

WebTransactions V7.5

Anschluss an OSD-Anwendungen

Kritik... Anregungen... Korrekturen...

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

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

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

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

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

Copyright und Handelsmarken

Copyright © Fujitsu Technology Solutions GmbH 2010.

Alle Rechte vorbehalten.

Liefermöglichkeiten und technische Änderungen vorbehalten.

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

Inhalt

1	Einleitung	7
1.1	Charakterisierung des Produkts	7
1.2	Architektur von WebTransactions for OSD	9
1.3	Dokumentation zu WebTransactions	11
1.4	Konzept und Zielgruppe dieses Handbuchs	14
1.5	Neue Funktionen	15
1.6	Darstellungsmittel	16
2	WebTransactions installieren	17
2.1	Installation	17
2.1.1	Windows	18
2.1.1.1	Installation über die Bedienoberfläche	18
2.1.1.2	Bedienerlose Installation	19
2.1.2	Solaris	21
2.1.3	Linux	22
2.1.4	BS2000/OSD	23
2.1.5	Installation von WebLab	23
2.2	Lizenzierung	24
3	Beispielsitzung	25
3.1	WebTransactions-Server verwalten	25
3.1.1	Browser einstellen	26
3.1.2	Administrationsprogramm starten	27
3.1.3	Lizenzen eingeben	28
3.1.4	Benutzer anlegen	31
3.1.5	Pool anlegen	32

3.1.6	Pool dem Benutzer zuweisen	34
3.2	Host-Anwendung an das WWW anbinden	35
3.2.1	Projekt anlegen	35
3.2.1.1	Basisverzeichnis anlegen	36
3.2.1.2	Automask-Template generieren	39
3.2.2	Projekt speichern	40
3.2.3	Sitzung starten	42
3.3	Globale Änderung der Darstellung	47
3.4	Formatspezifische Änderung der Darstellung	52
3.4.1	Formatspezifisches Template mit dem Capture-Verfahren generieren	53
3.4.2	Formatspezifisches Template bearbeiten	56
3.5	WebTransactions-Anwendung starten	61
3.5.1	Start-Template erzeugen	61
3.5.2	Sitzung starten mit WebLab	64
3.5.3	Alternative Möglichkeiten zum Start einer WebTransactions-Anwendung	65
4	Basisverzeichnis anlegen	67
4.1	Basisverzeichnis anlegen mit WebLab	67
4.2	Struktur eines Basisverzeichnisses	69
5	WebTransactions konfigurieren und starten	71
5.1	Zeichensatz konfigurieren	71
5.1.1	7-Bit-ASCII-Zeichensatz	72
5.1.2	8-Bit-Zeichensatz	73
5.1.3	Unicode-Zeichensatz	74
5.2	WebTransactions-Anwendung starten	75
6	Host-Anwendung ohne Bearbeitung integrieren	77
6.1	Master-Templates OSD.wmt und OSD_Pocket.wmt	78
6.2	Template AutomaskOSD.htm	80
6.2.1	Varianten von AutomaskOSD.htm erstellen (mit WebLab)	80
6.2.2	Aufbau von AutomaskOSD.htm	82
6.3	Template wtKeysOSD.htm	90

6.4	Template wtBrowserFunctions.htm	91
6.5	Template wtPKEYS.htm	91
7	Templates bearbeiten	93
7.1	Capture-Verfahren mit WebLab	94
7.1.1	Erkennungskriterien einzeln erfassen	94
7.1.2	Erkennungskriterien gesammelt erfassen	97
7.1.3	Erkennungskriterium bearbeiten	103
7.1.4	Capture-Datenbank bearbeiten	103
7.2	Individuelle Templates für Popup-Boxen	104
7.2.1	Ohne spezielle Popup-Behandlung: Identifikationsprobleme	105
7.2.2	Templates für Popups generieren	106
7.3	Sprechende Namen verwenden	110
8	Kommunikation steuern	115
8.1	Systemobjekt-Attribute	115
8.1.1	Übersicht	116
8.1.2	Zusammenspiel von Systemobjekt-Attributen und Methoden	133
8.2	Host-Objekte	135
8.2.1	Host-Datenobjekte	135
8.2.2	Host-Steuerobjekte	140
8.3	Unterstützung von Terminal-Funktionen durch den Browser	144
8.3.1	Unterstützte Terminal-Funktionen	144
8.3.2	Zusammenspiel von Host-Steuerobjekt WT_KEY.KEY und Template wtKeysOSD.htm	147
8.3.3	Zuordnung der Tasten in wtKeysOSD.js	149
8.3.4	Zusammenspiel von wtCommonBrowserFunctions.js und wt<browser>BrowserFunctions.js	154
8.3.5	Verwendung des Objekts WT_BROWSER	158
8.3.6	Unterstützung der P-Tasten	160
8.3.6.1	Definition in WTML	160
8.3.6.2	Template wtPKEYS.htm	161
8.3.6.3	PKEYS verwalten	166
8.3.6.4	PKEYs speichern	168
8.4	Start-Templates für OSD	169
8.4.1	OSD-spezifisches Start-Template des Start-Template-Sets (wtstartOSD.htm)	170
8.4.2	WTBean wtcStartOSD.wtc zur Generierung eines Start-Template	174

8.5	Neues OSD-Kommunikationsobjekt anlegen (wtcOSD)	176
9	Druck-/Asynchron-Unterstützung nutzen	179
9.1	Druck-/Asynchron-Unterstützung einschalten	179
9.2	Funktionsweise der Druck-/Asynchron-Unterstützung	180
9.3	Behandlung asynchroner Nachrichten	185
9.4	Druckunterstützung	189
9.4.1	Terminal-Hardcopy-Druck	190
9.4.2	Ausdruck von Host-Daten	193
9.4.3	Browserdarstellungs-Druck	198
9.4.4	Ausgelieferte Druckfunktionen (Browser-Plattform Windows)	199
9.4.4.1	Browser-Druck	199
9.4.4.2	Druck-Plugin WTAPrint	204
9.4.5	Konfiguration der Druckserver für Windows	210
	Fachwörter	213
	Abkürzungen	233
	Literatur	235
	Stichwörter	237

1 Einleitung

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

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

1.1 Charakterisierung des Produkts

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

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

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

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

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

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

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

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

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

1.2 Architektur von WebTransactions for OSD

Folgende Abbildung zeigt die Architektur von WebTransactions for OSD:

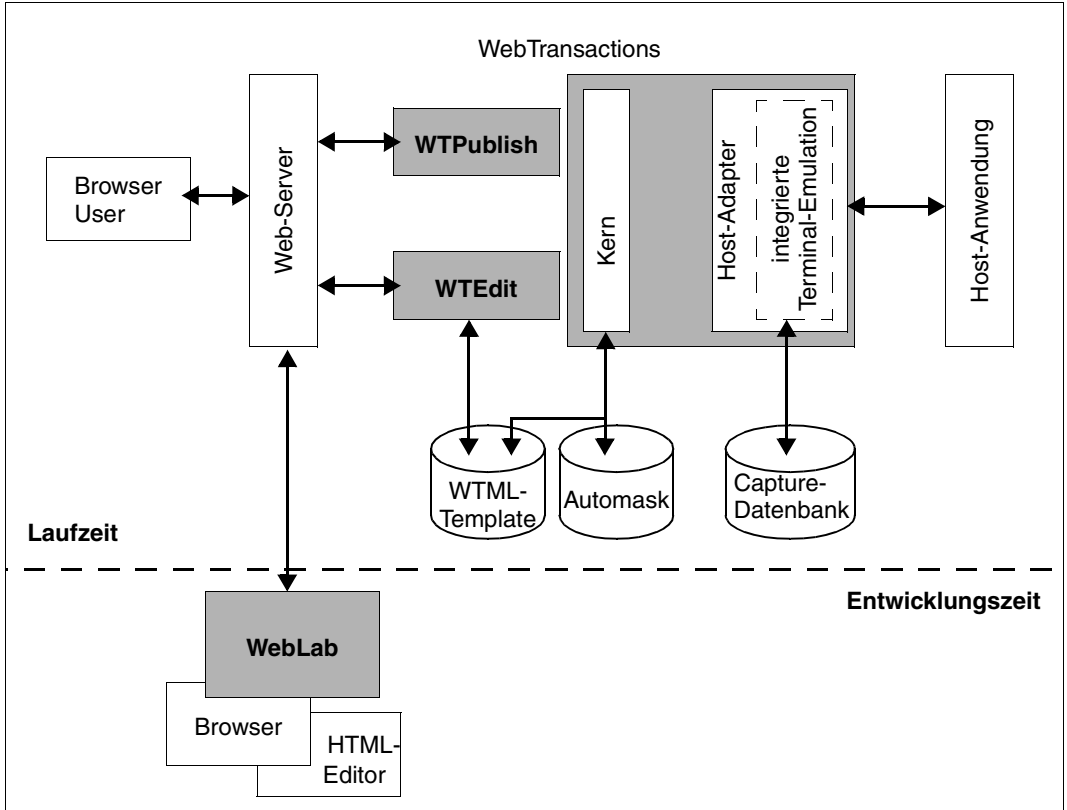


Bild 1: Architektur von WebTransactions for OSD

Host-Adapter mit integrierter Terminal-Emulation

WebTransactions for OSD benutzt zur Lauf- und zur Entwicklungszeit eine 9750-Emulation, die in den Host-Adapter integriert ist und die Kommunikation zwischen dem Kern von WebTransactions und der Host-Anwendung abwickelt.

WebLab

WebLab ist die Entwicklungsumgebung von WebTransactions, mit der Sie alle Schritte von der Anbindung einer Host-Anwendung, der Erzeugung und Nachbearbeitung der format-spezifischen Templates bis zum Test der Anwendung ausführen können.

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 Formatbeschreibungdatei (FLD-Datei). Die formatspezifischen Templates können Sie mit WebLab nachbearbeiten.

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 Erkennungskriterium gefunden, setzt WebTransactions das Bildschirmformat dynamisch um: mit dem Standard-Automask-Template.

Unicode-Unterstützung

Das Terminal-Protokoll 9750, das von WebTransactions for OSD unterstützt wird, wird durch die Unicode-Unterstützung in BS2000/OSD seinerseits Unicode-fähig. Sie müssen die Unicode-Unterstützung für 9750 explizit durch Setzen des entsprechenden Terminal-typs einschalten, andernfalls verhält sich WebTransactions for OSD wie in der Vorgängerversion (siehe Systemobjekt-Attribut `TERMINAL_TYPE`, [Seite 131](#)).

WebTransactions for OSD erzeugt Daten in UTF-8-Codierung, die zum Browser und zurück durchgereicht werden. Welche Regeln Sie dazu für die Templates einhalten müssen, entnehmen Sie dem [Abschnitt „Unicode-Zeichensatz“](#) auf [Seite 74](#).

1.3 Dokumentation zu WebTransactions

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

- Ein einführendes Handbuch, das für alle Liefereinheiten gilt:

Konzepte und Funktionen

Das Handbuch beschreibt alle zentralen Konzepte von WebTransactions:

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

Template-Sprache

Nach einem Überblick über WTML finden Sie

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

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

Anschluss an openUTM-Anwendungen über UPIC

Anschluss an MVS-Anwendungen

Alle Handbücher zu den Host-Adapttern 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 OSD-Dialoganwendungen an das Web anschließen wollen.

Die einzelnen Kapitel beschreiben die hierfür notwendigen Schritte. Wenn Sie noch nicht mit WebTransactions for OSD 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 OSD-spezifischen Informationen.

Gültigkeit der Beschreibung

WebTransactions for OSD ist auf den Systemplattformen BS2000/OSD, 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

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

Art der Neuerung	Beschreibung
Neuer Wert für das Systemobjekt-Attribut <code>HOST_CHARSET</code> : 9763-UNICODE	Seite 123
Neuer Wert für das Systemobjekt-Attribut <code>TERMINAL_TYPE</code> : UTF-8	Seite 131
Änderung der Systemobjekt-Attribute: – <code>END_MARK</code> – <code>LZE_CHAR</code>	Seite 119 Seite 125

1.6 Darstellungsmittel

Diese Dokumentation verwendet die folgenden Darstellungsmittel:

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

2 WebTransactions installieren

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



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

2.1 Installation

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

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



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

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

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

2.1.1 Windows

Für Windows steht WebTransactions nach dem Download als Windows Installationspaket (msi-Datei) `WebTransactionsOSD75.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 `WebTransactionsOSD75.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.d11` für die Java-Integration (optional)

Wenn Sie die Werte angegeben haben, wird die Installation gestartet und die gewünschten Komponenten werden 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\WebTransactionsOSD75.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\WebTransactionsOSD75.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. `webtransOSDV75.tar.gz`). Diese Datei müssen Sie zuerst dekomprimieren, mit dem Kommando:

```
gunzip -d webtransOSDV75.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 webtransOSDV75.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 these settings OK ? [y]
```

Bestätigen Sie Ihre Angaben, um die Installation abzuschließen.

2.1.4 BS2000/OSD

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

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

2.1.5 Installation von WebLab

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

Installationspaket auf den Entwicklungsrechner übertragen

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

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

WebLab auf Windows installieren

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

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

2.2 Lizenzierung

Nach der Installation müssen Sie noch die Anzahl der vorhandenen Lizenzen und den maschinenspezifischen Aktivierungsschlüssel konfigurieren. Dazu rufen Sie das Administrationsprogramm 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 OSD auf einem Rechner installiert haben (siehe hierzu auch [Kapitel „WebTransactions installieren“ auf Seite 17](#)), müssen Sie zuerst die Voraussetzungen schaffen, um mit WebTransactions und WebLab zu arbeiten. Diese Voraussetzungen schaffen Sie mit dem Administrationsprogramm, das im WebTransactions-Handbuch „Konzepte und Funktionen“ beschrieben ist.

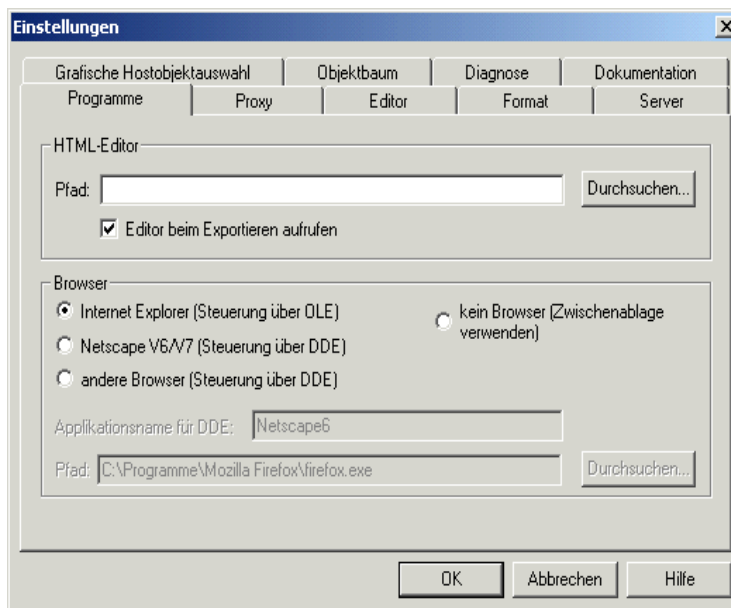
Als ersten Schritt stellen Sie in WebLab den Browser ein, den WebLab für die Bedienung der WebTransactions-Anwendung verwenden soll. Die Arbeit mit dem Administrationsprogramm gliedert sich dann in folgende Schritte:

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

3.1.1 Browser einstellen

Bevor Sie zu arbeiten beginnen, 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.



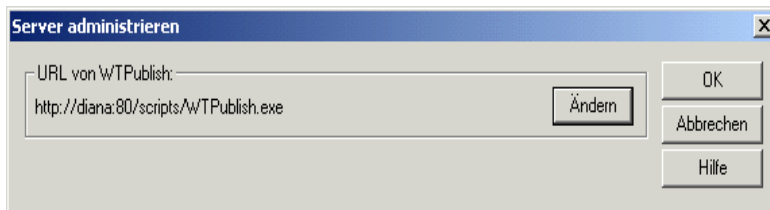
- ▶ Wählen Sie im unteren Bereich **Browser** den Browser aus, der auf Ihrem Rechner installiert ist, und wie er von WebLab verwendet werden soll.
- ▶ Bestätigen Sie Ihre Einstellung mit **OK**.

3.1.2 Administrationsprogramm starten

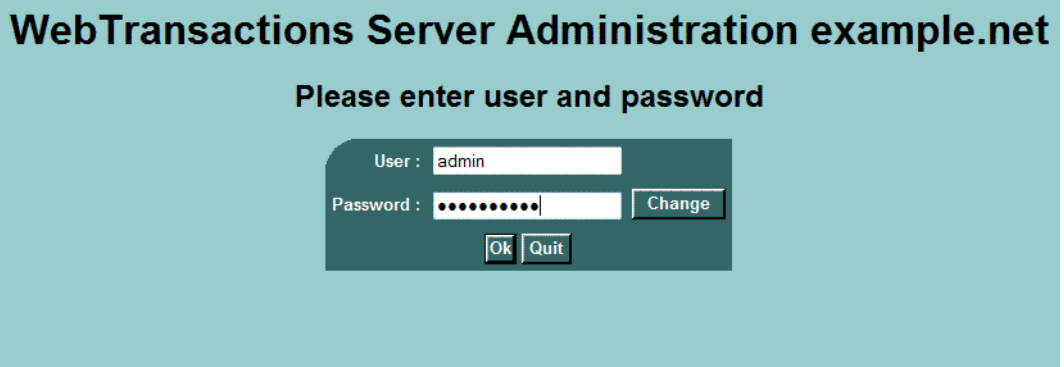
- ▶ Wählen Sie den Befehl **Administrieren/Server**, um zuerst das Administrationsprogramm zu starten. Das Dialogfeld **Server administrieren** wird am Bildschirm eingeblendet.
- ▶ Klicken Sie unter **URL von WTPublish** auf **Ändern**. Das Dialogfeld **URL von WTPublish** wird angezeigt.
- ▶ Wählen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

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

Licenses

Users

Pools


Applications

Sessions

Tools

Clusters

Print



Registration

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

Licenses

Info

Server-Id: **693619ac**

Key: **Invalid**

Licensed Servers: **1**

Licensed concurrent users: **3**

On demand user licenses: **0**

License logging:

Action

Enter new license

Servers:

Licenses:

On Demand Licenses:

Days:

Key:

Set

Save Load Refresh Exit

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

Die Registrierungsseite wird geöffnet:

The screenshot shows a registration form with the following fields and options:

- Type of License:** Radio buttons for Single Server and Cluster.
- Server-Id:** Text input field containing "fe7b19ac".
- Number of licenses:** Text input field.
- On demand licenses:** Check box, currently unchecked.
- Email address:** Text input field with the note "Key will be sent to this address!" below it.
- Platform:** Dropdown menu with "Please select..." selected.
- Installed adapters:** Text input field.
- Reason for registration:** Dropdown menu with "Change of registered licenses" selected.
- Name:** Text input field.
- Company:** Text input field.
- Department:** Text input field.
- Street:** Text input field.
- PC/City:** Two adjacent text input fields.
- Country:** Text input field.
- Sales Representative:** Text input field.
- Remarks:** Large text area with a vertical scrollbar.
- Request Key:** Button at the bottom of the form.

- ▶ 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 Clients (concurrent sessions) 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 WebLab-Benutzer einzutragen. Das Fenster **Users** wird im Browser angezeigt.

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

Users

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

Click on user name to access the user's properties

Save Load Refresh Exit

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

3.1.5 Pool anlegen

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

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

FUJITSU

Licenses

Users

Pools

Applications

Sessions

Tools

Clusters

Print

Pools

Directory	Virtual Path	Comment	Action
c:/inetpub/wwwroot/webtav75	webta		Remove Edit
d:/basedirs/manual	manual		Remove Edit
d:/basedirs/v71	basev71		Remove Edit
d:/basedirs/v75	basedirs/v75		Remove Edit
d:/home/cvs/v75/webta	regtest		Remove Edit

[Check here to create it](#)

Click on directory to access the pool's properties

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

Powered by **WebTransactions**

- 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, also ohne WebTransactions dafür aufrufen zu müssen, auf Dateien (z.B. Bilder, Aufrufseite, ...) von WebTransactions-Anwendungen in diesem Verzeichnis zugegriffen wird.



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

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

In Pools, die auf diese Weise angelegt wurden, können nun mit WebLab die Basisverzeichnisse angelegt werden. Allerdings kann bisher noch kein Benutzer außer `admin` 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.

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

Pool d:/basedirs/manual

Users (who can create applications here) Action

No users allowed yet

frank	
webtaman	Add

Click on user name to access the user's properties

Save Load Refresh Exit

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

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

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

3.2 Host-Anwendung an das WWW anbinden

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

1. Projekt anlegen
 - Basisverzeichnis anlegen
 - Automask generieren
2. Projekt speichern
3. Sitzung starten

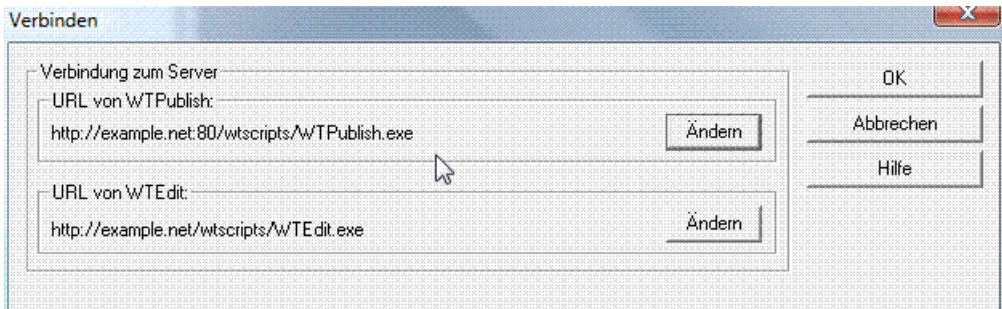
3.2.1 Projekt anlegen

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.

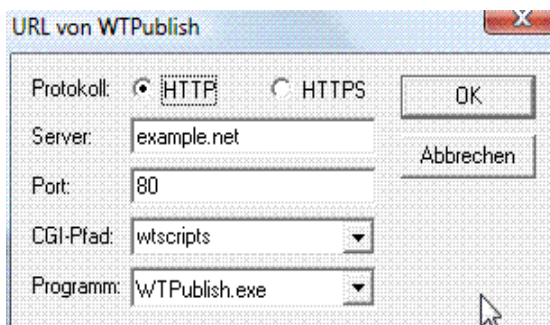
3.2.1.1 Basisverzeichnis anlegen

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

Server *example.net*

Port *80*

CGI-Pfad *wtscripts*

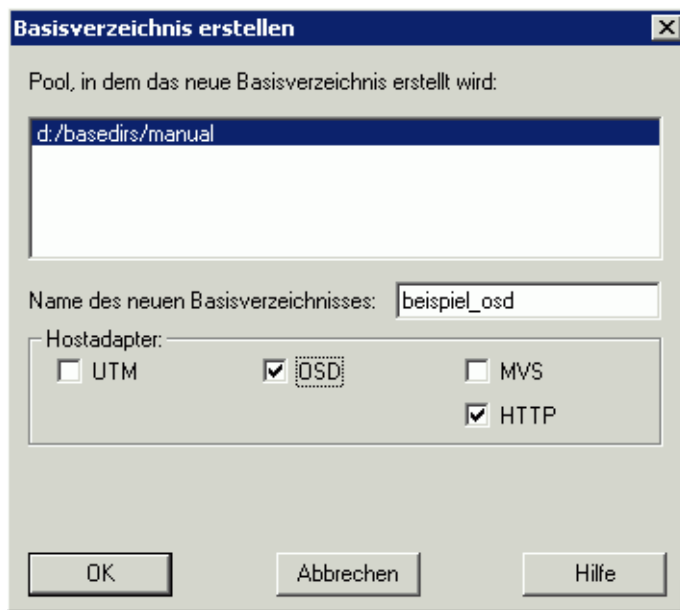
Programm *WTEdit.exe*

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

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



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

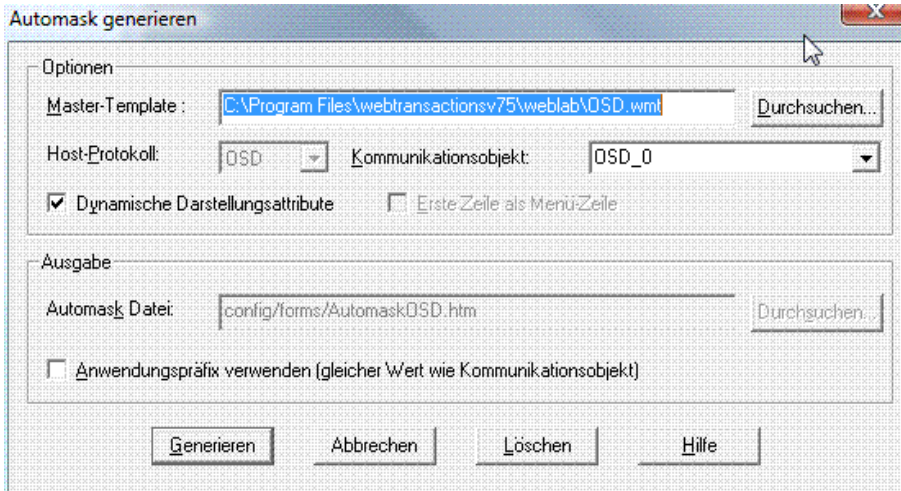


In der oberen Liste dieses Dialogfeldes werden die Pools angezeigt, unter denen der angemeldete Benutzer auf dem WebTransactions-Server Basisverzeichnisse angelegen darf.

- ▶ Wählen Sie mit einem Klick aus der Liste den Pool, den Sie im [Abschnitt „Pool anlegen“ auf Seite 32](#) angelegt haben.
- ▶ Geben Sie dann im Eingabefeld **Name des neuen Basisverzeichnisses** einen Namen an, hier *beispiel_osd*.
- ▶ Wählen Sie dann den passenden Host-Adapter, über den WebTransactions mit der Host-Anwendung kommuniziert, hier *OSD*. 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 *OSD_0*. Wenn Sie hier nichts angeben, wird die Voreinstellung *OSD_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.

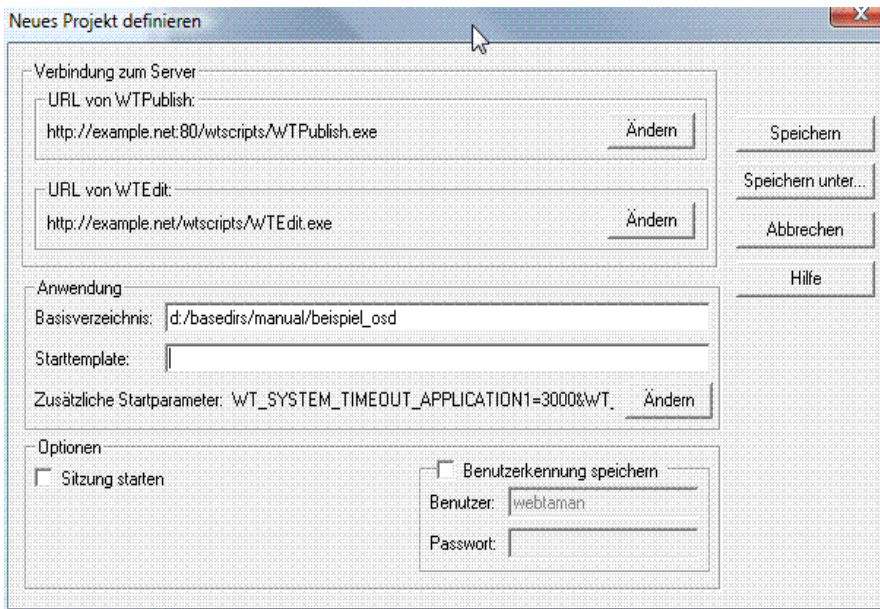
- ▶ In diesem Beispiel akzeptieren Sie alle weiteren Voreinstellungen und bestätigen das Dialogfeld mit **Generieren**. Das Dialogfeld wird geschlossen und Basisverzeichnis und Automask werden mit den angegebenen Werten am WebTransactions-Server generiert. Im WebLab-Hauptfenster wird unterhalb des Arbeitsbereichs 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 61](#)).

3.2.2 Projekt speichern

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



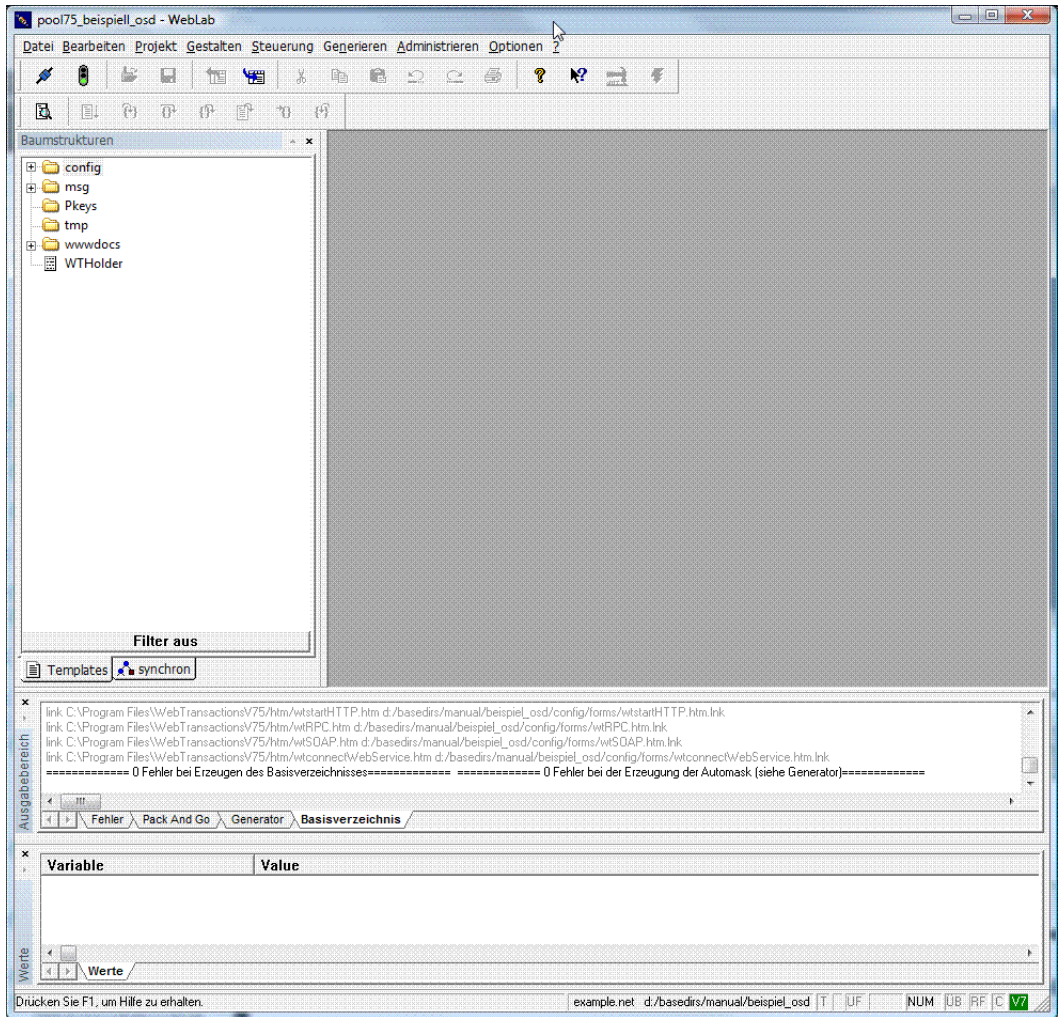
- ▶ In diesem Beispiel akzeptieren Sie alle Voreinstellungen und speichern das Projekt mit **Speichern unter...**

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

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

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

Anschließend sind Sie mit Ihrem neuen Basisverzeichnis verbunden. Eine Übersicht über die angelegten Verzeichnisse finden Sie im [Abschnitt „Struktur eines Basisverzeichnisses“ auf Seite 69](#).



3.2.3 Sitzung starten



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

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





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

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

 		main menu	
		[WebTransactions: test environment]	
status	base directory:	d:/base_dirs/manual/beispiel_osd	
workflow	PROTOCOL:	HTTP ▾	
	private WT_SYSTEM:	<input checked="" type="checkbox"/>	
	name of new communication object:	OSD_0	
	create new communication:	create	
	select HANDLE:	HTTP_0 ▾	
	continue communication:	continue	
	connect webService	webService	
	terminate session	quit	
system parameters	STYLE:		
	LANGUAGE:		
	DEFAULT_FORMAT:	wtstart	
	TIMEOUT_APPLICATION:	120 (2 minutes) ▾	
	TIMEOUT_USER:	600 (10 minutes) ▾	
	COMMUNICATION_INTERFACE_VERSION:	3.0 or higher ▾	
	WTML_VERSION:	3.0 or higher ▾	
	SEARCH_HOST_OBJECTS:	<input type="checkbox"/>	

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

- ▶ Wählen Sie in der **PROTOCOL**-Auswahlliste den Eintrag **OSD**.
- ▶ Geben Sie den Namen des Kommunikationsobjekts an, hier **OSD_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 OSD-spezifische Start-Template `wtstartOSD.htm` übernimmt die Kontrolle und zeigt das nächste Formular.

 		OSD communication
status	communication object:	WT_HOST.OSD_0
	connection:	down
workflow	destination:	main menu ▾ <input type="button" value="go to"/>
	access host:	<input type="button" value="open"/> <input type="button" value="run"/>
	parameters:	<input type="button" value="update"/> <input type="button" value="reset"/>
connection parameters	HOST_NAME:	<input type="text"/>
	SYM_DEST:	<input type="text"/>
	APPLICATION_PREFIX:	<input type="checkbox"/>
	STATION_NAME:	<input type="text"/>
	TERMINAL_TYPE:	9750 ▾
	PORT_NUMBER:	102
	FORMAT:	wtstartOSD
	AUTOMASK:	AutomaskOSD
	DISCONNECT:	wtstartOSD
	CAPTURE_FILE:	config/capture.sdb
	TRACE_LEVEL:	<input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input checked="" type="checkbox"/> E <input checked="" type="checkbox"/> M <input type="checkbox"/> L
	LOGFILE:	<input type="text"/>
	TRACEFILE:	<input type="text"/>
	NATIONAL_VARIANT:	International ▾
	FIRST_IO_TIMEOUT:	60 ▾
	MULTIPLE_IO_TIMEOUT:	2 ▾
	END_MARK:	~
Print/Asynchronous support:	<input type="radio"/> Start <input checked="" type="radio"/> Stop	
DISPLAY_EURO:	<input type="checkbox"/>	
FIELD_NAMES:	<input type="checkbox"/>	

Mit dem OSD-spezifischen Start-Template 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.

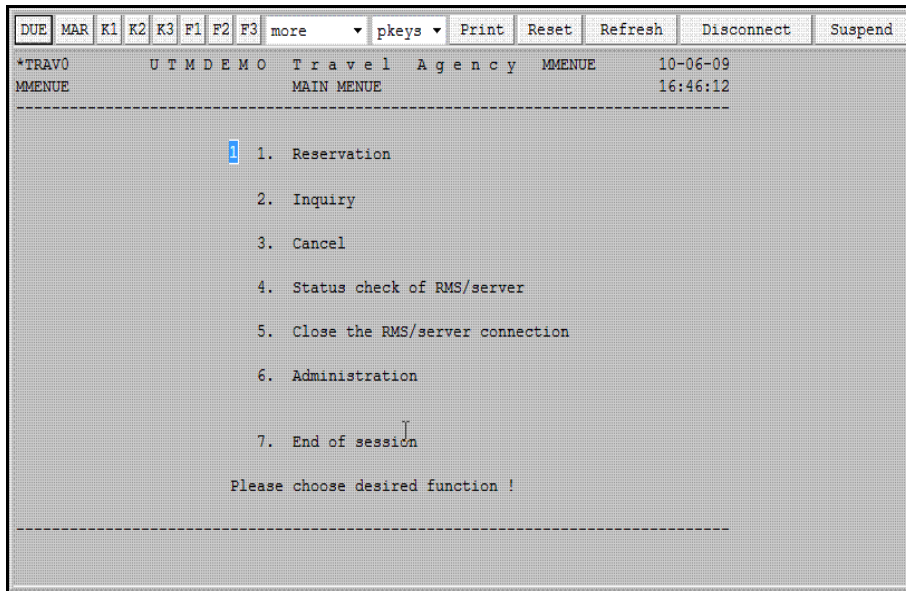
- ▶ Tragen Sie dazu bei **HOST_NAME** den Namen des Rechners ein, auf dem die Host-Anwendung läuft.
- ▶ Tragen Sie bei **SYM_DEST** den symbolischen Namen der Host-Anwendung ein, hier *Travel*. Der symbolische Name ist der Name, unter dem die Anwendung am Host-Rechner bekannt ist.
- ▶ Falls die Host-Anwendung einen spezifischen Terminal-Namen erwartet, können Sie diesen im Feld `STATION_NAME` eintragen.
- ▶ Aktivieren Sie den Knopf **run**, um die Verbindung zur Host-Anwendung zu öffnen. Es wird der erste Bildschirm der OSD-Anwendung mit `AutomaskOSD.htm` ausgegeben.

