss der Benutzer identifiziert werden: – In WebTransactions sind über Attribute des Systemobjekts der symbolische Name und die IP-Adresse des Rechners verfügbar, auf dem der Browser läuft. `CGI.REMOTE_HOST` enthält den symbolischen Namen und `CGI.REMOTE_ADDR` enthält die IP-Adresse. – Wenn der Browser-Rechner über einen Proxy-Server mit WebTransactions verbunden ist, liefern die Attribute `CGI.REMOTE_HOST` und `CGI.REMOTE_ADDR` jedoch den Namen und die IP-Adresse des Proxy-Servers. Integrieren Sie in diesem Fall ins Start-Template eine Abfrage, die den Benutzer zur Eingabe einer Kennung auffordert
sym. Name/IP-Adresse/Kennung ⟶ Session-Index (Multi Session API Server)	Im Start-Template können Sie einem speziellen symbolischen Namen/einer IP-Adresse oder einer Kennung einen speziellen Session-Index zuordnen.
Session-Index (Multi Session API Server) ⟶ MHIS Anzeige LU-Name	In der Konfiguration des Multi Session API Servers lässt sich einem speziellen Session-Index ein spezieller MHIS Anzeige LU-Name zuordnen.
MHIS Anzeige LU-Name ⟶ MVS Verbindung und LU#	Die MHIS-Konfiguration wiederum erlaubt es, dem MHIS Anzeige LU-Namen eine spezielle MVS Verbindung und eine MVS LU# zuzuordnen.
MVS Verbindung und LU# ⟶ MVS Anzeige LU-Name	Die MVS VTAM-Konfiguration schließlich erlaubt es, einer MVS Verbindung und MVS LU# einen speziellen MVS Anzeige LU-Namen zuzuordnen.

Abbildung eines speziellen Benutzers auf eine spezielle Drucker-LU

Information	Erläuterung
symb. Name/IP-Adresse/Kennung	siehe Tabelle für Anzeige-LU auf Seite 164.
symb. Name/IP-Adresse/Kennung ⟶ *printfile_name*	Im Start-Template können Sie einem speziellen symbolischen Namen/einer IP-Adresse oder einer Kennung einen speziellen *printfile_name* zuordnen.
printfile_name ⟶ MHIS Print Service Session Name	In der Konfiguration des MHIS lässt sich einem speziellen *printfile_name* ein spezieller MHIS Print Service Session Name zuordnen.
MHIS Print Service Session Name ⟶ MHIS Drucker LU-Name	Die MHIS-Konfiguration erlaubt es, dem MHIS Print Service Session Name einen speziellen MHIS Drucker LU-Namen zuzuordnen.
MHIS Drucker LU-Name ⟶ MVS Verbindung und LU#	Die MHIS-Konfiguration erlaubt es auch, dem MHIS Drucker LU-Name eine spezielle MVS Verbindung und MVS LU# zuzuordnen.
MVS Verbindung und LU# ⟶ MVS Drucker LU-Name	Die MVS VTAM-Konfiguration schließlich erlaubt es, einer MVS Verbindung und MVS LU# einen speziellen MVS Drucker LU-Namen zuzuordnen.

8.4.4 Browserdarstellungs-Druck

Diese Druckfunktion wird nicht vom Programm `WTAPrint.exe` bereitgestellt, das mit WebTransactions zur Verfügung gestellt wird und auf jedem Client installiert werden muss, sondern basiert auf der Druckfunktionalität der Web-Browser.

Die im Browser-Fenster dargestellten Informationen können Sie über die Druckfunktionen des Web-Browsers auslösen (Befehl *Print...* des File-Menüs).

8.4.5 Ausgelieferte Druckfunktionen (Browser-Plattform Windows)

WebTransactions enthält verschiedene Druckfunktionen, die Sie für den Terminal-Hardcopy-Druck (siehe Seite 157) und den Ausdruck von Host-Daten (siehe Seite 163 und Seite 160) einsetzen können:

– den Browser-Druck, der ohne das auf dem Client-PC installierte Druck-Plugin ablaufen kann.

– das Druck-Plugin `WTAPrint`

Das Template `wtKeysMVS.htm` generiert einen zusätzlichen Knopf **Print**, wenn es möglich ist, eine Druckfunktion auszulösen. Das ist der Fall, wenn eine der beiden folgenden Bedingungen erfüllt ist:

– Die Druck-/Asynchron-Unterstützung ist eingeschaltet (d.h. `WT_ASYNC` auf `"Yes"` gesetzt).

