FIX/Wt
Release 1.0.0

Handbuch für Anwendungsentwickler

Inhaltsverzeichnis

Vorwort

Dieses Handbuch beschreibt FIX/Wt in der Version 1.0.0.

FIX/Wt wird entwickelt von

Nonne & Schneider Informationssysteme GmbH
Friedrich List Straße 31
35398 Gießen

Tel: 0641/97477-0, Fax:0641/97477-77

Im Zuge der Weiterentwicklung von FIX/Wt können Leistungsmerkmale hinzukommen, verändert werden oder entfallen.

Handbuchversion: Edition 1.0.0a Nonne & Schneider Informationssysteme GmbH, 2012

I Allgemeiner Überblick

1 Was ist FIX/Wt

FIX/Wt (sprich FIX/Witty) ist eine Software mit der es möglich ist, FIX-Anwendungen im Web-Browser zu starten und zu bedienen. Von der Architektur her ersetzt FIX/Wt den FIX/Win Client. FIX/Wt ist dabei sowohl Client als auch Server.

Aus Sicht der FIX-Anwendung ist FIX/Wt - wie auch FIX/Win - ein Client, der sich mit der Anwendung verbindet. Änderungen an der Anwendung sind dazu nicht notwendig. Es genügt, die Anwendung mit einer modernen FIX Bibliothek zu binden. Damit kann die Anwendung sowohl über FIX/Wt, als auch über FIX/Win bedient werden.

Aus Sicht des Browsers ist FIX/Wt ein (WEB-)Server, der die Daten zur Darstellung des Bildschirms liefert und der Daten in Form von Feldwerten, Tastencodes und Mausklicks entgegen nimmt. Dieser WEB-Server kann entweder der in FIX/Wt eingebaute WEB-Server sein oder der Microsoft Internet Information Server.

2 Funktionsweise von FIX/Wt

FIX/Wt arbeitet, genau wie FIX/Win, mit virtuellen Bildschirmen. Das bedeutet, dass von FIX ein Characterbildschirm übertragen wird und FIX/Wt (FIX/Win) aus diesen Daten mit Hilfe von Zeichenoperationen wie Linien, Rechtecken und Bitmaps einen graphischen Bildschirm erzeugt und anzeigt. Der Unterschied zwischen FIX/Win und FIX/Wt besteht darin, dass hierfür statt der Funktionen des WIN32-APIs die Funktionen des Toolkits Wt (siehe http://www.webtoolkit.eu/wt) verwendet werden (daher auch der Name FIX/Wt). Dieses Toolkit erzeugt aus den Zeichenanweisungen JavaScript Code, der ein HTML 5 Canvas Element mit grafischen Elementen füllt. Diese Zeichenanweisungen werden vom Browser ausgeführt. Es werden also weder native HTML Elemente wie Tabellen, Text und Felder verwendet, noch wird eine komplette Grafik vom WEB-Server bereitgestellt. Dies macht den Ansatz performant und die Entwicklung von FIX/Wt durch die Ähnlichkeit mit FIX/Win überhaupt erst möglich.

Zur Erstellung von Dialogboxen und anderen grafischen Elementen, die zur Bedienung von FIX/Wt notwendig sind, verwendet FIX/Wt ebenfalls das Toolkit Wt, das diese Funktionalität bereit stellt.

An der Schnittstelle zu FIX hat sich in FIX/Wt nichts geändert. FIX/Wt verwendet somit gegenüber der Anwendung das gleiche Protokoll wie FIX/Win.

Ein großer Unterschied zu FIX/Win besteht darin, das FIX/Wt als (Web-)Server arbeitet. Das bedeutet, dass sich verschiedene Benutzer über einen HTTP-Zugang an ein und die selbe FIX/Wt Instanz verbinden. Für jeder Verbindung wird eine Session gestartet, über die eine FIX-Anwendung bedient werden kann.

3 Fehlende Features und weitere Unterschiede zu FIX/Win

FIX/Wt unterstützt nicht alle Features von FIX/Win. Auf einige Dinge wurde in der ersten Version bewusst verzichtet. Andere sind in dem Umfeld eines Browsers einfach nicht machbar. Für wieder andere steht eine Analyse der Machbarkeit noch aus.

Hinzu kommt die Tatsache, dass im Falle von FIX/Wt mit einer komplett anderen Architektur gearbeitet wird. So sind jetzt an der Anwendung nicht mehr zwei Rechner beteiligt (jeweils einer für FIX und FIX/Win) sondern drei (zusätzlich ein Rechner, auf dem der Browser gestartet wurde). Das verhindert Dinge, wie das Drucken aus der Anwendung oder aus FIX/Wt heraus oder das Ablegen von Dateien auf dem Rechner des Benutzers.

Folgende Features werden von FIX/Wt nicht oder nur eingeschränkt unterstützt:

3.1 Framework und Kernklassen

Die mit FIX/Win 3.0 eingeführte Trennung in Framework und Kernklassen wurde für FIX/Wt wieder aufgegeben. Das Binary von FIX/Wt besteht deshalb aus einer Datei und es ist nicht möglich, verschiedene Frameworks zu entwickeln. Das hat zwei Gründe:

- WT arbeitet mit threadlokalem Speicher. Wenn Code in eine andere DLL ausgelagert wird, besteht kein Zugriff auf diesen Speicher.
- Das Interesse und der Bedarf an der Entwicklung von eigenen Frameworks war wesentlich kleiner als zu Beginn angenommen. Abgesehen von einigen Nachfragen, war FIX/Win selbst das einzige Framework, das innerhalb der letzten Jahre die Kernklassen genutzt hat.

3.2 Felderfassung

Die in FIX/Win vorhandene Felderfassung wurde auf den Browser verlagert. Dabei wurde das Verhalten nicht 1:1 übernommen. Das bedeutet, dass sich die Felderfassung - insbesondere was komplexe Typen wie Datetime, Interval oder Felder mit Brüchen betrifft - unter FIX/Wt teilweise anderes verhält als unter FIX/Win.

3.3 Starten eines Editors

Das Starten eines Texteditor ist in einem Browser nicht möglich. Somit entfällt in FIX/Wt das mit FIX/Win 3.1 eingeführte Feature, das der Aufruf von editor() in FIX einen Texteditor auf der Frontendseite startet.

3.4 Verwendung von ActiveX Controls

ActiveX Controls stehen nur unter Windows zur Verfügung. Sie werden zwar vom

InternetExplorer unterstützt, können jedoch vom Benutzer aus Sicherheitsgründen verboten werden. Andere Browser unterstützen keine ActiveX Controls, so dass das Vorhandensein nicht vorausgesetzt werden kann.

3.5 Copy und Paste

Copy und Paste funktioniert in Feldern wie gewohnt. Das bedeutet, das mit Strg-C der markierte Teil in die Zwischenablage kopiert werden kann und mit Strg-V der markierte Teil durch den Inhalt der Zwischenablage ersetzt werden kann.

Zusätzlich zu dieser Funktionalität ist es möglich, die Copy und Paste Funktion über einen Menüpunkt im Kontextmenü auszulösen. Dabei wird immer der komplette Feldinhalt kopiert bzw. eingefügt. Die Verwendung der Zwischenablage funktioniert jedoch nur im InternetExplorer, wenn eine Sicherheitsabfrage bestätigt wurde. Alle anderen Browser verbieten aus Sicherheitsgründen den Zugriff auf die Zwischenablage. Bei diesen Browsern wird deshalb ein verstecktes Feld als Zwischenablage benutzt. Das hat zur Folge, dass die Werte von Feldern nur innerhalb der Session kopiert werden können. Der Zugriff von anderen Anwendungen oder Sessions ist nicht möglich.

Das Kopieren von Teilbereichen und das Senden von Texten der Zwischenablage als Events wurde noch nicht implementiert. Hier ist damit zu rechnen, das ähnliche Probleme und Einschränkungen auftreten.

3.6 Drucken des Bildschirms

Das Drucken des Bildschirminhaltes wurde noch nicht implementiert. Es ist abzuklären, ob dies funktionieren wird. Probleme sind darin zu sehen, dass der Bildschirminhalt nicht mehr in FIX/Wt vorliegt, sondern im Browser und dass kein Zugriff auf die dem Browser zugänglichen Drucker besteht.

Es ist jedoch möglich, über die Druckfunktion des Browsers einen Ausdruck vom aktuellen Zustand zu machen. Allerdings enthält dieser dann auch Iconbox und Statuszeile.

3.7 Umschalten des LookAndFeels zur Laufzeit

Aus Gründen der Einfachheit wurde auf dieses Feature verzichtet. Das bedeutet, dass nach dem Umschalten des LookAndFeels die Anwendung (also die Verbindung zu FIX/Wt) neu gestartet werden muss, damit die Auswirkungen sichtbar werden.

3.8 Entwicklermenü

Das Entwicklermenü steht unter FIX/Wt nicht zur Verfügung. Mit Grund dafür ist, dass das Nachladen von Einstellungen wie dem LookAndFeel und der Tastaturtabelle von FIX/Wt zu Gunsten von einer einfachen internen Struktur nicht unterstützt wird.

3.9 Verschieben von Fenstern

Dieses Feature wurde noch nicht implementiert. Ob es mit Hilfe der Möglichkeiten eines Browsers funktioniert eine Nachbildung dieses Features zu implementieren, ist noch offen und muss durch eine Analyse geklärt werden.

3.10 Verwendung von verschiedenen Farbvariationen

Dieses Feature wurde noch nicht in FIX/Wt implementiert. Es ist jedoch möglich, es in einer der nächsten Versionen von FIX/Wt aufzunehmen.

3.11 Anzeige eines Logos während des Verbindungsaufbaus

Dieses Feature wurde noch nicht in FIX/Wt implementiert. Es ist jedoch möglich, es in einer der nächsten Versionen von FIX/Wt aufzunehmen.