Das `AutomaskOSD.htm`-Template stellt Ihnen zur Kommunikation mit der OSD-Anwendung eine Knopfleiste zur Verfügung, welche die Sondertasten des 9750-Terminals abbildet (siehe [Abschnitt „Template AutomaskOSD.htm“ auf Seite 80](#)).

Wenn Sie jetzt die Verbindung zum Host beenden wollen, wählen Sie den Knopf **Disconnect**. Es wird wieder auf das Template `wtstartOSD` verzweigt (siehe auch [Abschnitt „OSD-spezifisches Start-Template des Start-Template-Sets \(wtstartOSD.htm\)“ auf Seite 170](#)). 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 *Travel* einzuloggen.
- ▶ Starten Sie die Travel-Anwendung mit **DUE**. Das nächste Format der Travel-Anwendung wird im Browser angezeigt.



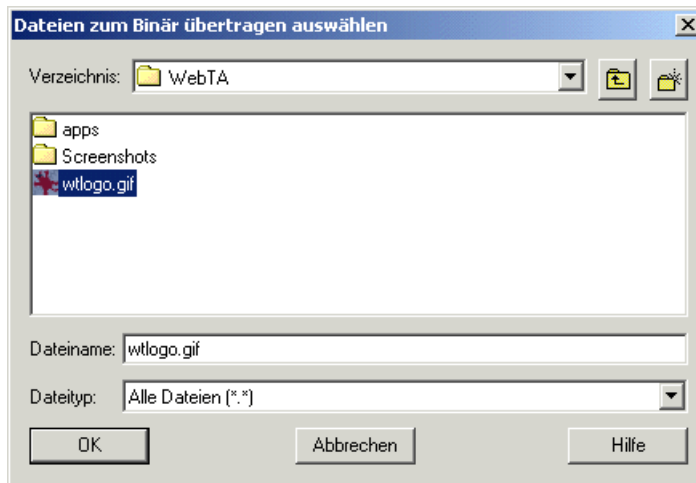
3.3 Globale Änderung der Darstellung

Als Beispiel für eine globale Änderung soll ein Firmenlogo in `AutomaskOSD.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 `wwdocs/image` im Basisverzeichnis nutzen.

Bild binär übertragen

Um das Firmenlogo mit WebLab in das Verzeichnis `wwdocs/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.

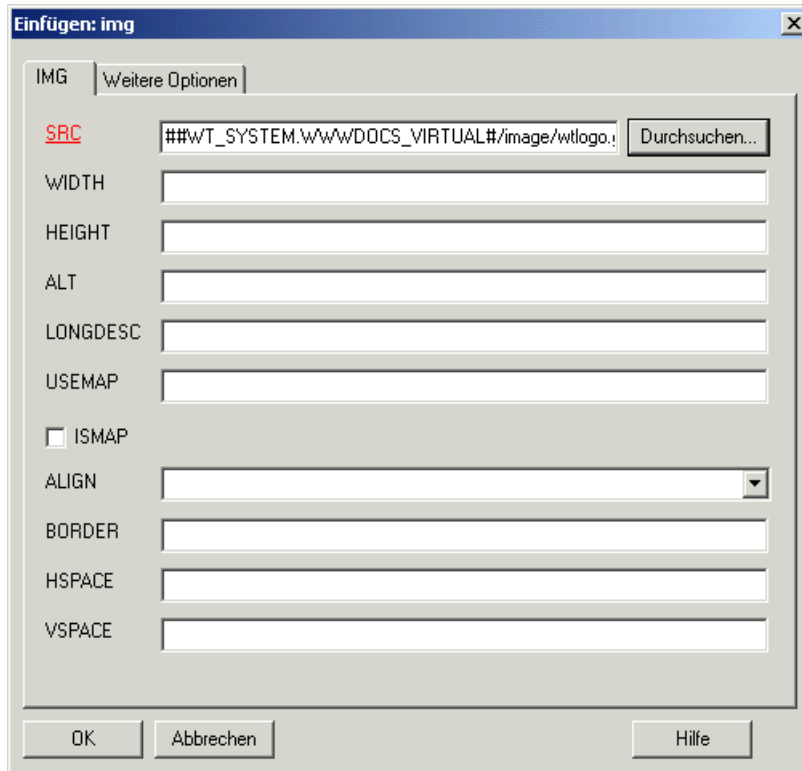


- ▶ Wählen Sie in diesem Dialogfeld das Verzeichnis, in dem Sie das Firmenlogo gespeichert haben und bestätigen Sie mit **OK**. Das Dialogfeld **Zielverzeichnis auswählen** wird am Bildschirm eingeblendet.
- ▶ Wählen Sie in diesem Dialogfeld das Verzeichnis `wwdocs/image` im Basisverzeichnis Ihrer WebTransactions-Anwendung 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 den Befehl **Datei/aktuelles Template öffnen**.
 - Damit laden Sie in den Arbeitsbereich von WebLab das Template `AutomaskOSD.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**. Das Dialogfeld **Einfügen:img** wird am Bildschirm eingeblendet.



In diesem Dialogfeld können Sie die Quelldatei 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 `wwdocs/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 `wwdocs` im Basisverzeichnis.

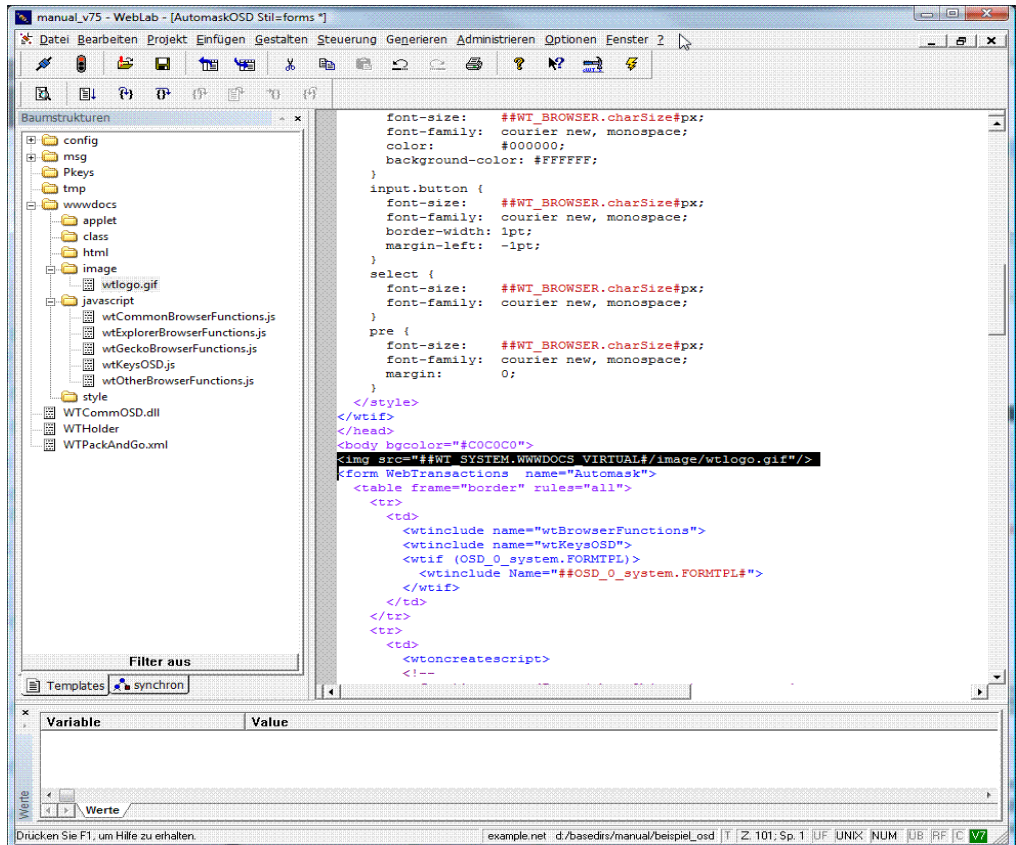
- ▶ Verwenden Sie den Knopf **Durchsuchen**, um die Bilddatei auszuwählen. Wenn sie sich unterhalb von `wwdocs` befindet, verwendet WebLab automatisch `WWWDOCS_VIRTUAL` und der Name wird wie folgt erzeugt:

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

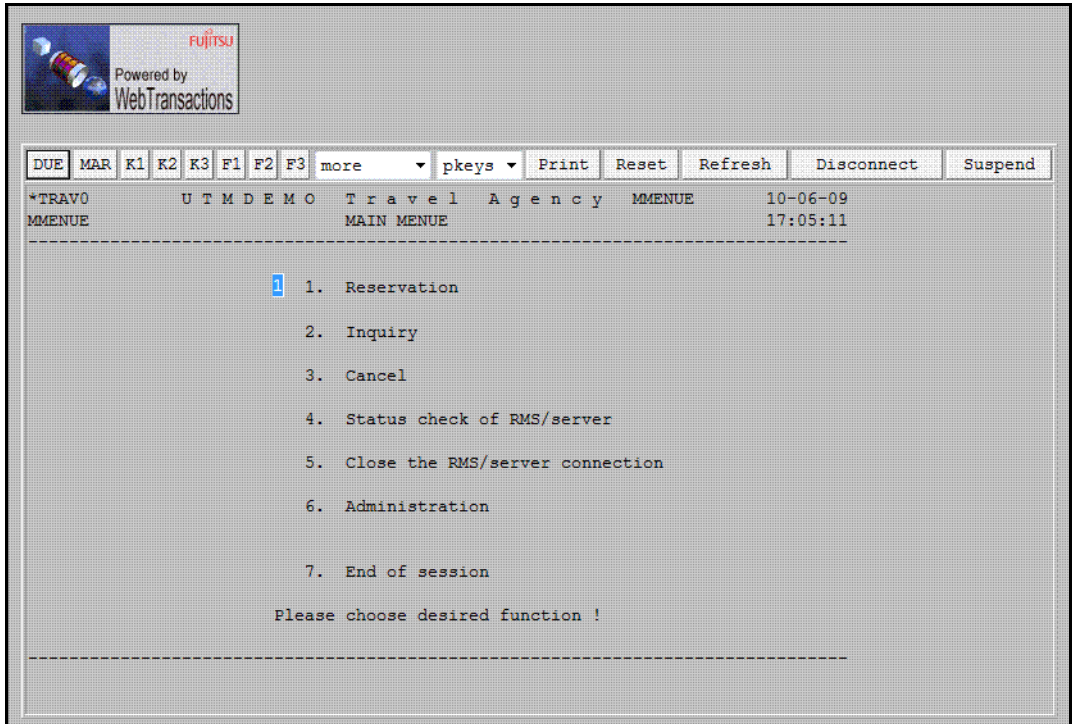
- Bestätigen Sie Ihre Angaben mit **OK**. WebLab fügt nach dem BODY-Tag folgende Zeile in das Automask-Template ein:

```

```



- ▶ Um sich die Änderung anzusehen, wählen Sie den Befehl **Steuerung/Im Browser aktualisieren**. Das Logo 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.

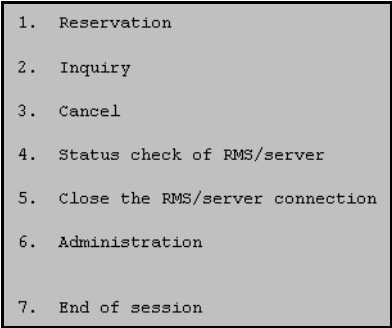
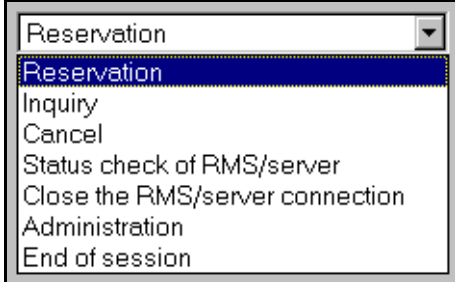


3.4 Formatspezifische Änderung der Darstellung

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
 <p>1. Reservation 2. Inquiry 3. Cancel 4. Status check of RMS/server 5. Close the RMS/server connection 6. Administration 7. End of session</p>	 <p>Reservation Reservation Inquiry Cancel Status check of RMS/server Close the RMS/server connection Administration End of session</p>

3.4.1 Formatspezifisches Template mit dem Capture-Verfahren generieren

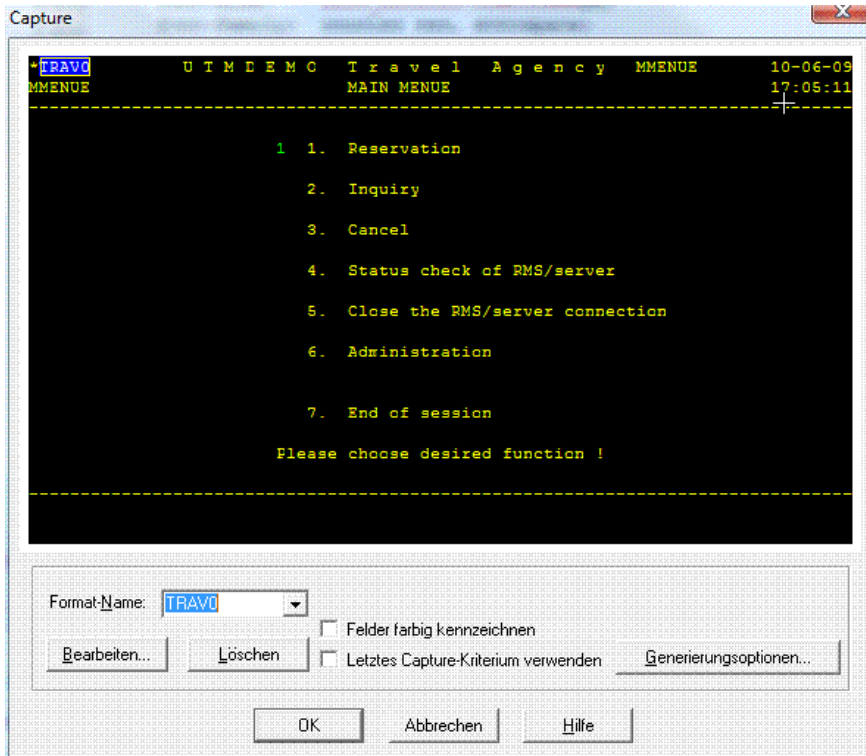
Um formatspezifische Templates für die Formate der Host-Anwendung zu erstellen, benutzen Sie das Capture-Verfahren in WebLab. Mit dem Capture-Verfahren erstellen Sie interaktiv Erkennungskriterien für die einzelnen Bildschirmformate, d.h. Muster zur Erkennung bekannter Formate.

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



Im Start-Template `wtstartOSD.htm` ist für das kommunikationsspezifische Systemobjekt-Attribut `CAPTURE_FILE` als Vorbelegung der Pfadname `config/capture.sdb` eingetragen (siehe [Abschnitt „Sitzung starten“ auf Seite 42](#)). 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 *TRAVO*. Damit legen Sie fest, dass dieses Format daran erkannt werden soll, dass TRAVO 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 *TRAVO* erstellt. Statt des Automask-Templates wird ab jetzt das format-spezifische Template zur Darstellung dieses Formats verwendet.

Generiertes Template

Der folgende Text zeigt den Abschnitt aus dem generierten Template *TRAVO.htm*, der die Felder des Formats darstellt. Den Aufbau eines vollständigen Templates entnehmen Sie dem [Abschnitt „Aufbau von AutomaskOSD.htm“ auf Seite 82](#).

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

```
<!-- ----- -->
<!-- begin of host screen section -->
<!-- ----- -->
<div style="color:##OSD_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 Auswertungoperator `##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 Auswertungoperator in einem Kommentar gespeichert.

```
<span class="screenline" id="SL1"><wtrem ** *TRAVO          U T M D E M O   T r a v e l
A g e n c y      **>\
##OSD_0.E_01_001_059.HTMLValue#\
<wtrem ** MMENU     **>\
##OSD_0.E_01_060_008.HTMLValue#\
<wtrem **          **>\
##OSD_0.E_01_068_005.HTMLValue#\
<wtrem ** 10-06-09 **>\
##OSD_0.E_01_073_008.HTMLValue#\</span>
<span class="screenline" id="SL2"><wtrem ** MMENU                               MAIN MENU
**>\
##OSD_0.E_02_001_041.HTMLValue#\
<wtrem **                               **>\
##OSD_0.E_02_042_031.HTMLValue#\
<wtrem ** 17:05:11 **>\
##OSD_0.E_02_073_008.HTMLValue#\</span>
<span class="screenline" id="SL3"><wtrem ** -----
----- **>\
```

```

##OSD_0.E_03_001_080.HTMLValue#</span>
<span class="screenline" id="SL4"><wtrem **
**>\
##OSD_0.E_04_001_080.HTMLValue#</span>
<span class="screenline" id="SL5"><wtrem **
**>\
##OSD_0.E_05_001_024.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="##(OSD_0.E_05_025_001.Visible == 'No') ? 'password' : 'text'#" ##(
WT_BROWSER.acceptClass ) ? 'class="box" style="width:9px"' : ''# name="E_05_025_001"
size="1" maxlength="1" markable="1" value="##OSD_0.E_05_025_001.Value#" />\
<wtrem ** 1. Reservation **>\
...
<wtrem ** Please choose desired function ! **>\
##OSD_0.E_20_025_032.HTMLValue#\
<wtrem **
**>\
##OSD_0.E_20_057_024.HTMLValue#</span>
<span class="screenline" id="SL21"><wtrem **
**>\
##OSD_0.E_21_001_080.HTMLValue#</span>
<span class="screenline" id="SL22"><wtrem ** -----
----- **>\
##OSD_0.E_22_001_080.HTMLValue#</span>
<span class="screenline" id="SL23"><wtrem **
**>\
##OSD_0.E_23_001_080.HTMLValue#</span>
<span class="screenline" id="SL24"><wtrem **
**>\
##OSD_0.E_24_001_080.HTMLValue#</span>

```

Alle Eingabefelder werden in einem Objekt verwaltet.

```

<wtoncreatescript>
<!--
wtInputFields = {E_05_025_001:OSD_0.E_05_025_001};
/-->
</wtoncreatescript></pre></div>
<!-- ----- -->
<!-- end of host screen section -->
<!-- ----- -->
</td>

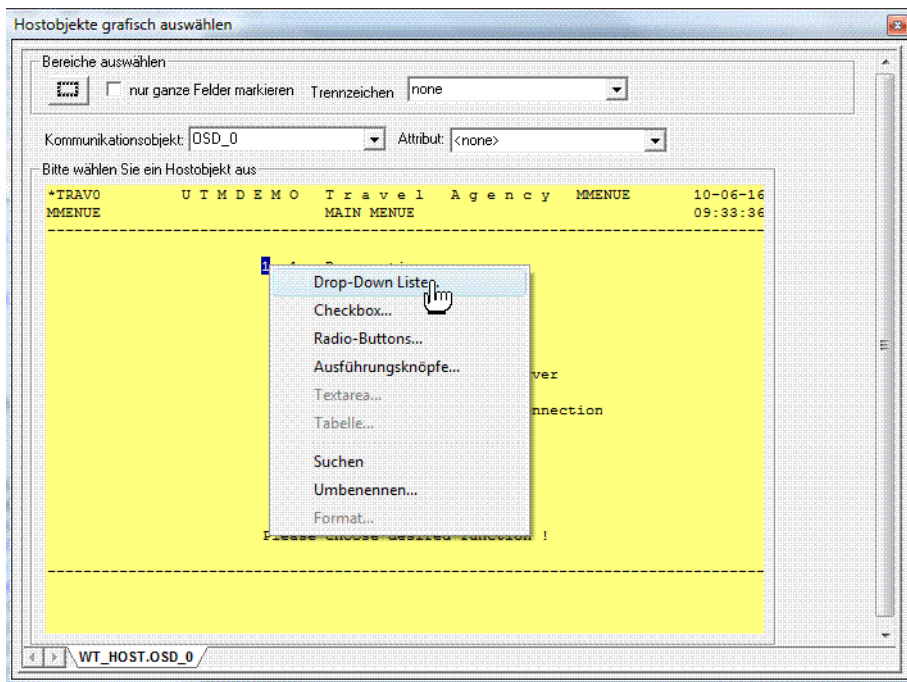
```

3.4.2 Formatspezifisches Template bearbeiten

- ▶ Wählen Sie den Befehl **Datei/aktuelles Template öffnen**, um das formatspezifische Template TRAV0 im Arbeitsbereich von WebLab zu öffnen.
- ▶ Wählen Sie dann den Befehl **Gestalten/Hostobjekte grafisch auswählen/Aus einem Kommunikationsobjekt**. Die grafische Hostobjektauswahl für das aktuelle Template wird am Bildschirm eingeblendet.

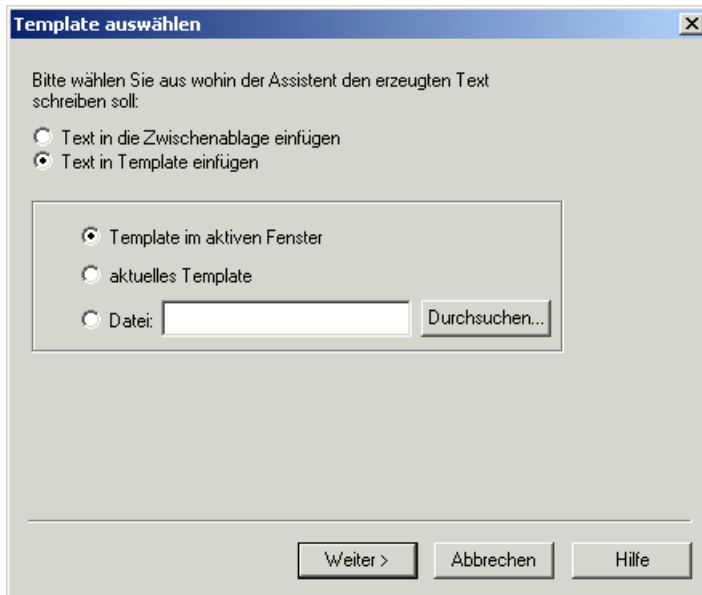
In diesem Dialogfeld wird das Format angezeigt, wie es auch in einer Emulation oder am Terminal dargestellt würde. Alle Ausgabefelder, die nicht veränderbar sind, sind grau hinterlegt. Das einzige Eingabefeld dieses Formats ist mit der Farbe weiß hinterlegt.

- ▶ 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**.

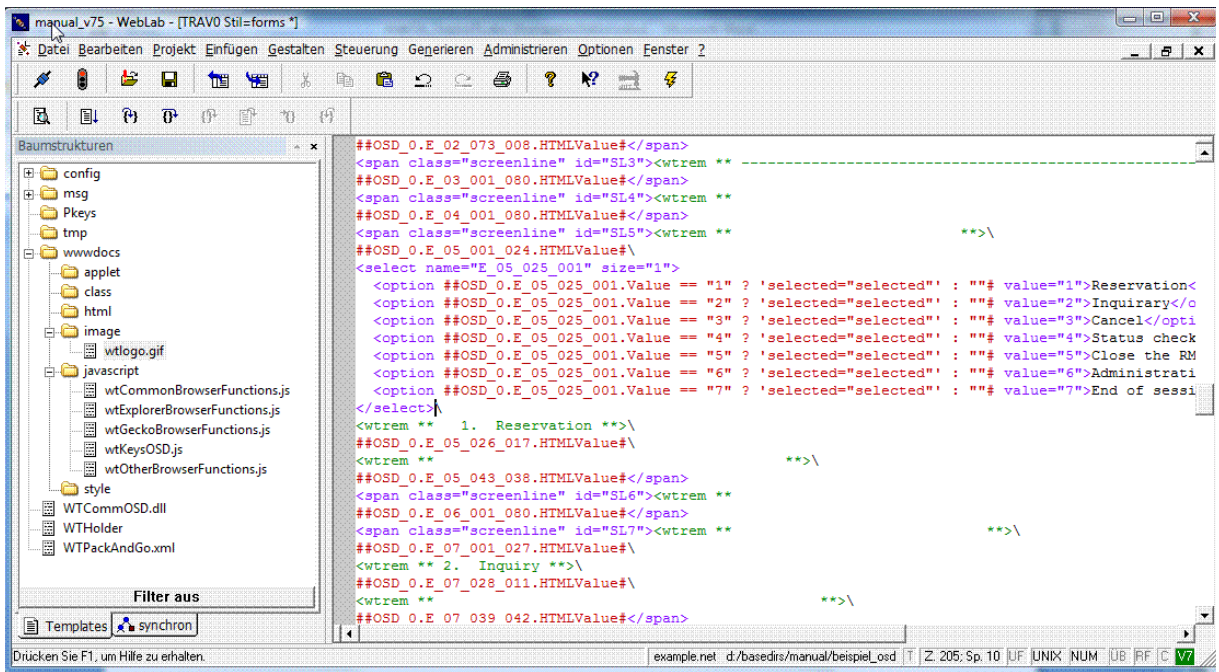
- ▶ Bestätigen Sie ansonsten die Voreinstellung mit **Weiter**.

Das zweite Dialogfeld **Werte zuordnen** wird am Bildschirm eingeblendet.

Intern	Wert an der Oberfläche
1	Reservation
2	Inquiry
3	Cancel
4	Status check of RMS
5	Close the RMS/server connection
6	Administration

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 51](#)) in die Eingabefelder ein. Mit der Schaltfläche **Hinzufügen** übernehmen Sie ein solches Wertepaar in die Liste.
- ▶ Bestätigen Sie am Ende der Liste Ihre Eingaben mit **Fertig stellen**. Der entsprechende HTML-Code für die Umsetzung einer Liste wird mit den angegebenen Werten in das Template *TRAV0.htm* eingefügt.
- ▶ Um sich diese Ersetzung anzusehen, blättern Sie im Template TRAV0, bis Sie zum Host-Abschnitt gelangen. Dieser Abschnitt beginnt mit dem Kommentar `begin of host screen section`.

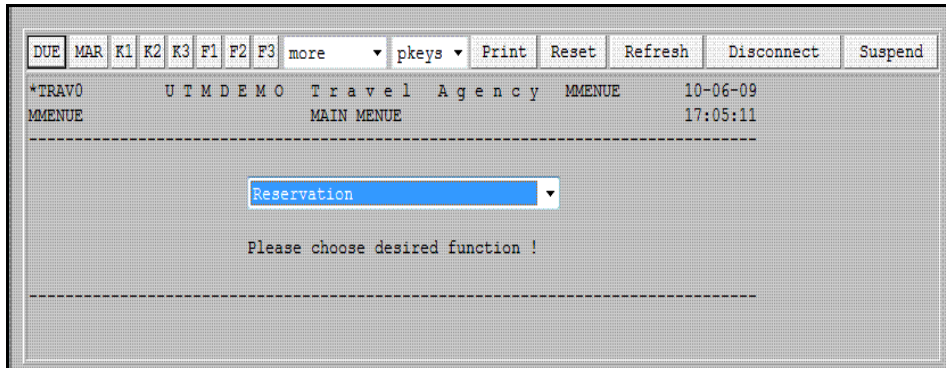


Das formatspezifische Template ist analog zum Automask-Template aufgebaut. Alle Felder des Host-Formats werden im Host-Abschnitt mit Namen aufgeführt, wobei sich jeder Name aus der Zeilen- und Spaltenposition zusammensetzt. Die obersten Felder sorgen für die Darstellung des Kopfes, die neue Liste (`select`-Tag)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

Da die Auswahl jetzt über die Liste erfolgt, sind die erläuternden Texte hinter der Liste nicht mehr notwendig, d.h. Sie können alle Host-Objekte und die Kommentare aus dem Template `TRAVO.htm` löschen.

- Löschen Sie alle Felder des Formats hinter der `select`-Liste bis einschließlich 7 End of Session. 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 `wtstartOSD.htm` verzweigt (siehe [Abschnitt „OSD-spezifisches Start-Template des Start-Template-Sets \(wtstartOSD.htm\)“ auf Seite 170](#)). Mit der Auswahl **main menu** und dem Knopf **go to** gelangen Sie zum allgemeinen Start-Template zurück. Hier können Sie mit **quit** die WebTransactions-Anwendung verlassen.

3.5 WebTransactions-Anwendung starten

Eine bearbeitete WebTransactions-Anwendung starten Sie mit WebLab auf die gleiche Weise, wie die automatische 1:1-Umsetzung (siehe [Abschnitt „Sitzung starten“ auf Seite 42](#) ff). Einziger Unterschied: Sie sollten sich vergewissern, dass im Template `wtstartOSD.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 zur WebTransactions-Anwendung bestehen.

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

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

- ▶ Geben Sie unter **Dateiname** das Verzeichnis und den Namen des Start-Templates ein, hier `config/forms/Start_Travel.htm`.
- ▶ Wählen Sie dann die Registerkarte **WT_SYSTEM-Attribute**.

In der Registerkarte **WT_SYSTEM-Attribute** legen Sie die wichtigsten Attribute des Systemobjekts fest. Für die Beispielsitzung sind die Voreinstellungen ausreichend.


- ▶ Wählen Sie die Registerkarte **OSD-Verbindungsparameter**.

The screenshot shows a dialog box titled 'Einfügen: wtcStartOSD' with four tabs: 'wtcStartOSD', 'WT_SYSTEM Attribute', 'OSD Verbindungsparameter', and 'Weitere Optionen'. The 'OSD Verbindungsparameter' tab is active. It contains the following fields and controls:

- Kommunikationsobjekt:** A dropdown menu with 'OSD_0' selected.
- Hostapplikation:** A dropdown menu with 'travel' selected, a small icon, and an 'Auswählen...' button.
- Name des Host-Rechners:** A dropdown menu with 'system1' selected, a small icon, and an 'Auswählen...' button.
- Präfix anwenden:** An unchecked checkbox.
- Capture-Datenbank:** A dropdown menu with 'config/travel.sdb' selected and a 'Durchsuchen...' button.

At the bottom of the dialog are three buttons: 'OK', 'Abbrechen', and 'Hilfe'.

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

- ▶ Geben Sie den Namen für das Kommunikationsobjekt ein, hier *OSD_0*.
 -  Beachten Sie, dass der Name des Kommunikationsobjekts mit dem Namen in der Automask übereinstimmen muss.
- ▶ Geben Sie den Namen der Host-Anwendung ein, hier *travel*.
- ▶ Geben Sie die Internet-Adresse oder den symbolischen Namen des Rechners ein, auf dem die Host-Anwendung läuft, hier *system1*.
- ▶ Geben Sie den Namen der Capture-Datenbank ein, hier *config/travel.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 alle Eigenschaften für die Verbindung zur OSD-Host-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:wtcStartOSD** wird geschlossen. Das Start-Template *Start_Travel.htm* wird generiert und im Arbeitsbereich von WebLab angezeigt. Gespeichert wird das Start-Template im Basisverzeichnis unter `config/forms`.

Das Start-Template *Start_Travel.htm* wird nun bei jedem Aufruf die hier einmal vorgenommenen Einstellungen vornehmen. Sie müssen diese Einstellungen nicht mehr bei jedem Sitzungs-Start erneut eingeben, wie es im [Abschnitt „Sitzung starten“ auf Seite 42](#) mit `wtstart.htm` und `wtstartOSD.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-Template 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 für den Benutzer bereit stellen.
- Sie können eine eigene Aufrufseite schreiben, in der WebTransactions über ein Formular oder einen Link gestartet wird.

Beispiel

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

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

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

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

Beispiel

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

4 Basisverzeichnis anlegen

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

Bei einer De-Installation von WebTransactions oder beim Installieren einer neuen Produktversion bleiben die individuellen Konfigurationen 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 einrichten. Anschließend muss er für diese Benutzerkennung einen oder mehrere Pools freigeben, damit Sie dort ein Basisverzeichnis anlegen können.

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

Gehen Sie folgendermaßen vor:

- ▶ Rufen Sie WebLab auf, z.B. über **Start/Programme/WebTransactions 7.5/WebLab**

- ▶ Als Einstieg zum Anlegen eines Basisverzeichnisses gibt es folgende zwei Möglichkeiten:
 - ▶ Wählen Sie den Befehl **Projekt/Neu...** und bestätigen Sie die Abfrage, ob ein Basisverzeichnis erstellt werden soll, mit **Ja** (siehe auch [Abschnitt „Projekt anlegen“ auf Seite 35](#)).

oder

- ▶ Wählen Sie den Befehl **Generieren/Basisverzeichnis ...** und geben Sie bei der folgenden Abfrage an, dass ein neues Projekt erstellt wird.

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

- ▶ Tragen Sie im Dialogfeld **Verbinden** die Verbindungsparameter ein und bestätigen Sie mit **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 **OSD**
 - Klicken Sie auf **OK**.
- ▶ Tragen Sie im Dialogfeld **Automask generieren** die gewünschten Optionen ein. Diese entsprechen den Optionen für die Generierung formatspezifischer Templates.
- ▶ Wählen Sie **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“ sowie im [Abschnitt „Struktur eines Basisverzeichnisses“ auf Seite 69](#) beschrieben.

Basisverzeichnis auf eine neue Version umstellen

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

4.2 Struktur eines Basisverzeichnisses

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



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

Unterverzeichnis Pkeys

Im Unterverzeichnis `Pkeys` wird standardmäßig die Zuordnung der programmierbaren Tasten (P-Tasten) abgelegt. Die Zuordnung der P-Tasten muss in einer XML-Datei vorliegen, die Sie z.B. mit dem Template `wtPKEYS.htm` erstellen und sitzungsspezifisch speichern können. Nach dem Starten einer WebTransactions-Sitzung kann eine P-Tasten-Zuordnung, die im Verzeichnis `Pkeys` vorliegt, in die aktuelle Emulation geladen werden.

5 WebTransactions konfigurieren und starten

WebTransactions for OSD wird auf den Plattformen BS2000/OSD, Solaris, Linux und Windows angeboten. Der Anschluss an den Host-Rechner erfolgt über eine 9750-Terminal-Emulation, die in den Host-Adapter integriert ist.

Die für den Anschluss notwendigen Konfigurationsdaten wie z.B. Host-, Anwendungs- und Rechnername werden mit Hilfe bestimmter Systemobjekt-Attribute gesetzt. Im [Abschnitt „Zusammenspiel von Systemobjekt-Attributen und Methoden“ auf Seite 133](#) sind die relevanten Attribute aufgelistet. Diese Attribute werden im Start-Template mit den entsprechenden Werten versorgt, siehe [Abschnitt „OSD-spezifisches Start-Template des Start-Template-Sets \(wtstartOSD.htm\)“ auf Seite 170](#).

5.1 Zeichensatz konfigurieren

Die 9750-Terminal-Emulation, die in WebTransactions for OSD integriert ist, unterstützt den klassischen 7-Bit-ASCII-Zeichensatz. Ist die Emulation als 9763-Terminal konfiguriert, werden auch mehrere 8 Bit Zeichensätze der Norm ISO-8859-x unterstützt. Das Terminal-Protokoll 9750 wird durch die Unicode-Unterstützung in BS2000/OSD zudem Unicode-fähig.

5.1.1 7-Bit-ASCII-Zeichensatz

Der 7-Bit-ASCII-Zeichensatz umfasst 128 Zeichen einschließlich Kontrollzeichen. Einige Zeichen können durch nationale Varianten ersetzt werden wie z.B. „|“ durch „?“, sodass mehrere 7-Bit-Varianten unterstützt werden.

WebTransactions unterstützt diese 7-Bit-Varianten durch das Systemobjekt-Attribut `NATIONAL_VARIANT`, das folgende Werte haben kann:

- Danish
- English (UK)
- English (USA)
- French
- French-Belgian
- German
- International
- Italian
- Norwegian
- Spanish
- Swedish
- Swiss

Der Host-Adapter setzt die 7-Bit-Varianten in den 8-Bit Zeichensatz ISO Latin-1 um, der vom Browser benutzt wird.

Dieser Zeichensatz enthält 256 Zeichen. Davon sind die ersten 128 identisch mit dem klassischen 7-Bit-ASCII-Zeichensatz. Die zweiten 128 Zeichen beinhalten dagegen etliche Sonderzeichen und Buchstaben für bestimmte Sprachen, z.B. „ö“, „ß“ oder „á“.

Die 9750-Emulation muss entsprechend dem Zeichensatz, der von der OSD-Host-Anwendung verwendet wird, konfiguriert sein.

5.1.2 8-Bit-Zeichensatz

WebTransactions unterstützt den 8-Bit-Zeichensatz (ISO-8859-x), wenn es ein 9763-Terminal emuliert, d.h., das Systemobjekt-Attribut `TERMINAL_TYPE` hat den Wert 9763.

Die Norm ISO-8859-x umfasst mehrere 8-Bit-Zeichensätze, die unterschiedliche Sprachen unterstützen:

ISO-Norm	Zeichensatz
ISO-8859-1	Latin-1
ISO-8859-2	Latin-2
ISO-8859-5	Kyrillisch
ISO-8859-7	Griechisch
ISO-8859-9	Türkisch

Zusätzlich werden die alten 7-Bit-Zeichensätze zwecks Kompatibilität unterstützt.

Einige OSD-Anwendungen fordern zu Beginn der Kommunikation mit dem Terminal einen „status request“, den das Terminal mit der Liste der 8-Bit-Zeichensätze beantwortet, die es unterstützt. Bei jedem Kommunikationsschritt schicken diese OSD-Anwendungen Informationen über den aktuellen Zeichensatz mit (7- oder 8-Bit). Der OSD-Host-Adapter von WebTransactions interpretiert diese Informationen und setzt das verbindungspezifische Systemobjekt-Attribut `HOST_CHARSET` entsprechend (siehe [Abschnitt „Systemobjekt-Attribute“ auf Seite 115](#)). Bei 7-Bit-Zeichensatz z.B. wird es auf ISO-8859-1 gesetzt, die nationale Variante wird entsprechend berücksichtigt.

Alle generierten Standard-Templates enthalten das folgende Script, welches das globale Systemobjekt-Attribut `CHARSET` mit dem verbindungspezifischen Wert von `HOST_CHARSET` versorgt:

```
WT_SYSTEM.CHARSET = host_system.HOST_CHARSET;
```



Sie müssen diese Wertzuweisung in Ihre selbst erstellten Templates entsprechend einbauen.

WebTransactions verwendet den Wert aus `WT_SYSTEM.CHARSET` beim Versenden der HTML-Seite an den Browser im HTTP-Protokoll (HTTP-Header-Feld `content-type`, siehe WebTransactions-Handbuch „Konzepte und Funktionen“). Damit können die Browser, die dieses Konzept unterstützen, „dynamisch den entsprechenden Zeichensatz einstellen.“

Das HTTP-Protokoll benutzt standardmäßig den 8-Bit ISO Latin-1-Zeichensatz. In HTTP V1.1 können Sie einen anderen Zeichensatz für den Browser spezifizieren. Je nach verwendetem Browser und Konfiguration kann es sein, dass Sie ggf. die automatische Übernahme des Wertes aus dem HTTP-Header `content-type/charset` oder eine feste Zeichensatz-Einstellung konfigurieren müssen.

Die Template-Sprache WTML unterstützt alle 8-Bit-Zeichensätze. D.h. Sie können mit einem entsprechenden Editor z.B. kyrillisch schreiben, wenn die OSD-Host-Anwendung diesen Zeichensatz unterstützt.

5.1.3 Unicode-Zeichensatz

Das Terminal-Protokoll 9750 ist aufgrund der Unicode-Unterstützung in BS2000/OSD Unicode-fähig. Sie müssen die Unicode-Unterstützung für 9750 explizit durch Setzen des entsprechenden Terminaltyps einschalten, andernfalls verhält sich WebTransactions for OSD wie beim Systemobjekt-Attribut `TERMINAL_TYPE`, [Seite 131](#) beschrieben.

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

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

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```
 - oder durch das Setzen des Attributs `charset` im HTTP-Header-Feld `Content-Type`.
- Die Templates dürfen ausschließlich ASCII-Zeichen enthalten. Enthält ein Template Sonderzeichen (z.B. Umlaute) direkt als ANSI-Zeichen in Zeichenketten, so müssen diese durch die HTML-Escape-Sequenzen ersetzt werden. Die Escape-Sequenzen bestehen ihrerseits nur aus ASCII-Zeichen und können daher in UTF-8 verwendet werden.
- Zeichenkettenoperationen können fehlerhafte Ergebnisse produzieren, wenn in der Zeichenkette Zeichen in UTF-8 enthalten sind (siehe auch die Hinweise auf [Seite 137](#)). In den von WebTransactions generierten Templates kommen solche Operationen nicht vor.

5.2 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 77](#) beschriebene automatische Umsetzung der Host-Formate verwenden, sind keine weiteren Schritte mehr notwendig. Nach Installation von WebTransactions (siehe [Seite 17](#)) und Einrichten eines Basisverzeichnisses (siehe [Seite 67](#)) ist Ihre WebTransactions-Anwendung bereits einsatzfähig. Für den Start der Host-Anwendung steht Ihnen z.B. das fertig ausgelieferte Start-Template-Set zur Verfügung, das in [Abschnitt „Start-Templates für OSD“ auf Seite 169](#) beschrieben ist.

Für den Produktivbetrieb sollten Sie mit Hilfe des WtBean `wtcStartOSD.wtc` in WebLab ein Anwendungs-spezifisches Start-Template erzeugen.

6 Host-Anwendung ohne Bearbeitung integrieren

Für die dynamische Umsetzung der 9750-Formate auf die Oberfläche eines Web-Browsers stellt WebTransactions for OSD 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:

- `OSD.wmt/OSD_pocket.wmt`
OSD-spezifische Master-Templates für ein generelles Layout der Formulare. Diese Templates werden bei der Generierung des Automask-Templates und der formatspezifischen Templates verwendet.
- `AutomaskOSD.htm`
Umsetzungs-Template für OSD-Host-Anwendungen, das in der Lage ist, alle Formate einer OSD-Anwendung automatisch in eine HTML-Seite mit einem Formular umzusetzen.
- `wtKeysOSD.htm`
Dieses Template fügt Bedienelemente ein, die Tasten des 9750-Terminals abbilden. `wtKeysOSD.htm` wird nicht nur bei der dynamischen Umsetzung verwendet. Auch die mit dem Capture-Verfahren erstellten formatspezifischen Templates nutzen `wtKeysOSD.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 `AutomaskOSD.htm` erstellen (siehe [Seite 80](#)).

Globale Anpassungen, z.B. für eine Corporate-Identity, können Sie im Template `AutomaskOSD.htm` durchführen. Die Anpassungen werden dann automatisch für alle Formate 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.

6.1 Master-Templates OSD.wmt und OSD_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 WTScrips 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 OSD werden das Standard-Master-Template `OSD.wmt` und `OSD_Pocket.wmt` mit ausgeliefert, die Sie individuell anpassen, aber auch unverändert einsetzen können. Die Standard-Master-Templates enthalten bereits alle WTML-Tags und WTScrips, die für alle Templates der jeweiligen Produktvariante gleich sind, z.B. eine Prüfung, ob ein privates System-Objekt existiert.

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

Einsatz von OSD.wmt oder OSD_pocket.wmt

Das Master-Template `OSD_Pocket.wmt` wurde speziell für den Einsatz auf dem Pocket Internet Explorer entwickelt. Sie können es für die Generierung eines Automask-Templates (siehe [Abschnitt „Template AutomaskOSD.htm“ auf Seite 80](#)) und für formatspezifische Templates (siehe [Abschnitt „Capture-Verfahren mit WebLab“ auf Seite 94](#)) verwenden.

`OSD_Pocket.wmt` bietet die folgenden zusätzlichen Funktionen:

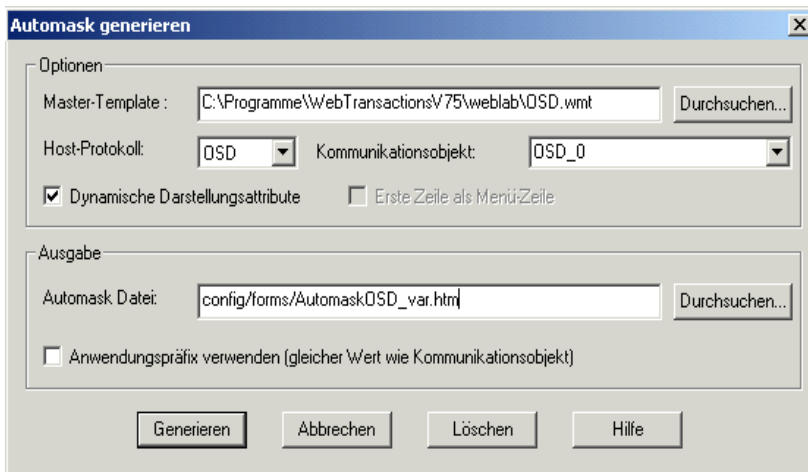
- Für den Bildschirm wird ein Frameset generiert. Es enthält zwei Frames, einen zur Kommando-Eingabe und einen für die Anzeige.
- Nur das Feld, in dem der Cursor steht, kann als Eingabefeld angesprochen werden.
- Die Zeilen auf dem Bildschirm werden automatisch umbrochen und farblich abgesetzt.
- Bei 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.

6.2 Template AutomaskOSD.htm

Das Template `AutomaskOSD.htm` erstellt dynamisch eine Abbildung des zuletzt vom Host mit `receive` empfangenen Formats. Es ist in der Lage, ohne Vorbereitung jedes beliebige 9750-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.

6.2.1 Varianten von AutomaskOSD.htm erstellen (mit WebLab)

Das Template `AutomaskOSD.htm` wird in der Regel beim Erzeugen eines Basisverzeichnis 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 OSD.wmt und OSD_Pocket.wmt“ auf Seite 78](#). Ein Master-Template wird immer benötigt. Voreinstellung ist das mitausgelieferte Master-Template `OSD.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: `OSD_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. Die Option ist für WebTransactions for OSD nicht relevant.

Automask Datei.

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

Voreingestellter Pfad:

`basedir/config/forms/AutomaskOSD.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 FLD-Datei und Template ein Präfix setzen. Der Dateiname setzt sich dann zusammen aus:

`commobj@formatname.flld` bzw. `commobj@formatname.htm`

6.2.2 Aufbau von AutomaskOSD.htm

Im Folgenden ist das Automask-Template dargestellt, das mit dem Master-Template OSD.wmt (siehe [Abschnitt „Master-Templates OSD.wmt und OSD_Pocket.wmt“ auf Seite 78](#)) generiert wurde.

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

```
<html>
<wtrem>*****</wtrem>
<wtrem>** WTML document: AutomaskOSD **</wtrem>
<wtrem>*****</wtrem>
<wtrem>**
<wtrem>** Document generation based on Master Template : **</wtrem>
<wtrem>** C:\Program Files\webtransactions\75\web\lab\OSD.wmt **</wtrem>
<wtrem>**
<wtrem>** Generated at Tue Jun 08 12:44:20 2010 **</wtrem>
<wtrem>**
<wtrem>** Options used by the generator : **</wtrem>
<wtrem>** - %OPTIONS: **</wtrem>
<wtrem>** CommObj = OSD_0 **</wtrem>
<wtrem>** NationalVariant = International - PartialFormatMode = No **</wtrem>
<wtrem>** - %OPTIONS: **</wtrem>
<wtrem>** self defined Tag = taggedInputCrossCall **</wtrem>
<wtrem>** - %OPTIONS: **</wtrem>
<wtrem>** self defined Tag = taggedInputCrossCall **</wtrem>
<wtrem>** - %OPTIONS: **</wtrem>
<wtrem>** self defined Tag = taggedOutputCrossCall **</wtrem>
<wtrem>** - %OPTIONS: **</wtrem>
<wtrem>** self defined Tag = taggedOutputCrossCall **</wtrem>
<wtrem>** - %LINES: **</wtrem>
<wtrem>** TaggedInput = Enabled - TaggedOutput = Enabled **</wtrem>
<wtrem>** DisplayAttributes = Dynamic - CursorInProtectedField = Yes **</wtrem>
<wtrem>** - %RECEIVES: **</wtrem>
<wtrem>** Parameters not specified **</wtrem>
<wtrem>*****</wtrem>
<wtrem>** WebTransactions V7.5 Fujitsu Technology Solutions 2010 **</wtrem>
<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)
  OSD_0 = WT_HOST.active || WT_HOST.OSD_0;
  if (OSD_0.WT_SYSTEM != null)
    OSD_0_system = OSD_0.WT_SYSTEM; // communication specific system object
```

```

else
  OSD_0_system = WT_SYSTEM;          // global system object
  //}}
  // propagate communication object to included WTML documents ////////////////
  wtCurrentComm = OSD_0;
  wtCurrentComm_system = OSD_0_system;

```

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

```

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

```

Ein ggf. vorhandenes PROLOG-Template wird ausgeführt.

```

<wtif (OSD_0_system.PROLOG)>
<wtinclude Name="##OSD_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 158).

```

<head>
<title>WebTransactions V7.5 - application ##OSD_0_system.SYM_DEST# on
##OSD_0_system.HOST_NAME#</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;
    }
  </style>
</wtif>

```

```

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 `wtKeysOSD.htm` und `wtBrowserFunctions.htm` aufgerufen zur Unterstützung einer möglichst genauen 1:1 Darstellung. Deren Bedienelemente werden somit Teil des Formulars.

```

<body bgcolor="#C0C0C0">
<form WebTransactions name="Automask">
  <table frame="border" rules="all">
    <tr>
      <td>
        <wtinclude name="wtBrowserFunctions">
          <wtinclude name="wtKeysOSD">
            <wtif (OSD_0_system.FORMTPL)>
              <wtinclude Name="##OSD_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 )          {

          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:#FFCOCO' :
'');
              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;
          if (WT_BROWSER.acceptInputAttributes)
            input +=
              (hostObject.Input == 'Numeric'? " numeric="1':"')
              + (hostObject.Markable == 'Yes'? " markable="1':"')
              + (hostObject.Type == 'Protected'? "
wtprotected="1':"');
          input += '"/>';
          document.write( input );
        }

        function taggedOutput( hostObject )        {

output = hostObject.HTMLValue;
          if ( hostObject.Visible == 'Yes' )
            {
              if ( hostObject.Inverse == 'Yes' )
                {
                  if ( hostObject.Color=='#000000' )
                    output = '<font color="#FFFFFF" style="\background-color:#000001\">'
+ output + '</font>';
                  else
                    output = '<font color="#000000" style="\background-color:' +
hostObject.Color + '\">' + output + '</font>';
                }
            }
        }
```

```

        else if (hostObject.Color !=
OSD_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:##OSD_0.WT_Color.Default = "\#000000"#"><pre>\

```

Als wesentliche Funktionalität von AutomaskOSD.htm wird in einer Schleife jedes Format-element auf je ein HTML-Element abgebildet. Hierfür stellt der Host-Adapter die Host-Objekte \$FIRST und \$NEXT zur Verfügung. Der Code des AutomaskOSD-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 = OSD_0.$FIRST.Name; OSD_0 && element != '$END'; element =
OSD_0.$NEXT.Name)
    {
        currentHostObject = OSD_0[element];
        if ( currentHostObject.StartLine != currentLine )
        {
            document.write ('</span>
<span class="screenline" id="SL',++currentLine,'">');
        }
        if ( currentHostObject.Type == 'Protected' && currentHostObject.Markable == 'No' )
        {
            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. Für alle Browser, die nicht beliebige Attribute beim `<input>`-Tag akzeptieren, wird das Attribut `markable` gesetzt, sofern das entsprechende Hostobjekt markierbar ist.

```

    <script type="text/javascript">
<!--
    <wtoncreatescript>
<!--
        for( element in wtInputFields )
        {
            if (!WT_BROWSER.acceptMaxLength)
                document.writeln('wtSetMaxLength('\', element, '\',', wtInputFields[ element
].Length, ' ');');
            if (!WT_BROWSER.acceptInputAttributes)
            {
                if (wtInputFields[ element ].Input == 'Numeric')
                    document.writeln('wtSetInputAttribute("numeric", "", element, ');');
                if (wtInputFields[ element ].Markable == 'Yes')
                {
                    document.writeln('wtSetInputAttribute("markable", "", element, ');');
                    if (wtInputFields[ element ].Type == 'Protected')
                        document.writeln('wtSetInputAttribute("wtprotected", "", element, ');');
                }
            }
        }
    }
//-->
</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('##OSD_0.WT_FOCUS.Field##OSD_0.WT_FOCUS.Offset&&wtCurrentComm_system.isOverwrite?', '+OSD_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( OSD_0[WT_POSTED.wt_cursor] )
  {
    if(WT_POSTED.wt_cursorOffset && WT_POSTED.wt_cursorOffset*1>0)
    {
      wtCursorField = OSD_0[WT_POSTED.wt_cursor];
      OSD_0.WT_FOCUS.Field =
'E'+wtCursorField.STARTLINE+'_'+(wtCursorField.STARTCOLUMN*1+WT_POSTED.wt_cursorOffset*1)+'_1';
    }
    else
      OSD_0.WT_FOCUS.Field = WT_POSTED.wt_cursor;
  }
  if (OSD_0_system.EDIT_MODE &&OSD_0_system.EDIT_MODE.match(/USER/))
    OSD_0_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];
  }

  wtMarkedFields = WT_POSTED.wt_markedFields.split(',');
  for( i=1; i<wtMarkedFields.length; i++)
    OSD_0[wtMarkedFields[i]].Modified = 'Yes';
  //}}
  //{{WebLab(processHostCommunication)
  if ( WT_POSTED.wt_special_key == 'Suspend' || OSD_0_system.SUSPEND )
  {
    OSD_0_system.SUSPEND = false;
```