– Der Browser unterstützt das HTML-Tag `<iframe>`.

8.4.5.1 Browser-Druck

Die Funktion für den Browser-Druck ist als Parameter des Inline WTBean `wtcMVS` implementiert. (siehe Abschnitt „Neues MVS-Kommunikationsobjekt anlegen (wtcMVS)“ auf Seite 144 und WebTransactions-Handbuch „Konzepte und Funktionen“).

Im WTBean `wtcMVS` ist eine ActiveX-Komponente enthalten, die automatisch geladen und installiert wird. Abhängig von der Browsereinstellung muss der Anwender bestätigen, dass das signierte ActiveX-Control installiert werden darf.

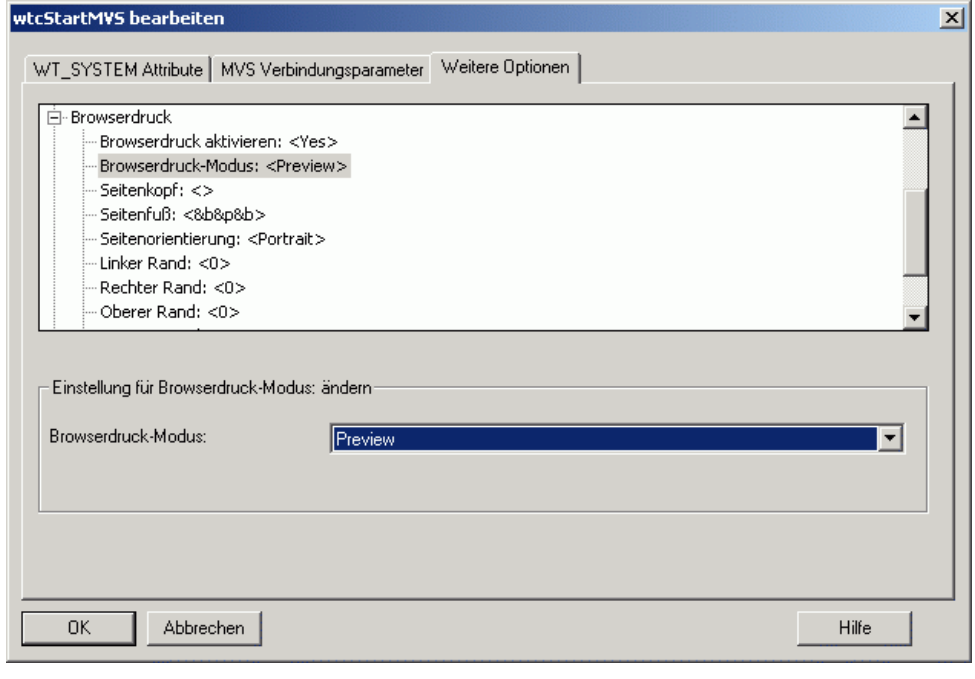
Beim Browser-Druck wird, im Gegensatz zum Browser-Darstellungsdruck, nicht das für den Benutzer sichtbare Formular gedruckt, sondern ein unsichtbares Dokument. Dieses wurde speziell zum Drucken vorbereitet, um das gleiche Ergebnis zu erzielen, wie die Druckfunktionen eines Terminals.

Einstellungen im Start-Template

Um den Browser-Druck zu nutzen, müssen Sie im Start-Template die folgenden Parameter setzen:

- Wählen Sie z.B. im Dialogfeld **wtcStartMVS bearbeiten** die Registerkarte **Weitere Optionen**.
- Klicken Sie unter **Globale Parameter** auf den Eintrag **Asynchrone Meldungen zulassen.**
- Markieren Sie die Option **Asynchrone Meldungen zulassen**. Der Wert des Eintrags wird von **No** auf **Yes** geändert.

  Wenn es ausreicht, dass nur bei jedem Dialogschritt geprüft wird, ob Druckdaten vorliegen, und wenn alle benutzten Browser das HTML Tag `iframe` unterstützen, können Sie auch auf die Unterstützung asynchroner Nachrichten verzichten.
- Klicken Sie dann unter **Browserdruck** auf den Eintrag **Browserdruck aktivieren**.
- Markieren Sie die Option **Browserdruck aktivieren**. Der Wert des Eintrags wird von **No** auf **Yes** geändert.