3.12 Verwendung der Schreibmarke (Caret)

In FIX/Wt wird die Schreibmarke nur noch in Feldern dargestellt. Ein Browser bietet ansonsten nicht die Möglichkeit, eine Schreibmarke zu verwenden. Damit besteht ein Problem bei der Darstellung von Choices. Hier wird der aktuellen Satz anhand der Position der Schreibmarke erkannt. Dieses Problem wird durch eine Erweiterung von FIX behoben, mit der es möglich ist, den aktuellen Satz durch ein Attribut hervorzuheben. (siehe FIX Releasenotes.html: Referenz 838 vom 28.09.2012: Bessere Darstellung von Choices)

3.13 Kontextmenüs

Kontextmenüs werden generell von FIX/Wt unterstützt. Sie sind jedoch im Gegensatz zu denen von FIX/Win nicht tastaturbedienbar. Abgesehen davon besteht ein Problem bei der Darstellung der Tasten in den Menüpunkten. Diese können nicht wie bei FIX/Win oder Windows rechts ausgerichtet dargestellt werden.

4 Anpassen der FIX-Anwendung

Aus technischer Sicht ist eine Anpassung der FIX-Anwendung an FIX/Wt nicht notwendig. FIX/Wt verwendet das gleiche Protokoll zur Kommunikation mit FIX.

Es gibt jedoch Features in einer Anwendung, die aufgrund der Umgebung nur in Zusammenhang mit FIX/Win funktionieren. Durch die Tatsache, das der Benutzer im Falle von FIX/Wt an einem entfernten Rechner vor einem Browser sitzt, sind viele Dinge, die im Falle von FIX/Win auf die eine Art gelöst wurden, anders zu lösen.

Die Probleme, die auftreten können, hängen stark von der jeweiligen Anwendung ab

und können sehr vielfältig sein. An dieser Stelle soll deshalb nur ein Beispiel angeführt werden:

Wenn eine FIX-Anwendung Daten für Excel erzeugt, diese über eine Netzwerkfreigabe bereitstellt und dann über FIX/Win Excel startet, dann wird das auf diese Weise in FIX/Wt nicht mehr funktionieren. Es ist zwar noch möglich, über FIX/Wt Excel zu starten, jedoch wird der Benutzer, der die Anwendung über den Browser startet, nichts davon mitbekommen, da er ja typischerweise an einem ganz anderen Rechner sitzt. Es ist also notwendig, diesen Start von Excel auf den Rechner zu verschieben, auf dem der Browser gestartet wurde. Hier steht jedoch die Netzwerkfreigabe nicht unbedingt zur Verfügung.

Damit die Anwendung entscheiden kann, wie und ob sie bestimmte Features bereitstellt, kann über die Funktion

```
int fx_client_type()
```

abgefragt werden, von welchem Frontend sie bedient wird. Die möglichen Werte sind in der FIX-Includedatei

```
fix/basics.h
```

definiert:

Macro	Wert
FRONT_TERMINAL	0x00000000
FRONT_FIXWIN	0x00000001
FRONT_FIXWT	0x00030001

Der letzte Wert steht erst in neueren Versionen von FIX zur Verfügung und ist die einzige Änderung in FIX, die FIX/Wt betrifft.

5 Softwarevoraussetzungen

Zum Start von FIX/Wt wird ein Rechner mit einem der folgenden Betriebssysteme benötigt:

- Microsoft Windows XP
- Microsoft Windows 7
- Microsoft Windows 2003
- Microsoft Windows 2008

Wenn FIX/Wt in einem Webserver installiert werden soll, muss dieser zusätzlich installiert werden:

- Microsoft Internet Information Services 7.5

Um eine FIX/Wt Session zu starten wird ein Browser benötigt, der HTML 5 Canvas Elemente unterstützt. Folgende Browser werden von FIX/Wt unterstützt:

- Mozilla Firefox
- Goggle Chrome
- Microsoft Internet Explorer (ab Version 9)
- Safari (nur iPad 2)

II Installation und Start

1 Installation

FIX/Wt wird als ZIP-Datei

```
FIXWT-FL-<patchdate>.zip
```

ausgeliefert. Diese ZIP-Datei stellt eine Vollversion dar und enthält neben den Binärprogrammen alle Konfigurationsdateien, die zum Betrieb notwendig sind. Zur Installation ist das ZIP-Archive zu entpacken. Danach müssen die Konfigurationsdateien den vorhandenen Gegebenheiten angepasst werden. Welche Einstellungen vorzunehmen sind, wird in einem eigenen Kapitel beschrieben.

Auf diese Weise entsteht ein Verzeichnis mit einer funktionierenden FIX/Wt Installation. Dieses Verzeichnis kann als Basis für die Erstellung eines eigenen Setup-Paketes genutzt werden, das zum Einsatz kommt, um FIX/Wt beim Kunden zu installieren.

Damit die Konfigurationsdateien nicht bei jeder Änderung von FIX/Wt erneut angepasst werden müssen, werden bei einer Änderung von FIX/Wt Update-Pakete in Form von ZIP-Dateien ausgeliefert. Diese Pakete enthalten alle Dateien, die seit der letzten Auslieferung einer Vollversion geändert wurden oder neu hinzu gekommen sind. Im Dateinamen des Archivs wird das Patchdate als Bestandteil verwendet, so das auf den ersten Blick klar ist, wann das Update erstellt wurde. Der Dateiname eines Update hat die Form:

```
FIXWT-UP-<patchdate>.zip
```

Die Distribution einer FIX/Wt Version besteht dann aus dem Hauptarchive FIXWT-FL-<patchdate>.zip und dem letzten Update-Paket. Damit ist man in der Lage, eine neue Installation aufzusetzen oder eine bestehende Installation zu aktualisieren. Um eine neue Installation aufzusetzen ist das Hauptarchive FIXWT-FL-<patchdate>.zip zu entpacken und über die Konfigurationsdateien den Gegebenheiten anzupassen. Um eine bestehende Installation zu aktualisieren, ist das letzte Update-Paket zu entpacken, dass alle Änderungen der vorherigen Pakete enthält. Dabei ist darauf zu achten, dass keine Konfigurationsdateien überschrieben werden, die in der bestehenden Installation bereits angepasst wurden. In diesem Fall ist zu prüfen, welche Änderungen die Dateien enthalten und die Änderungen sind in der angepassten Datei nachzuziehen. Um das Ganze zu unterstützen, enthält jedes Paket die Datei

```
changes.log
```

die für jede ausgelieferte Version dokumentiert, welche Dateien sich geändert haben, welche neu sind und welche gelöscht wurden. Um festzustellen, welches Patchdate eine installierte Version besitzt, kann die letzte Versionsangabe dieser Datei in der bestehenden Installation ausgelesen werden. Eine andere Möglichkeit besteht darin, die Versionsinformation aus den Binärdateien FixWtHttp.exe oder FixWtIIS.dll auszulesen.

Das zum Update beschriebene Verfahren besitzt den schon oben erwähnten Nachteil, dass bei einer Änderung von Konfigurationsdateien genau geprüft werden muss, ob eine Datei installiert werden kann. Dieser Fall erzeugt zwar Aufwand und ist nicht

komplett auszuschließen. Er tritt jedoch eher selten auf und in der Regel sind nur die Dateien

```
bin/FixWtHttp.exe
bin/FixWtIIS.dll
changes.log
```

von einer Änderung betroffen.

Oft kann es jedoch auch sein, dass die Dateien

```
resources\fixwt.xml
resources\fixwt_de.xml
```

in einem Update geändert wurden. Diese Dateien enthalten die Texte von FIX/Wt in Englisch und Deutsch und müssen geändert werden, wenn es notwendig ist, auf neue Meldungen zuzugreifen. Wenn die Installation weitere Sprachen verwendet, dann müssen die neuen Meldungen auch in die Ressourcendateien für diese Sprachen übertragen werden. Um das zu erleichtern, werden die neuen Meldungen und entfallenen Meldungen in changes.log dokumentiert. Da dazu das Werkzeug diff verwendet wird, erkennt man eine neue Meldung an dem Zeichen ">" und eine alte Meldung, die entfallen ist, an dem Zeichen "<".

Optional zu diesem Verfahren besteht auch die Möglichkeit, dass statt eines Updates eine neue Vollversion ausgeliefert wird. Dies ist dann der Fall, wenn die Menge der geänderten Dateien im Gegensatz zur letzten Vollversion stark gestiegen ist. In diesem Fall sind alle Dateien neu zu installieren. Auch hier muss darauf geachtet werden, dass eigene Anpassungen in dem neuen Stand nachgezogen werden.

2 Verzeichnisaufbau

bin

Alle Binärdateien von FIX/Wt.

- **FixWtIIS.dll** - Binary für IIS
- **FixWtHttp.exe** - Eigener Server
- **bmpackWt.exe** - Tool um Bitmaps zu packen

Hier können auch die Dateien der Benutzerbibliothek abgelegt werden.

config

Dateien zur Konfiguration von FIX/Wt.

- **<anwendung>.frc** - Anwendungsspezifische Einstellungen für alle Benutzer
- **fixwt.frc** - Allgemeine Einstellungen zu FIX/Wt.
- **fw_usr.frc** - Benutzerspezifische Einstellungen für alle Benutzer

- **klickchr.fix** - Definition von anklickbaren Grafikzeichen
- **stdicon.fix** - Einstellungen der Iconbox
- **stdkey.fix** - Tastaturtabelle
- **wt_config_http.xml** - Wt Einstellungen für FixWtHttp.exe
- **wt_config_iis.xml** - Wt Einstellungen für FixWtIIS.dll

config/users

In diesem Verzeichnis werden Einstellungen für jeden Benutzer abgelegt. Wenn ein Benutzer im Konfigurationsdialog speichert, werden die Einstellungen in dieses Verzeichnis gespeichert.

LookAndFeel

Die Einstellungen zu den verschiedenen LookAndFeels.