```
}
else
{
  try {
    OSD_0.send();
    OSD_0.receive();
    if( OSD_0_system.CAPTURED_FLD == "Yes" && OSD_0_system.APPLICATION_PREFIX )
      setNextPage( OSD_0_system.APPLICATION_PREFIX + '@' + OSD_0_system.FLD );
    else
      setNextPage( OSD_0_system.FLD );
    WT_SYSTEM.CHARSET = OSD_0_system.HOST_CHARSET;
  }
  catch (e) {
    if ( OSD_0.$CONNECTION.ALIVE == 'No' )
      if ( OSD_0_system.DISCONNECT )
        setNextPage(OSD_0_system.DISCONNECT);
      else
        setNextPage('wtstart');
    else if ( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT )
      setNextPage( WT_SYSTEM.COMMUNICATION_ERROR_FORMAT );
  }
}
//}}
//-->
</wtonreceivescript>
</body>
<wtif (OSD_0_system.EPILOG)>
  <wtinclude Name="##OSD_0_system.EPILOG#">
</wtif>
</html>
```

6.3 Template wtKeysOSD.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 `wtKeysOSD.htm` stellt Ihnen für die OSD-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 `wtKeysOSD.htm` einige Knöpfe, die keiner Taste eines Terminals entsprechen (DISCONNECT, REFRESH, SUSPEND und Print).

Die einzelnen Bedienelemente sind in der Tabelle auf [Seite 148](#) beschrieben.

`wtKeysOSD.htm` inkludiert die Datei `wtKeysOSD.js`, die die Abbildung der Sondertasten für WebTransactions für OSD 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 wtKeysOSD.js“ auf Seite 149](#).

Das Master-Template `OSD.wmt` enthält einen Aufruf (`<wtInclude>`) für `wtKeysOSD.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 `wtKeysOSD.htm` ein unsichtbares Eingabefeld an mit den Parametern `<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 in den Host-Adapter integrierte 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 Tabelle auf [Seite 148](#)). Alle Funktionen, die `wtKeysOSD.htm` und `wtKeysOSD.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`.

6.4 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 `OSD.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 `wtKeysOSD.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 „Unterstützte Terminal-Funktionen“ auf Seite 144](#) finden Sie eine Aufstellung, welcher Browser welche Funktion unterstützt.

6.5 Template wtPKEYS.htm

Ein 9750-Terminal bietet Ihnen zusätzlich die programmierbaren Tasten (P-Tasten), die Sie mit Funktionen belegen können.

Das Template `wtPKEYS.htm` stellt Ihnen eine einfache Möglichkeit zur Verfügung, die P-Tasten am Browser verfügbar zu machen. Das Template dient als Vorlage, Sie sollten es nach Ihren Bedürfnissen erweitern oder verändern. Die Beschreibung der Vorgehensweise finden Sie im [Abschnitt „Unterstützung der P-Tasten“ auf Seite 160](#).

`wtPKEYS.htm` inkludiert das Template `wtPkeyValues.htm`, das eine Tabelle für die Darstellung aller Sondertasten als Knöpfe enthält. Zusätzlich wird die Definition der Klasse `WT_PKEYS` im Template `wtPkeyFunctions.htm` verwendet. Die Klasse `WT_PKEYS` bietet Ihnen die Methoden zur Verwaltung der P-Tasten.

Das Template `wtPKEYS.htm` wird angezeigt, wenn Sie im Bildschirm der Host-Anwendung in der Auswahl-Liste `pkeys` den Befehl `P` auswählen. `wtPKEYS.htm` wird dann während der Sitzung anstelle des aktuellen Templates mit der Funktion `setNextPage()` aufgerufen.

7 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 94](#)).
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“.

7.1 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, die in der Capture-Datenbank verwaltet werden.

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“.

Die Capture-Datenbank erstellen Sie, indem Sie mit dem Capture-Verfahren für die einzelnen Formate Erkennungskriterien festlegen. Die Erkennungskriterien können Sie entweder einzeln für jedes Format festlegen oder gesammelt über eine Formatbeschreibungquelle.

7.1.1 Erkennungskriterien einzeln erfassen

Mit dieser Art des Capture-Verfahrens legen Sie für jedes Format die Erkennungskriterien einzeln fest.

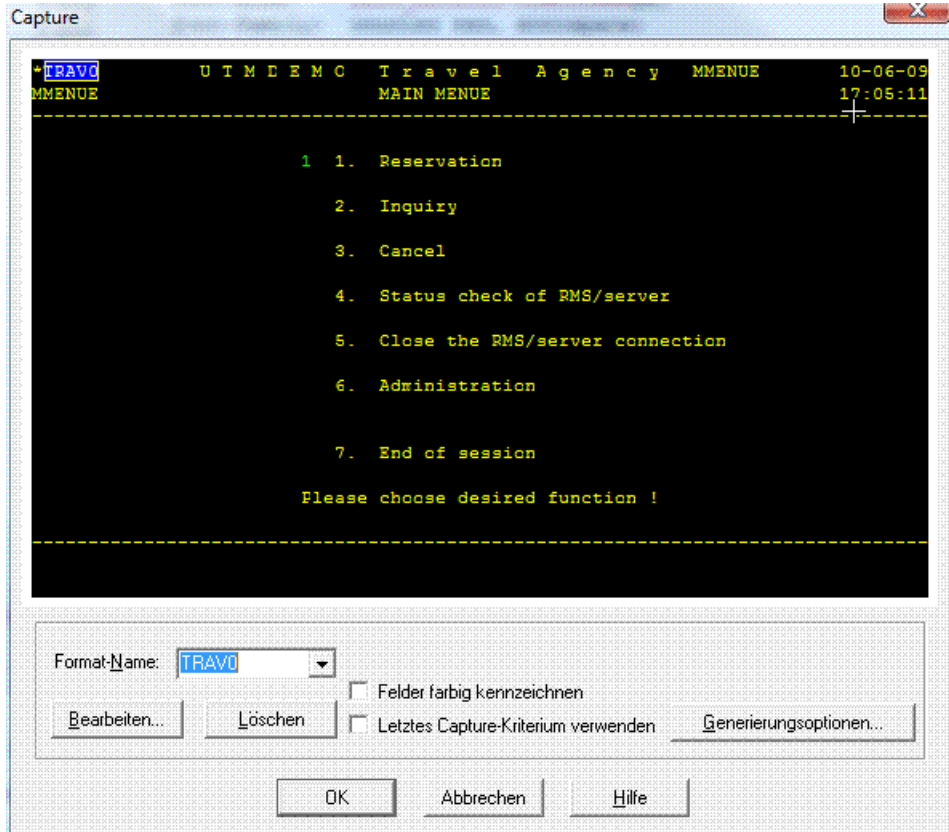
Vorgehensweise

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

Im Start-Template `wtstartOSD.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. 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.

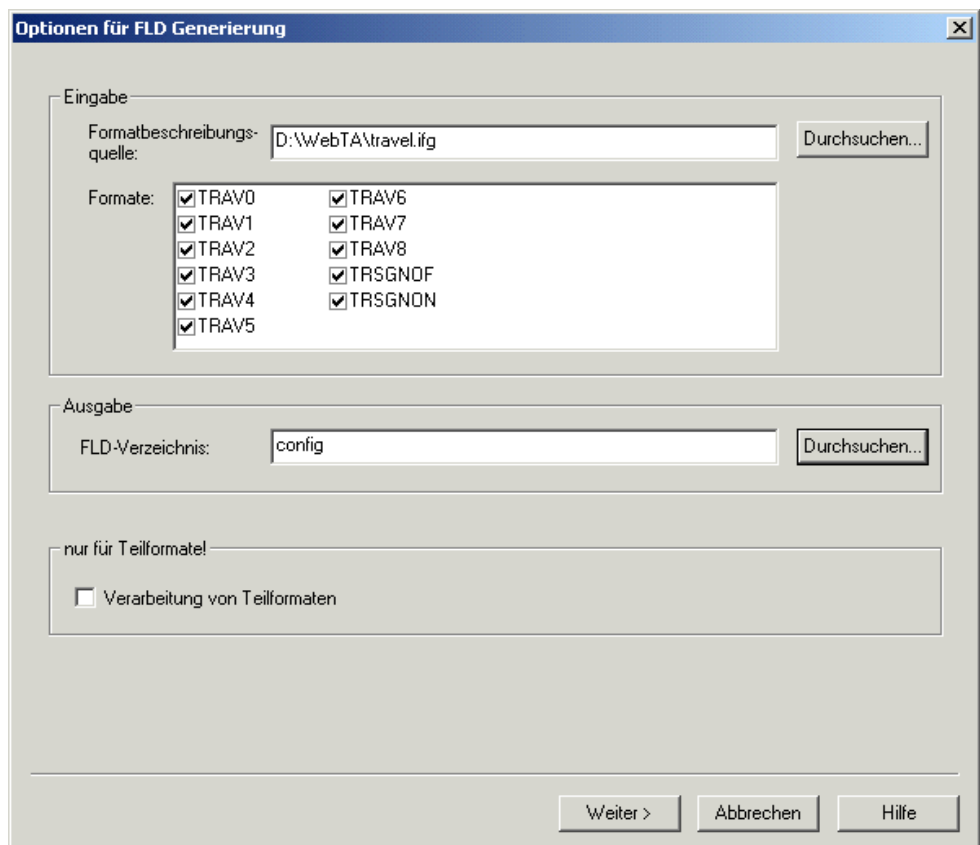
- ▶ Öffnen Sie dann mit der Schaltfläche **Generierungsoptionen** das Dialogfeld **Capture: Optionen für FLD und Template Generierung**. In diesem Dialogfeld legen Sie die Generierungsoptionen fest. In der Regel sind die Voreinstellungen ausreichend.
- ▶ Geben Sie die erforderlichen Generierungsoptionen 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 WebTransactions zur Laufzeit dann benötigt, wenn Sie sprechende IFG-Namen verwenden, siehe hierzu auch [Abschnitt „Sprechende Namen verwenden“ auf Seite 110](#). FLD-Dateien werden ansonsten 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.

7.1.2 Erkennungskriterien gesammelt erfassen

Mit dieser Art des Capture-Verfahrens legen Sie für alle erforderlichen Formate der Host-Anwendung die Erkennungskriterien gesammelt fest. Voraussetzung hierfür ist, dass Sie über eine Formatbeschreibungsource mit den erforderlichen Formaten verfügen. Die Formatbeschreibungsource erzeugen Sie mit dem Programm IFG2FLD, siehe hierzu auch [Abschnitt „Sprechende Namen verwenden“ auf Seite 110](#).

Vorgehensweise

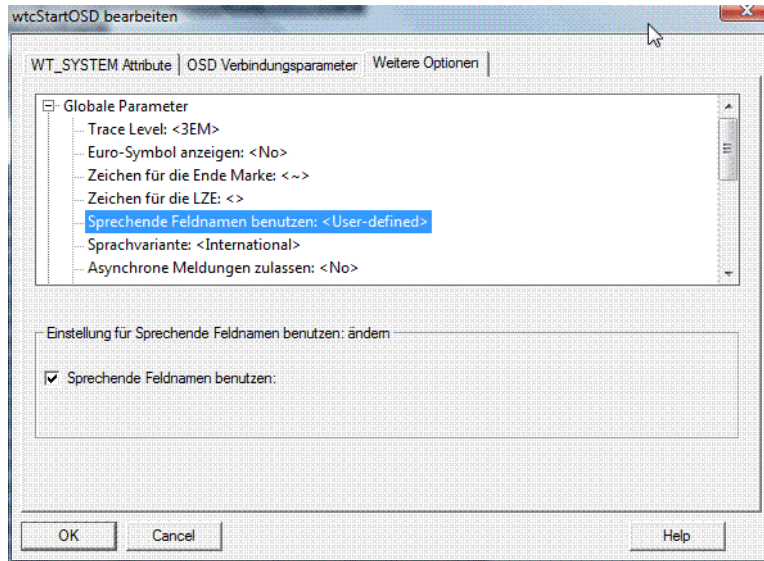
- ▶ Stellen Sie ggf. mit dem Befehl **Datei/Verbindung öffnen** eine Verbindung zum Basisverzeichnis her.
- ▶ Wählen Sie den Befehl **Generieren/Templates/aus IFG-Bibliothek**, um aus der Formatbeschreibungsource FLD-Dateien und Templates zu generieren. Das Dialogfeld **Optionen für FLD-Generierung** wird am Bildschirm eingeblendet.



- ▶ Geben Sie in der Gruppe **Eingabe** die Formatbeschreibungsource ein. Die dort beschriebenen Formate werden in der Liste **Formate** angezeigt.
- ▶ Wählen Sie dann in der Gruppe **Ausgabe** das Verzeichnis, in das die generierten FLD-Dateien geschrieben werden sollen.
- ▶ Bestätigen Sie mit **Weiter**. Das Dialogfeld **FLD und Template generieren** wird eingeblendet.

- ▶ Geben Sie hier in der Gruppe **Ausgabe** das Verzeichnis an, in das die generierten Templates geschrieben werden sollen.
- ▶ Markieren Sie die Option **Capture aus FLD-Dateien durchführen**, um die Erkennungskriterien gesammelt zu erfassen.
- ▶ Bestätigen Sie mit **Fertig stellen**.

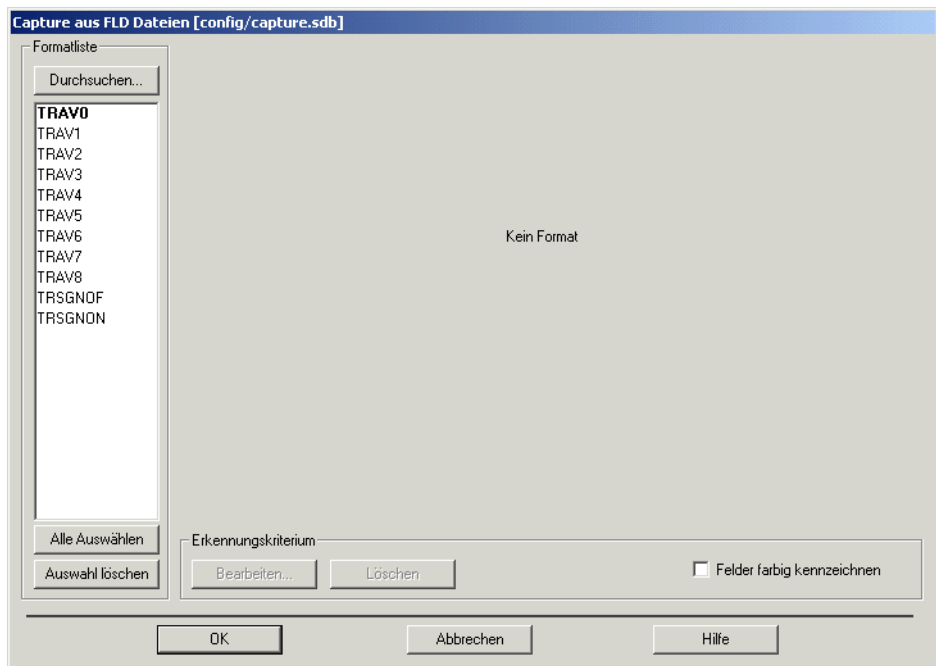
- ▶ Setzen Sie das kommunikationsspezifische Systemobjekt-Attribut `FIELD_NAMES` auf den Wert `User-defined`. Schalten Sie dazu beim Erzeugen oder Bearbeiten des anwendungsspezifischen Start-Templates auf der Registerkarte **Weitere Optionen** die Option **Globale Parameter / Sprechende Feldnamen benutzen** ein.



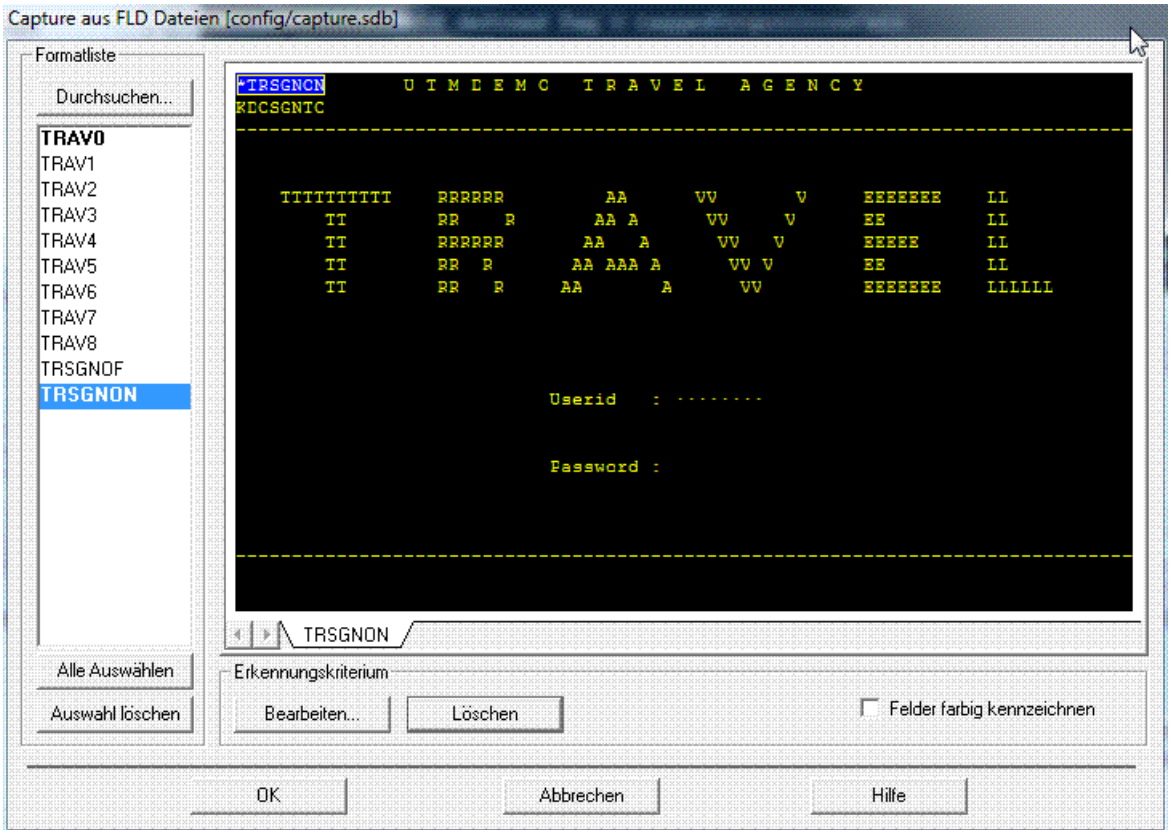
- ▶ In einem weiteren Dialogfeld geben Sie die Capture-Datenbank an, in der die Erkennungskriterien abgelegt werden sollen. Danach werden die FLD-Dateien und die Templates generiert.

i Wenn Sie zu einem früheren Zeitpunkt schon FLD-Dateien und Templates aus einer Formatbeschreibungquelle generiert haben, können Sie diese Schritte überspringen und die Erfassung der Erkennungskriterien direkt mit dem Befehl **Generieren/Capture/aus FLD-Dateien** starten.

Das Dialogfeld **Capture aus FLD Dateien** wird eingeblendet.



- ▶ Wählen Sie aus der Formatliste das erste Format, für das Sie Erkennungskriterien festlegen wollen. Das Format wird rechts im Dialogfeld angezeigt.

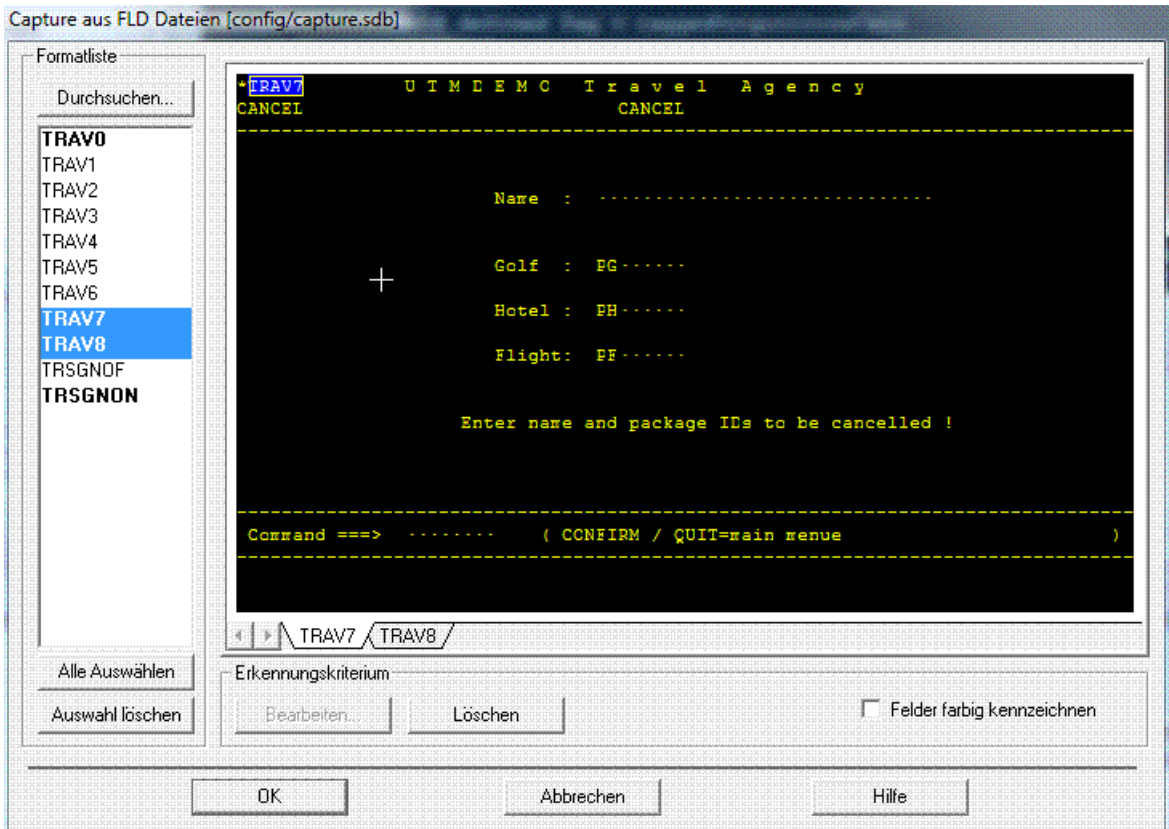


- ▶ Markieren Sie im angezeigten Format die Bereiche, die das Format eindeutig identifizieren. Markieren Sie ein oder mehrere Rechtecke durch Ziehen mit der Maus bei gedrückter Standard-Maustaste.

Sobald Sie für ein Format ein oder mehrere Erkennungskriterien markiert haben, werden diese in der Capture-Datenbank abgelegt und der Formatname in der **Formatliste** fett dargestellt. Im Gegensatz zur einzelnen Erfassung der Erkennungskriterien können Sie hier für die Formate keine eigenen Namen vergeben.

Sie können auch gleich für mehrere Formate die Erkennungskriterien erfassen.

- ▶ Markieren Sie dazu in der **Formatliste** im Dialogfeld **Capture aus FLD Dateien** die Formate, für die Sie Erkennungskriterien erfassen wollen. Das erste der markierten Formate wird im Dialogfeld angezeigt. Dahinter liegen in Registerkarten die weiteren markierten Formate zur Bearbeitung.



Dabei erhalten automatisch alle ausgewählten Formate dieselben Erkennungskriterien wie das erste Format. Zum Überprüfen können Sie die einzelnen Formate über den Karteikartenreiter anwählen.

7.1.3 Erkennungskriterium bearbeiten

Während des Capture-Verfahrens können Sie die einzelnen Erkennungskriterien bearbeiten. Wählen Sie dazu im Dialogfeld **Capture** bzw. **Capture aus FLD-Dateien** die Schaltfläche **Bearbeiten**. Das Dialogfeld **Capture 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.

7.1.4 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.

7.2 Individuelle Templates für Popup-Boxen

Host-Formate können Popup-Boxen enthalten.

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

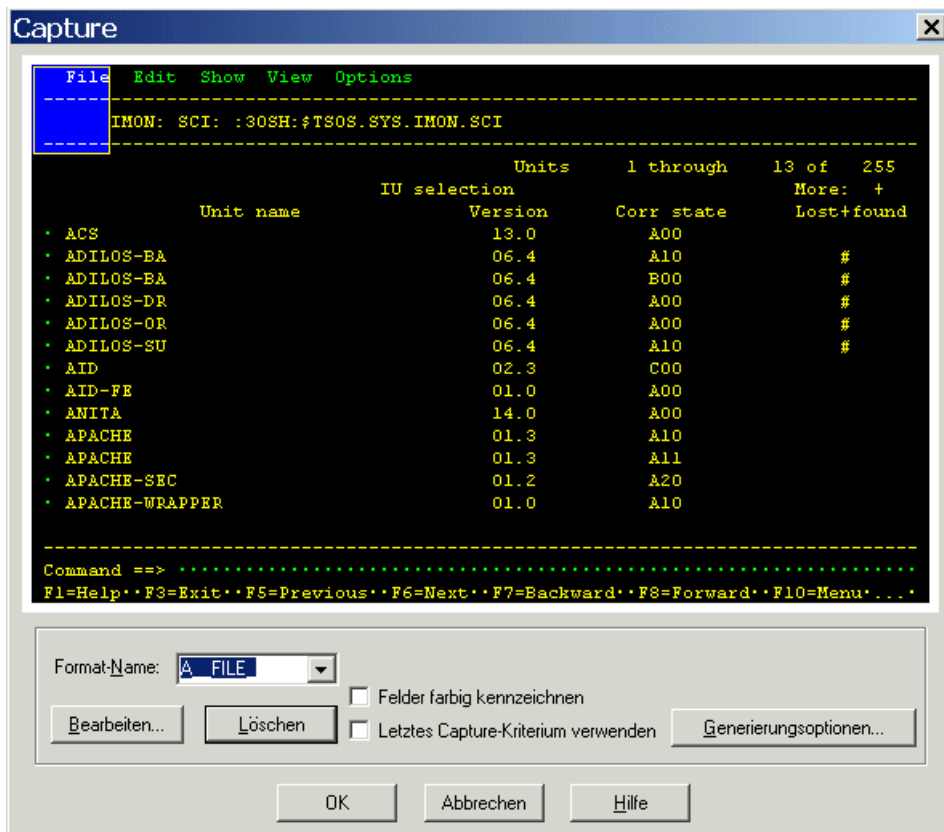
Für individuell angepasste Templates stellt WebTransactions Attribute des Systemobjekts für die Popup-Behandlung zur Verfügung. Falls zur Laufzeit der WebTransactions-Anwendung das Attribut `USE_POPUP_RECOGNITION` auf "Yes" gesetzt ist, werden diese Popups erkannt und als eigenständige Seiten an den 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 115](#).

Bevor im [Abschnitt „Templates für Popups generieren“ auf Seite 106](#) 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.

7.2.1 Ohne spezielle Popup-Behandlung: Identifikationsprobleme

Mit dem Capture-Verfahren von WebLab werden bestimmte Format-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 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 `AutomaskOSD.htm` darstellen.

Wenn Sie das Erkennungskriterium so wählen, dass es nicht überdeckt wird, kann Web-Transactions 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 bei der Capture-Funktion das Popup nicht angezeigt war, wäre das Popup im generierten Template nicht enthalten und würde im Browser nie angezeigt werden.

Falls bei der Capture-Funktion 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.

7.2.2 Templates für Popups generieren

Um die im vorhergehenden Abschnitt dargestellten Probleme zu vermeiden, bietet Web-Transactions 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 `wtcStartOSD` erzeugen (siehe [Abschnitt „WTBean wtcStartOSD.wtc zur Generierung eines Start-Template“ auf Seite 174](#)), können Sie dieses Attribut direkt im Start-Template setzen:

- ▶ Wählen Sie im Dialogfeld **Einfügen:wtcStartOSD** 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 `OSD_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.



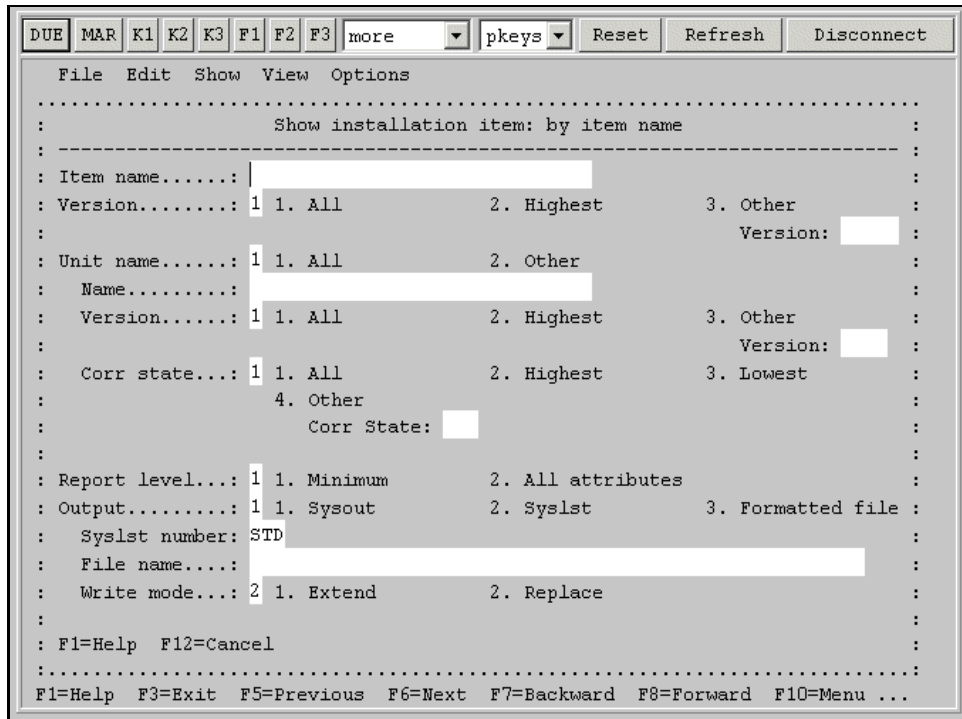
Beachten Sie, dass das Attribut in diesem Fall nur für die aktuelle Sitzung festgelegt ist. Es muss vor dem nächsten Start im Start-Template erneut gesetzt werden.

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 115](#). Diese Systemobjekt-Attribute sind auch für den Ablauf erforderlich.

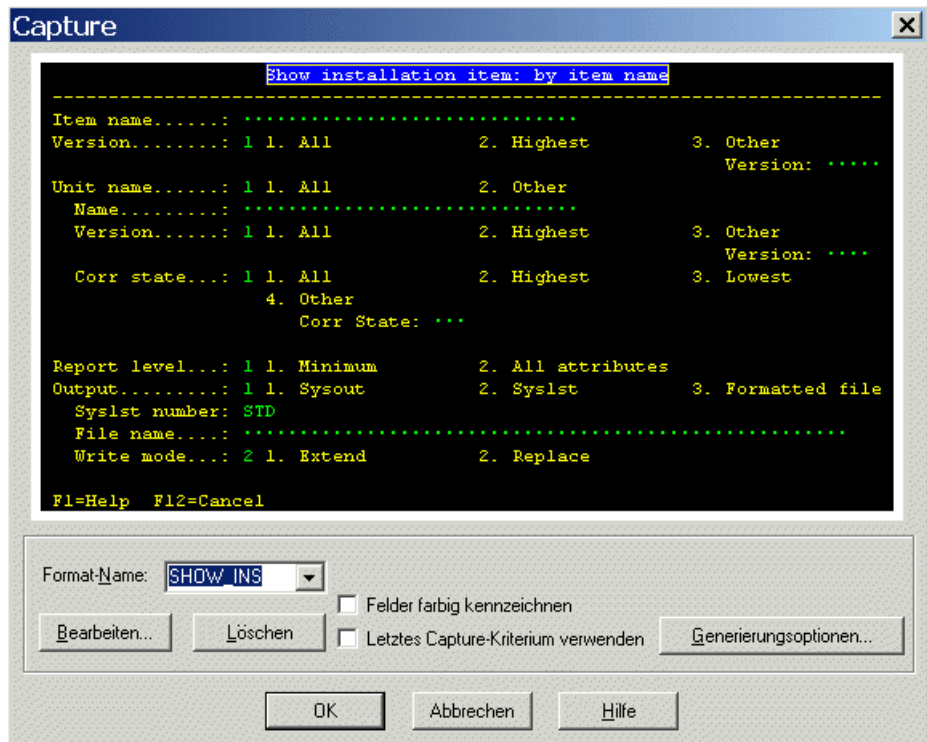
Vorgehen

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**, um die Popup-Box mit dem Capture-Verfahren zu erfassen. Das Dialogfeld **Capture** wird am Bildschirm mit der Popup-Box eingeblendet. Bitte beachten Sie hier den Unterschied zwischen dem Inhalt des gesamten Bildschirms und dem Inhalt des Bereiches der als Popup-Box erkannt wurde.



- ▶ 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:

The screenshot shows a window titled "Show installation item: by item name". It contains several form fields and options:

- Buttons: DUE, MAR, K1, K2, K3, F1, F2, F3, more (dropdown), pkeys (dropdown), Reset, Refresh, Disconnect.
- Item name: [text input]
- Version: 1. All, 2. Highest, 3. Other. Version: [text input]
- Unit name: 1. All, 2. Other
- Name: [text input]
- Version: 1. All, 2. Highest, 3. Other. Version: [text input]
- Corr state: 1. All, 2. Highest, 3. Lowest, 4. Other. Corr State: [text input]
- Report level: 1. Minimum, 2. All attributes
- Output: 1. Sysout, 2. Syslst, 3. Formatted file
- Syslst number: STD
- File name: [text input]
- Write mode: 2. 1. Extend, 2. Replace
- Footer: F1=Help F12=Cancel

- 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 `AutomaskOSD`).



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.

7.3 Sprechende Namen verwenden

Die Feldnamen eines Formats werden beim Capture-Verfahren generiert aus der Zeilen- und Spaltenposition des Feldes im Format und aus seiner Länge. Dies führt zu sehr abstrakten Namen, die wenig mit der Verwendung des Feldes und seinem Inhalt zu tun haben und den Programmierer bei der Bearbeitung des Templates daher kaum unterstützen.

Zusammen mit dem Capture-Verfahren können Sie auch eine so genannte Formatbeschreibungsource nutzen und so in die Templates die ursprünglichen Feldnamen generieren. Die Formatbeschreibungsource ist das Ergebnis eines IFG2FLD-Laufs, mit dem die Formatbeschreibungen aus der IFG-Bibliothek gelesen und in der Quelle abgelegt werden. Die Formatbeschreibungsource enthält alle Feldnamen, die der Entwickler der Host-Formate ursprünglich mit dem Formatierungssystem definiert hat. Falls die Feldnamen sich auf die Bedeutung des Feldes beziehen und somit „sprechender“ sind, können sie dem mit der Host-Anwendung vertrauten Entwickler eine wesentlich bessere Orientierungshilfe bieten.

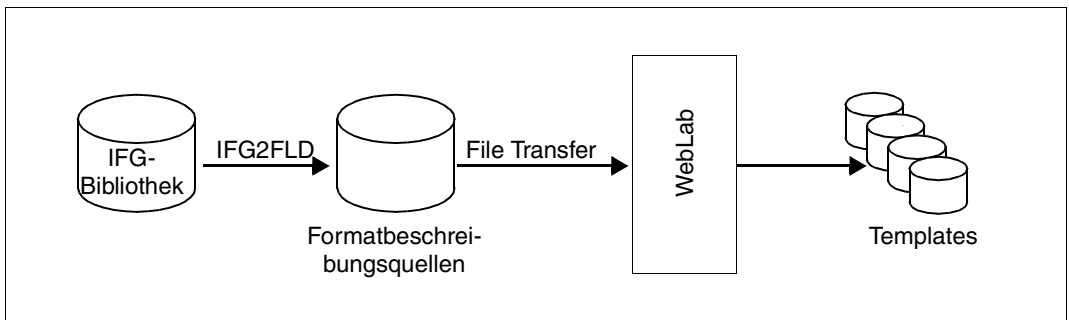


Bild 2: Templates aus einer Formatbeschreibungsource generieren

Die Formatbeschreibungsource können Sie beim Capture-Verfahren in WebLab angeben.

Vorgehen

Um bei der Template-Generierung sprechende Namen zu verwenden, gehen Sie folgendermaßen vor:

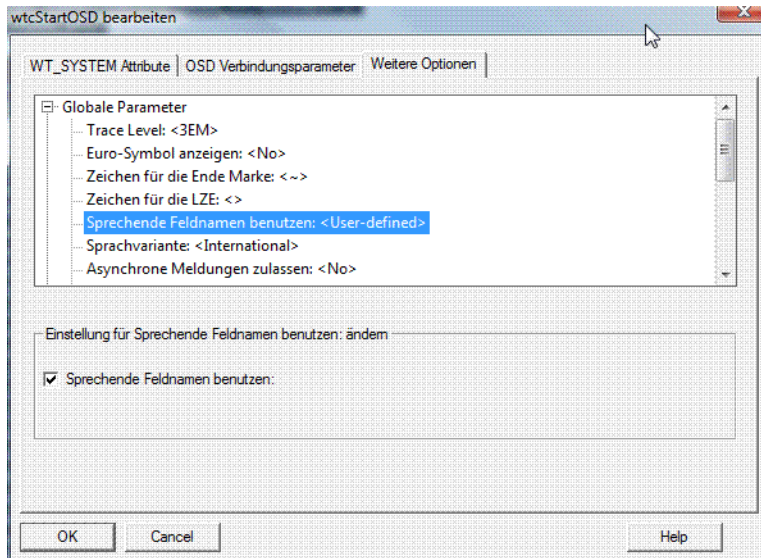
- ▶ Übertragen Sie eine der folgenden LMS-Bibliotheken binär auf den OSD-Rechner zu der Kennung, unter der die IFG-Bibliothek liegt.

Bibliothek	Bedeutung
WTifg2f1dFTP.lms	für die Übertragung mit FTP
WTifg2f1dopenFT.lms	für die Übertragung mit openFT oder mit dem Kommando TRANSFER-FILE im BS2000

- ▶ Loggen Sie sich am OSD-Rechner unter dieser Kennung ein.
- ▶ Führen Sie den IFG2FLD-Lauf durch wie im folgenden Abschnitt beschrieben.
- ▶ Übertragen Sie die Formatbeschreibungquelle im Textmodus auf den Rechner, auf dem WebLab läuft.
- ▶ Geben Sie beim Capture-Verfahren im Dialogfeld **Generierungsoptionen** die Formatbeschreibungquelle an.



Für den Ablauf müssen Sie das kommunikationsspezifische Systemobjekt-Attribut FIELD_NAMES auf den Wert User-defined setzen. Dies können Sie z.B. erreichen, indem Sie beim Erzeugen oder Bearbeiten eines anwendungsspezifischen Start-Templates auf der Registerkarte **Weitere Optionen** die Option **Globale Parameter/Sprechende Feldnamen benutzen** einschalten.



Anwendung von IFG2FLD

IFG2FLD ist in der OSD-LMS Bibliothek `WTifg2f1dFTP.lms` bzw. `WTifg2f1d0penFT.lms` enthalten. Diese Bibliotheken finden Sie im Verzeichnis `install_dir/lib`.

IFG2FLD ist ein OSD-Programm, deshalb müssen Sie die entsprechende Bibliothek zu einer OSD-Benutzererkennung übertragen. Der Ziel-Name der Bibliothek im BS2000 ist abhängig davon, welche Übertragungsart Sie wählen, entweder `WTIFG2FLD.LMS` oder `WTIFG2FLDOPENFT.LMS`. Anschließend können Sie IFG2FLD aus der übertragenen Bibliothek mit dem folgendem Kommando starten:

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

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

Kommando	Beschreibung
<code>MAPFILE IFG-Bibliothek</code>	weist die zu bearbeitende IFG-Bibliothek zu.
<code>PROFILE Benutzerprofil</code>	weist ein Benutzerprofil für die Konvertierung zu. Jede IFG-Bibliothek enthält zumindest ein Benutzerprofil um ein Format verarbeiten zu können. Das Benutzerprofil ist eine Menge von Standardwerten für die Handhabung des IFG, für die Eigenschaften der Formate, für die Eigenschaften der Felder des Formats und für die Eigenschaften, die die Programmierung betreffen.
<code>CONVERT Formatname [/version /ALL]</code>	nimmt das angegebene IFG-Format in die Formatbeschreibungsource auf. Optional können Sie die Version des Formats mit angeben. Wenn Sie die Version nicht angeben, wird das Format mit der höchsten Version aufgenommen. Wenn Sie statt einer Version den Wert <code>/ALL</code> angeben, werden alle Versionen des Formats in die Formatbeschreibungsource aufgenommen.
<code>CONVERT *ALL [/version /ALL]</code>	nimmt alle IFG-Formate in der IFG-Bibliothek in die Formatbeschreibungsource auf. Optional können Sie die Version der Formate mit angeben. Es werden dann nur die Formate dieses Versionsstandes aufgenommen. Wenn Sie die Version nicht angeben, werden die Formate der höchsten Version in die Formatbeschreibungsource aufgenommen. Wenn Sie statt einer Version den Wert <code>/ALL</code> angeben, werden alle Formate aller Versionen aufgenommen.
<code>END</code>	erzeugt die Formatbeschreibungsource und beendet IFG2FLD.

Beispiel

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

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

8 Kommunikation steuern

8.1 Systemobjekt-Attribute

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

Hier werden nur diejenigen Attribute beschrieben, die es speziell für OSD-Anbindungen gibt oder die zumindest für OSD-Anbindungen 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` („verbindungspezifisches 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 zum verbindungspezifischen und globalen Systemobjekt finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

8.1.1 Übersicht

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

Die Systemobjekt-Attribute können in folgende Kategorien eingeteilt werden:

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



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

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: <i>application_prefix@formatname . fld</i> bzw. <i>application_prefix@formatname . htm</i>	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: <i>AutomaskOSD</i> . Falls kein Erkennungskriterium gefunden und der Inhalt des Attributs <i>AUTOMASK</i> übernommen wurde, aber das dort genannte Template nicht vorhanden ist, wird das in <i>DEFAULT_FORMAT</i> hinterlegte Template verwendet. Falls zwar ein Erkennungskriterium gefunden wird, aber kein entsprechendes Template, wird ebenfalls das in <i>DEFAULT_FORMAT</i> hinterlegte Template verwendet (siehe WebTransactions-Handbuch „Konzepte und Funktionen“).	t


Attributname	Bedeutung	Erläuterung/Kategorie	
AUTOTAB	Automatischer Sprung der Schreibmarke in das folgende Feld bei Erreichen des Feldendes	Bei 9750-Terminals kann die Host-Anwendung die Reaktion der Schreibmarke bei Erreichen des Feldendes bestimmen. Diese Eigenschaft wird mit dem Attribut AUTOTAB der WebTransactions-Anwendung zur Verfügung gestellt. Das Attribut kann, abhängig vom aktuellen AUTOTAB-Zustand des Terminals, die Werte Yes und No annehmen. Das Automask-Template und alle mit dem Master-Template OSD.wmt generierten Templates werten dieses Attribut aus.	c
BYPASS	Bypass-Druckdatei Flag	Dieses Attribut wird von WebTransactions auf Yes gesetzt, wenn beim Aufruf von receive eine Bypass-Druckdatei aufbereitet wurde. 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 (... \Capture.sdb). Der Name kann als absoluter Pfadname oder als relativer Pfadname (in Bezug auf das Basisverzeichnis) angegeben werden, z.B.: – absolute Angabe: C:\osdappli1\config1\dialog.sdb – relative Angabe (Basisverzeichnis = osdappli): config1\dialog.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
COMMUNICATION_INTERFACE_VERSION	Schnittstellenversion	Globales WT_SYSTEM-Attribut: – 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 <u>nicht</u> 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_MESSAGE	Kurznachricht für Verbindungsaufbau	Maximal 80 Byte lange Kurznachricht für den Verbindungsaufbau, optional. Die Kurznachricht ist Teil des Connection Letters, der beim Verbindungsaufbau an den Partner übergeben wird. Eine Kurznachricht kann entweder als String oder hexadezimal angegeben werden: <i>host_system . CONNECTION_MESSAGE=C 'cccc'</i> oder <i>host_system . CONNECTION_MESSAGE=X 'xxxxxxxx'</i> Dieses Attribut kann z.B. dazu verwendet werden, ein eventuell benötigtes Zugriffs- oder Netzwerk-Passwort anzugeben.	o
CONNECTION_PASSWORD	Passwort für Verbindungsaufbau	Maximal 4 Byte langes Passwort für den Verbindungsaufbau zu DCAM-Anwendungen, optional. Dieses Passwort ist Teil des Connection Letters, der beim Verbindungsaufbau an den Partner übergeben wird. Das Passwort muss in BCAM mit XSTAT APPPW= generiert sein und kann entweder als String oder hexadezimal angegeben werden: <i>host_system . CONNECTION_PASSWORD=C 'cccc'</i> oder <i>host_system . CONNECTION_PASSWORD=X 'xxxxxxxx'</i>	o
DISCONNECT	Template für den Verbindungsabbau	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: <i>wtstart</i>	t
DISPLAY_EURO	Euro-Symbol anzeigen	Wenn dieses Attribut auf Yes gesetzt ist, dann wird das Zeichen, das dem Code X'A4' der ISO-8859-Codetabelle entspricht, als Euro-Zeichen ausgegeben. Bei DISPLAY_EURO=No (Voreinstellung) wird bei X'A4' das Währungssymbol (¤) angezeigt. DISPLAY_EURO=YES wirkt nur dann, wenn das Attribut TERMINAL_TYPE auf 9763 und das Attribut HOST_CHARSET auf ISO-8859-1, ISO-8859-2 oder ISO-8859-9 gesetzt ist.	t

Attributname	Bedeutung	Erläuterung/Kategorie	
END_MARK	Ersatzzeichen für die Endemarke	Vorbelegung: ~ (Tilde) Es kann ein Ersatzzeichen festgelegt werden. Es sind Zeichen außerhalb des Standard-Zeichensatzes möglich. Wenn der Terminal-Typ für die Emulation 9763-UNICODE ist, können Sie für das Attribut END_MARK am kommunikationsspezifischen System-Objekt auch Zeichen aus dem Unicode-Vorrat bis maximal U+FFFF angeben. Geben Sie diese als HTML-Fluchtsequenz an (&#lt;d>; oder &#x<x>;). Sinnvoller Wert: ◄ (◀)	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 verarbeitet 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	<p>Das Attribut enthält den Namen eines Templates (ohne die Endung '.htm'). Ist das Attribut definiert, so wird das entsprechende Template am Ende der generierten Templates inkludiert. Voreinstellung: keine Inkludierung</p> <p> Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter.</p> <p>Siehe auch PROLOG und FORMTPL</p>
FIELD_NAMES	IFG-Feldnamen benutzen	<p>Wenn dieses Attribut auf <code>User-defined</code> gesetzt ist, dann können die Feldnamen der IFG-Bibliothek benutzt werden, andernfalls müssen die von <code>WebTransactions</code> erzeugten Feldnamen verwendet werden. Mögliche Werte: <code>User-defined</code>, <code>Generic</code> Voreinstellung: <code>User-defined</code></p> <p> Beachten Sie, dass in den ausgelieferten Start-Templates und Beans <code>Generic</code> als Voreinstellung implementiert ist.</p>

Attributname	Bedeutung	Erläuterung/Kategorie
FIRST_IO_TIMEOUT	Timer für den receive-Aufruf	<p>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")</p> <p>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.</p> <p>Ist FIRST_IO_TIMEOUT größer als das globale Systemobjekt-Attribut TIMEOUT_APPLICATION, wird ein von TIMEOUT_APPLICATION abgeleiteter Wert verwendet:</p> <ul style="list-style-type: none"> - 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 <p>Der Wert ist in Sekunden, evtl mit einem Dezimaltrenner (Punkt oder Komma) anzugeben. Kleinste Einheit ist 500 Millisekunden.</p>
FLD	Format-Name	<p>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), dann wird FLD auf den Wert von AUTOMASK gesetzt (wird stets vom receive gesetzt).</p> <p>Siehe auch FORMAT</p>

Attributname	Bedeutung	Erläuterung/Kategorie	
FORMTPL	Formularfelder	<p>Das Attribut enthält den Namen eines Templates (ohne die Endung '.htm'). Ist das Attribut definiert, so wird das entsprechende Template am Anfang des <code>wtDataForm</code> in den generierten Templates inkludiert.</p> <p>Voreinstellung: keine Inkludierung</p> <p> Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter.</p> <p>Siehe auch <code>PROLOG</code> und <code>EPILOG</code>.</p>	t
HARDCOPY	Hardcopy-Druckdatei Flag	<p>Dieses Attribut wird von <code>WebTransactions</code> auf <code>Yes</code> gesetzt, wenn beim Aufruf von <code>receive</code> eine Hardcopy-Druckdatei aufbereitet wurde.</p> <p>Vor der Auswertung von <code>\$MESSAGE.PRINTING</code> sollte das Attribut vom Template wieder auf <code>No</code> gesetzt werden, wie im Template <code>wtasync.htm</code>, das von <code>WebTransactions</code> bereitgestellt wird.</p>	c, t


Attributname	Bedeutung	Erläuterung/Kategorie	
HOST_CHARSET	Zeichensatz der Host-Anwendung	<p>Der Host-Adapter versorgt dieses Attribut mit einer Angabe zum Zeichensatz, mit dem die OSD-Host-Anwendung arbeitet. Folgende Werte sind möglich:</p> <p>ISO-8859-1: verschiedene westeuropäische Sprachen wie Englisch, Deutsch, Französisch, auch Latin-1 genannt</p> <p>ISO-8859-2: verschiedene mitteleuropäische Sprachen, auch Latin-2 genannt</p> <p>ISO-8859-5: Kyrillisch</p> <p>ISO-8859-7: Griechisch</p> <p>ISO-8859-9: Türkisch</p> <p>UTF-8</p> <p>Wenn die Host-Anwendung in den Unicode-Modus umschaltet, interpretiert WebTransactions for OSD</p> <ul style="list-style-type: none"> – die Feldinhalte vom und zum BS2000/OSD als UTF-EBCDIC. – die entsprechenden Inhalte der Host-Objekte bei der Zuweisung als UTF-8-codiert und gibt sie bei der Auswertung UTF-8-codiert aus. <p>Alle Templates, die WebTransactions für diesen Host-Adapter generiert, weisen das Systemobjekt-Attribut HOST_CHARSET auf das globale Systemobjekt-Attribut CHARSET zu. Daher wird das Feld Content-Type im HTTP-Header richtig belegt und der Browser kann die Daten korrekt interpretieren (siehe Abschnitt „Unicode-Unterstützung“ auf Seite 10 und WebTransactions-Handbuch „Konzepte und Funktionen“).</p>	c, r
HOST_NAME	Name des Host-Rechners	<p>Name oder Internet-Adresse des Host-Rechners. Wenn Sie einen symbolischen Namen verwenden, muss dieser Name entweder lokal in der HOSTS-Datei oder mittels Domain Name Service (DNS) definiert sein. Bei Ablauf auf der Plattform BS2000/OSD sind ausschließlich die maximal 8 Zeichen langen BCAM-Namen erlaubt.</p>	o

Attributname	Bedeutung	Erläuterung/Kategorie
IGNORE_ASYNC	ASYNC-Bedingung ignorieren	<p>Normalerweise (IGNORE_ASYNC='NO') wird bei dem Methodenaufruf <code>send</code> 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 <code>Yes</code> gesetzt und WT_KEY.Key auf <code>Refresh</code>. Der folgende Methodenaufruf <code>receive</code> übernimmt die bereits vorliegenden Host-Daten und wartet nicht mit FIRST_IO_TIMEOUT und MULTIPLE_IO_TIMEOUT auf weitere Daten.</p> <p>Wenn IGNORE_ASYNC auf <code>Yes</code> gesetzt ist, wird unabhängig von inzwischen asynchron eingegangenen Daten gesendet. Manuelles Wiederholen des <code>send</code>-Aufrufes wird dadurch vermieden. WT_KEY.KEY bleibt unverändert, REFRESH_BY_ASYNC wird nicht auf <code>Yes</code> gesetzt, denn es wurde kein Refresh durchgeführt. IGNORE_ASYNC unterdrückt somit auch die Erkennung der asynchronen Nachricht.</p> <p>Voreinstellung ist NO. Siehe auch MULTIPLE_IO_TIMEOUT, REFRESH_BY_ASYNC</p>
LOCAL_PORT	Nummer des lokalen Ports	<p>Ist dieses Attribut angegeben, so wird dieser Port von der Socketverbindung als Ausgangsport auf dem lokalen System verwendet.</p>

Attributname	Bedeutung	Erläuterung/Kategorie
LZE_CHAR	Zeichen für die Darstellung des logischen Zeilen-Endes (LZE)	<p>Auf einem realen Terminal vom Typ 9750 oder einer entsprechenden Emulation kann die Host-Anwendung LZE-Zeichen ausgeben, die erst bei der Eingabe von Daten an die Host-Anwendung von Bedeutung sind. Damit lassen sich z.B. mehrere BS2000/OSD-Kommandos mit einer Datenfreigabe eingeben, die dann der Reihe nach ausgeführt werden.</p> <p>Wenn mit dem Attribut LZE_CHAR ein Zeichen vereinbart wird, kann WebTransactions diese Funktionalität unterstützen.</p> <p>Stellen Sie ein Zeichen ein, das ansonsten in der Host-Anwendung nicht benutzt wird, siehe auch END_MARK. Es sind Zeichen außerhalb des Standard-Zeichensatzes möglich.</p> <p>Wenn der Terminal-Typ für die Emulation 9763-UNICODE ist, können Sie für das Attribut LZE_CHAR am kommunikationsspezifischen System-Objekt auch Zeichen aus dem Unicode-Vorrat bis maximal U+FFFF angeben. Geben Sie diese als HTML-Fluchtsequenz an (&#x2039; oder &#x2039;).</p> <p>Sinnvoller Wert: &#x2039; (<)</p> <p>Voreinstellung: leer</p>

Attributname	Bedeutung	Erläuterung/Kategorie
MULTIPLE_IO_TIMEOUT	Timeout für vollständiges Bildschirmformat	<p>Vorbelegung: 2 (Sekunden)</p> <p>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.</p> <p>Die Empfangsverarbeitung wird jedoch beendet, sobald WebTransactions auf Grund des verwendeten Protokolls feststellen kann, dass die Host-Anwendung auf Eingabe wartet.</p> <p>Wenn Sie mit Druck-/Asynchron-Unterstützung arbeiten (siehe Seite 185ff), dann wird in zyklischen Abständen automatisch ein Refresh durchgeführt, sodass noch unvollständige Formate beim nächsten Refresh vollständig aufgebaut sein können.</p> <p>Ist sichergestellt, dass die Host-Anwendung keine Teilnachrichten benutzt, kann der Wert auf 0 gesetzt werden.</p> <p>Ziel ist es, grundsätzlich ohne MULTIPLE_IO_TIMEOUT das Schirmende zu erkennen, da dieser Timeout</p> <ul style="list-style-type: none"> – bei jedem Dialogschritt die verstreichende Zeit erhöht – groß genug eingestellt werden muss, um bei starker Last nicht fälschlicherweise zu früh das Schirmende anzunehmen. <p>Auf Basis des Transportprotokolls NEABT kann bei fast allen OSD-Host-Anwendungen (Ausnahme: z.B. Verbindung über OMNIS) erkannt werden, dass der Host wieder auf Eingabe wartet. Wenn dieses Protokoll-Element eintrifft, wird das weitere Warten abgebrochen.</p> <p>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.</p> <p>Siehe auch Attribute END_WAIT_CONDITION</p>

Attributname	Bedeutung	Erläuterung/Kategorie	
NATIONAL_VARIANT	Sprachvariante für Kommunikation zwischen WebTransactions und Host-Anwendung	Dieses Attribut spezifiziert die Sprachvariante des 7-Bit-Zeichensatzes. Mögliche Werte: <ul style="list-style-type: none"> – Danish – English (UK) – English (USA) – French – French-Belgian – German – International – Italian – Norwegian – Spanish – Swedish – Swiss 	t
NIL_MODE	Bei der Ausgabe von Host-Objekten NIL und SPACE unterschiedlich behandeln	Auf einem realen Terminal vom Typ 9750 oder einer entsprechenden Emulation werden Feldinhalte, die eine binäre Null (NIL) enthalten, mit einem mittigen Punkt dargestellt. WebTransactions ersetzt diese Zeichen bei den VALUE-Host-Objekt-Attributen (VALUE, HTMLVALUE, RAWVALUE) durch ein Leerzeichen, um die Ausgabe „browser-gerecht“ zu machen. Dadurch ist es nicht mehr möglich, bei der Eingabe zu unterscheiden, ob die SPACE's wirklich als SPACE gespeichert werden sollen, oder ob ein im Bildschirmpuffer existierendes NIL bestehen bleiben soll. Wenn das Attribut NIL_MODE auf true gesetzt ist, werden die NIL's nicht mehr durch SPACE ersetzt, sondern durch das Zeichen mit dem Code 0xB7 dargestellt, das in vielen Zeichensätzen als „Mittelpunkt“ vorhanden ist. Damit ist eine eindeutige Unterscheidung zwischen SPACE und NIL möglich. Das Attribut wird mit jedem receive wirksam, d.h. die Attribut-Änderung zwischen zwei Host-Objekt-Auswertungen hat keine Wirkung. Voreinstellung: false	t
OFFLINE_COMMUNICATION	Schalter zum Abspielen eines Tracefiles	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

Attributname	Bedeutung	Erläuterung/Kategorie	
OFFLINE_LOGFILE	Dateiname des zu erzeugenden Emulations-Trace	Dateiname des Emulations-Trace, der aufgezeichnet werden soll Default: WEBTADUMP.LOG andere Werte möglich siehe auch RECORD_HOST_COMMUNICATION	o
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 ein Leerzeichen enthält, dann werden Eingabefelder mit Leerzeichen gefüllt, andernfalls wird das Nilzeichen (0) als Füllzeichen verwendet (Voreinstellung)	t
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  Das Attribut wird nur vom generierten Standard-Template ausgewertet, nicht vom Host-Adapter. Siehe auch EPILOG und FORMTPL.	t
POPUP.COLUMN	Anfangsspalte des empfangenen Popups	Falls bei einem Aufruf von <code>receive</code> 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) - <i>Maximale Spaltenanzahl</i> Voreinstellung: 0.	c, r
POPUP.HEIGHT	Höhe des empfangenen Popups	Falls bei einem Aufruf von <code>receive</code> 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) - <i>Maximale Zeilenanzahl (inkl. Rahmen)</i> 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. Voreinstellung 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

Attributname	Bedeutung	Erläuterung/Kategorie	
POPUP.HMIDDLE	Parameter für die Erkennung von Popups	Enthält die verschiedenen Zeichen aus denen der horizontale Rahmen eines Popup-Fensters gebildet sein kann. Voreinstellung 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. Voreinstellung 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
POPUP.LINE	Zeilennummer der linken oberen Ecke des empfangenen Popups	Falls bei einem Aufruf von <code>receive</code> 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) - <i>Maximale Zeilenanzahl</i> 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. Voreinstellung 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. Voreinstellung 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. Voreinstellung ist ein Doppelpunkt „:.“.	t
POPUP.WIDTH	Breite des empfangenen Popups	Falls bei einem Aufruf von <code>receive</code> 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: 0 (falls kein Popup vorliegt) - <i>Maximale Spaltenanzahl</i> (inkl. Rahmen) Voreinstellung: 0.	c, r
PORT_NUMBER	Portnummer	Portnummer für die Kommunikation mit der OSD-Host-Anwendung über TCP/IP Voreinstellung: 102	o

Attributname	Bedeutung	Erläuterung/Kategorie	
RECEIVED_BLOCKS	Nachrichtenzahl	Anzahl der während des letzten <code>receive</code> -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 <code>MULTIPLE_IO_TIMEOUT</code> 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 <code>END_WAIT_CONDITION.EXPECTED_BLOCKS</code> auch verwendet werden, um die Wartezeiten wegen <code>MULTIPLE_IO_TIMEOUT</code> abzuberechnen.	c, r
RECORD_HOST_COMMUNICATION	Schalter für Emulations-Trace	Wert "Yes": Schaltet den Emulations-Trace ein Default: "No" siehe auch <code>OFFLINE_TRACEFILE</code>	o
REFRESH_BY_ASYNC	Automatisches Setzen von Refresh, wenn Asynchron-Nachrichten eingetroffen sind.	Falls dieses Attribut auf Yes gesetzt ist, wurde das Host-Steuerobjekt <code>WT_KEY.Key</code> während des letzten <code>send</code> automatisch auf den Wert <code>Refresh</code> gesetzt, weil eine Asynchron-Nachricht seit dem letzten <code>receive</code> eingetroffen ist. Damit wurde beim letzten Aufruf der Form <code>send</code> bzw. <code>receive</code> das alte Format aktualisiert. D.h. es kam <u>nicht</u> 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 <code>REFRESH_BY_ASYNC</code> - einen Hinweis für den Benutzer generieren. No (Vorbelegung) heißt, dass der Inhalt der Eingabefelder des Formats und der Wert von <code>WT_KEY.Key</code> unverändert zur Host-Anwendung gesendet wurde (und damit auch der Inhalt des Bildschirmpuffers). Siehe auch <code>IGNORE_ASYNC</code>	c
STATION_NAME	Stationsname	Stationsname, der zum Verbindungsaufbau mit der OSD-Host-Anwendung verwendet wird. Dieses Attribut müssen Sie nur dann mit einem Wert belegen, wenn die OSD-Anwendung spezielle Stationsnamen verwendet. Ist kein Wert angegeben, generiert WebTransactions automatisch einen Stationsnamen und liefert den generierten Namen in diesem Attribut zurück. Max. Länge: 8	o

Attributname	Bedeutung	Erläuterung/Kategorie	
SYM_DEST	Name der OSD-Host-Anwendung auf dem Host-Rechner	z.B. \$DIALOG	o
TERMINAL_TYPE	Terminaltyp	Terminaltyp wie er von WebTransactions emuliert wird. Mögliche Werte: 9750 (Vorbelegung) 9755 9763 (falls 8-Bit-Zeichensatz verwendet wird) 9763-UNICODE 9763-UNICODE schaltet die Unicode-Unterstützung ein. Ist dieser Wert gesetzt, gibt sich die in WebTransactions for OSD integrierte Emulation beim Verbindungsaufbau zu BS2000/OSD als Unicode-fähig zu erkennen. Die OSD-Host-Anwendung kann nun diese Fähigkeit nutzen. Weitere Details zur Unicode-Unterstützung siehe Seite 74 .	o
TRACE_LEVEL	Tracelevel	Steuert den Inhalt des WebTransactions-Tracefiles. Mögliche Werte: 0, 1, 2, 3, [E], [M], [L] 0...3 werden numerisch ausgewertet (\geq). 3 schließt 2 und 1 mit ein. 0 schließt alle Traces mit numerischem Tracelevel aus. E, M und L sind eigene Level, die unabhängig von 0...3 gefiltert werden. Dabei bedeuten: <ul style="list-style-type: none"> - 0, 1, 2, 3 unterschiedliche Tracelevels - E Ausgabe der Aufrufe der Emulationsfunktionen - M Ausgabe aller Host-Matrizen, d.h. der „rohen“ Maskendaten - L Ausgabe in eine Protokoll-Datei Voreinstellung: 3EM (= Maximal-Trace)	t
USE_POPUP_RECOGNITION	Flag für die Popup Erkennung	Ist auf Yes zu setzen, falls Popup-Fenster erkannt werden sollen. Voreinstellung: No.	t
WT_ASYNC	Ausdruck und asynchrone Meldungen	Ist auf Yes zu setzen, falls Ausdruckfunktionen und asynchrone Meldungen unterstützt werden sollen. Voreinstellung: No.	t

Attributname	Bedeutung	Erläuterung/Kategorie	
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 199). 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
WT_BROWSER_PRINT_OPTIONS.ORIENTATION	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Gibt die Seitenorientierung an, nur bei Verwendung des Internet Explorers. 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 Explorers (siehe Abschnitt „Variablen in Kopf- und Fußzeile“ auf Seite 201)	t
WT_BROWSER_PRINT_OPTIONS.FOOTER	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Text für die Fußzeile, nur bei Verwendung des Internet Explorers (siehe Abschnitt „Variablen in Kopf- und Fußzeile“ auf Seite 201)	t
WT_BROWSER_PRINT_OPTIONS.LEFT	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Linker Rand in mm, nur bei Verwendung des Internet Explorers	t
WT_BROWSER_PRINT_OPTIONS.RIGHT	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Rechter Rand in mm, nur bei Verwendung des Internet Explorers	t
WT_BROWSER_PRINT_OPTIONS.TOP	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Oberer Rand in mm, nur bei Verwendung des Internet Explorers	t
WT_BROWSER_PRINT_OPTIONS.BOTTOM	Steuerung des Browser-Drucks nur für Browser-Plattform Windows	Unterer Rand in mm, nur bei Verwendung des Internet Explorers	t

8.1.2 Zusammenspiel von Systemobjekt-Attributen und Methoden

Dieser Abschnitt informiert Sie darüber, welche OSD-spezifischen Attribute des Systemobjekts bei welchen Aktionen bzw. 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, bestimmen die folgenden Attribute des kommunikationsspezifischen Systemobjekts, die Sie z.B. im Start-Template setzen können:

System-Objekt-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: <i>application_prefix@formatname.flid</i>
HOST_NAME	Angabe des Rechnernamens oder seiner IP-Adresse
LOCAL_PORT	Nummer des lokalen Ports
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	Angabe der Portnummer für die OSD-Host-Anwendung
RECORD_HOST_COMMUNICATION	Schalter für Emulations-Trace
STATION_NAME	Angabe des Stationsnamens, falls die OSD-Host-Anwendung Stationsnamen verwendet
SYM_DEST	Angabe des Host-Anwendungsnamens
TERMINAL_TYPE	Angabe des Terminaltyps wie er von WebTransactions simuliert wird

Ein Aufruf von `open` schließt zuerst eine evtl. noch bestehende Verbindung. Anschließend wird die neue Verbindung aufgebaut.

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 Hostobjekt-Attribut `WT_KEY.Key` gesteuert. Die Templates versorgen `WT_KEY.Key` mit einem Wert für die Funktionen, die `wtKeysOSD.htm` zur Verfügung stellt (siehe [Abschnitt „Template wtKeysOSD.htm“ auf Seite 90](#)). 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 Hostobjekt-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 verbindungspezifischen 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 121 und „[MULTIPLE_IO_TIMEOUT](#)“ auf Seite 126 sowie der Attribute der `END_WAIT_CONDITION`-Gruppe ab Seite 119.

8.2 Host-Objekte

WebTransactions for OSD verwendet zwei Arten von Host-Objekten:

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

8.2.1 Host-Datenobjekte

Für den Datenaustausch zwischen WebTransactions und Host-Anwendung gibt es Host-Datenobjekte, die WebTransactions erzeugt, wenn ein neues Bildschirmformat (eventuell 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. Dabei können die Objekte über generische Namen oder über IFG-Feldnamen identifiziert werden.

Generische Namen für Host-Datenobjekte

Generische Namen werden immer vergeben. Wenn das verbindungspezifische Systemobjekt-Attribut `FIELD_NAMES` den Wert `User-defined` enthält, werden zusätzlich die mit dem IFG definierten Namen verwendet, siehe [Seite 136](#).

Für die generischen Namen ist für die Benennung dieser Objekte ihre Position im Bildschirmformat maßgebend, bestimmt nach Zeile und Spalte:

`E_yy_xxx_III` (für die Felder eines Bildschirmformats)

`F_yy_xxx_III` (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)
III	steht für die 3-stellige Anzahl von Bildschirmzeichen in einer Bildschirmzeile ab der Position <code>yy_xxx</code>

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 `///` 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 Feldgrenzen hinaus erstreckt, wird es bei Bildschirmformaten bei Spalte 80 abgeschnitten, bei Popups wird bei der letzten Spalte des Popups abgeschnitten.

Verkürzte Angabe von generischen Host-Datenobjekten

Sie können die Objektnamen verkürzt angeben, indem Sie die Längenangabe oder die führenden Nullen weglassen. Werden diese Möglichkeiten genutzt, ist allerdings ein genauer Überblick über die Verwendung der Elementnamen notwendig: So liefert z.B. `WT_FOCUS` den Elementnamen immer in der vollständigen Schreibweise. Die erste der folgenden Abfragebedingungen hätte also nie den Wert `TRUE`:

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

IFG-Namen von Host-Datenobjekten

Sie können in den Templates auch mit den (sprechenden) IFG-Namen an Stelle der generischen Namen arbeiten. Voraussetzung dafür ist,

- dass das kommunikationsspezifische Systemobjekt-Attribut `FIELD_NAMES` den Wert `User-defined` enthält
- und dass Sie zuvor mit WebLab zu einem Format eine `FLD-Datei` erzeugt und dabei die Formatbeschreibungsquelle verwendet haben. Diese müssen Sie beim Capture unter **Generierungsoptionen** angeben.

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_III oder F_yy_xxx_III	VALUE	<p>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 Hochkomma sowie kaufmännisches Und (&) werden für die Ausgabe in HTML umgesetzt.</p> <p>Bei Eingabefeldern (siehe Type Unprotected) kann der Inhalt des Attributs VALUE geändert werden. Das Verändern von VALUE bildet die Tastatureingabe eines Benutzers an einem Terminal ab.</p> <p>Unicode-Unterstützung Die folgenden Hinweise gelten auch für die Attribute HTMLVALUE und RAWVALUE.</p> <p><i>Unicode bei Zeichenkettenoperationen</i> Da der WebTransactions-Kern selbst Unicode nicht unterstützt, sollten Sie Operationen mit Zeichenketten nur vorsichtig einsetzen. Zeichenkettenoperationen können fehlerhafte Ergebnisse produzieren, wenn in der Zeichenkette Zeichen in UTF-8 enthalten sind. Das Attribut length liefert nicht die Anzahl der Zeichen, sondern die Anzahl der Bytes. Bei Vergleichen und Manipulationen mit solchen Zeichenketten ist also die Repräsentation der UTF-8-Zeichen zu berücksichtigen. In den Templates, die WebTransactions generiert, liegen solche Operationen nicht vor. Wird von einem Feld am Bildschirm nur eine Teilzeichenkette benötigt, können Sie dafür die Möglichkeiten von WebTransactions for OSD nutzen (siehe „Generische Namen für Host-Datenobjekte“ auf Seite 135).</p> <p><i>Unicode in der Diagnose</i></p> <ul style="list-style-type: none"> – In Traces und in der Online-Anzeige von Host-Objekten werden Unicode-Zeichen in einer Ersatzdarstellung angezeigt. – Host-Objekte können in WebLab nur mit 8-Bit-Zeichen überschrieben werden.
	HTMLVALUE	<p>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 Siehe auch Hinweise zu Unicode-Unterstützung oben.</p>

Objektname	Attributname	Bedeutung des Attributs
	RAWVALUE	Der Inhalt des Feldes wird als Sequenz von 8-Bit-Zeichen ohne Konvertierung zurückgegeben; lediglich binäre Nullen werden in Leerzeichen umgewandelt. Siehe auch Hinweise zu „Unicode-Unterstützung“ auf Seite 137 .
	STARTLINE	Zeile, in der das repräsentierte Bildschirmfeld beginnt, mögliche Werte: $1 \leq n \leq \text{letzte Bildschirmzeile}$ Diese hängt vom jeweiligen Bildschirmtyp ab: 24x80: $n \leq 25$ 32x80: $n \leq 33$ 43x80: $n \leq 44$ 27x132: $n \leq 28$
	NAME	Name des Feldes.
	STARTCOLUMN	Spalte, in der das Bildschirmfeld beginnt; mögliche Werte: $1 \leq n \leq \text{letzte Bildschirmspalte}$ Siehe auch Beispiel ¹
	LENGTH	Länge des Feldes, mögliche Werte: $1 \leq n \leq \text{letzte Bildschirmspalte}$
	TYPE	Angabe des Feldtyps Protected nicht überschreibbares Feld Unprotected Eingabefeld
	INPUT	Datentyp der Eingabe Alpha oder Numeric
	MARKABLE	Yes oder No
	MODIFIED	Yes oder No
	BLINKING	Yes oder No
	UNDERLINE	Yes oder No
	VISIBLE	Yes oder No
	INTENSITY	Normal oder Reduced
	INVERS	Yes oder No

Objektname	Attributname	Bedeutung des Attributs
	COLOR	Hier wird der RGB-Farbwert des betreffenden Feldes zurückgegeben. Der Wert hängt auch von den Einstellungen des Host-Objekts WT_COLOR ab, siehe dort. Wenn die Attribute von WT_COLOR nicht gesetzt wurden, werden folgende Werte zurückgegeben: #000000: Kein Farbattribut gesetzt #0000FF: Blau #FF0000: Rot #FFC0CB: Magenta #008000: Grün #40E0D0: Türkis #FFFF00: Gelb #FFFFFF: 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: $1 \leq n \leq \text{maximale Bildschirmbreite}$

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 ...

8.2.2 Host-Steuerobjekte


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

- Die Systemzeile (WT_SYSTEM_LINE)
- die Reihenfolge der Felder eines aktuellen Bildschirms
- das Feld, in dem sich der Cursor befindet
- Angabe, welche Datenfreigabetaste benutzt wird

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
\$CONNECTION	ALIVE	Prüfen, ob die Verbindung zum Host noch aktiv ist. Dieses Attribut ist als eine Ergänzung zu \$MESSAGE.WAITING zu verstehen. WAITING kann unter Umständen nach einem Verbindungsabbau den Wert NO enthalten, sodass der Verbindungsabbau nicht bemerkt wird, wenn bei asynchroner Arbeit nur WAITING geprüft wird.
\$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_III)	

Objektname	Attribut	Bedeutung des Attributs
\$MESSAGE	PRINTING	<p>Wenn dieses Attribut ausgewertet wird, sendet WebTransactions die älteste zum Druck anstehende Datei an den Browser (mit dem MIME-Typ <code>webta/hardcopy-print</code> oder MIME-Typ <code>webta/bypass-print</code>).</p> <p> Beachten Sie, dass mit der Auswertung von <code>\$MESSAGE.PRINTING</code> die Interpretation des Templates abgebrochen und die HTML-Generierung unterdrückt wird.</p>
	PRINTFILE_NAME	<p>Dieses Attribut liefert den Namen der Datei, die als nächste zum Drucken ansteht.</p> <p>Wenn keine Datei zum Drucken vorliegt, wird ein Leerstring zurückgeliefert.</p>
	WAITING	<p>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.</p> <p>Falls ja, wird der Wert von <code>\$MESSAGE.WAITING</code> intern auf "Yes" gesetzt. WebTransactions liest diese asynchrone Meldung dann beim nächsten Aufruf von <code>receive</code>.</p>
\$NEXT	Name	<p>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 einschließlich der Systemzeile (<code>WT_SYSTEM_LINE</code>).</p> <p>Existiert kein weiteres Objekt mehr, wird der Name <code>\$END</code> zurückgegeben.</p>
	Zusätzlich alle Attribute der dynamischen Host-Datenobjekte (<code>E_yy_xxx_III</code>)	
\$SCREEN	CONTENT	<p>Alle Zeichen des gesamten Bildschirms in einem String. Diese Angabe kann dazu verwendet werden, zwei Bildschirme zu vergleichen.</p>

Objektname	Attribut	Bedeutung des Attributs
WT_COLOR	DEFAULT	RGB-Farbwert für Felder, für die kein Farbattribut gesetzt ist. Mögliche Werte: #000000: Schwarz (Voreinstellung) #0000FF: Blau #FF0000: Rot #FFC0CB: Magenta #008000: Grün #40E0D0: Türkis #FFFF00: Gelb #FFFFFFE: Weiß
	BLUE	RGB-Farbwert für Felder mit dem Farbattribut <code>blue</code> Voreinstellung: #0000FF
	RED	RGB-Farbwert für Felder mit dem Farbattribut <code>red</code> Voreinstellung: #FF0000
	MAGENTA	RGB-Farbwert für Felder mit dem Farbattribut <code>pink</code> Voreinstellung: #FFC0CB
	GREEN	RGB-Farbwert für Felder mit dem Farbattribut <code>green</code> Voreinstellung: #008000
	CYAN	RGB-Farbwert für Felder mit dem Farbattribut <code>turquoise</code> Voreinstellung: #40E0D0
	YELLOW	RGB-Farbwert für Felder mit dem Farbattribut <code>yellow</code> Voreinstellung: #FFFF00
	WHITE	RGB-Farbwert für Felder mit dem Farbattribut <code>white</code> Voreinstellung: #FFFFFFE
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 <code>E_yy_xxx_III</code> oder als sprechender Name. 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 <code>E_yy_xxx</code> .

Objektname	Attribut	Bedeutung des Attributs
WT_KEY	KEY	Gibt an, welche Sondertaste in der Terminal-Emulation ausgeführt werden soll, wenn <code>send</code> ausgeführt wird. Die erlaubten Werte sind im Abschnitt „Zusammenspiel von Host-Steuerobjekt WT_KEY.KEY und Template wtKeysOSD.htm“ auf Seite 147 beschrieben. Entsprechende Knöpfe werden durch <code>wtKeysOSD.htm</code> in die aktuellen Templates inkludiert. (<code><wtInclude ...></code>).
	PKEY_n	Ermöglicht den direkten Zugriff auf den Inhalt der P-Taste <i>n</i> . Wenn diesem Attribut ein Wert zugewiesen wird, wird die entsprechende P-Taste in der Emulation geladen. Das Auswerten dieses Attributs gibt den aktuellen Inhalt zurück. In der Entwicklungsumgebung WebLab werden nur die P-Tasten im Objektbaum angezeigt, denen ein Inhalt zugewiesen wurde. Die Verwendung dieses Attributs ist im Abschnitt „Unterstützung der P-Tasten“ auf Seite 160 beschrieben. Mögliche Werte für <i>n</i> : 1 bis 255
WT_SYSTEM_LINE	Name	Name des Feldes in der Systemzeile
	Zusätzlich alle Attribute der dynamischen Host-Datenobjekte (E _{yy} _xxx _{III})	

8.3 Unterstützung von Terminal-Funktionen durch den Browser

Bei WebTransactions for OSD können Sie die Formate der Host-Anwendung ohne Nachbearbeitung am Browser darstellen (1:1-Umsetzung). Diese Funktionalität ist im Master-Template `OSD.wmt` enthalten (siehe [Seite 78](#)). Die Templates, die Sie über das Master-Template generiert haben, inkludieren die Templates `wtBrowserFunctions.htm` und `wtKeysOSD.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

`wtKeysOSD.htm` enthält die Knöpfe für die OSD-spezifischen Standard-Tasten und inkludiert die JavaScript-Datei `wtKeysOSD.js`, die die Abbildung der Sondertasten für WebTransactions for OSD enthält. In dieser Datei können Sie die Abbildung der Tasten entsprechend Ihren Wünschen anpassen oder erweitern (siehe [Abschnitt „Zuordnung der Tasten in wtKeysOSD.js“ auf Seite 149](#)).

8.3.1 Unterstützte Terminal-Funktionen

Folgende Terminal-Funktionen stehen zur Verfügung:

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

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

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

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

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

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

Wenn Sie den Internet Explorer ab V4.0 oder einen auf dem Gecko basierenden Browser (z.B. Netscape ab V6) verwenden, haben die Tasten des 9750-Terminals folgende Entsprechung¹:

Tasten bei Verwendung des Internet Explorer	entsprechende Taste am 9750-Terminal
ENTER	ENTER
F1 ... F12	F1 ... F12
Shift+F1 ... Shift+F12	F13 ... F24
STRG+F1 ... STRG+F12	K1 ... K12
STRG+Shift+F1 STRG+Shift+F2	K13 K14
STRG+1 ... STRG+9, STRG+0	P1 ... P9, P10
STRG+Shift+1 ... STRG+Shift+9, STRG+Shift+0	P11 ... P19, P20
STRG+Shift+F12	MAR
EM	EM
INS	INS

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

8.3.2 Zusammenspiel von Host-Steuerobjekt WT_KEY.KEY und Template wtKeysOSD.htm

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 wtKeysOSD.htm in die aktuellen Templates inkludiert (<wtInclude ...>). wtKeysOSD.htm inkludiert die JavaScript-Datei wtKeysOSD.js, die die Abbildung der Sondertasten für WebTransactions for OSD enthält (siehe [Abschnitt „Zuordnung der Tasten in wtKeysOSD.js“ auf Seite 149](#)).

Die folgende Tabelle zeigt:

- die Bedienelemente, die das Template wtKeysOSD.htm und die inkludierte Datei wtKeysOSD.js standardmäßig zur Verfügung stellen
- die Werte, welche für die damit verbundenen Funktionen im Host-Steuerobjekt WT_KEY hinterlegt werden.

Taste auf 9750-Terminal	Wert von WT_KEY.KEY	Bedeutung
DÜ	DUE	Datenfreigabe an den Host
DÜ2	DUE2	wie DÜ Datenfreigabe an den Host aber mit anderen Übertragungskennzeichen
K1 ... K14	K1 ... K14	Kurznachricht an die Host-Anwendung; es werden keine weiteren Daten übertragen.
F1 ... F24	F1 ... F24	wie DÜ Datenfreigabe an den Host aber mit anderen Übertragungskennzeichen
P1 ... P20	PKEY1 ... PKEY20	programmierbare Taste wird ausgeführt. Die programmierbare Taste kann entweder von der OSD-Anwendung oder durch Zuweisen eines Wertes auf ein Host-Steuerobjekt-Attribut WT_KEY.PKEY n geladen worden sein.
MAR		Markiert ein Feld als ausgewählt.
	Disconnect	Die Verbindung zum Host wird getrennt. Der folgende Aufruf <code>receive</code> versorgt daraufhin das 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.
	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 126). Die Funktion Refresh führt nicht zu einem Dialogschritt mit der Host-Anwendung: Bei den Aufrufen von <code>send</code> und <code>receive</code> wird die Host-Kommunikation unterdrückt.
	ClearScreen	Bildschirm löschen
	Print	Anfordern eines Terminal-Hardcopy-Drucks (siehe Abschnitt „Terminal-Hardcopy-Druck“ auf Seite 190).

8.3.3 Zuordnung der Tasten in wtKeysOSD.js

Alle Tastatureingaben werden von dem verwendeten Browser angenommen. Für die anwendungsspezifische Zuordnung von speziellen Funktionstasten stellt WebTransactions als Schnittstelle die Datei `wtKeysOSD.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 154](#)).

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

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

In der Tabelle `wtKeyMappingTableInput` wird für jede Tasten-Abbildung ein Objekt mit mehreren Attributen angelegt (siehe Tabelle auf [Seite 150](#)). Mit diesen Attributen wird beschrieben

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

Beispiel