Alle weiteren Optionen unter **Browserdruck** werden nur ausgewertet, wenn Sie **Browserdruck aktivieren** auf **Yes** gesetzt haben:

Browserdruck-Modus

Automatic	Der Ausdruck erfolgt sofort auf dem Standard-Drucker.
Preview	Es wird zuerst eine Druck-Vorschau geöffnet.

Die folgenden Einstellungen werden nur bei Verwendung des Internet Explorer ausgewertet. Bei allen anderen Browsern müssen die Seiteneinstellungen über das zugehörige Dialogfeld des Browsers vorgenommen werden.

Seitenkopf
Text für den Seitenkopf (siehe „Variablen in Kopf- und Fußzeile“ unten)

Seitenfuß
Text für den Seitenfuß (siehe „Variablen in Kopf- und Fußzeile“ unten)

Seitenorientierung

Portrait	Hochformat
Landscape	Querformat

Linker Rand, **Rechter Rand**, **Oberer Rand**, **Unterer Rand**
Randbreiten in mm

Variablen in Kopf- und Fußzeile

Sie können in der Kopf- und in der Fußzeile die folgenden variablen Angaben verwenden:

&w	Fenstertitel
&u	Adresse der Seite (URL)
&d	Datum in Kurzformat (z.B. 15.04.2003)
&D	Datum in Langformat (z.B. Dienstag, 15. April 2003)
&t	Uhrzeit (wie in der Systemsteuerung eingestellt)
&T	Uhrzeit im 24-Stunden-Format
&p	Aktuelle Seitennummer
&P	Gesamtseitenzahl des Dokuments
&&	Kaufmännisches Und (&)
&b	Der Text nach diesem Kürzel wird zentriert
&b&b	Der Text nach dem ersten `&b` wird zentriert, der Text nach dem zweiten `&b` wird rechtsbündig ausgegeben

Beispiel

Der folgende Eintrag in der Kopfzeile

```
Seite &p von &P&bWebTransactions Browser Print&b&d
```

liefert im Ausdruck in der obersten Zeile

Seite *nummer* WebTransactions Browser Print *aktuelles Datum*

Konvertierungsdatei wtc_bp_print.cnv

Mit Hilfe der Konvertierungsdatei `wtc_bp_print.cnv` können Sie Druck-Steuersequenzen in HTML-Code umsetzen. Die Datei liegt im Basisverzeichnis der zugehörigen WebTransactions-Anwendung. Die Konvertierung der Druck-Steuersequenzen erfolgt beim Einlesen der Druckdatei im Template `wtc_bp_Print.htm`. In diesem Template wird die Konvertierungsdatei als zweiter (optionaler) Parameter beim Aufruf des Userexits `Getfile` angegeben (siehe WebTransactions-Handbuch „Template-Sprache“).

Im Auslieferungszustand enthält die Konvertierungsdatei folgende Einträge:

```
* Printer conversion file
00=20
7F="?"
"<"="&lt;"
">"="&gt;"
```

Jede Zeile in der Konvertierungsdatei muss die Form *suchtext=ersetzungstext* haben.

`suchtext` und `ersetzungstext` sind die alte und neue Zeichenfolge. Die Einträge sind wie folgt aufgebaut:

– Sie bestehen entweder aus zweistelligen Hexadezimal-Codes oder einer Zeichenkette, die in Doppelhochkommas eingeschlossen sein muss.

– Innerhalb der Zeichenkette können wieder zweistellige Hexadezimal-Codes verwendet werden, denen ein Gegenschrägstrich (\) vorangestellt sein muss.