LookAndFeel/Fixwt-LAF-[0-3]

Einstellungen für ein bestimmtes LookAndFeel

- **bitmap-[M|L|H].[png|bif]** - gepacktes Bitmap mit Grafikzeichen
- **keybitmap-[M|L|H].[png|bif]** - gepacktes Bitmap mit Grafikzeichen für Keylabels
- **coltab.fix** - Farbdefinitionsdatei
- **LookAndFeel.frc** - Einstellungen zum LookAndFeel
- **browserspec.<nr>** - Browserspezifische Abweichungen der Einstellungen aus LookAndFeel.frc
- **LookAndFeel-[M|L|H].css** - CSS Einstellungen für dieses LookAndFeel (Iconbox, Statuszeile, Meldungsfenster, Eingabefeld)
- **sInEmpty-[M|L|H].png** - Bitmap für Hintergrund der Statuszeile
- **sInConfig-[M|L|H].png** - Bitmap für Config-Button der Statuszeile
- **sInMsgwin-[M|L|H].png** - Bitmap für Meldungsfenster-Button der Statuszeile
- **sInMsgwinFill-[M|L|H].png** - Bitmap für Meldungsfenster-Button der Statuszeile, wenn Meldungen vorhanden
- **keyFrmActive-[M|L|H].png** - Bitmap für Rahmen eines Keylabels im gedrückten Zustand
- **keyFrmInActive-[M|L|H].png** - Bitmap für Rahmen eines Keylabels im nicht gedrückten Zustand
- **keyFrmRollover-[M|L|H].png** - Bitmap für Rahmen eines Keylabels, wenn sich der Mauszeiger darüber befindet

- **paButton_[0-7]-[M|L|H].png** - Bitmaps zur Darstellung von Paintareas vom Typ Button für 8 verschiedene Attribute
- **paMenuitem_[0-7]-[M|L|H].png** - Bitmaps zur Darstellung von Paintareas vom Typ Menueitem für 8 verschiedene Attribute
- **paTableheader_[0-7]-[M|L|H].png** - Bitmaps zur Darstellung von Paintareas vom Typ Tableheader für 8 verschiedene Attribute
- **pa_VarButton_[0-7]-[M|L|H].png** - Bitmaps zur Darstellung von Paintareas vom Typ Varbutton für 8 verschiedene Attribute

LookAndFeel/IconBoxWt-LAF-[0-3]

Verzeichnis mit Bitmaps der Iconbox.

resources

Wt Ressourcen für FIX/Wt

- **wt.xml** - Texte von Wt in Defaultsprache
- **fixwt.xml** - Texte von FIX/Wt in Defaultsprache
- **wt_<lang>.xml** - Texte von Wt in Sprache <lang>
- **fixwt.xml** - Texte von FIX/Wt in Sprache <lang>
- **common.css** - allgemeine CSS-Einstellungen, die über alle Themes und LookAndFeels gleich sind.

resources/themes

Verschiedene Themes zur Darstellung von Wt-Dialogen

3 Anlegen einer Programmtabelle

Zum Starten von FIX-Programmen über FIX/Wt sind die FIX-Programme in die Programmtabelle einzutragen. Diese Tabelle besitzt den gleichen Aufbau wie in FIX/Win und liegt im Hauptverzeichnis von FIX/Wt. Es besteht lediglich ein Unterschied bei der Auswertung, wobei von FIX/Wt die Angabe des Gruppenverzeichnisses ignoriert wird. Damit ist es möglich, eine bestehende Programmtabelle von einer FIX/Win Installation zu kopieren.

Die Programmtabelle hat folgenden Aufbau:

```
<Programm>, <Gruppe>, <Res-Datei>, <X>, <Y>, <Cmd>, <Hostname>
```

Dabei haben die einzelnen Komponenten folgende Bedeutung:

- <Programm> ist der Name, unter dem die FIX-Anwendung später in einer Auswahlbox angewählt werden kann oder der mit dem Parameter program übergeben wird. Der Name muss in der Programmtabelle eindeutig sein.
- <Gruppe> dieser Wert wird von FIX/Wt ignoriert.
- <Res-Datei> bezeichnet eine Ressourcedatei, in der alle vom Benutzer einstellbaren Optionen einer Anwendung abgelegt werden.
- <X>, <Y> definiert die zur Verfügung stehende Größe des FIX/Wt-Bildschirms in Zellen. Üblicherweise werden hier die Werte 80,25 eingetragen. Bei einer Erhöhung der Werte müssen die Masken der FIX-Anwendung verändert werden, so dass diese den größeren Platz auch nutzen.
- <Cmd> bezeichnet ein Startskript auf Seite der Anwendung, das eine Ablaufumgebung aufbaut und die Anwendung startet.
- <Hostname> ist der Hostname des Rechners, auf dem das Startskript liegt. Alternativ kann hier auch eine IP-Adresse angegeben werden. In beiden Fällen ist es möglich, den Port anzugeben, an den sich FIX/Win verbinden soll. Dazu ist die Nummer des Ports durch ":" getrennt hinter den Hostnamen oder die IP-Adresse zu schreiben. Fehlt die Angabe, dann wird der Port verwendet, der dem Service exec zugeordnet wurde (in der Regel 512).

Kommentare können - wie in allen anderen Konfigurationsdateien auch - mit einem # eingeleitet werden, wobei das Zeilenende das Ende des Kommentars bildet.

Der Basisname der Programmtabelle kann entweder beim Start einer Session mitgegeben werden oder es wird program verwendet. Dieser Basisname wird um eine sprachspezifische Endung ergänzt: _<sprache>.fix, wobei die Sprache aus der Spracheinstellung des Browsers gebildet wird. Wird eine Datei dieses Namens nicht im Hauptverzeichnis von FIX/Wt gefunden, dann wird eine Datei mit der Endung .fix verwendet.

Auf diese Weise ist es möglich, je nach Sprache verschiedene FIX-Programme (die über die Parameter im Kommando der Programmtabelle die Sprache der Anwendung festlegen) zu starten.

4 Anlegen von Benutzerverzeichnissen

Für jeden Benutzer, der seine Einstellungen bezüglich Stil und Größe in FIX/Wt speichern können soll, ist ein Verzeichnis unterhalb von

```
config/users
```

anzulegen. Als Name des Verzeichnisses ist der Benutzername zu verwenden.

Wenn es dieses Verzeichnis nicht gibt, kann der Benutzer keine eigenen Einstellungen vornehmen und bekommt beim Aufblenden des Konfigurationsdialogs eine entsprechende Meldung.

Wenn der Benutzer die Einstellungen speichert, werden die Dateien fw_usr.frc und <anwendung>.frc in das Verzeichnis des Benutzer kopiert, wobei die Einstellungen mit denen aus dem Konfigurationsdialog überschrieben werden. Einstellungen, die nicht im Konfigurationsdialog stehen, können dann vom Administrator durch Bearbeiten der Dateien für einen bestimmten Benutzer vorgenommen werden.

5 Starten von FIX/Wt

FIX/Wt kann als eigener Webserver gestartet werden oder als ISAPI-Extension im Microsoft Internet Information Server.

Start als Webserver

Zum Starten als eigener Webserver ist die Datei

```
FixWtHttp.exe
```

im Verzeichnis bin auszuführen. Zum Beenden kann eine Konsole zur Konfiguration durch Eingabe der URL

```
http://<servername>:8080/fixwt/config.html
```

in einem Browser geöffnet werden, wobei <servername> durch den Namen des Rechners zu ersetzen ist, auf dem FixWtHttp.exe gestartet wurde. Durch Anklicken des Stop-Buttons wird der Prozess beendet.

Start als ISAPI-Extension

Um FIX/Wt als ISAPI-Extension zu starten ist die Installation entweder in einem Verzeichnis unterhalb von wwwroot vorzunehmen oder es ist ein virtuelles Verzeichnis im IIS anzulegen, das auf die Installation von FIX/Wt zeigt. Dabei sollten die Rechte so gesetzt werden, dass der IIS darauf zugreifen kann.

Die Benutzerbibliothek, deren Pfad konfiguriert wird, sollte an dieser Stelle zu finden

sein. Alle anderen DLLs, die die Benutzerbibliothek benutzt müssen nach

```
c:/Windows/SysWOW64/inetsrv
```

kopiert werden.

Weiterhin sollten folgende Dinge im IIS konfiguriert werden:

- Verzeichnis durchsuchen aktivieren: Doppelklick auf Icon "Verzeichnis durchsuchen", Im rechten Panel "Aktionen" auf "Aktivieren" klicken.
- Zugriff auf bin erlauben - bin aus "Ausgeblendete Segmente" entfernen: Doppelklick auf Icon "Anforderungsfilterung"; Tab "Ausgeblendete Segmente" auswählen; Verzeichnis "bin" entfernen.
- ISAPI Module aktivieren 1: Doppelklick auf Icon "Handlerzuordnung"; Rechtsklick auf ISAPI-dll; Menüpunkt "Featureberechtigung bearbeiten"; "Lesen", "Script", "Ausführen" ankreuzen -> Zustand wechselt auf "Aktiviert".
- ISAPI Module aktivieren 2: Obersten Knoten im Baum auswählen; Icon "ISAPI und CGI Einschränkungen" doppelt klicken; Rechte Maustaste Menüpunkt "Featureeinstellungen bearbeiten"; Checkboxen für CGI und ISAPI aktivieren.
- Einstellungen für WT laut WT Dokumentation:
  - Anwendungspools; rechte Maustaste auf entsprechendem Pool; Menüpunkt "Erweiterte Einstellungen".
  - Muss: Allgemein/32 Bit Anwendung aktivieren: TRUE (auf 64 Bit Systemen)
  - Muss: Prozessmodell/Maximale Anzahl von Arbeitsprozessen: 1
  - Sollte: Wiederverwendung/Regelmäßiges Zeitintervall (Minuten): 0 - sonst wird die Wt Applikation oft neu gestartet.
  - Sollte: Wiederverwendung/Limit für den privaten Speicher (KB), Limit für den virtuellen Speicher (KB), Anforderungslimit: 0 - sonst wird die Applikation von Zeit zu Zeit beendet.