```
wtKeyMappingTableInput = [
  { sl:'title of my select list'},
  { la:'Select', ac:doToggleMark, kc:VK_F12, mk:MK_CTRL+MK_SHIFT },
  { la:'ENTER', ac:'DUE2' },
  { co:'P1', ac:'P1', kc:VK_F1, mk:MK_ALT }
];
```

Aus dieser Definition entsteht folgende Abbildung:

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

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

title of my select list	—	ohne Funktion
Select	—	ruft die Funktion <code>doToggleMark()</code> auf
ENTER	—	sendet den Funktions-Code <code>DUE2</code> an WebTransactions

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

Bezeichnung	Attribut	Bedeutung
la	label	Beschriftung z.B. für den Eintrag in der Auswahl-Liste. Existiert dieses Attribut nicht, wird für die Liste kein Eintrag erzeugt. Die entsprechende Taste aber trotzdem auf eine Funktionalität abgebildet.
co	comment	Kommentar, dieses Attribut wird nicht ausgewertet. Dieses Attribut ist als Alternative zum Attribut <code>la</code> gedacht; durch Ändern des Attributes <code>la</code> nach <code>co</code> kann z.B. eine Taste aus der Auswahl-Liste entfernt werden.
ac	action	<p>Auszuführende Aktion bei Auslösen der zugeordneten Taste oder Auswahl aus einer Liste.</p> <p>Ist dieses Attribut vom Typ <code>string</code>, dann wird der Inhalt nach <code>wt_special_key.value</code> übertragen und an <code>WebTransactions</code> gesendet. Es wird also das Formular an <code>WebTransactions</code> übertragen und als Sonderfunktion der Wert von <code>ac</code> mitgegeben (z.B. <code>F1</code> als Funktionstaste).</p> <p>Ist dieses Attribut vom Typ <code>function</code>, dann wird eine Client-seitige Funktion mit diesem Namen aufgerufen. Diese Funktion muss definiert sein.</p> <p>Die JavaScript-Dateien <code>wt<browser>BrowserFunctions.js</code> stellen folgende Funktionen zur Verfügung:</p> <ul style="list-style-type: none"> - <code>doCursorHome</code> - <code>doCursorUp</code> - <code>doCursorDown</code> - <code>doCursorLeft</code> - <code>doCursorRight</code> - <code>doTab</code> - <code>doBackTab</code> - <code>doToggleMark</code> - <code>doToggleInsert</code>. <p>Die Implementierung dieser Funktionen kann abhängig von den Fähigkeiten des Browsers auch leer sein (siehe Abschnitt „Callback-Funktionen des Key mapping“ auf Seite 155).</p> <p>Ist dieses Attribut nicht definiert, kann keine Aktion durchgeführt werden. Die Tastatur-Eingabe wird dem Browser zur Bearbeitung überlassen.</p>
kc	key code	<p>Nummer, die im Tastatortreiber der gedrückten Taste zugeordnet ist. Für viele Tasten existiert in <code>wtCommonBrowserFunctions.js</code> ein Symbol, die Namen beginnen mit <code>VK_</code>.</p> <p>Für Tastenkombinationen ist zusätzlich der <code>modifier key (mk)</code> von Bedeutung.</p>

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

Aufbau von wtKeysOSD.js

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

Das Objekt `wtKeyMappingTableInput` wird als Literal angelegt.

```
wtKeyMappingTableInput = [
```

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

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

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

```
{ co:'MAR', ac:doToggleMark, kc:VK_F12, mk:MK_CTRL+MK_SHIFT },
{ co:'MAR', ac:doToggleMark, kc:VK_MAR, mk:0 },
{ co:'END', ac:doEnd, kc:VK_END },
{ 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:'SDZ', ac:doPageUp, kc:VK_PGUP },
{ co:'SNZ', ac:doPageDown, kc:VK_PGDN },
{ co:'HOME', ac:doCursorHome, kc:VK_HOME },
{ co:'TAB', ac:doTab, kc:VK_TAB },
{ co:'BACKTAB', ac:doBackTab, kc:VK_TAB, mk:MK_SHIFT },
```


Die Attribute `la` kennzeichnen die Einträge in der Auswahl-Liste.

```
{ la:'K1', ac:'K1', kc:VK_F1, mk:MK_CTRL },
{ la:'K2', ac:'K2', kc:VK_F2, mk:MK_CTRL },
{ la:'K3', ac:'K3', kc:VK_F3, mk:MK_CTRL },
...
{ la:'K14', ac:'K14', kc:VK_F2, mk:MK_CTRL+MK_SHIFT },
{ la:'F1', ac:'F1', kc:VK_F1 },
{ la:'F2', ac:'F2', kc:VK_F2 },
...
{ la:'F24', ac:'F24', kc:VK_F12, mk:MK_SHIFT },
{ la:'ClearScr', ac:'ClearScreen' },
{ la:'DUE2', ac:'DUE2' },
{ la:'InsClip', ac:doInsertClipboard, kc:VK_V, mk:MK_CTRL+MK_SHIFT },
```

Die folgenden Attribute `la` kennzeichnen die Auswahlliste für die P-Tasten.

```
{ la:'P1', ac:'PKEY1', kc:VK_1, mk:MK_CTRL, sl:'pkeys' },
{ la:'P2', ac:'PKEY2', kc:VK_2, mk:MK_CTRL },
...
{ la:'P20', ac:'PKEY20', kc:VK_0, mk:MK_CTRL+MK_SHIFT },
```

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

```
{ la:'P', ac:'PKEYS' }
];
```

8.3.4 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 Basisverzeichnis 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 `wtKeysOSD.js` aufgerufen. Sie enthält die Definition von Variablen, damit die Tasten in den anderen Dateien `*.js` symbolisch angesprochen werden können.

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

Key mapping Funktionen

```
function wtCreateKeyMap()
```

erzeugt aus der Tabelle `wtKeyMappingTableInput` eine zur Laufzeit einfacher und schneller zugreifbare Struktur. Der Aufruf erfolgt aus `wtKeys0SD.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 `wtKeys0SD.htm`. Durch Weglassen des Aufrufs dieser Funktion in `wtKeys0SD.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 150](#)).

Die meisten dieser Funktionen liefern `false` als Ergebnis, um zu signalisieren, dass keine allgemeine Behandlung für diese Tastatur-Eingabe zur Verfügung steht. Deshalb soll hier auf das Standardverhalten des entsprechenden Browsers zurückgegriffen werden. Dieses Standardverhalten wird in den Dateien `wt<browser>BrowserFunctions` gegebenenfalls durch Funktionen gleichen Namens überladen (siehe [Seite 157](#)).

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 150) 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 150) 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 `wtKeysOSD.htm`“ auf Seite 90). 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 150](#)) 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 Cursors 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 Cursors im Feld).

```
function doCursorHome()
```

Positioniert den Cursor an den Anfang des ersten Eingabefeldes.

```
function doTab()
```

springt auf den Anfang des nächsten Eingabefeldes.

```
function doBackTab()
```

springt auf den Anfang des vorhergehenden Eingabefeldes.

```
function doToggleMark()
```

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

```
function doToggleInsert()
```

Umschalten zwischen Einfüge- und Überschreib-Modus.

8.3.5 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 OSD.wmt oder OSD_Pocket.wmt generiert wurden (z.B. AutomaskOSD.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 Schriftgrößen zu bieten (z.B. abhängig von der am Bildschirm eingestellten Auflösung).

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



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

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

Beispiel für die Verwendung von WT_BROWSER.charSize

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

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

Diese Angabe wird automatisch übernommen, da beim Sitzungsstart das Attribut WT_POSTED.wtCharSize ausgewertet wird. Alle Größeneinstellungen in AutomaskOSD.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>
```

8.3.6 Unterstützung der P-Tasten

Ein 9750-Terminal bietet Ihnen zusätzlich programmierbare Tasten (die P-Tasten), die Sie mit Funktionen belegen können. Um die P-Tasten auch aus WTML programmierbar zu gestalten, bietet WebTransactions for OSD einen zusätzlichen Satz von Objekten am Host-Steuerobjekt `WT_KEY` (siehe [Abschnitt „Host-Steuerobjekte“ auf Seite 140](#)).

Diese Objekte sind mit `PKEYn` bezeichnet, wobei n einen Wert von 1 bis 255 haben kann. `PKEY`-Objekte werden nur dann im Objektbaum der Entwicklungsumgebung WebLab angezeigt, wenn die entsprechenden P-Tasten für die Emulation definiert wurden.

Obwohl an einem 9750-Terminal nur 20 P-Tasten unterstützt werden, können Sie in WebTransactions for OSD 255 `PKEY`-Objekte definieren. In der JavaScript-Datei `wtKeysOSD.js` ist die Programmierung für die P-Tasten `P-1` bis `P-20` vorbereitet. Sie können diese Datei entsprechend Ihren Wünschen anpassen oder erweitern (siehe [Abschnitt „Zuordnung der Tasten in wtKeysOSD.js“ auf Seite 149](#)).

8.3.6.1 Definition in WTML

Für die Definition der `PKEY`-Objekte in WTML sind die folgenden Punkte zu beachten:

- Jeder `PKEY` kann mit einer Folge von bis zu 255 Zeichen definiert werden.
- Für die Terminal-Funktionen sind die folgenden Ersatzdarstellungen möglich. Die Ersatzdarstellungen werden beim Auslesen der P-Tasten wiederhergestellt.
 - `[AM]`, `[EM]`,
 - `[DUE]`, `[DUE2]`,
 - `[Pn]` mit $1 \leq n \leq 20$
 - `[F1]`, `[F2]`, `[F3]`, `[F4]`, `[F5]`
 - `[Kn]` mit $1 \leq n \leq 14$
 - `[SMR]`, `[SML]`, `[SM0]`, `[SMU]`, `[SNZ]`, `[SBA]`, `[SZA]`, `[SDZ]`, `[TAR]`, `[TAL]`
 - `[EFG]`, `[AFG]`, `[AFZ]`, `[EFZ]`, `[LZF]`, `[LSP]`, `[LVD]`, `[MAR]`, `[FAZ]`, `[LZE]`, `[RS]`
- Die Nummer eines `PKEY` darf keine führenden Nullen enthalten.
- Aus Kompatibilitätsgründen zum Protokoll 810 gelten folgende Regeln:
 - Alle `PKEYs` (1 bis 255) können immer definiert und benutzt werden.
 - Aber nur `PKEY1` bis `PKEY20` können gekettet und von der Host-Anwendung ausgelesen werden (Terminal-Funktion `LPBE` = Lesen P- Bereiche).

- Hinweis zu Unicode-Zeichen:

PKEY-Objekte können nur von der Host-Anwendung mit Unicode-Zeichen versorgt werden. Das Fenster **PKEYS-ASSIGNMENT** unterstützt Unicode-Zeichen nicht.