  Beispiel

  ```
  "\195H"="</pre><pre class=pb>"
  ```

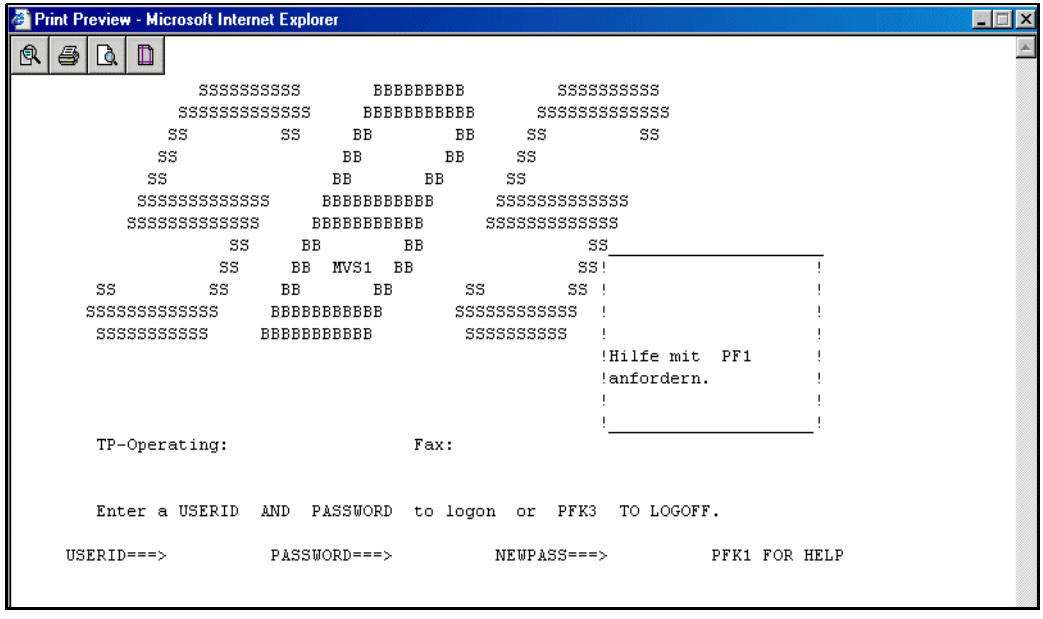
  ersetzt die Steuerzeichenfolge durch einen vorher im Template definierten CSS-Seitenumbruch.

– Gegenschrägstriche und Doppelhochkommas in Zeichenketten müssen mit Gegenschrägstrichen entwertet werden (\\ und \").

Die Konvertierungs-Regeln werden von oben nach unten angewendet. Zusätzliche Änderungen können im Template `wtc_bp_Print.htm` ergänzt werden (z.B. mit der Methode `replace` der Klasse `String`).

Druck-Vorschau

Ist im Start-Template die Option **Browserdruck-Modus** auf **Preview** gesetzt, erhalten Sie vor dem Start des Druckvorgangs eine Seitenansicht.



Die Schaltfläche wird nur angezeigt, wenn die Sitzung mit WebLab gestartet wurde. Sie öffnet ein weiteres Browser-Fenster, das alle Steuer- und Binärzeichen innerhalb des Ausdrucks markiert:

- Zeichen < 0x20: hexadezimal in rot
- Zeichen > 0x7F (8-Bit-Zeichen): zusätzlich hexadezimal blau in Klammern
- Leerzeichen: gelb hinterlegte Punkte
- Am Ende der Anzeige werden gefundene Sequenzen von Steuerzeichen ausgegeben. Diese Angaben können für die Konvertierungsdatei verwendet werden (siehe Abschnitt „Konvertierungsdatei wtc_bp_print.cnv“ auf Seite 169)

druckt bei Verwendung des Internet Explorer das Dokument auf dem eingestellten Standard-Drucker aus. Bei allen anderen Browsern wird das Dialogfeld **Drucken** aufgerufen.

öffnet bzw. schließt die Druck-Vorschau des Browsers. Die Schaltfläche wird nur im Internet Explorer angezeigt.

öffnet die Einstellungen für die Seiteneigenschaften. Die Schaltfläche wird nur im Internet Explorer angezeigt.

8.4.5.2 Druck-Plugin WTAPrint

Das Druck-Plugin `WTAPrint` steht auf dem WebTransactions-Server zum Download zur Verfügung. Verwenden Sie hierfür die Seite `Wtdownload.htm`. Diese Seite wurde bei der Installation von WebTransactions im Dokumentenverzeichnis des Web-Servers unter `webtav75` angelegt. Nach dem Download kann `WTAPrint` auf dem Browser-Rechner installiert werden.