6 Lizenzprüfung

FIX verwendet zur Lizenzprüfung die IP-Adresse des Gerätes vor dem der Benutzer sitzt. Von diesem Gerät aus kann der Benutzer 6 Sitzungen starten. Für FIX/Wt könnte die IP-Adresse, die der Browser liefert verwendet werden. In folgenden Fällen handelt es sich dabei jedoch nicht um die Adresse des Gerätes vor dem der Benutzer sitzt:

- Der Browser wird über einen Terminalserver-Client gestartet. In diesem Fall besitzen die Browser von allen Benutzern die IP-Adresse des Terminal-Servers.
- Der Browser befindet sich in einem anderen Netzwerk. In diesem Fall besitzt der Browser die IP-Adresse des Routers. Wenn mehrere Benutzer, die sich hinter dem Router befinden, einen Browser starten, besitzen alle die gleiche IP-Adresse.

Da in diesen Fällen die gleiche Adresse für verschiedene Geräte verwendet wird, kann diese Adresse nicht verwendet werden.

Da es keine Möglichkeit gibt, die Adresse des Rechners, vor dem der Benutzer sitzt, in allen Fällen sicher zu ermitteln, verwendet FIX/Wt folgendes Verfahren zur Erzeugung einer Adresse, die zur Lizenzprüfung verwendet wird:

Alle vier Stellen der IP-Adresse werden aus Zufallszahlen gebildet und die IP-Adresse

wird dem Browser zugewiesen.

Die Zuweisung der so generierten IP-Adresse an den Browser geschieht über ein Cookie. Wenn ein Browser kein Cookie von einer vorherigen Verbindung besitzt, dann wird eine neue IP-Adresse generiert und ein Cookie an den Browser vergeben. Beim der zweiten Verbindung wird das Cookie gelesen und die darin gespeicherte Adresse verwendet. Da die gleiche Adresse verwendet wird, belegt die zweite Verbindung keine zusätzliche Lizenz und zählt nur als weitere von den 6 möglichen Sessions.

Die Cookies laufen um 00:00 Uhr ab. Wenn danach der Browser noch läuft und es wird eine zweite Verbindung hergestellt, dann bekommt der Browser eine neue IP-Adresse und ein neues Cookie und belegt somit eine weitere Lizenz. Die erste Verbindung läuft dann noch unter der alten IP-Adresse. FIX/Wt prüft jedoch nach 20 Felderfassungen, ob die Zeit abgelaufen ist. Wenn ja, dann wird der Benutzer mit einer Meldung darauf hingewiesen, dass er die Anwendung verlassen soll und sich neu verbinden soll. Bei einer neuen Verbindung wird die Lizenz dann freigegeben und die von der zweiten Session belegte Lizenz wird mit verwendet.

Dieses Verfahren besitzt zwei Probleme, die jedoch aufgrund der Auswirkungen nicht ins Gewicht fallen und sich in einem Fall auch umgehen lassen:

- Es werden durch den Zufallsgenerator für zwei verschiedene Browser die gleichen IP-Adressen generiert. Abgesehen davon, dass die Wahrscheinlichkeit hierfür gering ist (<Anzahl User>:4228250625), sind die Auswirkungen vertretbar: Die beiden Browser teilen sich die 6 möglichen Sitzungen.
- Ein Browser unterstützt keine Cookies. Dieser Fall kann vermieden werden, indem die Benutzer angewiesen werden, ihren Browser entsprechend zu konfigurieren. Wenn er doch eintritt, dann kommt es dazu, dass diesem Browser bei jeder Verbindung eine neue IP-Adresse zugewiesen wird und er jedes mal eine neue Lizenz belegt.

III Arbeiten mit FIX/Wt

1 Starten einer Session

Um mit FIX/Wt zu arbeiten, muss ein Browser gestartet werden und eine Verbindung zu FIX/Wt in Form einer Session aufgebaut werden. Je nach Webserver ist eine andere URL im Browser anzugeben:

Eigener Webserver

```
http://<servername>:8080/fixwt
```

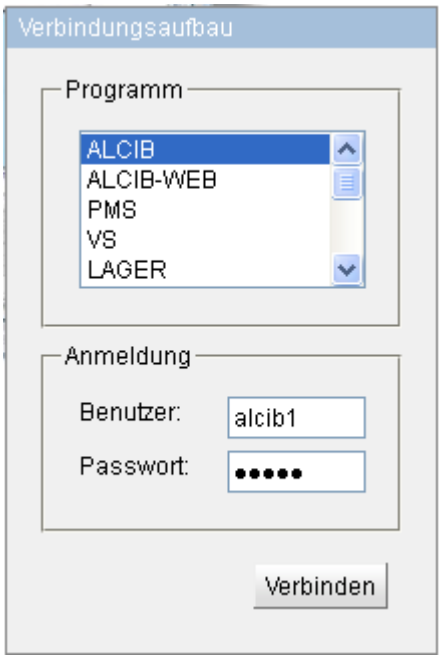
ISAPI-Extension

```
http://<rechnername>/Verzeichnis/bin/FixWtIIS.dll
```

Zusätzlich können in beiden Fällen weitere Parameter (über ? und & als Trenner; wie in URLs üblich) mitgegeben werden:

- **user=<name>** - Der Benutzername. Er wird zum einen als Vorgabe für den Dialog zum Verbindungsaufbau verwendet und zum anderen, um das Design-Theme aus den Einstellungen zu lesen.
- **password=<text>** - Das Passwort, das als Vorgabe für den Dialog zum Verbindungsaufbau verwendet wird. Da dieses Passwort im Klartext in der URL steht, sollte diese Option nur für Testzwecke verwendet werden.
- **program=<name>** - Das Programm aus der Programmtabelle. Dieser Wert wird in dem Dialog zum Verbindungsaufbau voreingestellt.
- **prgtab=<name>** - Der Basisname der Programmtabelle. Wenn der Wert nicht angegeben wird, dann wird "program" verwendet.

Wenn der Verbindungsaufbau zu FIX/Wt erfolgreich war, wird in dem FIX/Wt Prozess eine neue Session gestartet. Als erstes wird in dieser Session der Dialog zum Verbindungsaufbau eingeblendet. Hier kann ein Programm aus der Programmtabelle ausgewählt werden und der Benutzer und das Passwort eingegeben werden. Nach dem Drücken des Buttons "Verbinden" wird von FIX/Wt die Verbindung zu der FIX-Anwendung aufgebaut.



2 Bedienen von FIX/Wt

Im Prinzip wird eine FIX-Anwendung in FIX/Wt genauso bedient wie in FIX/Win. Es gibt jedoch einige Unterschiede, die hier erläutert werden sollen.

2.1 Tastenbedienung

FIX/Wt verwendet die gleiche Tastaturtabelle wie FIX/Win. Damit ist die Tastenbelegung gleich.

Browser fangen jedoch gewisse Tasten ab und es gibt keine Möglichkeit, die Funktion dieser Tasten zu unterbinden und daraus entsprechende Events für die FIX-Anwendung zu generieren. Da es eine Vielzahl von Tastenkombinationen gibt, ist es unmöglich, alle Kombinationen zu testen. Von manchen Tasten sind auch nur bestimmte Browser betroffen. Deshalb wird hier nur eine Liste der wichtigsten Tasten beschrieben, die jedoch im Laufe der Zeit erweitert wird.

Taste	**Funktion**	**Browser**
Strg-F4	Tab/Fenster schließen	alle
Alt-F4	Browser schließen	alle
Menu	Kontextmenü einblenden	alle
Strg-N	Neues Fenster	Chrome
Strg-P	Drucken	InternetExplorer
Strg-T	Neuer Tab	Chrome

Andererseits gibt es bestimmte Tasten, die für den Browser wichtig sind, die jedoch von FIX/Wt abgefangen werden, weil in der Tastaturtabelle ein Event dafür definiert ist.

Diese Funktionen des Browsers können über dessen Menü ausgelöst werden. Eine andere Möglichkeit besteht darin, vor dem Drücken der Taste durch Anklicken der Adressleiste diese zu aktivieren. Hier eine Liste der bis jetzt bekannten Tasten:

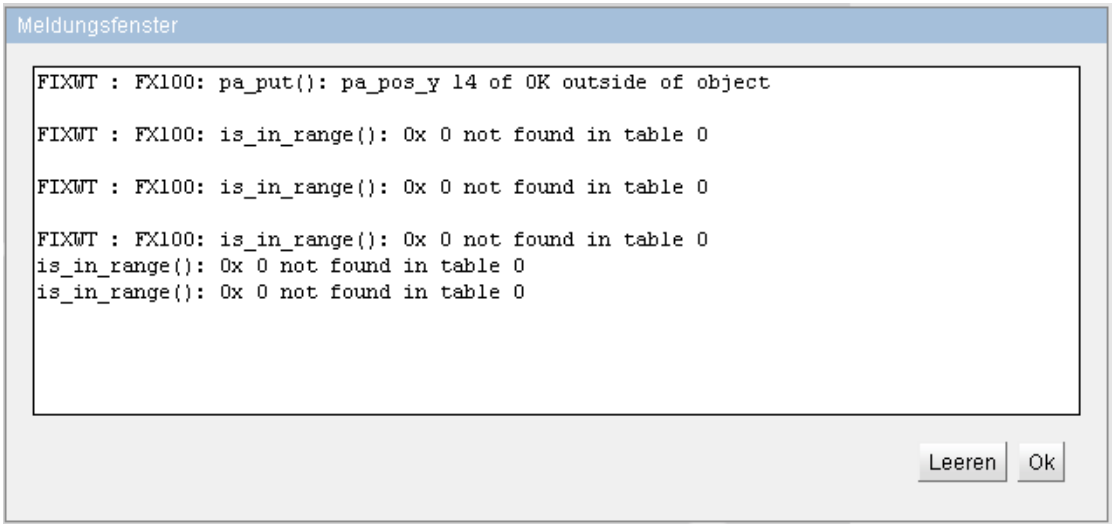
Taste	Funktion	Browser
F5	Aktualisieren	alle
F11	Vollbild	alle
Strg-P	Drucken	Chrome, Firefox
Strg-F	Suchen	alle

2.2 Hauptmenü

Da der Browser schon ein Menü besitzt und auch die Anwendung, wurde auf ein Hauptmenü von FIX/Wt verzichtet. Deshalb sind die Funktionen über Icons der Statusleiste zugänglich.