Beispiel

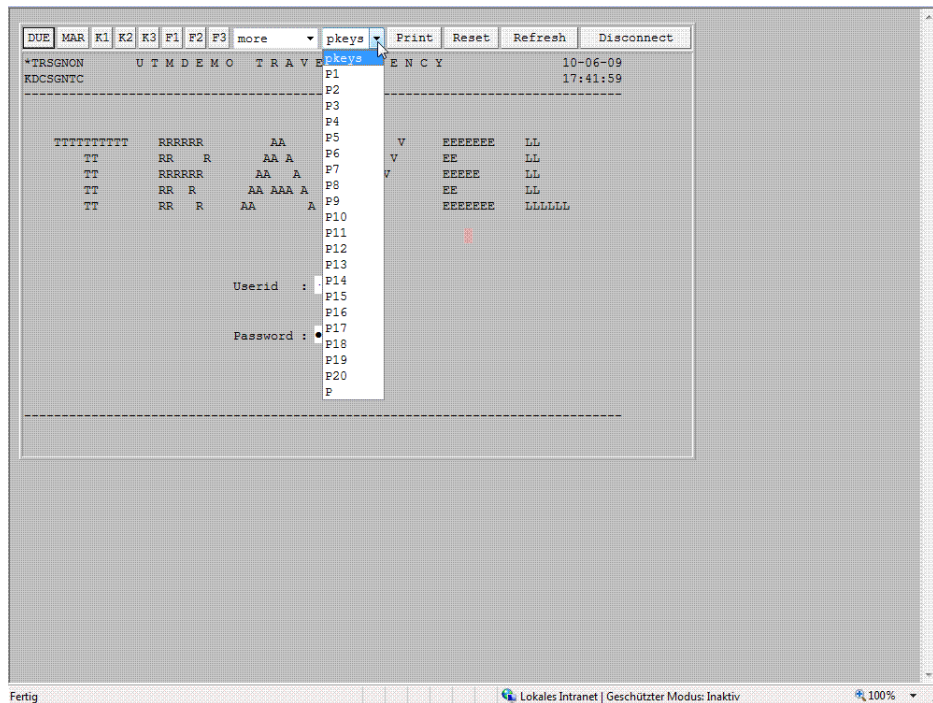
```
...  
WT_KEY.PKEY1 = 'SH-USER-STAC[EM][DUE]';  
WT_KEY.KEY = 'PKEY1';  
host.send();  
...
```

8.3.6.2 Template wtPKEYS.htm

Das ausgelieferte Template `wtPKEYS.htm` stellt Ihnen eine einfache Möglichkeit zur Verfügung, die P-Tasten am Browser zu programmieren. Das Template dient als Vorlage, Sie sollten es nach Ihren Bedürfnissen erweitern oder verändern. Die Vorlage sieht z.B. nur vor, die P-Tasten in einer frei zu benennenden Datei abzuspeichern. Um für die Benutzer bei jedem neuen Sitzungsstart den letzten Inhalt der P-Tasten automatisch wiederherzustellen, könnten die P-Tasten in einer Datei gespeichert werden, die sich aus dem Rechnernamen oder der Benutzerkennung des Benutzers ergibt.

Um das Template aufzurufen und P-Tasten zuzuweisen, gehen Sie wie folgt vor:

- ▶ Wählen Sie am Bildschirm Ihrer Host-Anwendung den Befehl **pkeys/P**.



Durch diese Eingabe wird in der laufenden Sitzung `wtPKEYS.htm` anstelle des aktuellen Templates mit der Funktion `setNextPage()` aufgerufen.

PKEYS - Assignment

P1					
P2					
P3					
P4					
P5					
P6					
P7					
P8					
P9					
P10					
P11					
P12					
P13					
P14					
P15					
P16					
P17					
P18					
P19					
P20					

AM	EM	LZE	DUE	DUE2
AFG	AFZ	LSP	LVD	LZF
EFG	EFZ	FAZ	MAR	RS
F1	F2	F3	F4	F5
K1	K2	K3	K4	K5
K6	K7	K8	K9	K10
K11	K12	K13	K14	
P1	P2	P3	P4	P5
P6	P7	P8	P9	P10
P11	P12	P13	P14	P15
P16	P17	P18	P19	P20
SBA	SZA	SDZ	SNZ	
SMO	SMU	SML	SMR	
TAL	TAR			

File:

In diesem Bildschirm können Sie den aufgeführten PKEYs Funktionen zuweisen.



Die in diesem Schirm bearbeiteten Daten werden in einem eigenen Bearbeitungsspeicher verwaltet. Erst wenn Sie die **Set EMU**-Schaltfläche drücken, werden die Werte in die Emulation programmiert.

Sie haben zwei Möglichkeiten:

- Sie tragen Ihre Angaben direkt in die betreffende Zeile ein.
- Sie klicken auf den Knopf mit der betreffenden Funktion. Die zugehörige Ersatzdarstellung wird an der Cursor-Position eingetragen, sofern der Browser die Ermittlung der Schreibmarke im Eingabefeld ermöglicht. Andernfalls wird die Ersatzdarstellung an das Ende des gerade benutzten Eingabefeldes angehängt. Die Zuweisung der Funktionen zu den Knöpfen ist im Template `wtPkeyValues.htm` abgelegt, das von `wtPKEYS.htm` inkludiert wird. Die Funktionstasten haben folgende Bedeutung:

AFG	Zeichen ausfügen
AFZ	Zeile ausfügen
AM	Anfangsmarke
DUE oder DUE1	Datenübertragung mit DU1 starten

DUE2	Datenübertragung mit DU2 starten
EFG	Zeichen-Einfüge-Modus
EFZ	Zeile einfügen
EM	Endemarke
F1 .. F5	Funktionstasten
FAZ	Feldtrennzeichen in den Ausgangszustand zurücksetzen (Feldbeginnzeichen FBZ und Anzeigesteuerzeichen ASZ); z.B. durch MAR erfolgte Markierungen
K1 .. K14	Kurznachricht
LSP	Löschen Bild
LVD	Variablen löschen
LZE	Logisches Zeilenende
LZF	Löschen bis Zeilen-/Feldende
MAR	Feld markieren
P1 .. P20	Programmiertaste
RS	Zurücksetzen der Emulation (z.B. nach EFG)
SBA	Cursor an den Bildanfang
SDZ	Cursor an den Anfang der vorhergehenden Zeile
SML	Cursor links
SMO	Cursor oben
SMR	Cursor rechts
SMU	Cursor unten
SNZ	Cursor an den Anfang der nächsten Zeile
SZA	Cursor an den Zeilenanfang
TAL	Tabulator links
TAR	Tabulator rechts

Mit den Aktionsknöpfen im Template steuern Sie, wie die P-Tasten-Zuweisung weiter bearbeitet werden soll:

Set EMU

lädt die Zuweisung der P-Tasten aus dem Bearbeitungsspeicher in Ihre Emulation.

Load EMU

liest die Inhalte der aktuell definierten PKEYs aus der Emulation in den Bearbeitungsspeicher.

Reset

setzt den Bildschirm-Inhalt zurück.

Back

verzweigt wieder zurück zum Bildschirm der Host-Anwendung. Nicht gespeicherte Zuweisungen werden dabei verworfen.

Load File

Lädt den Inhalt der P-Tasten aus der XML-Datei, die Sie unter **File** angegeben haben, in den Bearbeitungsspeicher. Die XML-Datei muss die Zuweisung der P-Tasten enthalten und im Basisverzeichnis der WebTransactions-Anwendung unter Pkeys vorliegen.

Save File

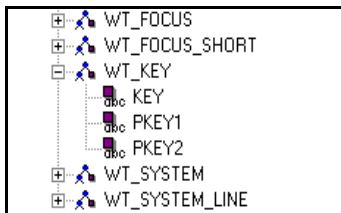
Speichert die Zuweisung in der XML-Datei, die Sie unter **File** angegeben haben. Die Angabe in **File** erfolgt ohne das Suffix `.xml`. Die XML-Datei wird im Basisverzeichnis der WebTransactions-Anwendung unter Pkeys angelegt.



Während einer Sitzung in der Emulation programmierte P-Tasten sind nach Sitzungsende verloren. Wenn Sie bei der nächsten Sitzung die Zuweisung wieder verwenden wollen, müssen Sie die XML-Datei erneut mit **Load File** und **Set EMU** laden. Welche Möglichkeiten Sie haben, die Inhalte der P-Tasten für die nächste Sitzung weiterzuverwenden, entnehmen Sie dem [Abschnitt „PKEYs speichern“ auf Seite 168](#).

8.3.6.3 PKEYS verwalten

Die PKEYs werden als Objekte unter WT_KEY angelegt.



Um die Objekte zu bearbeiten, stehen Ihnen im Template `wtPkeyFunctions.htm` die folgenden Methoden zur Verfügung. Die Methoden sind in der Klasse `WT_PKEYS` definiert. `wtPkeyFunctions.htm` wird auch von `wtPKEYS.htm` inkludiert.

Methoden eines Objekts der Klasse WT_PKEYS

`WT_PKEYS([number])`

Der Konstruktor legt unter dem Objekt `WT_KEY` die angegebene Anzahl von `PKEY`-Objekten an.

Voreinstellung: 20

`setDirectory(directory)`

legt das Verzeichnis fest, in dem die XML-Datei mit der `PKEY`-Zuweisung abgelegt werden soll.

Voreinstellung: `basisverzeichnis/Pkeys`

`setFileName(file)`

Name für die XML-Datei mit den `PKEY`-Inhalten. Die Datei wird relativ zu dem Verzeichnis angelegt, das mit `setDirectory()` festgelegt wurde.

`getFileName()`

liefert den Namen der Datei, der mit `setFileName()` festgelegt wurde.

`setPassword(password)`

gibt das Passwort an, mit dem die XML-Datei geschützt werden soll

`readFromFile()`

liest die `PKEYs` aus der Datei, die mit `setDirectory()` und `setFileName()` festgelegt wurde.

`saveToFile()`

schreibt die Inhalte der `PKEY`-Zuweisung in die Datei, die mit `setFileName()` festgelegt wurde. Nach einem Aufruf von `setClusterParameter()` wird die Datei auch auf die anderen Cluster Member verteilt.

`loadFromEmu()`

liest die Inhalte der `PKEYs` aus der Emulation.

`setToEmu()`

lädt die Inhalte der PKEYs in die Emulation.

`getKeyValue(number)`

ermittelt den Inhalt des angegebenen PKEY aus der Instanz von WT_PKEY.

`setKeyValue(number, value)`

setzt den PKEY, der mit *number* angegeben ist, in der Instanz von WT_PKEY auf den Wert *value*.

`initClusterParameter (file, user, password, clusterName)`

legt eine Übergabedatei für alle Werte an, die für die Verteilung von Dateien in einer Cluster-Konfiguration benötigt werden (siehe WebTransactions-Handbuch „Konzepte und Funktionen“).

file Name der Übergabedatei, wird in der Methode `setClusterParameter()` verwendet. Die Datei wird im XML-Format mit dem Suffix `.xml` angelegt.

user, *password*

Kennung und Passwort, mit dem Dateien im Cluster verteilt werden können.

clusterName

Name der Cluster-Konfiguration.

Die Methode `initClusterParameter()` muss zur Vorbereitung einmal auf dem Cluster Controller aufgerufen werden. Die entstandene Datei *file.xml* kann dann mit Hilfe von WebLab auf die Cluster Member verteilt werden. Sie müssen diesen Vorgang wiederholen, wenn sich die Cluster-Definition geändert hat oder Zugriffsrechte verändert wurden. Die Kennung *user* und das Passwort *password* müssen auf allen beteiligten Rechnern Zugriff auf das entsprechende Basisverzeichnis haben.

`setClusterParameter(file)`

legt fest, dass `saveToFile()` die Cluster-Definition, die mit `initClusterParameter()` erstellt wurde, nutzen soll, um die PKEYs auf die Cluster Member zu verteilen.

8.3.6.4 PKEYs speichern

Bei einer Terminal-Emulation, die auf einem Client-PC installiert ist, werden die Inhalte der PKEYs lokal am PC gespeichert und unterliegen dem Schutz der Windows-Benutzererkennung. Sie sind also bei einem erneutem Aufruf dieser Emulation wieder verfügbar und (je nach Betriebssystem) vor dem Zugriff durch andere Benutzer geschützt.

Dies gilt nicht für WebTransactions, da hier die Emulation im Host-Adapter auf dem Server läuft. Host-Adapter oder Emulation haben keine Kenntnis darüber, von welchem PC aus und mit welcher Benutzererkennung der Anwender der aktuellen Sitzung sich an seinem Client-PC angemeldet hat.

Es sind verschiedene Maßnahmen möglich, um den letzten Inhalt der P-Tasten bei der nächsten Sitzung für den gleichen Nutzer wiederherstellen zu können und den Inhalt der P-Tasten gegen unberechtigten Zugriff zu schützen. Diese Maßnahmen sind abhängig von den jeweiligen Gegebenheiten beim Kunden.

Für eine persistente Speicherung der PKEY-Inhalte haben Sie verschiedene Möglichkeiten:

- Sie können die PKEY-Zuordnung verwenden, die Sie mit dem `WTBean wtcSingleSignOn` erstellt haben. Sie müssen dann die PKEY-Zuordnung in dem Bereich des `WebTransactions`-Objektbaumes anlegen, der von `wtcSingleSignOn` benutzerspezifisch gespeichert wird (siehe Dokumentation des `WTBean`).
- Sie können die PKEY-Zuordnung so ablegen, dass sie ein-eindeutig einem Benutzer zuzuordnen ist, z.B.
 - in einer Datei, deren Name aus IP-Adresse und Namen des PC abgeleitet wird. Dieses Vorgehen ist nicht möglich bei einem Betrieb über PROXY.
 - in einer Datei, deren Name dann in einem Cookie gespeichert wird. Es muss sichergestellt sein, dass der Dateiname nur einmal zugeordnet wird. Dieses Vorgehen ist nicht möglich, wenn Cookies nicht erlaubt sind.
 - in einer Datei, deren Name aus der Benutzererkennung abgeleitet ist. Mögliche Einschränkungen bei diesem Vorgehen ergeben sich durch folgende Fragen:
 - Ist `VBScript` erlaubt?
 - Ist die Benutzererkennung der Windows-Anmeldung eindeutig im Benutzerkreis der `WebTransactions`-Anwendung?
 - direkt in einem Cookie. Dieses Vorgehen ist nicht möglich, wenn Cookies nicht erlaubt sind. Einschränkungen können sich durch die maximale Länge der Cookies ergeben.

8.4 Start-Templates für OSD

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 OSD“ wird das Start-Template `wtstartOSD.htm` ausgeliefert. Dieses OSD-spezifische Start-Template wird in [Abschnitt „OSD-spezifisches Start-Template des Start-Template-Sets \(wtstartOSD.htm\)“ auf Seite 170](#) 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 OSD-Anwendung sollten Sie ein eigens dafür generiertes Start-Template verwenden. Bei der Generierung werden Sie vom WTBean `wtcStartOSD.wtc` unterstützt.

8.4.1 OSD-spezifisches Start-Template des Start-Template-Sets (wtstartOSD.htm)

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

- Sie können Verbindungsparameter festlegen und eine Verbindung zu einer OSD-Anwendung öffnen. Wenn Sie hier die Option **run** wählen, wird die Verbindung zur Host-anwendung 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 `wtstartOSD.htm` erneut angezeigt. Es enthält nun - falls die Verbindung erfolgreich geöffnet werden konnte - zusätzliche Knöpfe für die Kommunikation mit der OSD-Anwendung. Wenn Sie hier die Option **close** wählen, wird die Verbindung zur Host-Anwendung beendet.
- Wenn Sie die Option **receive** wählen, wird das Template `wtstartOSD.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

 OSD communication	
status	communication object: WT_HOST.OSD_0
	connection: down
workflow	destination: main menu ▾ go to
	access host: open run
	parameters: update reset
connection parameters	HOST_NAME: <input type="text"/>
	SYM_DEST: <input type="text"/>
	APPLICATION_PREFIX: <input type="checkbox"/>
	STATION_NAME: <input type="text"/>
	TERMINAL_TYPE: 9750 ▾
	PORT_NUMBER: <input type="text" value="102"/>
	FORMAT: <input type="text" value="wtstartOSD"/>
	AUTOMASK: <input type="text" value="AutomaskOSD"/>
	DISCONNECT: <input type="text" value="wtstartOSD"/>
	CAPTURE_FILE: <input type="text" value="config/capture.sdb"/>
	TRACE_LEVEL: <input type="radio"/> 0 <input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input checked="" type="checkbox"/> E <input checked="" type="checkbox"/> M <input type="checkbox"/> L
	LOGFILE: <input type="text"/>
	TRACEFILE: <input type="text"/>
	NATIONAL_VARIANT: International ▾
	FIRST_IO_TIMEOUT: <input type="text" value="60"/> ▾
	MULTIPLE_IO_TIMEOUT: <input type="text" value="2"/> ▾
	END_MARK: <input type="text" value="~"/>
	Print/Asynchronous support: <input type="radio"/> Start <input checked="" type="radio"/> Stop
DISPLAY_EURO: <input type="checkbox"/>	
FIELD_NAMES: <input type="checkbox"/>	

Im Abschnitt **connection parameters** können Sie die gleichnamigen Attribute des Systemobjekts auf die gewünschten Werte setzen (siehe [Abschnitt „Systemobjekt-Attribute“ auf Seite 115](#)).

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

destination

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

access host

Hier werden die Aktionen angeboten, die aktuell mit der Sitzung durchgeführt werden können. Ist noch keine Verbindung geöffnet, so stehen 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 offen ist)

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 / resume 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 Knopfs **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.

8.4.2 WtBean `wtcStartOSD.wtc` zur Generierung eines Start-Template

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

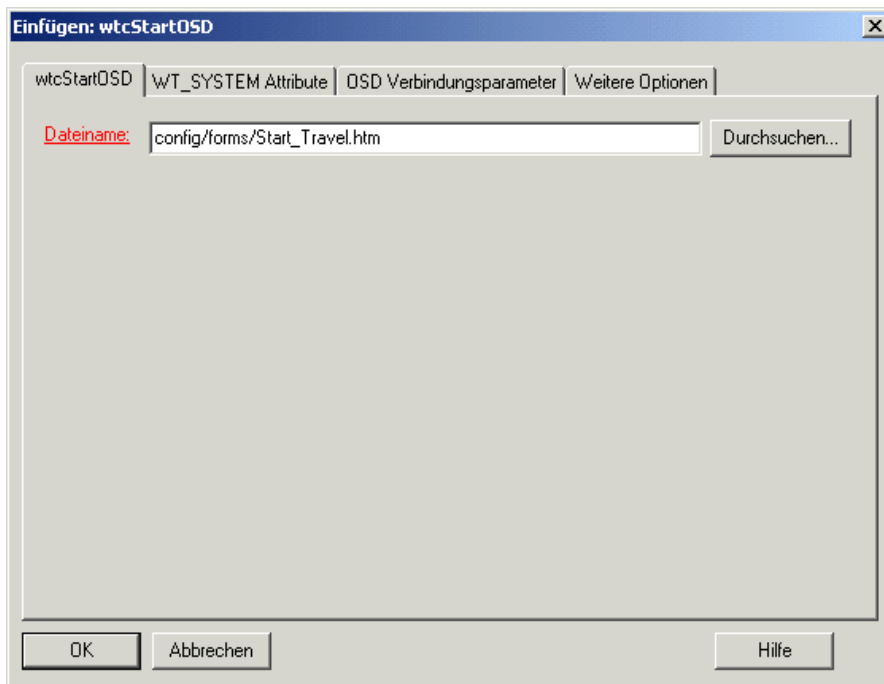
`wtcStartOSD.wtc` enthält das inline WtBean `wtcOSD.wtc`, mit dem Sie in einem Template ein neues OSD-Kommunikationsobjekt anlegen, und damit eine Verbindung zu einer OSD-Anwendung aufbauen (siehe Abschnitt „[Neues OSD-Kommunikationsobjekt anlegen \(wtcOSD\)](#)“ auf Seite 176).



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

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

- In der Registerkarte **wtcStartOSD** legen Sie den Namen des zu generierenden Start-Template fest.
- In der Registerkarte **WT_SYSTEM-Attribute** legen Sie die wichtigsten Attribute des Systemobjekts fest.
- In der Registerkarte **OSD-Verbindungsparameter** legen Sie die wichtigsten Verbindungsparameter fest.
- In der Registerkarte **Weitere Optionen** können Sie in einer Baumstruktur alle Parameter für die Verbindung zur OSD-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 generierten Start-Templates.

8.5 Neues OSD-Kommunikationsobjekt anlegen (wtcOSD)

Um in einem Template ein neues OSD-Kommunikationsobjekt anzulegen, und damit eine Verbindung zu einer OSD-Anwendung aufzubauen, wird das WtBean `wtcOSD` mitausgeliefert. Dieses WtBean können Sie auch dazu nutzen, um mehrere Verbindungen parallel zu öffnen. `wtcOSD` 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/wtcOSD** rufen Sie das WtBean zur Bearbeitung auf. WebLab generiert das Dialogfeld **Einfügen:wtcOSD**:

The screenshot shows a dialog box titled "wtcOSD bearbeiten". It has two tabs: "OSD Verbindungsparameter" (selected) and "Weitere Optionen". Under "OSD Verbindungsparameter", there are four fields: "Kommunikationsobjekt:" (dropdown menu showing "OSD_0"), "Hostapplikation:" (dropdown menu showing "travel" and "Auswählen..." button), "Name des Host-Rechners:" (dropdown menu showing "system1" and "Auswählen..." button), and "Capture-Datenbank:" (dropdown menu showing "config/capture.sdb" and "Durchsuchen..." button). There is also a checkbox "Präfix anwenden:" which is unchecked. At the bottom are "OK", "Abbrechen", and "Hilfe" buttons.

In diesem Dialogfeld können Sie die Parameter für das neue Kommunikationsobjekt bearbeiten. Die wichtigsten Parameter sind auf der ersten Registerkarte **OSD 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 geschützten Bereiche sind grau hinterlegt dargestellt. Auf diese Bereiche können Sie nur über die Oberfläche des WTBean Einfluss nehmen. Sie wählen dazu im Kontextmenü der Startzeile des WTBean (pink hinterlegt) den Befehl **WTBean bearbeiten** (siehe WebTransactions-Handbuch „Konzepte und Funktionen“).

9 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 180](#) beschriebene Frameset. In der Konfiguration der eingesetzten Web-Browser muss JavaScript zugelassen sein.

9.1 Druck-/Asynchron-Unterstützung einschalten

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

Wenn Sie für den Start der WebTransactions-Sitzung das ausgelieferte Start-Template `wtstartOSD.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 wtccOSD`, 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.

9.2 Funktionsweise der Druck-/Asynchron-Unterstützung

Druck- und Asynchronunterstützung funktionieren nach dem gleichen Prinzip.

- Unterstützt der Browser das HTML-Tag `<i frame>` 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 `<i frame>` und wird die Druck-/Asynchron-Unterstützung eingeschaltet (d.h. `WT_ASYNC` wird auf "Yes" gesetzt), wird nur inline, also mit `<i frame>`, 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 Web-Transactions-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 Druckerfordernungen
    ... (siehe Seite 191)

    Behandlung asynchroner Nachrichten
    ... (siehe Seite 187)

    //-->
  </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`:

```

<!-- ----- -->
<!-- begin of wtBrowserFunctions.htm -->
<!-- ----- -->
<wtOnCreateScript>
<!--
    for( wtCurrentComm_Name in WT_HOST )
    {
        if( WT_HOST[ wtCurrentComm_Name ] == wtCurrentComm )
            break;
    }
//-->
</wtOnCreateScript>
<input type="hidden" name="wt_cursor" value="" />
<input type="hidden" name="wt_cursorOffset"
value="##wtCurrentComm.WT_FOCUS&&wtCurrentComm.WT_FOCUS.Offset ?
wtCurrentComm.WT_FOCUS.Offset : 0#" />
<input type="hidden" name="wt_markedFields" value="" />
<input type="hidden" name="wt_modifiedFields" value="" />
<input type="hidden" name="wt_isOverwrite"
value="##wtCurrentComm_system.isOverwrite?1:0#" />
<script type="text/javascript">
<!--
    ##wtCurrentComm_system.NIL_MODE?'wtWantNils = true;':''#
    ##wtCurrentComm_system.AUTOTAB&&wtCurrentComm_system.AUTOTAB=='No'?'wtWantAutoTab =
false;':''#
    wtCharSize=##WT_BROWSER.charSize#;
    wtRefreshTimer = 5000; // milliseconds
//-->
</script>
<wtIf ( wtCurrentComm_system.WT_ASYNC == 'Yes' && WT_BROWSER.acceptIframe )>
    <wtrem>support of asynchronous message and printing with browsers supporting <iframe>//
/////////</wtrem>
    <iframe id="asyncFrame" style="border:0;height:0;position:absolute;"
src="about:blank"></iframe>
    <script type="text/javascript">

```

```

<!--
function AutomaticRefresh()
{
    try{
        document.getElementById('asyncFrame').contentDocument.location =
'##WT_SYSTEM.HREF_ASYNC#&WT_ASYNC_PAGE=wtasync';
    }catch(e){
        try{
            document.getElementById('asyncFrame').src =
'##WT_SYSTEM.HREF_ASYNC#&WT_ASYNC_PAGE=wtasync';
        }catch(f){}
    }
    setTimeout('AutomaticRefresh()', wtRefreshTimer);
}

##(wtCurrentComm_system.HARDCOPY == 'Yes' || wtCurrentComm_system.BYPASS == 'Yes') ?
'AutomaticRefresh()'; :
'setTimeout("AutomaticRefresh()", wtRefreshTimer);'#
//-->
</script>
<wtElse><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()', wtRefreshTimer);
}
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()", wtRefreshTimer);'#
}
//-->
</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>
...
<!-- ----- -->
<!-- end of wtBrowserFunctions.htm -->
<!-- ----- -->
```

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

9.3 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 180](#) vorgestellte Frameset `wframes` 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

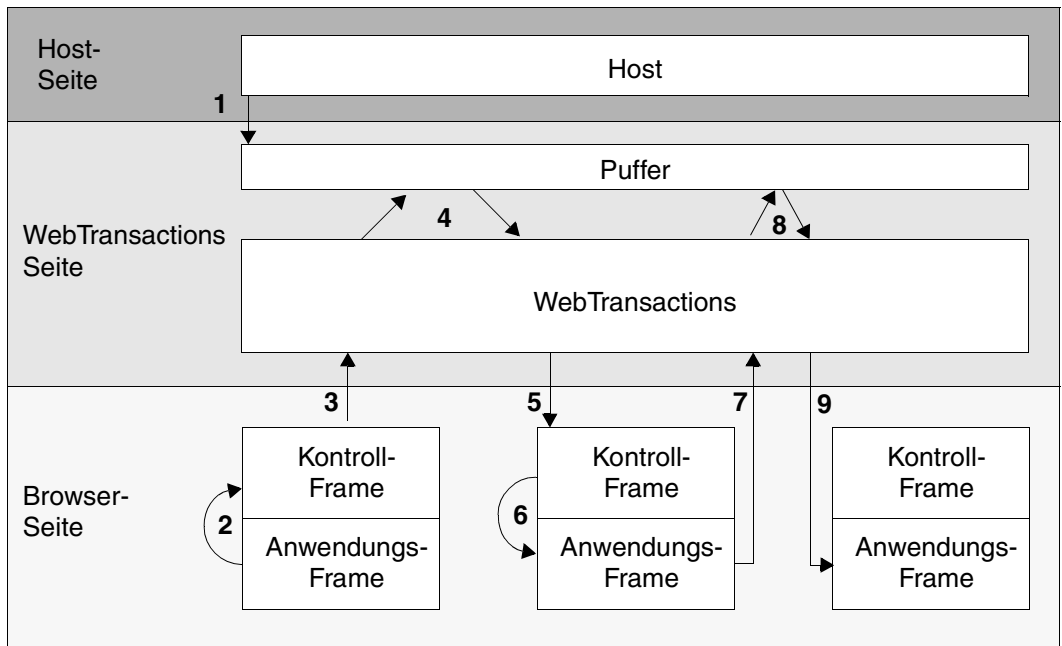


Bild 3: Behandlung asynchroner Nachrichten

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 einen Refresh des Anwendungs-Frames auslöst (siehe „In `wtasync.htm` definierte Behandlung asynchroner Nachrichten“ auf Seite 187).

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>");
```

9.4 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.
- Browserdarstellungs-Druck
Dabei wird die im Web-Browser dargestellte Information ausgedruckt.
- Ausdruck von Host-Daten
Dabei werden Informationen ausgedruckt, die von der Host-Anwendung aufbereitet und gesendet wurden.

Für Terminal-Hardcopy-Druck und Browserdarstellungs-Druck kann man wiederum unterscheiden, ob der Ausdruck vom Anwender am Web-Browser angestoßen wird (anwender-getrieben), 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 „Druck-Plugin WTAPrint“ auf Seite 205](#) 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.

9.4.1 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 Anwendungsschirm aus.

Konzept

Grundlage für die Behandlung des Terminal-Hardcopy-Drucks ist das in [Abschnitt „Funktionsweise der Druck-/Asynchron-Unterstützung“ auf Seite 180](#) beschriebene Frameset `wtframes` aus Anwendungs-Frame und Kontroll-Frame.

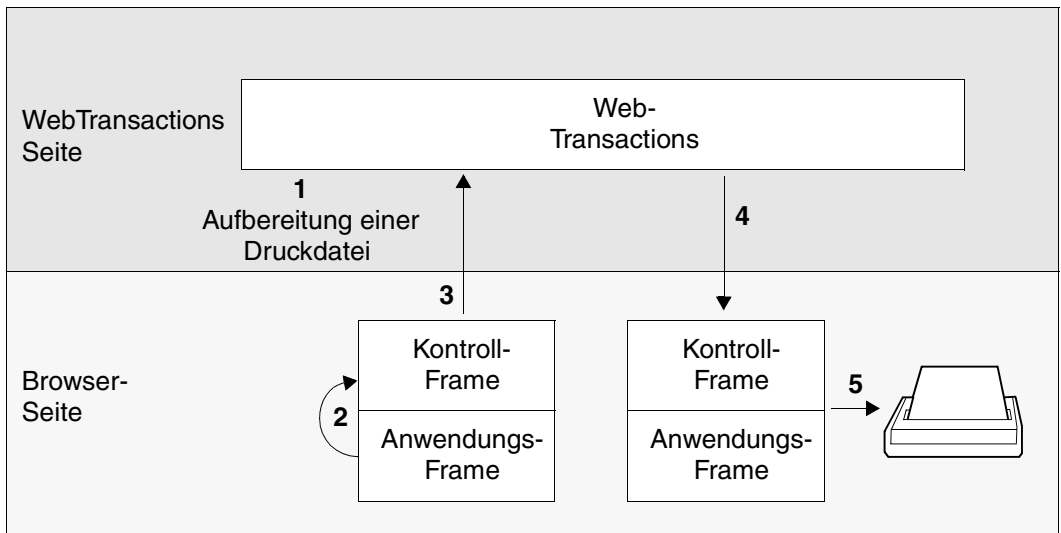


Bild 4: Terminal-Hardcopy-Druck

Wenn ein Terminal-Hardcopy-Ausdruck angefordert ist, stellt WebTransactions eine entsprechende Druckdatei zur Verfügung (Schritt 1). Getriggert durch das Script 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 199](#) 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 verbindungspezifischen 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 `wtKeysOSD.htm` definiert, das vom Umsetzungstemplate `AutomaskOSD.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.

Von der Host-Anwendung ausgelöster Terminal-Hardcopy-Druck

Ein Terminal-Hardcopy-Druck wird auch dann ausgelöst, wenn WebTransactions von der Host-Anwendung eine Nachricht empfängt, die eine entsprechende Hardcopy-Escape-Sequenz enthält. In einem solchen Fall behandelt WebTransactions die empfangene Nachricht wie gewohnt, d.h. eine entsprechende HTML-Seite wird erzeugt und zur Anzeige an den Browser geschickt. Darüber hinaus wird eine Druckdatei generiert, die eine alphanumerische Darstellung des aktuellen Schirms enthält (vgl. Schritt 1 in der Abbildung auf [Seite 186](#)).

9.4.2 Ausdruck von Host-Daten

OSD-Anwendungen sind in der Lage, Druckinformationen gezielt zu spezifischen Druckern zu schicken. Das Konzept dieses Bypass-Drucks umfasst die folgenden zwei Möglichkeiten:

1. der Drucker hat den gleichen Namen wie das Terminal, das mit der Host-Anwendung verbunden ist. Verwendet wird der „Stationsname“ in OSD-Terminologie. In der WebTransactions-Umgebung entspricht das Terminal dem WebTransactions-Server. In diesem Fall wird die Druckdatei direkt an den WebTransactions-Server geschickt.
2. der Drucker hat einen eigenen Stationsnamen. Das Terminal wird nur für die Drucksteuerung verwendet und ist der Host-Anwendung bekannt.

Für Informationen über den Stationsnamen gibt es verschiedene Schnittstellen in der Host-Anwendung.

Beispiel für eine OSD/openUTM-Umgebung

1. KDCS INFO

Dieser KDCS-Aufruf fordert Informationen von openUTM an. Mit der Operationsmodifikation SI wird der Stationsname erfragt.

2. KDCS INIT

Mit diesem KDCS-Aufruf meldet man sich bei einem openUTM-Programm an. Die Operationsmodifikation PU gibt die Informationen für die aktuelle Konversation zurück. Für Details siehe openUTM-Handbuch „[Anwendungen programmieren mit KDCS für COBOL, C und C++](#)“.

Drucker mit gleichem Namen wie Terminal

Die zu druckenden Informationen werden von der Host-Anwendung an WebTransactions geschickt. Der Host-Adapter OSD erkennt, dass ein Bypass-Druck ausgelöst wurde. Er schreibt die zu druckenden Informationen in eine Datei und setzt das Attribut `BYPASS` des Systemobjekts auf den Wert "Yes". Diese Datei wird im temporären Verzeichnis der entsprechenden WebTransactions-Sitzung abgelegt. Wie beim Hardcopy-Druck wird der Kontroll-Frame in regelmäßigen Abständen im Hintergrund abgeschickt und bei WebTransactions angefragt, ob eine Druckdatei vorliegt (siehe [Abschnitt „Terminal-Hardcopy-Druck“ auf Seite 190](#)). Falls ja, wird die aufbereitete Datei an den Browser geschickt. Im Browser sollte ein entsprechender MIME-Typ definiert sein, um ein zugeordnetes Druckprogramm zu starten (siehe hierzu auch [Abschnitt „Konfiguration der Web-Browser“ auf Seite 205](#)).



Die Daten der BYPASS-Nachricht werden von WebTransactions standardmäßig unabhängig vom Inhalt (transparent) übertragen. Dies hat den Vorteil, dass sowohl Grafik- als auch Textdateien gedruckt werden können.

Bypass-Druck mit Konvertierung

Eine OSD-Anwendung bereitet einen Bypass-Druck häufig nur für „einfache“ Drucker wie z.B. vom Typ 9001 auf. Wenn am Client ein anderer Druckertyp angeschlossen ist, dann kann es notwendig sein, dass WebTransactions Informationen an der Druckdatei verändert anstatt sie nur durchzureichen. Diese Änderungen können z.B. darin bestehen, eine andere maximale Zeilenlänge zu definieren oder bestimmte Zeichenfolgen der Druckdatei zu konvertieren. In diesem Fall müssen Sie im Browser zusätzliche Options- und Konvertierungsdateien konfigurieren, siehe [Seite 206](#).

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

```
// Support of bypass print ////////////////////////////////////////
else if (host_system.BYPASS == "Yes")
    {
        host_system.BYPASS = "No";        // Reset flag before
        if (host_system.WT_BROWSER_PRINT.toLowerCase() != 'yes')
            host.$MESSAGE.PRINTING;        // calling host.$MESSAGE.PRINTING
        else
            {
                exits = new WT_Userexit ('WTSysExit');
                if (!host_system.PRINTFILE_NAME)
                    WT_SYSTEM._printFile = WT_SYSTEM.BASEDIR + "/tmp/" +
WT_SYSTEM.SESSION + "/" + host.$MESSAGE.PRINTFILE_NAME;
                else
                    {
                        var i = host_system.PRINTFILE_NAME.replace(/\\/g, '/')
                    .lastIndexOf('/');
                        files = exits.Getdir (host_system.PRINTFILE_NAME.substring(0,i),
host_system.PRINTFILE_NAME.substring(i+1) + '*');
                        if (files)
                            {
                                filesArray = files.split('\n');
                                WT_SYSTEM._printFile =
host_system.PRINTFILE_NAME.substring(0,i+1)+ filesArray[0];
                            }
                        else
                            WT_SYSTEM._printFile = WT_SYSTEM.BASEDIR + "/tmp/" +
WT_SYSTEM.SESSION + "/" + host.$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-Bypass-Druckdatei zum Ausdruck ansteht, setzt WebTransactions das Attribut `BYPASS` des verbindungs-spezifischen Systemobjekts auf "Yes". Wird auf diese Weise eine Druckanforderung angezeigt, wird in `wtasync.htm` zunächst der Wert von `BYPASS` 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 `BYPASS` **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.

Drucker mit anderem Namen als Terminal

In diesem Fall können Sie auf ein existierendes Druckserver-Produkt für den Host-to-LAN-Druck zurückgreifen. Hiermit können Text- und Grafik-Dateien gedruckt werden.



Diese Lösung ist nur im Intranet einsetzbar. Außerdem muss der Druckertyp (PCL, Postscript...) entsprechend im OSD-System konfiguriert sein.

Einsatz eines existierenden Druckserver-Produkts für den Host-to-LAN-Druck

Um Druckaufträge von OSD-Anwendungen auf einen bestimmten LAN-Drucker zu schicken, müssen Sie einen Druckserver auf dem PC installieren, an dem der Drucker angeschlossen ist. Auf diese Weise sind Netzwerk-Drucker für Host-kontrollierte Druckfunktionen zugänglich. Auf der OSD-Seite gilt als Software-Voraussetzung das Produkt RSO V3.0A oder Folgeversionen.