Wenn Sie `WTAPrint` mit dem ebenfalls zum Download bereitstehenden Installations-Programm `WTAPrint2000` installieren, werden die benötigten Registrierungseinträge für den Internet Explorer erzeugt. `WTAPrint` ist für alle Web-Browser einsetzbar, die auf Windows-Plattformen laufen. Mit Hilfe von `WTAPrint` können Sie auch Druckdaten mit Austauschregeln verändern. Dieses Vorgehen kann für die Unterstützung bestimmter Drucker notwendig sein.

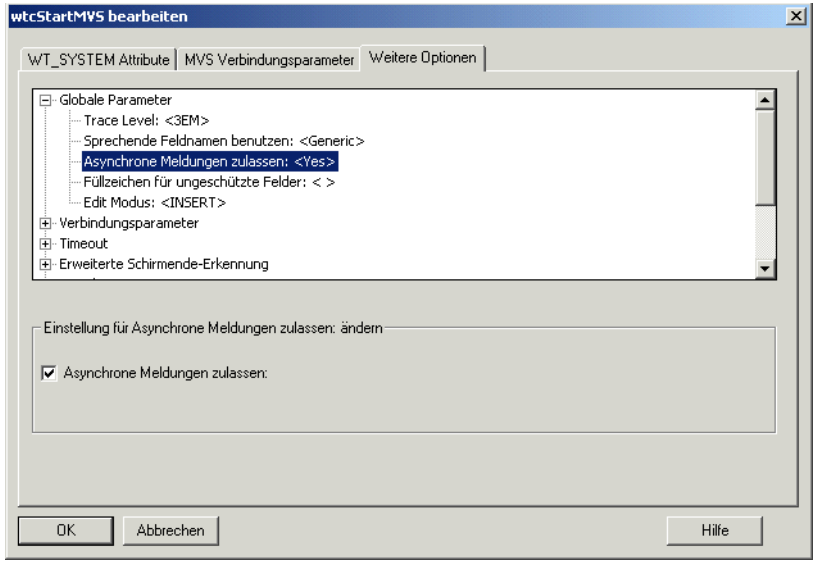
Die Aktivierung des Druck-Plugin `WTAPrint` ist als Parameter des Inline WTBean `wtcMVS` implementiert (siehe Abschnitt „Neues MVS-Kommunikationsobjekt anlegen (wtcMVS)“ auf Seite 144 und WebTransactions-Handbuch „Konzepte und Funktionen“).

Einstellungen im Start-Template

Um den den Plugin-Druck zu nutzen, müssen Sie im Start-Template die folgenden Parameter setzen:

- Wählen Sie z.B. im Dialogfeld **wtcStartMVS bearbeiten** die Registerkarte **Weitere Optionen**.
- Klicken Sie unter **Globale Parameter** auf den Eintrag **Asynchrone Meldungen zulassen**.
- Markieren Sie die Option **Asynchrone Meldungen zulassen**. Der Wert des Eintrags wird von **No** auf **Yes** geändert.

  Wenn es ausreicht, dass bei jedem Dialogschritt geprüft wird, ob Druckdaten vorliegen, und wenn alle benutzten Browser das HTML Tag `<iframe>` unterstützen, können Sie auch auf die Unterstützung asynchroner Nachrichten verzichten.



► Überprüfen Sie den Eintrag **Browserdruck aktivieren** unter **Browserdruck**. Der Eintrag muss in diesem Fall den Wert **No** haben.

Konfiguration der Web-Browser

Die WebTransactions-Druckfunktionen schicken Druckdateien mit einem speziellen MIME-Typ an den Browser:

`webta/hardcopy-print`
: für Terminal-Hardcopy-Druck

`webta/hardcopy-print`
: Bypass-Druck für Host-Daten-Druck

Dieser MIME-Typ muss in der Konfiguration der Browser definiert werden. Für die gängigen Web-Browser (Internet Explorer ab V4.0 mit allen Windows Versionen, Netscape Navigator ab V6, Mozilla) gehen Sie hierfür wie folgt vor:

► Installieren Sie `wtaprint2000.exe`.

Die oben aufgelisteten Web-Browser erkennen dann `wtaprint.exe` als Anwendung zum Drucken.

Wenn Ihr Browser ein Standard-Suffix für jeden MIME-Type vorsieht, können Sie beispielsweise das Suffix `.whp` angeben.