Anzeige des Meldungsfensters

Das Meldungsfenster kann über das Icon am rechten Rand der Statuszeile eingeblendet werden. Im Gegensatz zu FIX/Win wird es nicht angezeigt, wenn neue Meldungen ausgegeben werden. Statt dessen ändert sich das Icon (Standard: Symbol mit Warndreieck) so dass der Benutzer erkennen kann, dass Meldungen ausgegeben wurden. Dieses Icon wird nach dem Schließen des Meldungsfensters wieder auf das normale Icon zurückgesetzt.

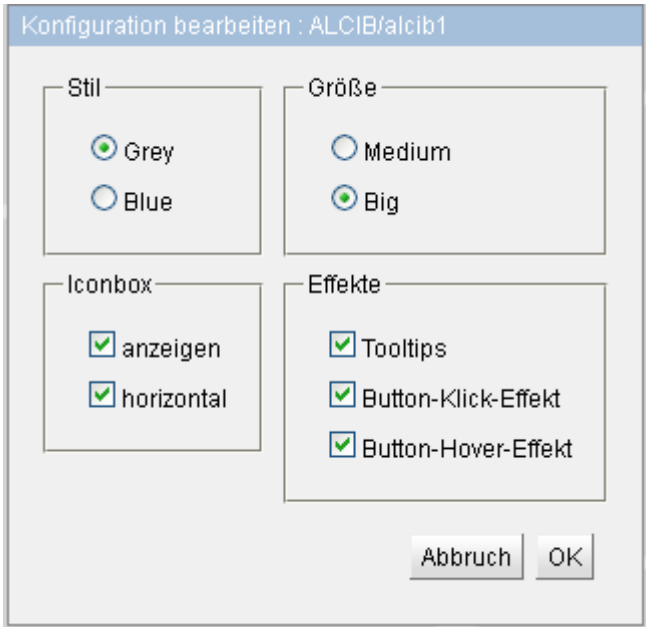


Anzeige des Dialogs mit Einstellungen

Über das zweite Icon der Statuszeile kann die Dialogbox mit Einstellungen geöffnet werden. Hier können Stil, Größe und Anzeige der Iconbox eingestellt werden. Im Gegensatz zu FIX/Win ist es nicht möglich, die Statuszeile auszublenden, da sie ja den Zugang zu den Einstellungen bietet.

Da die Effekte für Buttons sowie die Tooltips von FIX/Wt gesteuert werden, müssen dafür jedes mal Daten zum Browser übertragen werden. Bei langsamen Verbindungen ist es deshalb ratsam die Effekte über den Dialog abzuschalten.

Nach dem Beenden der Dialogbox werden die Einstellungen für den Benutzer gespeichert. Im Gegensatz zu FIX/Win wirken sich die Einstellungen erst beim nächsten Verbindungsaufbau aus.



IV Konfiguration von FIX/Wt

1 Gruppenverzeichnis

Das Konzept des Gruppenverzeichnisses wurde in FIX/Wt zugunsten einer einfachen Struktur aufgegeben. Die Einstellungen sind deshalb an den Dateien im Installationsverzeichnis vorzunehmen. Das bedeutet, dass alle Anwendungen, die über FIX/Wt gestartet werden, mit den gleichen Einstellungen arbeiten.

Es ist jedoch möglich, FIX/Wt mehrfach zu installieren, wobei jede Installation andere Einstellungen verwendet.

Der Aufbau des Verzeichnisses mit den einzelnen Dateien ist im Abschnitt "2 Verzeichnisaufbau auf Seite 12" nachzulesen.

1.1 Verfahren zum Einlesen der Dateien

Obwohl FIX/Wt kein Gruppenverzeichnis mehr verwendet, ist das grundsätzliche Verfahren zum Einlesen der Dateien erhalten geblieben.

Dateien aus dem Unterverzeichnis config

Diese Dateien werden nicht gesucht. Das bedeutet, dass diese Dateien direkt aus dem Verzeichnis config geladen werden.

Dateien aus den Unterverzeichnissen Fixwt-LAF und IconBoxWt-LAF

Die Dateien des LookAndFeels und der Iconbox besitzen andere Unterverzeichnisse wie bei FIX/Win. Für die Iconbox wird IconBoxWt-LAF verwendet und für das LookAndFeel Fixwt-LAF.

Für diese Dateien gibt es eine Suchreihenfolge, die den Stil und die Größe berücksichtigt. Die Größe wird durch einen Buchstaben gekennzeichnet und geht in den Dateinamen mit ein, indem dieser Buchstabe und zusammen mit dem Zeichen - ans Ende gestellt wird. Folgende Buchstaben werden verwendet (analog zu FIX/Win):

Buchstabe	Größe
-M	Medium
-L	Large
-H	Huge

Beispiel

bitmap-H.png enthält die Bitmaps für die Größe Huge und bitmap-L.png für die Größe Large.

Der Stil wird durch eine Nummer von 0-3 repräsentiert und geht in den Namen des Verzeichnisses mit ein. Es wird immer zuerst versucht, die Datei im stilabhängigen Verzeichnis zu lesen. Wenn sie dort nicht vorhanden ist, dann wird die Datei aus dem stilunabhängigen Verzeichnis gelesen, das die Nummer des Stils nicht im Namen enthält.

Für Dateien des LookAndFeels ergibt sich damit folgende Suchreihenfolge:

```
LookAndFeel/Fixwt-LAF-<style>/<dateibasis>-<size>.<ext>
LookAndFeel/Fixwt-LAF/<dateibasis>-<size>.<ext>
```

Für Dateien der Iconbox ergibt sich damit folgende Suchreihenfolge:

```
LookAndFeel/IconBoxWt-LAF-<style>/<dateibasis>-<size>.<ext>
LookAndFeel/IconBoxWt-LAF/<dateibasis>-<size>.<ext>
```

Bei einigen Dateien wird noch ein zusätzlicher Suchschritt eingefügt, indem die Größe weggelassen wird. Dabei wird folgende Suchreihenfolge verwendet:

```
LookAndFeel/Fixwt-LAF-<style>/<dateibasis>-<size>.<ext>
LookAndFeel/Fixwt-LAF-<style>/<dateibasis>.<ext>
LookAndFeel/Fixwt-LAF/<dateibasis>-<size>.<ext>
LookAndFeel/Fixwt-LAF/<dateibasis>.<ext>
```

Dieser Fall wird bei der Beschreibung der Konfigurationen in den folgenden Abschnitten explizit erwähnt

Beispiel

Wenn Stil 1 eingestellt ist und Größe Large, dann wird für die Datei mit Semigrafikzeichen folgende Suchreihenfolge verwendet:

```
LookAndFeel/Fixwt-LAF-1/bitmap-L.bmp
LookAndFeel/Fixwt-LAF/bitmap-L.bmp
```

2 CSS Dateien in FIX/Wt

FIX/Wt verwendet als Toolkit Wt. Dieses Toolkit erzeugt zur Darstellung der Benutzeroberfläche HTML-Seiten, deren Elemente über CSS definiert werden können. Dazu werden mehrere CSS-Dateien verwendet, die verschiedene Aufgaben übernehmen. Die folgende Liste bietet eine Übersicht. In den einzelnen Kapiteln wird genauer auf die Einstellungen in den einzelnen Dateien eingegangen.

resources/common.css - Allgemeine CSS Einstellungen. Diese Datei wird als erstes geladen. Die Einstellungen gelten global für alle LookAndFeels und Themes, so lange sie nicht von anderen überschrieben werden.

resources/themes/<name>/wt.css - CSS Einstellungen zur Definition eines WT-Themes. Hier stehen alle Einstellungen, die die allgemeinen Elemente von Wt (Dialogboxen, Felder, Buttons, ...) betreffen. In Wt können verschiedene Themes verwendet werden, die über CSS das Aussehen dieser Elemente definieren. Neben

dieser Datei existiert noch die Datei **wt_ie.css**, die Abweichungen für den InternetExplorer definiert.

LookAndFeel\Fixwt-LAF-[0-3]\lookAndFeel-[M|L|H].css - CSS Datei, die Definitionen für speziell von FIX/Wt bereitgestellte Elemente enthält. Diese Datei gibt es für jedes LookAndFeel (0-3) in allen Größen (M|L|H), so dass die Einstellungen getrennt vorgenommen werden können.

3 Konfiguration von Wt

Das Toolkit Wt wird über eine xml-Datei konfiguriert. Je nach Art des Webservers wird eine andere Datei von FIX/Wt dafür verwendet.

Start als Webserver:

```
config/wt_config_http.xml
```

Start als ISAPI-Extension:

```
config/wt_config_iis.xml
```

Die darin enthaltenen Einstellungen sind in der Dokumentation von Wt erklärt. Deshalb folgt hier nur die Beschreibung von einigen für FIX/Wt wichtigen Einstellungen.

log-file - Wt schreibt eine Log-Datei. Diese Einstellung definiert den Pfad der Datei relativ zum aktuellen Verzeichnis. Bei Start als ISAPI-Extension sollte man beachten, dass das aktuelle Verzeichnis bin ist.

log-config - Definiert, welche Meldungen in die Log-Datei geschrieben werden. Der Kommentar in der Datei erklärt den Aufbau. Folgende Einträge sind sinnvoll: ***** - alles, *** -debug** - keine Debugmeldungen, *** -debug -info** - keine Debug- und keine Infomeldungen.

session-id-length - Die Länge der id für Sessions. In der Praxis sind längere IDs sinnvoll; für das Debuggen eher kurze, die man sich leicht merken kann.

debug - Diese Einstellung legt fest, wie sich Wt bei einer JavaScript-Exception verhalten soll. Der Wert "stack" ist hier wenig informativ, da der Stack immer gleich ist. Wenn "false" angegeben wird, wird der Fehler in einer Meldungsbox ausgegeben. Bei "true" wird er an den Debugger des Browsers (z.B. Firebug) weitergegeben und von diesem gemeldet.