Die Konfiguration der Druckserver ist im [Abschnitt „Konfiguration der Druckserver für Windows“ auf Seite 210](#) beschrieben.

Zuordnung eines Druckers zu einem WebBrowser-Client

Um die Druckinformation auf einem Drucker auszugeben, der sich „in der Nähe“ des Anwenders befindet, muss die Konfiguration der OSD-Anwendung eine Namenskonvention verwenden, die diese Korrelation in irgendeiner Weise widerspiegelt. Beispielsweise könnte der Drucker, der sich in der Nähe eines Terminals mit dem Stationsnamen STATABCD befindet, mit dem Namen PRNABCD konfiguriert sein. Die Zuordnungskonvention könnte also wie folgt lauten: Wurde der Druckauftrag von einem Terminal mit dem Stationsnamen STATxxxx gestartet, dann schicke den Ausdruck an den Drucker PRNxxxx.

Damit eine solche Zuordnungskonvention auch beim Zugriff über das Web ausgenutzt werden kann, müssen Sie sicherstellen, dass ein bestimmter Anwender am Browser einen bestimmten, individuellen Stationsnamen verwendet, der im OSD-System konfiguriert ist. Innerhalb von WebTransactions-Templates sind der Host-Name und die IP-Adresse des Rechners, auf dem der Browser läuft, über Attribute des Systemobjekts zugänglich. Bei der CGI-Schnittstelle sind dies folgende Attribute:

WT_SYSTEM.CGI.REMOTE_HOST enthält den Host-Namen

WT_SYSTEM.CGI.REMOTE_ADDR enthält die IP-Adresse

Dadurch kann in den WebTransactions-Templates jedem Browser-Rechner ein spezieller Stationsname zugeordnet werden.

Beispiel 1

Im folgenden Beispiel werden die WebTransactions-Clients (Rechner, die über Web-Browser auf die WebTransactions-Anwendung zugreifen) durch ihre Stationsnamen identifiziert. Jedem WebTransactions-Client wird ein spezieller Stationsname zugeordnet, der im OSD-System konfiguriert wurde.

Name des WebTransactions Client (Host-Name des Rechners,auf dem der Browser läuft)	Stationsname
D241PBOH	STATPBOH
D241PCRS	STATPCRS
D241PDAN	STATPDAN

Entsprechendes WTScript (ins Start-Template einzufügen):

```
<wtOnCreateScript>
  WT_SYSTEM.STATION_NAME = "STAT" + WT_SYSTEM.CGI.REMOTE_HOST.substring(4,4)
</wtOnCreateScript>
```

Beispiel 2

Im folgenden Beispiel werden die WebTransactions-Clients (Rechner, die über Web-Browser auf die WebTransactions-Anwendung zugreifen) durch ihre IP-Adresse identifiziert.

IP-Adresse des WebTransactions Client (IP-Adresse des Rechners, auf dem der Browser läuft)	Stationsname
133.125.22.3	STAT001
133.122.54.6	STAT002
135.12.8.1	STAT003

Entsprechendes WTScript (ins Start-Template einzufügen):

```
<wtOnCreateScript>
  if (WT_SYSTEM.CGI.REMOTE_ADDR == "133.125.22.3")
    WT_SYSTEM.STATION_NAME = STAT001;
  else if (WT_SYSTEM.CGI.REMOTE_ADDR == "133.122.54.6")
    WT_SYSTEM.STATION_NAME = STAT002;
  else if (WT_SYSTEM.CGI.REMOTE_ADDR == "135.12.8.1")
    WT_SYSTEM.STATION_NAME = STAT003;
</wtOnCreateScript
```

Zuordnungsproblem bei Web-Anschluss über Proxy-Server

Das oben geschilderte Zuordnungsverfahren funktioniert jedoch nicht, wenn der Browser-Rechner über einen Proxy-Server mit WebTransactions verbunden ist. In diesem Fall steht in den Attributen `CGI.REMOTE_HOST` und `CGI.REMOTE_ADDR` lediglich der Name und die IP-Adresse des Proxy-Servers.

Dieses Problem können Sie lösen, indem Sie einzelne Benutzer über Kennungen identifizieren. Hierfür integrieren Sie ins Start-Template eine Abfrage, die den Benutzer zur Eingabe einer Kennung auffordert.

Beispiel

```

<html>
<head>
<title>Identification Page</TITLE>
</head>
<body style="font-family: courier" bgcolor="#C0C0C0">
<wtDataForm name="Identify">

Please identify yourself

<input type = "TEXT" name = "USERID" size = "4" maxlength = "4" >

</wtDataForm>
</body>
</html>

<wtOnReceiveScript>

WT_SYSTEM.STATION_NAME= "STAT"  + WT_POSTED.USERID;

..... open application with the correct station name .....
</wtOnReceiveScript>

```

9.4.3 Browserdarstellungs-Druck

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

Vom Anwender ausgelöster Browserdarstellungs-Druck

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

Von der Host-Anwendung ausgelöster Browserdarstellungs-Druck

Der Ausdruck der im Browser dargestellten Informationen wird vom Netscape Navigator automatisch ausgelöst, wenn in einem client-seitigen JavaScript die Methode `self.print()` aufgerufen wird.

Dies können Sie dadurch erreichen, indem Sie in ein `wtOnCreate`-Script folgende Anweisung integrieren:

```
document.write ("<SCRIPT>self.print();</SCRIPT>");
```

9.4.4 Ausgelieferte Druckfunktionen (Browser-Plattform Windows)

WebTransactions enthält verschiedene Druckfunktionen, die Sie für den Terminal-Hardcopy-Druck (siehe [Seite 190](#)) und den Ausdruck von Host-Daten (siehe [Seite 193](#)) einsetzen können:

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

Das Template `wtKeysOSD.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>`.

9.4.4.1 Browser-Druck

Die Funktion für den Browser-Druck ist als Parameter des Inline WtBean `wtcOSD` implementiert (siehe [Abschnitt „Neues OSD-Kommunikationsobjekt anlegen \(wtcOSD\)“ auf Seite 176](#) und [WebTransactions-Handbuch „Konzepte und Funktionen“](#)).

Im WtBean `wtcOSD` 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.

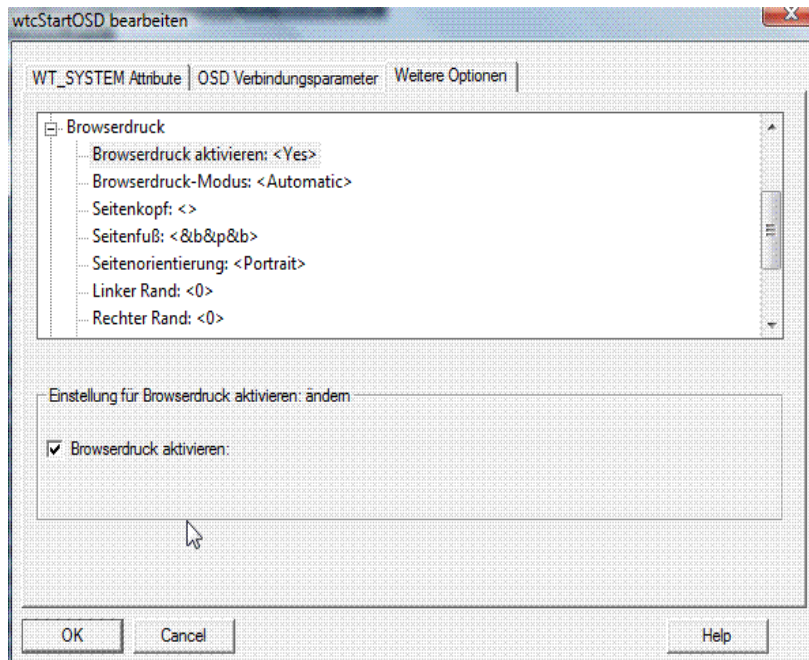
Einstellungen im Start-Template

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

- ▶ Wählen Sie z.B. im Dialogfeld **wtcStartOSD 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 Explorers 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](#)“ auf Seite 201)

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

Für den Internet Explorer können Sie 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* von *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>"
```

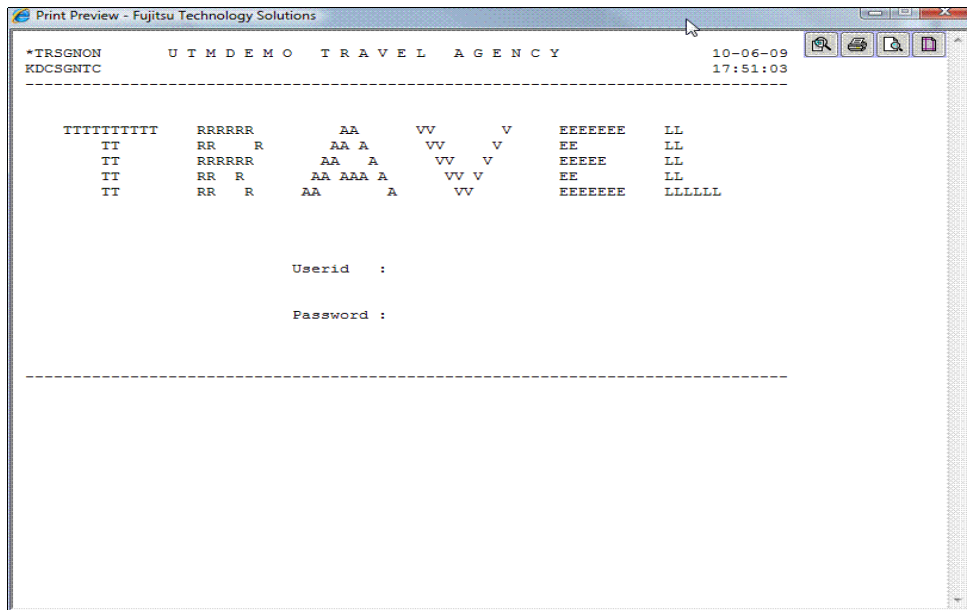
ersetzt die Steuerzeichenfolge durch einen vorher im Template definierten CSS-Seitenumbruch.

- 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 202).



druckt bei Verwendung des Internet Explorers 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.

9.4.4.2 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.

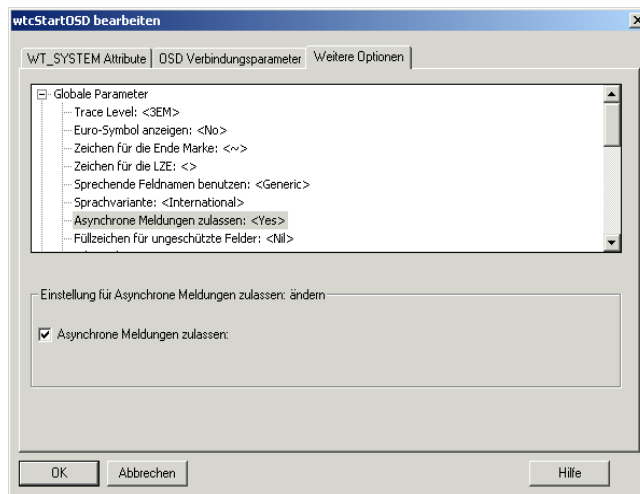
Die Aktivierung des Druck-Plugin `WTAPrint` ist als Parameter des Inline `WTBean` `wtcOSD` implementiert (siehe [Abschnitt „Neues OSD-Kommunikationsobjekt anlegen \(wtcOSD\)“](#) auf [Seite 176](#) und WebTransactions-Handbuch „Konzepte und Funktionen“).

Einstellungen im Start-Template

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

- ▶ Wählen Sie z.B. im Dialogfeld **wtcStartOSD 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/bypass-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 eines 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.

Diese Datei muss folgenden Aufbau haben:

```
[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 206](#)).

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 206](#)).

charset

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

Für die verschiedenen Zeichensätze sind die folgenden Identifizier 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 Fonts, 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.

Beispiele

1. Konfiguration für Terminal-Hardcopy-Druck:

MIME-Typ:

```
webta/hardcopy-print
```

Zugeordnetes Programm:

```
WTAPrint.exe "%1" HARDCOPY 0
```


2. Konfiguration für Bypass-Druck:

MIME-Typ:

webta/bypass-print

Zugeordnetes Programm:

WTAPrint.exe "%1" BYPASS 0

WTAPrint.exe "%1" BYPASS 3 "C:\TEMP\PRINT FILE"

3. Konfiguration für Bypass-Druck mit Konvertierung:

MIME-Typ:

webta/bypass-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 (xxx=...) mehrfach auftritt, dann wird die letzte dieser Zeichenfolgen zum Ersetzen verwendet.

9.4.5 Konfiguration der Druckserver für Windows

Um Netzwerk-Drucker für OSD-Anwendungen zugänglich zu machen, müssen Sie einen Druckserver auf dem Client-PC installieren und konfigurieren. Ein Druckserver ist auf der CD-Rom mit Windows enthalten.

Die folgende Abbildung zeigt die Arbeitsschritte, die auf OSD- und Windows-Seite ausgeführt werden müssen:

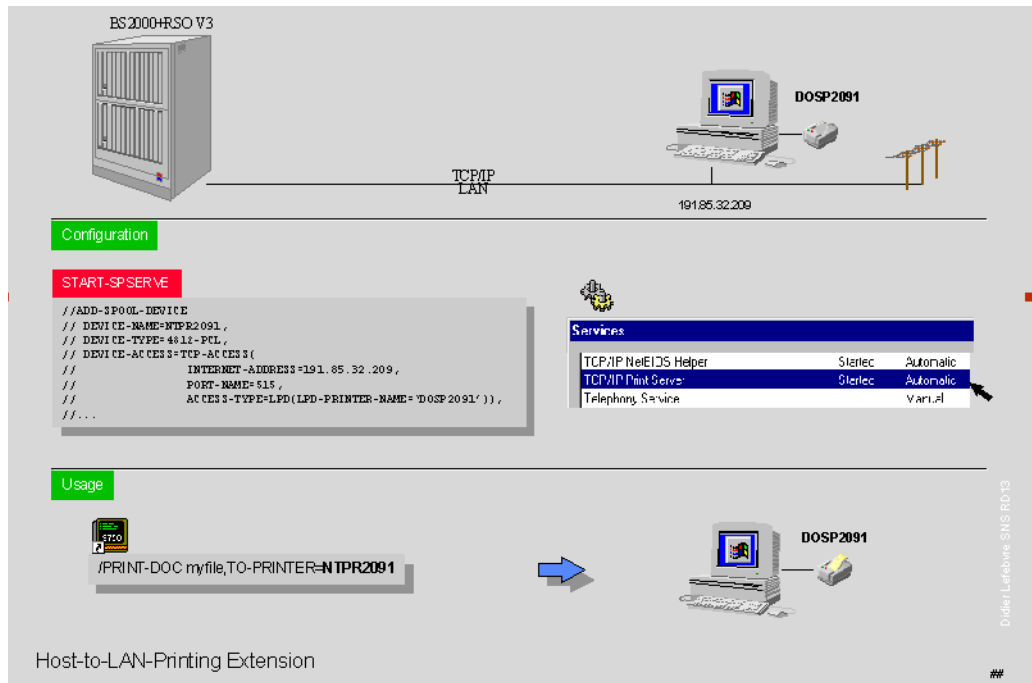


Bild 5: Arbeitsschritte zum Einrichten eines Druckers

OSD-Konfiguration mit dem RSO-Dienstprogramm SPSEVERE

DEVICE-NAME

Stationsname des Druckers für die OSD-Anwendung

DEVICE-TYPE

Druckertyp, z.B. PCL4, Postscript...

INTERNET-ADDRESS

IP-Adresse des Client-PC

LPD-PRINTER-NAME

Name des Druckers auf dem Client-PC

Windows:

- ▶ Falls es noch keinen Dienst „TCP/IP Druckdienste“ gibt, installieren Sie diesen Dienst über
Start/Einstellungen/Systemsteuerung/Software/Windows-Komponenten hinzufügen/entfernen/Weitere Datei- und Druckdienste für das Netzwerk
bzw.
/Details/Druckdienste für Unix
- ▶ Aktivieren Sie den Dienst „TCP/IP Druckdienste“.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *->kursiver* Schrift ausgezeichnet.

aktiver Dialog

Beim aktiven Dialog greift WebTransactions aktiv in die Steuerung des Dialogablaufs ein, d.h., das nächste zu verarbeitende *->Template* wird von der Template-Programmierung bestimmt. Mit den *->WTML*-Sprachmitteln können Sie z.B. mehrere *->Host-Formate* in einer *->HTML*-Seite zusammenfassen. Dabei wird am Ende eines Host- *->Dialogschritts* keine Ausgabe an den *->Browser* geschickt, sondern unmittelbar der Folgeschritt gestartet. Ebenso sind innerhalb **eines** Host-Dialogschritts mehrere Interaktionen zwischen Web- *->Browser* und WebTransactions möglich.

Array

->Datentyp, der eine endliche Menge von Werten eines Datentyps enthalten kann. Der Datentyp kann sein

- *->skalar*
- eine *->Klasse*
- ein Array

Die Werte im Array werden durch einen numerischen Index angesprochen, der mit 0 beginnt.

Asynchrone Nachricht

Versteht WebTransactions als Nachricht, die ans Terminal geschickt wird, ohne dass sie vom Anwender ausdrücklich angefordert worden wäre - d.h. ohne dass der Anwender auf irgendeine Taste gedrückt oder auf ein Oberflächenelement geklickt hätte.

Attribut

Definiert eine Eigenschaft eines *->Objekts*.

Ein Attribut kann z.B. Farbe, Größe oder Position eines Objekts oder selbst wieder ein Objekt sein. Attribute werden auch als *->Variablen* verstanden und können abgefragt und verändert werden.

Aufrufseite

Eine ->*HTML*-Seite, die Sie benötigen, um eine ->*WebTransactions-Anwendung* zu starten. Auf dieser Seite steht der Aufruf, der *WebTransactions* mit dem ersten ->*Template* startet, dem Start-Template.

Ausdruck

Kombination aus ->*Literalen*, ->*Variablen*, Operatoren und Ausdrücken, deren Auswertung jeweils ein bestimmtes Ergebnis liefert.

Auswertungsoperator

WebTransactions versteht den Auswertungsoperator als Operator, der die angesprochenen ->*Ausdrücke* durch ihr Ergebnis ersetzt (Objekt-Attribut-Auswertung). Der Auswertungsoperator wird in der Form `##ausdruck#` angegeben.

Automask-Template

Ein *WebTransactions*- ->*Template*, das von *WebLab* implizit beim Erzeugen eines Basisverzeichnisses oder explizit mit dem Befehl **Automask erzeugen** erstellt wird. Es wird verwendet, wenn kein formatspezifisches Template identifiziert werden kann. Ein Automask-Template enthält die Anweisungen, die für die dynamischen Formatabbildungen und zur Kommunikation notwendig sind. Es können verschiedene Varianten von Automask-Templates erstellt und über das System-Objekt-Attribut `AUTOMASK` ausgewählt werden.

Basisverzeichnis

Das Basisverzeichnis liegt auf dem *WebTransactions*-Server und ist die Grundlage einer ->*WebTransactions-Anwendung*. Im Basisverzeichnis liegen die ->*Templates* und alle Dateien oder Verweise auf Programme (Links), die für den Ablauf einer *WebTransactions-Anwendung* benötigt werden.

BCAM-Applikationsname

Entspricht dem *openUTM*-Generierungsparameter `BCAMAPPL` und ist der Name der ->*openUTM-Anwendung*, über den ->*UPIC* die Verbindung aufnehmen kann.

Benutzerkennung

Bezeichner für einen Benutzer. Einer Benutzerkennung können ein ->*Passwort* (zur ->*Zugangskontrolle*) und Zugriffsrechte (->*Zugriffskontrolle*) zugeordnet werden.

Berechtigungsprüfung

siehe ->*Zugangskontrolle*.

Browser

Programm, das zum Abrufen und Darstellen von ->*HTML*-Seiten erforderlich ist. Browser sind z.B. Microsoft Internet Explorer oder Mozilla Firefox.

Browser-Plattform

Betriebssystem des Rechners, auf dem ein ->*Browser* als Client für WebTransactions läuft.

Browserdarstellungs-Druck

Beim Browserdarstellungs-Druck von WebTransactions wird die im ->*Browser* dargestellte Information ausgedruckt.

Capture-Verfahren

Damit WebTransactions in der Ablaufphase die empfangenen ->*Formate* identifizieren kann, können Sie während einer ->*Sitzung* in WebLab für jedes Format einen bestimmten Bereich markieren und das Format benennen. Der Formatname und das ->*Erkennungskriterium* werden in der ->*Capture-Datenbank* gespeichert. Für das Format wird ein ->*Template* unter gleichem Namen generiert. Das Capture-Verfahren ist die Grundlage für die Bearbeitung formatspezifischer Templates für die Liefereinheiten WebTransactions for OSD und MVS.

Capture-Datenbank

Die Capture-Datenbank von WebTransactions enthält alle Formatnamen und die zugehörigen ->*Erkennungskriterien*, die mit dem ->*Capture-Verfahren* erzeugt wurden. Reihenfolge und Erkennungskriterien der Formate können mit WebLab bearbeitet werden.

CGI

(Common Gateway Interface)

Normierte Schnittstelle für den Programmaufruf auf ->*Web-Servern*. Im Gegensatz zur statischen Ausgabe einer zuvor festgelegten ->*HTML-Seite* ermöglicht diese Schnittstelle den dynamischen Aufbau von HTML-Seiten.

Client

Anforderer und Nutzer von Diensten.

Cluster

Menge von identischen ->*WebTransactions-Anwendungen* auf verschiedenen Servern, die zu einem Lastverbund zusammengeschlossen sind.

Dämon

Bezeichnung für einen Prozesstyp in Unix-/POSIX-Systemen, der keine Ein-/Ausgaben auf Terminals durchführt und im Hintergrund abläuft.

Datentyp

Festlegung, wie der Inhalt eines Speicherplatzes zu interpretieren ist. Ein Datentyp hat einen Namen, eine Menge zulässiger Werte (Wertebereich) und eine bestimmte Anzahl von Operationen, die die Werte dieses Datentyps interpretieren und manipulieren.

Dialog

Beschreibt die gesamte Kommunikation zwischen Browser, WebTransactions und ->*Host-Anwendung*. Er umfasst in der Regel mehrere ->*Dialogzyklen*. Bei WebTransactions werden mehrere Dialogarten unterschieden:

- ->*passiver Dialog*
- ->*aktiver Dialog*
- ->*synchronisierter Dialog*
- ->*nicht-synchronisierter Dialog*

Dialogzyklus

Zyklus, der beim Ablauf einer ->*WebTransactions-Anwendung* folgende Schritte umfasst:

- eine ->*HTML-Seite* aufbauen und an den ->*Browser* schicken
- auf Antwort vom Browser warten
- Antwortfelder auswerten und evtl. zur Weiterverarbeitung an die ->*Host-Anwendung* schicken

Während des Ablaufs einer ->*WebTransactions-Anwendung* werden mehrere Dialogzyklen durchlaufen.

Distinguished Name

Der Distinguished Name (DN) in ->*LDAP* setzt sich hierarchisch aus mehreren Teilen zusammen (z.B. „Land, unterhalb von Land: Organisation, unterhalb von Organisation: Organisationseinheit, darunter: Gebräuchlicher Name“). Die Summe dieser Teile identifiziert ein Objekt innerhalb des Directory-Baums eindeutig.

Durch diese Hierarchie wird die eindeutige Benennung von Objekten selbst in einem weltweiten Directory-Baum sehr einfach:

- Der DN "Land=DE/Name=Emil Mustermann" reduziert das Eindeutigkeits-Problem auf das Land DE.
- Der DN "Organisation=FTS/Name=Emil Mustermann" reduziert es auf die Organisation FTS.
- Der DN "Land=DE/Organisation=FTS/Name=Emil Mustermann" reduziert es auf die Organisation FTS innerhalb des Landes DE.

Dokumentenverzeichnis

Verzeichnis des ->*Web-Servers*, in dem Dokumente liegen, auf die über das Netz zugegriffen werden kann. WebTransactions legt in diesem Verzeichnis Dateien zum Herunterladen ab, wie z.B. den WebLab-Client oder allgemeine Start-Seiten.

Domain Name Service (DNS)

Verfahren zur symbolischen Adressierung von Rechnern in Netzen. Bestimmte Rechner im Netz, die DNS- oder Name-Server, führen eine Datenbank mit allen bekannten Rechnernamen und IP-Nummern in ihrer Umgebung.

Dynamische Daten

Werden in WebTransactions durch das WebTransactions-Objektmodell abgebildet, z.B. als ->*Systemobjekt*, Host-Objekt oder Nutzereingaben am Browser.

Eigenschaft

Definiert die Beschaffenheit von ->*Objekten*, z.B. könnten Kundename und Kundennummer Eigenschaften eines Objekts „Kunde“ sein. Diese Eigenschaften können innerhalb des Programms gesetzt, abgefragt und verändert werden.

EJB

(Enterprise JavaBean)

Industriestandard auf Basis von Java, mit dem innerhalb einer verteilten, objektorientierten Umgebung selbstentwickelte oder auf dem Markt gekaufte Server-Komponenten zur Erstellung von verteilten Programmsystemen genutzt werden können.

EHLAPI

(Enhanced High Level Language API)

Programmschnittstelle z.B. von Terminal-Emulationen für die Kommunikation mit der SNA-Welt. Auf dieser Schnittstelle basiert die Kommunikation zwischen Transit-Client und dem SNA-Rechner, die über das Produkt TRANSIT abgewickelt wird.

Erkennungskriterium

Über Erkennungskriterien werden ->*Formate* einer ->*Terminal-Anwendung* identifiziert und Sie können auf die Daten des Formats zugreifen. Als Erkennungskriterium wählen Sie jeweils einen oder auch mehrere Bereiche des Formats, deren Inhalt das Format eindeutig identifiziert.

Felddatei (*.fld-Datei)

Enthält in WebTransactions die Struktur des Datensatzes eines ->*Formats* (Metadaten).

FHS

(Format **H**andling **S**ystem)
Formatierungssystem für BS2000/OSD-Anwendungen.

Field

Kleinster Teil eines ->*Service* und Element eines ->*Records* oder ->*Puffers*.

Filter

Programm oder Programmteil (z.B. eine Bibliothek) zur Umsetzung eines Formats in ein anderes (z.B. XML-Dokumente in ->*WTS*cript-Datenstrukturen).

Format

Optische Darstellung auf alphanumerischen Bildschirmen, wird auch Maske oder Schirm genannt.

In WebTransactions wird ein Format jeweils durch eine ->*Felddatei* und ein Template repräsentiert.

Formatbeschreibungquellen

Beschreibung mehrerer ->*Formate* in einer oder mehreren Dateien, die aus einer Format-Bibliothek (FHS/IFG) erzeugt wurden oder direkt am ->*Host* vorliegen für die Nutzung „sprechender“ Namen in Formaten.

Formattyp

(nur relevant bei FHS-Anwendungen und Kommunikation über UPIC)
Spezifiziert den Typ des Formats: #Format, +Format, -Format oder *Format.

Funktion

Benutzerdefinierte Code-Teile mit einem Namen und ->*Parametern*. Durch eine Beschreibung der Funktionsschnittstelle (oder Signatur) können Funktionen in Methoden aufgerufen werden.

Holder Task

Prozess, Task oder Thread in WebTransactions, je nach Betriebssystem-Plattform. Die Anzahl der Tasks entspricht der Anzahl der Benutzer. Die Task wird beendet, wenn sich der Benutzer abmeldet oder durch Timeout. Ein Holder Task entspricht genau einer ->*WebTransactions-Sitzung*.

Host

Rechner, auf dem die ->*Host-Anwendung* läuft.

Host-Adapter

Dienen dazu, bestehende ->*Host-Anwendungen* an WebTransactions anzuschließen. Sie sorgen zur Laufzeit z.B. für den Auf- und Abbau von Verbindungen und für die Umsetzung der ausgetauschten Daten.

Host-Anwendung

Anwendung, die mit WebTransactions integriert ist.

Host-Plattform

Betriebssystem des Rechners, auf dem die ->*Host-Anwendung* läuft.

Host-Daten-Druck

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

Host-Datenobjekt

Bezeichnet in WebTransactions ein ->*Objekt* der Datenschnittstelle zur ->*Host-Anwendung*, das ein Feld mit all seinen Feldattributen repräsentiert. Es wird von WebTransactions nach dem Empfang von Daten der Host-Anwendung angelegt und existiert bis zum nächsten Datenempfang oder bis zum Beenden der ->*Sitzung*.

Host-Steuerobjekt

In WebTransactions enthalten Host-Steuerobjekte Informationen, die nicht nur ein einzelnes Feld betreffen, sondern das gesamte ->*Format*. Dazu gehören z.B. das Feld, in dem sich der Cursor befindet, die aktuelle Funktionstaste oder globale Formatattribute.

HTML

(Hypertext Markup Language)
Siehe ->*Hypertext Markup Language*

HTTP

(Hypertext Transfer Protocol)
Protokoll zur Übertragung von ->*HTML*-Seiten und Daten.

HTTPS

(Hypertext Transfer Protocol Secure)
Protokoll zur gesicherten Übertragung von ->*HTML*-Seiten und Daten.

Hypertext

Dokument mit Verweisen auf andere Stellen im gleichen oder in anderen Dokumenten, in die z.B. durch Anklicken mit der Maus gesprungen werden kann.

Hypertext Markup Language

Standardisierte Auszeichnungssprache für Dokumente im WWW.

JavaBean

Java-Programm (oder ->*Klasse*) mit genau festgelegten Konventionen für die Schnittstellen, die eine Wiederverwendung in mehreren Anwendungen ermöglichen.

KDCDEF

openUTM-Werkzeug für die Generierung von ->*openUTM-Anwendungen*.

Klasse

Enthält die Definition der ->*Eigenschaften* und ->*Methoden* eines ->*Objekts*. Sie ist das Modell für die Instanziierung von Objekten und definiert deren Schnittstellen.

Klassen-Template

Ein Klassen-Template in WebTransactions enthält für die gesamte Objektklasse (z. B. Eingabe- oder Ausgabefeld) gültige, immer wiederkehrende Anweisungen. Klassen-Templates werden durchlaufen, wenn auf ein ->*Host-Datenobjekt* der ->*Auswertungoperator* oder die *toString*-Methode angewendet wird.

Kommunikationsobjekt

Steuert eine Verbindung zu einer ->*Host-Anwendung* und enthält Information über den aktuellen Zustand der Verbindung, über die zuletzt empfangenen Daten etc.

Konvertierungswerkzeuge

Dienstprogramme, die mit WebTransactions ausgeliefert werden. Mit den Konvertierungswerkzeugen werden die Datenstrukturen von ->*openUTM-Anwendungen* analysiert und in Dateien abgelegt. Diese Dateien können Sie dann in WebLab als ->*Formatbeschreibungsqellen* verwenden, um WTML-Templates und ->*FLD-Dateien* zu generieren.

Die Basis für die Konvertierung können Cobol-Datenstrukturen oder IFG-Formatbibliotheken sein. Für Drive-Programme wird das Konvertierungswerkzeug mit dem Produkt Drive ausgeliefert.

LDAP

(Lightweight **D**irectory **A**ccess **P**rotocol)

Der X.500-Standard definiert als Zugriffsprotokoll DAP (Directory Access Protocol). Speziell für den Zugang zu X.500-Verzeichnisdiensten vom PC aus hat sich jedoch der Internet-Standard LDAP durchgesetzt.

Bei LDAP handelt es sich um ein vereinfachtes DAP-Protokoll, das nicht alle Möglichkeiten von DAP zulässt und mit DAP nicht kompatibel ist. Praktisch alle X.500-Verzeichnisdienste unterstützen neben DAP auch LDAP. In der Praxis kann es zu Verständigungsproblemen kommen, da es diverse Dialekte von LDAP gibt. Die Unterschiede der Dialekte sind in der Regel gering.

Literal

Zeichenfolge, die einen festen Wert repräsentiert. Literale dienen dazu, in Source-Programmen konstante Werte unmittelbar anzugeben („wörtliche“ Wertangabe).

Master-Template

WebTransactions-Template, das als Schablone für die Generierung der Automask und der formatspezifischen-Templates verwendet wird.

Message Queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete Warteschlangen (Queues) ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen, die Übermittlung der Nachricht wird garantiert, unabhängig davon, ob gerade eine Netzverbindung besteht oder nicht.

Methode

Objektorientierter Begriff für *->Funktion*. Eine Methode wirkt auf das *->Objekt*, in dem sie definiert ist

Modul-Template

Dient in WebTransactions dazu, *->Klassen*, *->Funktionen* und Konstanten global für eine komplette *->Sitzung* zu definieren. Ein Modul-Template wird mit Hilfe der Funktion `import()` geladen.

MT-Tag

(Master-Template-Tag)

Spezielle Tags in *->Master-Templates* für die dynamischen Teile eines Master-Templates.

Multi-Tier-Architektur

Allen Client-/Server-Architekturen liegt eine Gliederung in einzelne Software-Komponenten, auch Schichten oder Tiers genannt, zugrunde: Man spricht von 1-Tier, 2-Tier-, 3-Tier und auch von Multi-Tier-Modellen. Man kann die Aufgliederung auf der physischen oder der logischen Ebene betrachten:

- Logische Software-Tiers liegen vor, wenn die Software in modulare Komponenten mit klaren Schnittstellen gegliedert ist.
- Physische Tiers liegen dann vor, wenn die (logischen) Softwarekomponenten im Netz auf verschiedene Rechner verteilt sind.

Mit WebTransactions sind Multi-Tier-Modelle sowohl auf physischer als auch logischer Tiers-Ebene möglich.

Name/Value-Paar

In den vom ->*Browser* geschickten Daten die Kombination z.B. von einem ->*HTML*-Eingabefeldnamen mit seinem Wert.

nicht-synchronisierter Dialog

Der nicht-synchronisierte Dialog von WebTransactions erlaubt es, den Prüfmechanismus des ->*synchronisierten Dialogs* zeitweise auszuschalten. So lassen sich ->*Dialoge* zwischenschieben, die außerhalb des synchronisierten Dialogs liegen und keinen Einfluss auf den logischen Zustand der ->*Host-Anwendung* haben. Dadurch können Sie z.B. in einer ->*HTML*-Seite eine Schaltfläche anbieten, um Hilfeinformationen aus der laufenden Host-Anwendung anzufordern und in einem separaten Fenster anzuzeigen.

Objekt

Elementare Einheit innerhalb eines objektorientierten Softwaresystems. Jedes Objekt hat einen Namen, über den es angesprochen werden kann, ->*Attribute*, die seinen Zustand definieren und ->*Methoden*, die auf das Objekt angewandt werden können.

openUTM

(Universal Transaction Monitor)

Transaktionsmonitor von Fujitsu Technology Solutions, verfügbar für BS2000/OSD, verschiedenste Unix- und Windows-Plattformen.

openUTM-Anwendung

->*Host-Anwendung*, die Dienstleistungen zur Verfügung stellt, die Aufträge von Terminals, ->*Client-Programmen* oder anderen Host-Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine openUTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Einheit bildet.

Mit openUTM-Anwendungen kann sowohl über das Client/Server-Protokoll ->*UPIC* als auch über die Terminal-Schnittstelle (9750) kommuniziert werden.

openUTM-Client (UPIC)

Mit dem Produkt openUTM-Client (UPIC) können Sie Client-Programme für openUTM erstellen. openUTM-Client (UPIC) steht z.B. für Unix-, BS2000/OSD- und Windows-Plattformen zur Verfügung.

openUTM-Teilprogramm

Die Dienste einer ->*openUTM-Anwendung* werden durch ein oder mehrere openUTM-Teilprogramme realisiert. Sie sind über ->*Transaktionscodes* ansprechbar und enthalten spezielle openUTM-Funktionsaufrufe (z.B. KDCS-Aufrufe).

Parameter

Daten, die an eine ->*Funktion* oder ->*Methode* zur Verarbeitung übergeben werden (Eingabeparameter) oder Daten, die als Ergebnis von einer Funktion oder Methode zurückgeliefert werden (Ausgabeparameter).

passiver Dialog

Beim passiven Dialog von WebTransactions wird der Dialogablauf von der ->*Host-Anwendung* gesteuert, d.h., die Host-Anwendung bestimmt das nächste zu verarbeitende ->*Template*. Ein Anwender, der über WebTransactions auf die Host-Anwendung zugreift, durchläuft die gleichen Schritte wie beim Zugriff über ein Terminal. Passive Dialogsteuerung verwendet WebTransactions bei einer automatischen Umsetzung der Host-Anwendung oder wenn jedes Format der Host-Anwendung genau einem individuellen Template entspricht.

Passwort

In einer Anwendung für eine ->*Benutzererkennung* eingetragene Zeichenkette zur Authentisierung (->*Zugangsschutz*).

polling

Zyklische Abfrage auf Zustandsänderungen.

Pool

WebTransactions bezeichnet hiermit ein freigegebenes Verzeichnis, in dem WebLab ->*Basisverzeichnisse* anlegen und pflegen kann. Den Zugriff auf dieses Verzeichnis steuern Sie mit der Administration.