Zuordnung des Druck-Programms

Sie müssen diesen MIME-Typen ein geeignetes Druck-Programm zuordnen. Sie verwenden dazu folgende Syntax:

```
WTAPrint.exe  "%1" method target [filename] [option-file]
```

`"%1"`
: Der Browser ersetzt `"%1"` automatisch durch den Namen der auszudruckenden Datei.

method
: entsprechend dem zugeordneten MIME-Typ.

`HARDCOPY`
: die Druckdaten werden als reiner Text interpretiert und mittels Windows Spool Technik für den gewählten Drucker aufbereitet.

`BYPASS`
: die Druckdaten sind bereits für den gewählten Drucker aufbereitet und werden an den Drucker gesendet.

target
: Mögliche Werte für target: 0,1,2 oder 3

- 0 Druckerauswahl (bei jedem Ausdruck öffnet sich eine Dialogbox zur Auswahl eines Druckers).
- 1 Der Standard-Drucker wird verwendet.
- 2 Der Druck wird als Datei gesichert (bei jedem Ausdruck öffnet sich eine Dialogbox zur Angabe eines Pfades).
- 3 Der Druck wird an die Datei angefügt, deren voller Pfad mit *filename* angegeben ist.

filename
: voller Pfadname der Datei, an die der Ausdruck angefügt werden soll (nur zulässig, falls für *target* der Wert 3 gesetzt ist).

option-file

voller Pfadname der Datei, in der Optionen für die Konvertierung der Druckdatei definiert sind.

```
[Options]
[ConversionFile=conversion-file]
[MaxLengthLine=maxlength-line]
[MaxLengthPage=maxlength-page]
[CharSet=charset]
[Font=font]
```

Beachten Sie, dass Sie hier bei `[options]` die eckigen Klammern mit eingeben müssen. Alle weiteren Klammern in dieser Syntax kennzeichnen, wie gewohnt, optionale Angaben.

conversion-file

vollständiger-Pfadname-der-Konvertierungsdatei. Mit Hilfe dieser Datei können Sie Zeichenfolgen umsetzen, siehe Beispiel 3. Die Datei muss im selben Verzeichnis liegen wie `WTAPrint`. Sie darf maximal 1023 Zeilen enthalten; jede Zeile darf maximal 255 Zeichen lang sein und muss die Form *hex-alt*=*hex-neu* haben. *hex-alt* und *hex-neu* sind die alte und neue Zeichenfolge in vollständiger hexadezimaler Notation.

maxlength-line

Maximale Zeilenlänge. Wenn Sie diesen Wert nicht angeben, wird die Druckdatei nach der längsten Zeile durchsucht.
Kleinster errechneter Wert ist dabei 80.

Die Option `MaxLengthLine` wird nur ausgewertet, wenn für *method* der Wert `HARDCOPY` vorliegt (siehe Seite 173).

maxlength-page

Maximale Seitenlänge. Wenn Sie diesen Wert nicht angeben, wird die Druckdatei nach der längsten Seite durchsucht.
Kleinster errechneter Wert ist dabei 60.

Die Option `MaxLengthPage` wird nur ausgewertet, wenn für *method* der Wert `HARDCOPY` vorliegt (siehe Seite 173).

charset

Identifier für den Zeichensatz an der Windows Schnittstelle `CreateFont`. Voreinstellung: `1` (`DEFAULT_CHARSET`).

Für die verschiedenen Zeichensätze sind die folgenden Identifier anzugeben:

```
ANSI_CHARSET                0
ARABIC_CHARSET            178
CHINESEBIG5_CHARSET       136
DEFAULT_CHARSET             1
EASTEUROPE_CHARSET        238
GB2312_CHARSET            134
GREEK_CHARSET             161
HANGEUL_CHARSET           129
HANGUL_CHARSET            129
HEBREW_CHARSET            177
JOHAB_CHARSET             130
OEM_CHARSET               255
RUSSIAN_CHARSET           204
SHIFTJIS_CHARSET          128
SYMBOL_CHARSET              2
THAI_CHARSET              222
TURKISH_CHARSET           162
VIETNAMESE_CHARSET        163
```

font Name eines Font, der in dem Windows-System verfügbar sein muss. Wenn Sie diesen Wert nicht angeben, wird der erste Font verwendet, der die Bedingungen der Optionen `MaxLengthLine`, `MaxLengthPage` und `CharSet` erfüllt. `MaxLengthLine` und `MaxLengthPage` werden dabei auf eine Font-Größe umgerechnet.