4 Konfiguration der Iconbox

Die Konfiguration der Iconbox basiert auf der von FIX/Win. Sie verwendet ebenfalls das Konzept von Iconset und Iconlist (siehe Handbuch von FIX/Win). Folgende Unterschiede bestehen jedoch:

- Die Bitmaps müssen im png-Format vorliegen und auch in der Definitionsdatei so angegeben werden
- Für die Darstellung von inaktiven Bitmaps müssen Bitmaps mit dem Postfix gray_ vorliegen

- Die Darstellung von Rahmen wird über CSS konfiguriert. Bitmaps für Rahmen sind nicht notwendig

4.1 Icondefinitionsdatei

Die Konfigurationsdatei wird aus

```
config/stdicon.fix
```

eingelesen. Die Datei hat folgenden Aufbau:

```
# Kommentar
ICONLIST <LISTID>
<ICONID> <EVENTFOLGE> <BITMAPDATEI> <HINWEISTEXT> ;
oder SPACE <SIZE> ;
oder SPACE <SIZE> <BITMAPDATEI>;
...
END
...
ICONSET <SETID>
      <ICONID> ;
      ....
END
```

Für Kommentare in der Definitionsdatei ist das Zeichen # zu verwenden. Der gesamte Text hinter diesem Zeichen bis zum Zeilenende ist Kommentar.

Die Definition einer Iconliste beginnt mit dem Schlüsselwort ICONLIST. Danach ist die ID der Liste anzugeben. Für IDs von Iconlisten sind ganze positive Zahlen von 1 bis 2147483647 zu verwenden. Die ID 0 wird intern für die leere Iconliste verwendet. Für jedes Icon und jeden Zwischenraum folgt dann eine Definition mit Werten, die durch ein Semikolon abzuschließen ist. Die Icons und die Zwischenräume werden später in der gleichen Reihenfolge dargestellt, in der sie in der Iconliste definiert sind.

Die Definition eines Icons besteht aus der Angabe einer Icon-ID, einer Folge von maximal fünf FIX-Events, einer Bitmapdatei zur Darstellung des Icons und einem Hinweistext, der als Tooltip angezeigt wird. Als Icon-ID können alle positiven ganzen Zahlen bis 65535 verwendet werden. Der Name der Bitmapdatei ist ohne Pfad anzugeben. Die Datei wird im Unterverzeichnis LookAndFeel/IconBoxWt-LAF[0-3] erwartet und muss im png-Format vorliegen. Wenn sie dort nicht zu finden ist, wird versucht, sie aus dem Unterverzeichnis LookAndFeel/IconBoxWt-LAF zu laden. Zusätzlich zu dieser Datei muss eine Datei mit gleichem Namen und dem Postfix gray_ vorhanden sein, die das Icon in gegrautem Zustand darstellt.

Enthält der Hinweistext Leerzeichen, muss er in " " eingefasst werden. Die Zeichen werden gemäß Windows ANSI Code interpretiert. Als Eventfolge sind ein bis fünf durch Leerzeichen getrennte Eventbezeichner anzugeben. Als Eventbezeichner ist der zweibuchstabige Name des FIX-Events zu verwenden (so wie er in keys.h auf FIX Seite definiert ist, nur ohne das Postfix K_) oder die Angabe des Eventcodes mit einem vorangestellten '~' Zeichen.

Ein Zwischenraum zwischen zwei Icons kann durch Verwendung des Schlüsselwortes SPACE anstelle eines Icons definiert werden. Hinter diesem Schlüsselwort ist die Größe des Zwischenraums in Pixeln anzugeben. In der Mitte des Leerraums wird dann eine Linie gezeichnet.

Alternativ kann zusätzlich der Name einer Bitmap angegeben werden. Die Bitmap wird

aus dem Unterverzeichnis LookAndFeel/IconBoxWt-LAF[0-3] geladen und auf die angegebene Pixelbreite gestreckt oder gestaucht. Wenn für die IconBox statt der horizontalen die vertikale Darstellung gewählt wird, dann wird versucht eine andere Bitmap für diese Darstellung zu laden. Dazu wird der angegebene Name um den Postfix V_ erweitert (also beispielsweise V_space.png statt space.png). Existiert die Datei nicht, wird die angegebene Datei zur Darstellung des Zwischenraums verwendet. In beiden Fällen wird der angegebene Abstand als Pixelhöhe interpretiert und das Bitmap wird entsprechend gestreckt oder gestaucht.

Zum Abschluss der Definition einer Iconliste ist das Schlüsselwort END anzugeben.

4.2 Übernahme der Dateien von FIX/Win

Da es nur wenige Unterschiede zu FIX/Win gibt, besteht die Möglichkeit, die Dateien zu übernehmen. Dazu ist wie folgt vorzugehen:

- config/stdicon.fix aus dem FIX/Win Gruppenverzeichnis in das Verzeichnis von FIX/Wt kopieren
- In der Icondefinitionsdatei die Endung .bmp durch .png ersetzen
- Die Bitmaps für die Icons aus dem Gruppenverzeichnis von FIX/Win in das entsprechende Unterverzeichnis des FIX/Wt Verzeichnisses LookAndFeel kopieren. Beim Unterverzeichnis ist der Verzeichnisname IconBoxWt-LAF-<style> statt IconBox-LAF-<style> (FIX/Win) zu verwenden.
- Die Bitmaps für die Rahmen (frm*.bmp) können dabei ausgelassen werden. Statt dessen sind aber zwei Bitmap für den Hintergrund der Iconbox anzulegen. Für die horizontale Iconbox ist H_empty.png zu erzeugen und für die vertikale V_empty.png.
- Konvertieren der Bitmaps vom bmp-Format in das png-Format. Hierzu werden im Internet zahlreiche Freeware-Programme angeboten, die das im Batchbetrieb erledigen.
- Erzeugen von Bitmaps in Graustufen. Wenn diese Bitmaps mit dem Namen gray-<icon>.png in FIX/Win noch nicht vorhanden sind, dann müssen sie jetzt erzeugt werden. Dazu kann ebenfalls ein entsprechendes Freeware-Programm oder ein anderes Grafikprogramm verwendet werden.

4.3 Konfiguration über CSS

Für die Iconbox werden verschiedene CSS Klassen definiert, über die das Aussehen angepasst werden kann. Typischerweise findet die Definition dieser Klassen in der Datei

```
LookAndFeel/Fixwt-LAF-<style>/lookAndFeel<size>.css
```

statt, die abhängig von Stil und Größe geladen wird. Somit ist es möglich je nach Stil und Größe ein anderes Aussehen der Iconbox zu verwenden.

Folgende Klassen werden verwendet:

- **iconbox** - Klasse für den Container, der die Iconbox enthält
- **iconbox-space** - Klasse für ein Element, dass einen Zwischenraum darstellt
- **iconbox-button** - Klasse für ein Icon.
- **iconbox-button:hover** - Klasse für ein Icon, wenn sich der Mauszeiger darüber befindet

- **iconbox-button:active** - Klasse für ein Icon im angeklickten Zustand
- **iconbox-button-gray** - Klasse für ein Icon im nicht anklickbaren (disabled) Zustand

5 Konfiguration der Statuszeile

Das Aussehen der Statuszeile kann über die Datei

```
LookAndFeel/Fixwt-LAF-<style>/lookAndFeel<size>.css
```

festgelegt werden. Folgende CSS-Klassen beziehen sich auf die Statuszeile:

- **statusline** - Klasse für den Container, der die Statuszeile enthält
- **statusline-text** - Klasse für den Text der Statuszeile
- **statusline-button** - Klasse für einen Button der Statuszeile
- **statusline-button:hover** - Klasse für einen Button der Statuszeile, wenn sich der Mauszeiger darüber befindet
- **statusline-button:active** - Klasse für einen Button der Statuszeile in gedrücktem Zustand

Weiterhin werden folgende Bitmaps im png-Format zur Darstellung der Statuszeile verwendet, die alle im Verzeichnis LookAndFeel/Fixwt-LAF-<style> gesucht werden:

- **slnEmpty-<size>.png** - Bitmap zur Darstellung des Hintergrundes. Dieses Bitmaps wird auf die Breite der Zeile gestreckt.
- **slnMsgwin-<size>.png** - Bitmap für den Button zum Einblenden des Meldungsfensters
- **slnMsgwinFill-<size>.png** - Bitmap für den Button zum Einblenden des Meldungsfensters, wenn neue Texte im Meldungsfenster ausgegeben wurden.
- **slnConfig-<size>.png** - Bitmap für den Button zum Einblenden des Konfigurationsdialogs.

Die Angabe <size> bezeichnet die im LookAndFeel eingestellte Größe (M|L|H). Es kann also für jede Größe ein Bitmap definiert werden. Es ist aber auch möglich, die Größenangabe wegzulassen. Dieses Bitmap wird dann verwendet, wenn kein Bitmap mit Größenangabe vorhanden ist.

6 Konfiguration des Meldungsfensters

Das Aussehen des Meldungsfensters kann ebenfalls über CSS in der Datei

```
LookAndFeel/Fixwt-LAF-<style>/lookAndFeel<size>.css
```

vorgenommen werden. Folgende Klassen sind von Bedeutung:

- **.messagewindow-scrollarea** - Klasse für den Scrollbereich des Meldungsfensters
- **.messagewindow-test** - Klasse für den Text des Meldungsfensters

7 Erstellen einer Tastenbelegung

7.1 Tastenbelegungsdatei

Die Belegung der Tastatur wird über die Tastenbelegungsdatei in

```
config/stdkey.fix
```

definiert. Das Format entspricht dem von FIX/Win. Somit kann eine bestehende Tastenbelegungsdatei aus dem Gruppenverzeichnis von FIX/Win in das Installationsverzeichnis von FIX/Wt kopiert werden.