Posted-Objekt (wt_Posted)

Enthält in WebTransactions eine Liste der vom ->*Browser* zurückgeschickten Daten. Dieses ->*Objekt* wird von WebTransactions angelegt und lebt nur für die Dauer eines ->*Dialogzyklus*.

posten

Daten versenden

Projekt

Enthält in der WebTransactions-Entwicklungsumgebung verschiedene Einstellungen einer ->*WebTransactions-Anwendung*, die in einer Projektdatei (Endung .wtp) gespeichert werden. Sie sollten für jede WebTransactions-Anwendung, die Sie entwickeln, ein Projekt anlegen und zum Bearbeiten immer dieses Projekt öffnen.

Protokoll

Vereinbarungen über Verhaltensregeln und Formate bei der Kommunikation unter entfernten Partnern gleichen logischen Niveaus.

Protokolldatei

- openUTM-Client: Datei, in die bei abnormalem Beenden einer Conversation openUTM-Fehlermeldungen geschrieben werden.
- In WebTransactions werden Protokolldateien als Trace-Dateien bezeichnet.

Prozess

Der Begriff „Prozess“ wird als Oberbegriff für Prozess (Solaris, Linux und Windows) und Task (BS2000/OSD) verwendet.

Puffer

Definition eines Datensatzes, der von einem ->*Service* übertragen wird. Der Puffer dient zum Senden und zum Empfangen von Nachrichten. Zusätzlich gibt es einen speziellen Puffer für die Ablage der ->*Erkennungskriterien* und für Daten zur Darstellung am Bildschirm.

Roaming Session

->*WebTransactions-Sitzung*, die nacheinander oder gleichzeitig von verschiedenen ->*Clients* aus angesprochen werden kann.

Record

Definition eines Datensatzes, der in einem ->Puffer übertragen wird. Er beschreibt einen Teil des Puffers, der ein- oder mehrfach vorkommen kann.

Service-Anwendung

->WebTransactions-Sitzung, die abwechselnd von verschiedenen Benutzern aufgerufen werden kann.

Service-Knoten

Instanz eines ->Service. Beim Entwickeln und beim Ablauf einer ->Methode kann ein Service mehrfach instanziiert werden. Beim Modellieren und Code bearbeiten werden diese Instanzen als Service-Knoten bezeichnet.

Sichtbarkeit von Variablen

->Objekte und ->Variablen unterschiedlicher Dialogarten werden von WebTransactions in unterschiedlichen Adressräumen verwaltet. Das bedeutet, dass Variablen eines ->synchronen Dialogs im ->asynchronen Dialog oder im Dialog mit einer entfernten Anwendung nicht sichtbar und damit auch nicht zugreifbar sind.

Sitzung

Beginnt ein Endanwender mit einer ->WebTransactions-Anwendung zu arbeiten, so wird für ihn auf dem WebTransactions-Server eine WebTransactions-Sitzung eingerichtet. Diese Sitzung enthält alle für diesen Benutzer geöffneten Verbindungen zum ->Browser, zu speziellen ->Clients und ->Hosts.

Eine Sitzung kann gestartet werden

- durch Eingabe eines URL von WebTransactions im Browser.
- durch die Methode `START_SESSION` der Client/Server-Schnittstelle `WT_REMOTE`.

Eine Sitzung endet

- mit einer entsprechenden Eingabe des Benutzers im Ausgabebereich dieser ->WebTransactions-Anwendung (nicht über Standard-Buttons des Browsers).
- durch Überschreiten der konfigurierten Zeit, die WebTransactions auf eine Antwort von der ->Host-Anwendung oder vom ->Browser wartet.
- durch Terminierung mit Hilfe der WebTransactions-Administration.
- durch die Methode `EXIT_SESSION` der Client/Server-Schnittstelle `WT_REMOTE`.

Eine WebTransactions-Sitzung ist eindeutig durch eine ->WebTransactions-Anwendung und eine Session Id bestimmt. Während ihrer Lebensdauer existiert zu jeder WebTransactions-Sitzung auf dem WebTransactions-Server genau ein ->Holder Task.

Skalar

->*Variable*, die nur aus einem einzelnen Wert besteht - im Gegensatz zu einer ->*Klasse*, einem ->*Array* oder einer anderen komplexen Datenstruktur.

SOAP

(ursprünglich **S**imple **O**bject **A**ccess **P**rotocol)

Das ->*XML*-basierte SOAP-Protocol realisiert einen einfachen und transparenten Mechanismus, mit dem strukturierte und typisierte Informationen zwischen Rechnern in einer dezentralisierten, verteilten Umgebung ausgetauscht werden können.

SOAP stellt ein modulares Paketmodell sowie Mechanismen zum Verschlüsseln von Daten innerhalb von Modulen zur Verfügung. Dies ermöglicht die unkomplizierte Beschreibung der externen Schnittstellen eines ->*Web-Service*.

Stil

Realisiert in WebTransactions ein anderes Layout für ein ->*Template*, z.B. mit mehr oder weniger Grafikelementen für unterschiedliche ->*Browser*. Der Stil kann während einer ->*Sitzung* jederzeit geändert werden.

synchronisierter Dialog

Beim synchronisierten Dialog (Standardfall) überprüft WebTransactions automatisch, ob die Daten, die vom Web-Browser eingehen, auch wirklich die Antwort auf die letzte an den ->*Browser* geschickte ->*HTML*-Seite sind. Wenn z.B. der Anwender am Web-Browser über die Schaltfläche **Zurück** oder die History-Funktion zu einer „alten“ HTML-Seite der aktuellen ->*Sitzung* wechselt und diese zurückschickt, erkennt WebTransactions, dass die Daten nicht zum aktuellen ->*Dialogzyklus* passen und reagiert mit einer Fehlermeldung. Die zuletzt an den Browser gesendete Seite wird automatisch erneut an den Browser geschickt.

Systemobjekt (*wt_System*)

Das Systemobjekt von WebTransactions enthält ->*Variablen*, die während einer gesamten ->*Sitzung* existieren und erst am Ende einer Sitzung oder durch explizites Löschen wieder entfernt werden. Es ist immer sichtbar und identisch für alle Namensräume.

TAC

Siehe ->*Transaktionscode*

Tag

->*HTML*-, ->*XML*- und ->*WTML*-Dokumente bestehen aus Tags und dem eigentlichen Inhalt. Mit den Tags werden Auszeichnungen im Dokument durchgeführt z.B. Überschriften, Text Hervorhebungen (fett, kursiv) oder Quellangaben für Grafikdateien.

TCP/IP

(Transport **C**ontrol **P**rotocol/**I**nternet **P**rotocol)

Sammelname für eine Protokollfamilie in Rechnernetzen, die unter anderem im Internet verwendet wird.

Template

Vorlage für die Generierung von spezifischem Code. Ein Template enthält feste Teile, die bei der Generierung unverändert übernommen werden und variable Teile, die bei der Generierung durch die jeweils aktuellen Werte ersetzt werden. Ein Template ist eine ->WTML-Datei mit speziellen Tags zur Steuerung der dynamischen Generierung einer ->HTML-Seite und zur Verarbeitung der am ->Browser eingegebenen Werte. Es können mehrere Sätze von Templates parallel gehalten werden. Diese repräsentieren unterschiedliche Stile (z.B. viel/wenig Grafik, Java-Benutzung etc.).

WebTransactions nutzt verschiedene Arten von Templates:

- ->Automask-Templates für die automatische Umsetzung der ->Formate von MVS- und OSD-Anwendungen
- eigene Templates, die vom Programmierer selbst geschrieben werden, z.B. zur Steuerung eines ->aktiven Dialogs
- formatspezifische Templates, die für eine spätere Nachbearbeitung generiert werden
- Include-Templates, die in andere Templates eingefügt werden
- ->Klassen-Templates
- ->Master-Templates für ein einheitliches Layout fester Bereiche bei der Generierung der Automask und formatspezifischer Templates
- Start-Template, das als erstes Template einer WebTransactions-Anwendung durchlaufen wird

Template-Objekte

->Variablen zur Zwischenspeicherung von Werten für einen ->Dialogzyklus in WebTransactions.

Terminal-Anwendung

Anwendung auf einem ->Host-Rechner, auf die über die 9750- oder 3270-Schnittstelle zugegriffen wird.

Terminal-Hardcopy-Druck

Beim Terminal-Hardcopy-Druck von WebTransactions wird die alphanumerische Darstellung des ->Formats gedruckt, wie es von einem Terminal oder einer Terminal-Emulation dargestellt würde.

Transaktion

Verarbeitungsschritt zwischen zwei Sicherungspunkten (innerhalb eines Vorgangs), der durch die ACID-Bedingungen gekennzeichnet ist (**A**tomicity, **C**onsistency, **I**solation und **D**urability). Die in einer Transaktion beabsichtigten Änderungen an der Anwenderinformation werden entweder alle oder gar nicht durchgeführt (Alles-oder-Nichts-Regel).

Transaktionscode/TAC

Name, über den ein openUTM-Vorgang oder ein ->*openUTM-Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem openUTM-Teilprogramm bei der openUTM-Konfigurierung zugeordnet. Einem Teilprogramm können auch mehrere TACs zugeordnet sein.

UDDI

(**U**niversal **D**escription, **D**iscovery and **I**ntegration)

Umfasst Verzeichnisse, die Beschreibungen von ->*Web-Services* enthalten. Diese Informationen stehen Web-Usern allgemein zur Verfügung.

Unicode

Von der International Standardisation Organisation (ISO) und dem Unicode-Konsortium genormter alphanumerischer Zeichensatz zur Codierung von Zeichen – Buchstaben, Ziffern, Satzzeichen, Silbenzeichen, Sonderzeichen sowie Ideogrammen. Unicode fasst alle weltweit bekannten Textzeichen in einem einzigen Zeichensatz zusammen.

Unicode ist hersteller- und systemunabhängig. Es verwendet Zeichensätze der Länge zwei oder vier Bytes für die Codierung jedes Textzeichens. Diese Zeichensätze werden bei ISO als UCS-2 (Universal Character Set 2) beziehungsweise UCS-4 bezeichnet. Statt der durch ISO definierten Bezeichnung UCS-2 wird häufig die Bezeichnung UTF-16 (Unicode Transformation Format 16 Bit) verwendet, ein vom Unicode-Konsortium definierter Standard.

Neben der Nutzung von UTF-16 ist auch der Einsatz von UTF-8 (Unicode Transformation Format 8 Bit) weit verbreitet. UTF-8 ist inzwischen die globale Zeichen-Codierung im Internet.

UPIC

(**U**niversal **P**rogramming **I**nterface for **C**ommunication)

Trägersystem für openUTM-Clients, das über die X/Open-Schnittstelle CPI-C die Client-Server-Kommunikation zwischen CPI-C-Client-Anwendung und der openUTM-Anwendung ermöglicht.

URI

(**U**niform **R**esource **I**dentifier)

Oberbegriff für alle Namen und Adressen die im Internet Objekte referenzieren. Die allgemein gebräuchlichen URIs sind ->*URLs*.

URL

(Uniform Resource Locator)

Beschreibung von Ort und Zugriffsart einer Ressource im Internet.

Userexit

In C/C++ implementierte Funktion, die der Programmierer aus einem ->*Template* aufruft.

Variable

Speicherplatz für variable Werte, der einen Namen und einen ->*Datentyp* benötigt.

Vorgang

In ->*openUTM* Bearbeitung eines Auftrags durch eine ->*openUTM-Anwendung*. Es gibt Dialog-Vorgänge und Asynchronvorgänge. Dem Vorgang werden von openUTM eigene Speicherbereiche zugeordnet. Ein Vorgang setzt sich aus einer oder mehreren ->*Transaktionen* zusammen.

Web-Server

Rechner und Software zum Bereitstellen von ->*HTML*-Seiten und dynamischen Daten über ->*HTTP*.

Web-Service

Dienst, der im Internet bereitgestellt wird, z.B. ein Währungsumrechnungs-Programm, und über das SOAP-Protokoll angesprochen werden kann. Die Schnittstelle eines Web-Service ist in ->*WSDL* beschrieben.

WebTransactions-Anwendung

Anwendung, die ->*Host-Anwendungen* für den Internet-/Intranet-Zugriff integriert. Eine WebTransactions-Anwendung besteht aus

- einem ->*Basisverzeichnis*
- einem Start-Template
- den ->*Templates*, die die Umsetzung zwischen ->*Host* und ->*Browser* steuern
- protokollspezifischen Konfigurationsdateien

WebTransactions-Plattform

Betriebssystem des Rechners, auf dem WebTransactions läuft.

WebTransactions-Server

Rechner, auf dem WebTransactions läuft.

WebTransactions-Sitzung

Siehe ->*Sitzung*.

WSDL

(Web Services Description Language)

Bietet ->XML-Sprachregeln für die Beschreibung von ->Web-Services. Ein Web-Service wird dabei durch eine Auswahl von Ports definiert.

WTBean

In WebTransactions werden ->WTML-Komponenten mit selbstbeschreibender Schnittstelle als WTBeans bezeichnet. Es wird zwischen inline und standalone WTBeans unterschieden:

- ein inline WTBean entspricht einem Teil eines WTML-Dokuments
- ein standalone WTBean ist ein eigenständiges WTML-Dokument

Verschiedene WTBeans gehören zum Produktumfang von WebTransactions, weitere WTBeans stehen Ihnen auf der WebTransactions-Homepage zum Download zur Verfügung:

ts.fujitsu.com/products/software/openseas/webtransactions.html

WTML

(WebTransactions Markup Language)

Auszeichnungs- und Programmiersprache für WebTransactions ->Templates. WTML besteht aus ->WTML-Tags, die ->HTML erweitern, und der server-seitigen Programmiersprache ->WTScript, die z.B. den Datenaustausch mit ->Host-Anwendungen ermöglicht. WTML wird von WebTransactions und nicht vom ->Browser ausgeführt (serverside scripting).

WTML-Tag

(WebTransactions Markup Language-Tag)

Spezielle Tags von WebTransactions zur Generierung der dynamischen Teile einer ->HTML-Seite mit Daten aus der ->Host-Anwendung.

WTScript

Server-seitige Programmiersprache von WebTransactions. WTScripts stehen ähnlich wie client-seitige JavaScripts in Bereichen, die mit speziellen Tags eingeleitet und beendet werden. Statt ->HTML-SCRIPT-Tags verwenden Sie hierfür jedoch ->WTML-Tags: wtOnCreateScript und wtOnReceiveScript. Damit zeigen Sie an, dass diese Scripts von WebTransactions und nicht vom ->Browser ausgeführt werden sollen und signalisieren zusätzlich den gewünschten Ausführungszeitpunkt. OnCreate-Scripts werden ausgeführt, bevor die Seite an den Browser geschickt wird. OnReceive-Scripts werden erst ausgeführt, nachdem die Antwort vom Browser empfangen wurde.

XML

(e**X**tensible **M**arkup **L**anguage)

Definiert eine Sprache zur logischen Strukturierung von Dokumenten mit dem Ziel, diese einfach zwischen verschiedenen Anwendungen auszutauschen.

XML-Schema

Ein XML-Schema im allgemeinen Sinn definiert die zulässigen Elemente und Attribute einer XML-Beschreibung. XML-Schemata können verschiedene Formate haben, z.B. DTD (**D**ocument **T**ype **D**efinition), XML Schema (**W**3**C**-Standard) oder XDR (**X**ML **D**ata **R**educed).

Zugangskontrolle

Prüfung, ob ein Benutzer berechtigt ist, unter einer bestimmten Benutzerkennung mit der Anwendung zu arbeiten.

Zugriffskontrolle

Überwachung der Zugriffe auf die Daten und ->*Objekte* einer Anwendung.

Abkürzungen

BO	B usiness O bject
CGI	C ommon G ateway I nterface
DN	D istinguished N ame
DNS	D omain N ame S ervice
EJB	E nterprise J ava B ean
FHS	F ormat H andling S ystem
HTML	H ypertext M arkup L anguage
HTTP	H ypertext T ransfer P rotocol
HTTPS	H ypertext T ransfer P rotocol S ecure
IFG	I nteraktiver F ormat G enerator
ISAPI	I nternet S erver A pplication P rogramming I nterface
LDAP	L ightweight D irectory A ccess P rotocol
LPD	L ine P rinter D aemon
MT-Tag	M aster- T emplate- T ag
MVS	M ultiple V irtual S torage
OSD	O pen S ystems D irection
SGML	S tandard G eneralized M arkup L anguage
SOAP	S imple O bject A ccess P rotocol

SSL	S ecure S ocket L ayer
TCP/IP	T ransport C ontrol P rotocol/ I nternet P rotocol
Upic	U niversal P rogramming I nterface for C ommunication
URL	U niform R esource L ocator
WSDL	W eb S ervices D escription L anguage
wtc	W eb T ransactions C omponent
WTML	W eb T ransactions M arkup L anguage
XML	e Xtensible M arkup L anguage

Literatur

WebTransactions-Handbücher

Unter der Web-Adresse <http://manuals.ts.fujitsu.com> stehen Ihnen sämtliche Handbücher zum Download zur Verfügung.

**WebTransactions
Konzepte und Funktionen**
Einführung

**WebTransactions
Template-Sprache**
Referenzhandbuch

**WebTransactions
Client-APIs für WebTransactions**
Benutzerhandbuch

**WebTransactions
Anschluss an openUTM-Anwendungen über UPIC**
Benutzerhandbuch

**WebTransactions
Anschluss an MVS-Anwendungen**
Benutzerhandbuch

**WebTransactions
Zugriff auf dynamische Web-Inhalte**
Benutzerhandbuch

**WebTransactions
Web-Frontend für Web-Services**
Benutzerhandbuch

Sonstige Handbücher

Die Handbücher sind online unter <http://manuals.ts.fujitsu.com> zu finden oder in gedruckter Form gegen gesondertes Entgelt unter <http://manualshop.ts.fujitsu.com> zu bestellen.

openUTM

Anwendungen programmieren mit KDCS für COBOL, C und C++
Benutzerhandbuch

Stichwörter

\$FIRST (Host-Steuerobjekt) 140

\$MESSAGE (Host-Steuerobjekt)
 PRINTFILE_NAME 141
 PRINTING 141
 WAITING 141

\$NEXT (Host-Steuerobjekt) 141

\$SCREEN (Host-Steuerobjekt) 141

7-Bit-ASCII-Zeichensatz 72

9750-Bildschirm,dynamische Umsetzung 77

9750-Terminal 10, 71, 74, 160
 P-Tasten 161

9750-Terminal-Emulation 71

9763-UNICODE 131

A

Aktiver Dialog 213, 216

Aktualisieren

 Basisverzeichnis 68

Anlegen

 Basisverzeichnis 67

 Projekt 35

Anwendung

 starten 75

Anwendungsframe 180

Anwendungsname 133

APPLICATION_PREFIX (Systemobjekt-
 Attribut) 116, 133

Architektur

 WebTransactions 9

Array 213

Asynchron-Unterstützung, Konzept 186

Asynchrone Nachricht 185, 213

Attribut 213

Aufrufseite 214

Ausdruck 214

Auswertungsoperator 214

Automask 10

AUTOMASK (Systemobjekt-Attribut) 80, 116

Automask-Template 214

 generieren (Beispiel) 39

AutomaskOSD.htm 80

 Varianten 80

Automatische Umsetzung, Beispiel 42

AUTOTAB (Systemobjekt-Attribut) 117

B

Basisdatentyp 213

Basisverzeichnis 214

 anlegen 67

 auf eine neue Version umstellen 68

 Beispiel 36

BCAM-Applikationsname 214

BCAMAPPL 214

Bedienungselemente 147

Beispiel

 WebTransactions-Anwendung 25

Benutzererkennung 214

Berechtigungsprüfung siehe Zugangskontrolle

Bildschirmfeld-Inhalt 137

BLINKING (Host-Datenobjekt-Attribut) 138

Browser 214

Browser-Druck 199

 Konvertierungsdatei 202

 Start-Template 199, 204

Browser-Plattform 215

Browserdarstellungs-Druck 198, 215

BYPASS (Systemobjekt-Attribut) 117

C

Capture-Datenbank [215](#)
 Pfadname [117](#)
Capture-Funktion [94](#)
Capture-Verfahren [215](#)
CAPTURE_FILE (Systemobjekt-Attribut) [117](#)
CAPTURED_FLD (Systemobjekt-Attribut) [117](#)
CGI (Common Gateway Interface) [215](#)
CHARSET (Systemobjekt-Attribut) [123](#)
ClearScreen [148](#)
Client [215](#)
close [133](#)
Cluster [215](#)
COLOR (Host-Datenobjekt-Attribut) [139](#)
COMMUNICATION_INTERFACE_VERSION
 (Systemobjekt-Attribut) [117](#)
CONNECTION_MESSAGE (Systemobjekt-
 Attribut) [118](#)
Content-Type (HTTP-Header-Feld) [123](#)
Cursor-Position (Feldinformation) [140](#)

D

Dämon [215](#)
Daten
 dynamisch [217](#)
Datenfreigabe an den Host [148](#)
Datenfreigabetaste [140](#)
Datensatzstruktur [217](#)
Datentyp [216](#)
Diagnose
 Unicode [137](#)
Dialog [216](#)
 aktiv [213](#), [216](#)
 Arten [216](#)
 nicht synchron [216](#)
 passiv [216](#)
 synchron [216](#)
Dialogzyklus [216](#)
Disconnect [148](#)
DISCONNECT (Systemobjekt-Attribut) [118](#), [148](#)
DISPLAY_EURO (Systemobjekt-Attribut) [118](#)
Distinguished Name [216](#)
Dokumentenverzeichnis [217](#)
Domain Name Service (DNS) [217](#)

Druck von Host-Daten [193](#)
Druck-/Asynchron-Unterstützung [179](#)
 einschalten [179](#)
 Funktionsweise [180](#)
Druck-Unterstützung [189](#)
 Browserdarstellungs-Druck [198](#)
 Host-Daten-Druck [193](#)
 Terminal-Hardcopy-Druck [190](#)
 Web-Browser-Konfiguration [205](#)
Druckserver [195](#), [210](#)
DÜ [148](#)
DÜ2 [148](#)
DUE [148](#)

E

EHLAPI [217](#)
Eigenschaft [217](#)
Einfügen
 inline WtBean [176](#)
 standalone WtBean [174](#)
Eingabe-Datentyp [138](#)
EJB [217](#)
Elementnamen-Attribute [140](#)
END_MARK (Systemobjekt-Attribut) [119](#)
END_WAIT_CONDITION (Systemobjekt-
 Attribut) [126](#)
END_WAIT_CONDITION.
 CURSOR_IN_COLUMN (Systemobjekt-
 Attribut) [119](#)
END_WAIT_CONDITION.CURSOR_IN_LINE
 (Systemobjekt-Attribut) [119](#)
END_WAIT_CONDITION.
 CURSOR_NOT_IN_COLUMN (Systemobjekt-
 Attribut) [119](#)
END_WAIT_CONDITION.
 CURSOR_NOT_IN_LINE (Systemobjekt-
 Attribut) [119](#)
END_WAIT_CONDITION.
 FLD_DIFFERENT_FROM (Systemobjekt-
 Attribut) [119](#)
END_WAIT_CONDITION.FLD_EXPECTED
 (Systemobjekt-Attribut) [119](#)
END_WAIT_CONDITION.MATCH_OPERATION
 (Systemobjekt-Attribut) [119](#)

END_WAIT_CONDITION.
 MATCH_STARTCOLUMN (Systemobjekt-
 Attribut) 119
END_WAIT_CONDITION. MATCH_STARTLINE
 (Systemobjekt-Attribut) 119
END_WAIT_CONDITION. MATCH_VALUE (Sys-
 temobjekt-Attribut) 119
END_WAIT_CONDITION.EXPECTED_BLOCKS
 (Systemobjekt-Attribut) 119
EPILOG (Systemobjekt-Attribut) 120
Erfassen, einzelnes Erkennungskriterium 94
Erkennungskriterium 94, 217
 einzeln erfassen 94
Erstellen
 Capture-Datenbank 94
Erstes Template siehe Start-Template

F

F1 ... F24 148
Felddatei 217
Felder-Reihenfolge 140
Feldtyp 138
Festlegen
 Formularfelder 122
 Nachspann 120
 Vorspann 128
FHS 218
Field 218
FIELD_NAMES (Systemobjekt-Attribut) 120
Filter 218
FIRST_IO_TIMEOUT (Systemobjekt-
 Attribut) 121
FLD (Systemobjekt-Attribut) 121
FLD-Datei 96
fld-Datei 217
Font-Größe 158
Format 218
 #Format 218
 *Format 218
 +Format 218
 -Format 218
Formatbeschreibungsquelle 94, 218
Formate individuell aufbereiten 93
Formattyp 218

FORMTPL (Systemobjekt-Attribut) 122
Formularfelder festlegen 122
function
 doBackTab() 157
 doCursorDown() 157
 doCursorHome() 157
 doCursorLeft() 157
 doCursorRight() 157
 doCursorUp() 157
 doTab() 157
 doToggleInsert() 157
 doToggleMark() 157
 wtCreateKeyMap() 155
 wtCreateKeySelectList() 155
 wtHandleKeyboard(modifier, keyCode) 155
Funktion 218

G

Generieren
 Start-Template 174
getFileName (WT_PKEYS) 166
getKeyValue (WT_PKEYS) 167
Griechischer Zeichensatz 73

H

HARDCOPY (Systemobjekt-Attribut) 122
Holder Task 218
Host 218
Host-Adapter 219
Host-Anwendung 219
Host-Daten-Druck 193, 219
Host-Datenobjekt 135, 219
 Namen 135
 verkürzte Angabe 136
Host-Objekt 135
Host-Plattform 219
Host-Rechner, Name/Internetadresse 133
Host-Steuerobjekt 140, 219
 WT_KEY 147
HOST_CHARSET (Systemobjekt-Attribut) 123
HOST_NAME (Systemobjekt-Attribut) 123, 133
HTML 220
HTMLVALUE (Host-Datenobjekt-Attribut) 127,
 137

HTTP 219
HTTPS 219
Hypertext 219
Hypertext Markup Language (HTML) 220

I

IFG-Namen 136
IFG2FLD
 Anwendung 112
IGNORE_ASYNC (Systemobjekt-Attribut) 124
Individuelles Template
 Popup-Unterstützung 106
initClusterParameter (WT_PKEYS-Klasse) 167
Inline WtBean 230
 einfügen 176
INPUT (Host-Datenobjekt-Attribut) 138
Installation
 bedienerlos 19
 BS2000/OSD 23
 Linux 22
 Solaris 21
 über die Bedienoberfläche 18
 WebLab 23
 WebTransactions 17
 Windows 18
Integrierte Terminal-Emulation 9
INTENSITY (Host-Datenobjekt-Attribut) 138
INVERS (Host-Datenobjekt-Attribut) 138
ISO Latin-1 72
ISO-8859-1 73
ISO-8859-2 73
ISO-8859-3 73
ISO-8859-4 73
ISO-8859-5 73
ISO-8859-x 71, 73

J

JavaBean 220

K

K1 ... K14 148
KDCDEF 220
Klasse 220
Klassen-Template 220

Kommunikationsobjekt 220
 anlegen 176
 Verbindungsparameter 176
Konfiguration
 Kopplung 71
 Zeichensatz 71
Kontrollframe 180
Konvertierungswerkzeuge 220
Kopplung 71
Kyrillischer Zeichensatz 73

L

Latin-1 73
Latin-2 73
LDAP 221
LENGTH (Host-Datenobjekt-Attribut) 138
Literal 221
Lizenzen
 eingeben (Beispiel) 28
Lizenzierung 24
loadFromEmu 166
LOCAL_PORT (Systemobjekt-Attribut) 124, 133
LZE_CHAR (Systemobjekt-Attribut) 125

M

MARKABLE (Host-Datenobjekt-Attribut) 138
Master-Template 221, 227
 OSD.wmt 78
 OSD_Pocket.wmt 78
 Tags 221
Message Queuing 221
Methode 221
MIME-Typ 205
MODIFIED (Host-Datenobjekt-Attribut) 138
Modul-Template 221
MT-Tag 221
Multi-Tier-Architektur 222
MULTIPLE_IO_TIMEOUT (Systemobjekt-Attribut) 126

N

Nachricht
 empfangen 134
 senden 134

- Nachspann festlegen 120
NAME (Host-Datenobjekt-Attribut) 138
Name/Value-Paar 222
NATIONAL_VARIANT (Systemobjekt-Attribut) 72, 127
Nationale Buchstaben 72
Nicht synchronisierter Dialog 216, 222
NIL_MODE (Systemobjekt-Attribut) 127
- O**
Objekt 222
OFFLINE_COMMUNICATION (Systemobjekt-Attribut) 127, 133
OFFLINE_LOGFILE (Systemobjekt-Attribut) 128, 133
OFFLINE_TRACEFILE (Systemobjekt-Attribut) 128, 133
open 133
openUTM 222
 Vorgang 229
openUTM-Anwendung 223
openUTM-Client (UPIC) 223
openUTM-Teilprogramm 223
Operationen 216
OSD.wmt 78
OSD_Pocket.wmt 78
- P**
P-Tasten 160
 Zuweisung 161
P1 ... P20 148
PADDING_CHARACTER (Systemobjekt-Attribut) 128
Parameter 223
Passiver Dialog 216, 223
Passwort 223
PKEY-Objekte
 Unicode 161
PKEY1 ... PKEY20 148
PKEYS-ASSIGNMENT 161
polling 223
Pool 224
Popup-Box 104, 106
Popup-Erkennung 109
POPUP.COLUMN (Systemobjekt-Attribut) 128
POPUP.HEIGHT (Systemobjekt-Attribut) 128
POPUP.HEND (Systemobjekt-Attribut) 129
POPUP.HMIDDLE (Systemobjekt-Attribut) 129
POPUP.HSTART (Systemobjekt-Attribut) 128
POPUP.LINE (Systemobjekt-Attribut) 129
POPUP.VEND (Systemobjekt-Attribut) 129
POPUP.VMIDDLE (Systemobjekt-Attribut) 129
POPUP.VSTART (Systemobjekt-Attribut) 129
POPUP.WIDTH (Systemobjekt-Attribut) 129
PORT_NUMBER (Systemobjekt-Attribut) 129, 133
Posted-Objekt 224
Posten 224
Print (wtKeysOSD.htm) 148
Projekt 224
 anlegen 35
 Beispiel 35, 40
 speichern 40
PROLOG (Systemobjekt-Attribut) 128
Protokoll 224
Protokolldatei 224
Prozess 224
Puffer 224
- R**
RangeLength (Host-Datenobjekt-Attribut) 139
RangeName (Host-Datenobjekt-Attribut) 139
RangeStartColumn (Host-Datenobjekt-Attribut) 139
RAWVALUE (Host-Datenobjekt-Attribut) 127, 138
readFromFile (WT_PKEYS-Klasse) 166
receive 134
RECEIVED_BLOCKS (Systemobjekt-Attribut) 130
Record 225
RECORD_HOST_COMMUNICATION (Systemobjekt-Attribut) 130, 133
Refresh 148
REFRESH_BY_ASYNC (Systemobjekt-Attribut) 130
RSO-Dienstprogramm 210

S

saveToFile (WT_PKEYS-Klasse) 166
Schirmende-Erkennung 119
send 134
Service-Anwendung 225
Service-Knoten 225
setClusterParameter (WT_PKEYS-Klasse) 167
setDirectory (WT_PKEYS-Klasse) 166
setFileName (WT_PKEYS-Klasse) 166
setKeyValue (WT_PKEYS-Klasse) 167
setPassword (WT_PKEYS-Klasse) 166
setToEmu (WT_PKEYS-Klasse) 167
Sichtbarkeit 225
Sitzung 225

- Start-Templates 169
- starten 42
- starten (WebLab) 64
- WebTransactions 225

Skalar 226
SOAP 226
Sondertasten 143, 147

- Abbildung 90

Sonderzeichen 72, 137
Speichern

- Projekt 40

SPSERVE 210
Standalone WTBean 230

- einfügen 174

Start-Template 227

- generieren 174
- Systemobjekt-Attribute setzen 171
- wtstartOSD.htm 170

Start-Template-Set 169
STARTCOLUMN (Host-Datenobjekt-Attribut) 138
Starten

- Sitzung 42
- Start-Template 169

STARTLINE (Host-Datenobjekt-Attribut) 138
STATION_NAME (Systemobjekt-Attribut) 130, 133
Stationsname 193
status request 73
Stil 226

SYM_DEST (Systemobjekt-Attribut) 131, 133
Synchronisierter Dialog 216, 226
Systemobjekt 226

- Attribute im Start-Template setzen 171
- OSD-spezifische Attribute 115
- Zusammenspiel Attribute u. Aufrufe 133

Systemobjekt-Attribut

- CHARSET 123

T

TAC 228
Tag 226
Tastenunterstützung 146
Tastenzuordnung

- definieren 149
- wtkeysOSD.js 149

TCP/IP 227
Teilnachrichten-Folge 126
Template 227

- für Popup-Box 104
- individuell generieren 93
- Klasse 220
- Master 227
- Start 227

Template-Objekt 227
Templates 10, 74
Terminal-Anwendung 227
Terminal-Funktionen 144
Terminal-Hardcopy-Druck 190, 227
TERMINAL_TYPE (Systemobjekt-Attribut) 131, 133
Terminalverbindung

- öffnen 133

Thread 218
tmp-Unterverzeichnis 69
TRACE_LEVEL (Systemobjekt-Attribut) 131
Traces 137
Transaktion 228
Transaktionscode 228
Türkischer Zeichensatz 73
TYPE (Host-Datenobjekt-Attribut) 138

U

UDDI 228

- Umlaute 74
- UNDERLINE (Host-Datenobjekt-Attribut) 138
- Unicode 228
 - bei Zeichenkettenoperationen 137
 - in der Diagnose 137
 - PKEY-Objekte 161
- Unicode-Unterstützung 74, 137
 - einschalten 131
- Unicode-Zeichensatz 74
- Unterverzeichnis
 - tmp 69
- UPIC 228
- URI 228
- URL 229
- USE_POPUP_RECOGNITION (Systemobjekt-Attribut) 131
- Userexit 229
- UTM siehe openUTM

- V**
- VALUE (Host-Datenobjekt-Attribut) 127, 137
- Variable 229
- Verbindung
 - mehrere öffnen 176
- VISIBLE (Host-Datenobjekt-Attribut) 138
- Vorgang (openUTM) 229
- Vorspann festlegen 128

- W**
- web server 229
- Web-Browser
 - Konfiguration für Druck-Unterstützung 205
- Web-Service 229
- WebLab 10, 137
 - installieren 23
 - Varianten von AutomaskOSD erstellen 80
- WebTransactions
 - Anwendung starten 75
 - Architektur 9
 - Beispiel-Anwendung 25
- WebTransactions-Anwendung 229
 - starten 75
- WebTransactions-Plattform 229
- WebTransactions-Server 229

- WebTransactions-Sitzung 225
- Wertebereich eines Datentyps 216
- WSDL 230
- WT_ASYNC (Systemobjekt-Attribut) 131
- WT_BROWSER 158
 - Font-Größe 158
- WT_BROWSER_PRINT (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.BOTTOM (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.FOOTER (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.HEADER (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.LEFT (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.MODE (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.ORIENTATION (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.RIGHT (Systemobjekt-Attribut) 132
- WT_BROWSER_PRINT_OPTIONS.TOP (Systemobjekt-Attribut) 132
- WT_COLOR (Host-Steuerobjekt) 142
- WT_FOCUS (Host-Steuerobjekt) 136, 142
- WT_FOCUS_SHORT (Host-Steuerobjekt) 142
- WT_KEY (Host-Steuerobjekt) 90, 134, 143
- WT_KEY.KEY 148
- WT_KEY.Key 90, 134
- WT_KEY.PKEY 160
- WT_PKEYS 166
- WT_SYSTEM_LINE (Host-Steuerobjekt) 143
- WTAPrint 199
- wtasync.htm 181
 - Behandlung asynchroner Nachrichten 187
 - Druckunterstützung 191, 194
- WTBean 230
 - wtcOSD 176
- wtBrowserFunctions.htm 144
 - Druck-/Asynchron-Unterstützung 182
- wtc_bp_print.cnv 202
- wtc_bp_Print.htm 202
- wtCommonBrowserFunctions.js 154

wtcOSD [176](#)
wtcStartOSD [174](#)
wtcUTM [176](#)
wtframes.htm [180](#)
wtKeyMappingTableInput [149](#)
wtKeysOSD.htm [90](#), [144](#)
 Bedienungselemente [147](#)
wtKeysOSD.js [90](#), [147](#), [149](#), [152](#)
WTML [230](#)
WTML-Tag [230](#)
wtPkeyFunctions.htm [166](#)
wtPKEYS.htm [91](#), [161](#)
wtPkeyValues.htm [163](#)
WTScript [230](#)
wtstartOSD.htm [169](#)
WWW-Browser [214](#)
WWW-Server [229](#)

X

XML [231](#)
XML-Schema [231](#)

Z

Zeichenkettenoperationen [74](#), [137](#)
 Unicode [137](#)
Zeichensatz [71](#)
 für Emulation konfigurieren [72](#)
Zugangskontrolle [231](#)
Zugriffskontrolle [231](#)