*Beispiel***e**

- Konfiguration für Terminal-Hardcopy-Druck:

1. MIME-Typ:
   `webta/hardcopy-print`

   Zugeordnetes Programm:
   `WTAPrint.exe "%1" HARDCOPY 0`

2. Konfiguration für Bypass-Druck mit Konvertierung:

   MIME-Typ:
   `webta/hardcopy-print`

Zugeordnetes Programm:

```
WTAPrint.exe "%1" HARDCOPY 0 "C:\webta\conversion.ini"
```

Options-Datei `conversion.ini`:

```
[Options]
MaxLengthLine=90
ConversionFile=C:\webta\webtaprint.cnv
```

Konvertierungs-Datei `webtaprint.cnv`:

```
65=8080
66=8182
6567=8A
```

Für diese Datei gelten folgende Regeln:

- Es muss eine vollständige, hexadezimale 2-Ziffern-Darstellung verwendet werden. Leerzeichen oder andere nicht-hexadezimale Zeichen dürfen nicht enthalten sein.
- Es wird immer die längste passende Zeichenfolge in der Konvertierungsdatei genommen. Wenn z.B. die Eingabedatei die Folge `6567` enthält, dann wird dies in diesem Beispiel auf `8A` umgesetzt (nicht auf `808067`).
- Falls eine Zeichenfolge (*xxxx*=...) mehrfach auftritt, dann wird die letzte dieser Zeichenfolgen zum Ersetzen verwendet.

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

(**C**ommon **G**ateway **I**nterface)
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 Kundenname und Kundennummer Eigenschaften eines Objekts „Kunde“ sein. Diese Eigenschaften können innerhalb des Programms gesetzt, abgefragt und verändert werden.

EJB

(**E**nterprise **J**ava**B**ean)
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.

EHLLAPI

(**E**nhanced **H**igh **L**evel **L**angugage **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

(**F**ormat **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.

Formatbeschreibungsquellen

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

(**H**yper**t**ext **M**arkup **L**anguage)
Siehe *->Hypertext Markup Language*

HTTP

(**H**ypertext **T**ransfer **P**rotocol)
Protokoll zur Übertragung von *->HTML*-Seiten und Daten.

HTTPS

(**H**ypertext **T**ransfer **P**rotocol **S**ecure)
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 ->*Auswertungsoperator* 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 ->*Formatbeschreibungsquellen* 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

(**L**ightweight **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

(**M**aster-**T**emplate-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

(**U**niversal **T**ransaction **M**onitor)
Transaktionsmonitor von Fujitsu Technlogy 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 ->*Benutzerkennung* 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 Anwort 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

(**T**ransport **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* eingegeben 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

(**U**niform **R**esource **L**ocator)
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

(**W**eb **S**ervices **D**escription **L**anguage)
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

(**W**eb**T**ransactions **M**arkup **L**anguage)
Auszeichnungs- und Programmiersprache für WebTransactions ->*Templates*. WTML besteht aus ->*WTML-Tags*, die ->*HTML* erweitern, und der server-seitigen Programmiersprache ->*WTScript*, die z.B. den Datenaustauch mit ->*Host-Anwendungen* ermöglicht. WTML wird von WebTransactions und nicht vom ->*Browser* ausgeführt (serverside scripting).

WTML-Tag

(**W**eb**T**ransactions **M**arkup **L**anguage-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 (W3C-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**yper**t**ext **M**arkup **L**anguage
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
HTTPS	**H**yper**t**ext **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-Tag
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

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**X**tensible **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 openUTM-Anwendungen über UPIC
Benutzerhandbuch

WebTransactions
Anschluss an OSD-Anwendungen
Benutzerhandbuch

WebTransactions
Zugriff auf dynamische Web-Inhalte
Benutzerhandbuch

WebTransactions
Web-Frontend für Web-Services
Benutzerhandbuch

Sonstige Literatur

EDITION PRINT for Windows NT (MATERNA GmbH)
Installation and Configuration
Benutzerhandbuch

Zielgruppe
Das Handbuch richtet sich an alle, die einen Druck-Server der Produktlinie EDITION PRINT einsetzen.
Inhalt
Das Handbuch beschreibt, wie Sie EDITION PRINT-Produkte installieren, konfigurieren, starten und einsetzen.

Stichwörter

X

Z