Jede Zeile der Datei definiert die Belegung einer Taste und hat folgenden Aufbau:

```
<Tastenname> <Normal> <Shift> <Strg> <Shift+Strg> <Alt>
<Shift+Alt> <Strg+Alt> [<Objektklasse>]
```

<Tastenname> bezeichnet die Taste, die auf der Tastatur oder an der Maus gedrückt wird. Die anderen Werte geben FIX-Events an, die gesendet werden, wenn die Taste allein, zusammen mit Shift, zusammen mit Strg, zusammen mit Shift und Strg, zusammen mit Alt, zusammen mit Shift und Alt oder zusammen mit Strg und Alt gedrückt wird. Der Wert für Objektklasse muss nicht angegeben werden. In diesem Fall gilt die Definition für alle Objekte. Wenn hier jedoch eine oder mehrere Klassen angegeben werden, dann gilt die Definition nur für diese Klassen. So kann eine Taste durch mehrere Definitionen für verschiedene Objektklassen anderes definiert werden. Folgende Bezeichner können für die Objektklassen verwendet werden:

```
MASK SUBMASK ROLLMASK TABLE SELO CHOICE MENUE HELPTEXT
```

Für die Events ist der zweibuchstabige Eventcode von FIX zu verwenden (so wie er in keys.h auf FIX Seite definiert ist, nur ohne das Postfix K_) oder die Angabe des Eventcodes mit einem vorangestellten '~' Zeichen.

Die Tastennamen sind der folgenden Tabelle zu entnehmen:

Tastenname	**Taste auf deutscher Tastatur**
END	Ende
HOME	Pos1
INSERT	Einfg
DELETE	Entf
LEFT	Pfeil links
RIGHT	Pfeil rechts
UP	Pfeil hoch
DOWN	Pfeil runter
PRIOR	Bild hoch
NEXT	Bild runter
BACK	rückwärts Entfernen
TAB	Tabulator Taste
F1 ... F12	Funktionstasten
NUMPAD0	...
NUMPAD9	Nummernblock 0-9
MULTYPLY	* Nummernblock

ADD	+ Nummernblock
SUBTRACT	- Nummernblock
DIVIDE	/ Nummernblock
DECIMAL	, Nummernblock
RETURN	Eingabetaste
LBUTT	linke Maustaste
MBUTT	mittlere Maustaste
RBUTT	rechte Maustaste
APPS	Menütaste (rechts neben Windows-Taste)
WHEELFW	Mausrad vorwärts
WHEELBW	Mausrad rückwärts
ESCAPE	ESC-Taste
A..Z	Buchstabentasten
0..9	Zahlentasten

7.2 Tastenlabels

Für die Tastenlabels gelten die gleichen Regeln wie bei FIX/Win. Allerdings müssen die Bitmaps im png-Format vorliegen und im Verzeichnis Fixwt-LAF-<style> statt Fixwin-LAF-<style> abgelegt werden. Beim Kopieren der Bitmaps aus einem Gruppenverzeichnis von FIX/Win sind also entsprechende Konvertierungen vorzunehmen. Das gilt auch für die Dateien zur Darstellung der Rahmen (keyFrm*.bmp). Neu ist die Datei keyFrmRollover-<size>.png. Dieser Rahmen wird verwendet, wenn sich die Maus über einem Keylabel befindet.

7.3 Steuerung des infield-Editing

Anders als bei FIX/Win können die Tasten, die zur Felderfassung dienen, (in der jetzigen Version) nicht definiert werden. Statt dessen wurden die Tasten in dem JavaScript Handler, der für die Verarbeitung zuständig ist, fest kodiert.

Der Eventhandler arbeitet nach folgendem Verfahren:

- Es wird zwischen Tasten unterschieden, die an den Browser gesendet werden (und dann oft im Feld landen) und zwischen Tasten, die an FIX/Wt gesendet werden.
- Folgende Tasten werden an FIX/Wt gesendet:
  - Alle Funktionstasten - egal ob in Kombination mit Strg, Shift oder Alt. Ausnahmen: Tasten, die vom System (nicht vom Browser) abgefangen werden. D.h. dass Strg-F4 (Fenster schließen) nicht an FIX/Wt gesendet wird, weil es vom Windows-System abgefangen wird. Allerdings wird F5 (Refresh im Browser) an FIX/Wt gesendet und der Refresh unterbleibt.
  - Tab, Enter, Esc, Up, Down, PgUp, PgDown, Home, Pos1 - ebenfalls in allen Kombinationen (nicht jedoch Left und Right).
  - Alle Tasten, die in Kombination mit Strg gedrückt werden, allerdings nicht Strg-A, Strg-V, Strg-C, Strg-X, die im Feld benötigt werden.
  - Alle Tasten, die in Kombination mit Alt gedrückt werden. Auch hier besteht das Problem, das diese Tasten vom (Windows)system abgefangen werden. So öffnet Alt-A beispielsweise ein Menü des Firefox.
- Alle anderen Tasten werden an den Browser gesendet und damit an das Feld.

8 Verwendung von Semigrafikzeichen

FIX/Wt verwendet genauso wie FIX/Win Semigrafikzeichen zur Darstellung von Rahmen und Linien.

Allerdings müssen die gepackten Dateien im png-Format vorliegen und statt im Verzeichnis LookAndFeel/Fixwin-LAF-<style> im Verzeichnis LookAndFeel/Fixwt-LAF-<style> installiert werden. Ein Unterschied im Format der .bif-Dateien besteht nicht.

Für die ungepackten Bitmaps ist weiterhin das bmp-Format zu verwenden. Der Dateiname dieser Bitmaps hat (wie auch bei FIX/Win) die Form

```
xxx-Y.bmp
```

xxx ist der Grafikcode im Bereich 32-255 und Y das Attribut:

- 0 - normal (NORMAL)
- 1 - invertiert (INVERS)
- 2 - halbhell (LOW)
- 3 - unterstrichen (UNDERLINE)
- 4 - blinkend (BLINK)
- 5 - fett (BOLD)
- 6 - gegraut (GRAYED)
- 7 - unsichtbar (NOTVISIBLE)

8.1 Packen von Bitmaps

Zum Packen kann das Tool

```
bin/bmpbackWt.exe
```

benutzt werden. Es arbeitet ähnlich wie bmpack von FIX/Win, unterstützt jedoch nicht das Entpacken von Bitmaps und verwendet kein Gruppenverzeichnis.

Das Programm ist entweder im Hauptverzeichnis von FIX/Wt oder im Unterverzeichnis bin zu starten. Die Dateien werden aus

```
LookAndFeel-src/Fixwt-LAF[-0|-1|-2|-3]
```

gelesen. Wie bei FIX/Win müssen die einzelnen Bitmaps im bmp-Format in den Unterverzeichnissen

```
bitmap-[M|L|H]
keybitmap-[M|L|H]
```

vorliegen. Beim Packen werden die Dateien in den entsprechenden Unterverzeichnissen

```
LookAndFeel/Fixwt-LAF[-0|-1|-2|-3]
```

abgelegt. Wenn die Unterverzeichnisse nicht existieren, dann werden sie erzeugt. Für die gepackten Bitmaps wird das von FIX/Wt geforderte png-Format verwendet.

Das Zusammenstellen der Bitmapsätze, die gepackt werden, erfolgt über die grafische Oberfläche des Tools. Mittels des Buttons "Speichern" können die Einstellungen in einer

Datei gespeichert werden und zu einem späteren Zeitpunkt mit dem Button "Laden" wieder geladen werden. Damit das funktioniert, muss das Unterverzeichnis

```
bpckcfg
```

existieren.

bmpbackWt unterstützt folgende Kommandozeilenparameter:

- **/C <datei>** - Laden der Einstellungen aus der Konfigurationsdatei mit dem Namen <datei>. Der Pfad ist nicht anzugeben. Alle Dateien liegen im Unterverzeichnis bpckcfg.
- **/A** - Durch Angabe dieses Schalters beginnt das Packen direkt nach dem Start
- **/X** - Durch Angabe dieses Schalters wird bmpbackWt nach dem Packen beendet.

Neben dem Tool bmpbackWt wurde das Tool bmpack erweitert, das zusammen mit (neueren) FIX/Win Versionen (ab 4.5.0 201201101) ausgeliefert wird. Wenn der Schalter

- **/WT <Pfad>** - Erzeugen von Bitmaps für FIX/Wt

angegeben wird, dann werden zusätzlich zu den Bitmaps für FIX/Win auch die Bitmaps für FIX/Wt erzeugt. <Pfad> ist dabei der Pfad zur FIX/Wt Installation, die das Unterverzeichnis lookAndFeel beinhaltet, indem die Ergebnisse abgelegt werden. Wenn die entsprechenden Unterverzeichnisse nicht existieren, dann werden sie erzeugt.

9 Erstellen von Farbzuordnungtabellen

Der Aufbau von Farbzuordnungtabellen entspricht dem von FIX/Win, so dass ein Austausch zwischen beiden möglich ist. Prinzipiell werden auch die mit Version 4.5 eingeführten Farbvariationen unterstützt. Auf das Verwenden einer anderen Variation als 0 sollte bei der Verwendung von FIX/Wt jedoch verzichtet werden, da die übrigen Teile (Benutzerbibliothek) dieses Features erst in einer künftigen Version angepasst werden.

9.1 Aufbau der Farbzuordnungstabelle

Definition von Farben

Eine Zeile der Farbzuordnungstabelle definiert eine Farbe für ein Element von FIX/Wt. Sie hat folgenden Aufbau:

```
<Element> <Textfarbe> <Hintergrund> <Attribut> <Line1> ...
<Line4>
```

<Element> bezeichnet das Element, für das die Farbe definiert wird. Folgende Elemente können verwendet werden:

Element	Beschreibung
BACKGROUND	Allgemeiner Hintergrund

FIELD_NORMAL	Feld mit Attribut "normal"
FIELD_INVERS	Feld mit Attribut "invers"
FIELD_LOW	Feld mit Attribut "low"
FIELD_UNDERLINE	Feld mit Attribut "underline"
FIELD_BLINK	Feld mit Attribut "blink"
FIELD_GRAYED	Feld mit Attribut "grayed"
FIELD_BOLD	Feld mit Attribut "bold"
FIELD_INVISIBLE	Feld mit Attribut "invisible"
TABLE_NORMAL	Tabellenfeld mit Attribut "normal"
TABLE_INVERS	Tabellenfeld mit Attribut "invers"
TABLE_LOW	Tabellenfeld mit Attribut "low"
TABLE_UNDERLINE	Tabellenfeld mit Attribut "underline"
TABLE_BLINK	Tabellenfeld mit Attribut "blink"
TABLE_GRAYED	Tabellenfeld mit Attribut "grayed"
TABLE_BOLD	Tabellenfeld mit Attribut "bold"
TABLE_INVISIBLE	Tabellenfeld mit Attribut "invisible"
SPECIAL_NORMAL	Special-Feld mit Attribut "normal"
SPECIAL_INVERS	Special-Feld mit Attribut "invers"
SPECIAL_LOW	Special-Feld mit Attribut "low"
SPECIAL_UNDERLINE	Special-Feld mit Attribut "underline"
SPECIAL_BLINK	Special-Feld mit Attribut "blink"
SPECIAL_GRAYED	Special-Feld mit Attribut "grayed"
SPECIAL_BOLD	Special-Feld mit Attribut "bold"
SPECIAL_INVISIBLE	Special-Feld mit Attribut "invisible"
MENU_NORMAL	Menüeintrag mit Attribut "normal"
MENU_INVERS	Menüeintrag mit Attribut "invers"
MENU_LOW	Menüeintrag mit Attribut "low"
MENU_UNDERLINE	Menüeintrag mit Attribut "underline"
MENU_BLINK	Menüeintrag mit Attribut "blink"
MENU_GRAYED	Menüeintrag mit Attribut "grayed"
MENU_BOLD	Menüeintrag mit Attribut "bold"
MENU_INVISIBLE	Menüeintrag mit Attribut "invisible"
CHOICE_NORMAL	Choice-Eintrag mit Attribut "normal"
CHOICE_INVERS	Choice-Eintrag mit Attribut "invers"
CHOICE_LOW	Choice-Eintrag mit Attribut "low"
CHOICE_UNDERLINE	Choice-Eintrag mit Attribut "underline"
CHOICE_BLINK	Choice-Eintrag mit Attribut "blink
CHOICE_GRAYED	Choice-Eintrag mit Attribut "grayed"
CHOICE_BOLD	Choice-Eintrag mit Attribut "bold"
CHOICE_INVISIBLE	Choice-Eintrag mit Attribut "invisible"
SELO_NORMAL	Selo-Eintrag mit Attribut "normal"
SELO_INVERS	Selo-Eintrag mit Attribut "invers"
SELO_LOW	Selo-Eintrag mit Attribut "low"
SELO_UNDERLINE	Selo-Eintrag mit Attribut "underline"
SELO_BLINK	Selo-Eintrag mit Attribut "blink"
SELO_GRAYED	Selo-Eintrag mit Attribut "grayed"

SELO_BOLD	Selo-Eintrag mit Attribut "bold"
SELO_INVISIBLE	Selo-Eintrag mit Attribut "invisible"
LABEL_NORMAL	Selolabel mit Attribut "normal"
LABEL_INVERS	Selolabel mit Attribut "invers"
LABEL_LOW	Selolabel mit Attribut "low"
LABEL_UNDERLINE	Selolabel mit Attribut "underline"
LABEL_BLINK	Selolabel mit Attribut "blink"
LABEL_GRAYED	Selolabel mit Attribut "grayed"
LABEL_BOLD	Selolabel mit Attribut "bold
LABEL_INVISIBLE	Selolabel mit Attribut "invisible"
HEAD_NORMAL	Überschrift mit Attribut "normal"
HEAD_INVERS	Überschrift mit Attribut "invers"
HEAD_LOW	Überschrift mit Attribut "low"
HEAD_UNDERLINE	Überschrift mit Attribut "underline"
HEAD_BLINK	Überschrift mit Attribut "blink"
HEAD_GRAYED	Überschrift mit Attribut "grayed"
HEAD_BOLD	Überschrift mit Attribut "bold"
HEAD_INVISIBLE	Überschrift mit Attribut "invisible"
TEXT_NORMAL	Text mit Attribut "normal"
TEXT_INVERS	Text mit Attribut "invers"
TEXT_LOW	Text mit Attribut "low"
TEXT_UNDERLINE	Text mit Attribut "underline"
TEXT_BLINK	Text mit Attribut "blink"
TEXT_GRAYED	Text mit Attribut "grayed"
TEXT_BOLD	Text mit Attribut "bold"
TEXT_INVISIBLE	Text mit Attribut "invisible"
PA_TEXT_NORMAL	Textlabel mit Attribut "normal"
PA_TEXT_INVERS	Textlabel mit Attribut "invers"
PA_TEXT_LOW	Textlabel mit Attribut "low"
PA_TEXT_UNDERLINE	Textlabel mit Attribut "underline"
PA_TEXT_BLINK	Textlabel mit Attribut "blink"
PA_TEXT_GRAYED	Textlabel mit Attribut "grayed"
PA_TEXT_BOLD	Textlabel mit Attribut "bold"
PA_TEXT_INVISIBLE	Textlabel mit Attribut "invisible"
PA_HEADER_NORMAL	Tabellenüberschrift mit Attribut "normal"
PA_HEADER_INVERS	Tabellenüberschrift mit Attribut "invers"
PA_HEADER_LOW	Tabellenüberschrift mit Attribut "low"
PA_HEADER_UNDERLINE	Tabellenüberschrift mit Attribut "underline"
PA_HEADER_BLINK	Tabellenüberschrift mit Attribut "blink"
PA_HEADER_GRAYED	Tabellenüberschrift mit Attribut "grayed"
PA_HEADER_BOLD	Tabellenüberschrift mit Attribut "bold"
PA_HEADER_INVISIBLE	Tabellenüberschrift mit Attribut "invisible"
PA_BUTTON_NORMAL	Button mit Attribut "normal"
PA_BUTTON_INVERS	Button mit Attribut "invers"
PA_BUTTON_LOW	Button mit Attribut "low"
PA_BUTTON_UNDERLINE	Button mit Attribut "underline"

PA_BUTTON_BLINK	Button mit Attribut "blink"
PA_BUTTON_GRAYED	Button mit Attribut "grayed"
PA_BUTTON_BOLD	Button mit Attribut "bold"
PA_BUTTON_INVISIBLE	Button mit Attribut "invisible"
PA_VARBUTTON_NORMAL	Varbutton mit Attribut "normal"
PA_VARBUTTON_INVERS	Varbutton mit Attribut "invers"
PA_VARBUTTON_LOW	Varbutton mit Attribut "low"
PA_VARBUTTON_UNDERLINE	Varbutton mit Attribut "underline"
PA_VARBUTTON_BLINK	Varbutton mit Attribut "blink"
PA_VARBUTTON_GRAYED	Varbutton mit Attribut "grayed"
PA_VARBUTTON_BOLD	Varbutton mit Attribut "bold"
PA_VARBUTTON_INVISIBLE	Varbutton mit Attribut "invisible"

<Textfarbe> und <Hintergrund> definieren die Farbe des Textes und die Hintergrundfarbe. <Attribut> definiert ein Attribut für den Text. Folgende Werte sind zulässig:

Attribut	**Bedeutung**
NORMAL	Normaler Text
UNDERLINE	Text Unterstrichen
ITALIC	Kursiver Text
BOLD	Fettschrift

In <Line1> ... <Line4> können bis zu 4 Linienfarben angegeben werden, die zum Zeichnen von beispielsweise Rahmen verwendet werden. Je nach Element können diese Werte weggelassen werden, da sie zum Zeichnen nicht benötigt werden. Wie welche Linien in welchem Element gezeichnet werden wird im nächsten Abschnitt beschrieben. Für die Farben können entweder vordefinierte Farbwerte verwendet werden:

Farbwerte	**Farbwerte**
WHITE	BLACK
GRAY1	GRAY2
BLUE1	BLUE2
RED1	RED2
GREEN1	GREEN2
CYAN1	CYAN2
MAGENTA1	MAGENTA2
BROWN	YELLOW

oder eine Angabe im RGB Format in der Form

```
RGB_<rot><grün><blau>
```

wobei <rot>, <grün> und <blau> für den Farbanteil als hexadezimaler Wert zwischen 00 und FF steht (Beispiel RGB_0f23Fb).

Oder es kann eine durch DEFINE definierte Farbe für eine Variation verwendet werden.

Definition von Farbvariationen

Die Farben zur Darstellung von Objekten werden als Farbvariationen definiert. Eine Farbvariation besitzt eine Nummer von 0 bis 3. FIX/Wt unterstützt zur Zeit

Farbvariationen nicht vollständig. Trotzdem, muss mindestens die Variation 0 definiert werden, da diese Variation von allen Objekten benutzt wird, für die keine Farbvariation angegeben wurde

Die bisherigen Angaben definieren die allen Variationen gemeinsamen Angaben. Nach diesen Angaben erfolgen die Definitionen der Variationen. Die Definition einer Variation wird eingeleitet durch:

```
VARIATION [0|1|2|3]
```

Gleichzeitig beendet das Schlüsselwort VARIATION die allen Variationen gemeinsamen Angaben. Nach der Angabe der Variation können durch

```
READCOMMON
```

die gemeinsamen Angaben für diese Variation definiert werden. Im einfachsten Fall besteht die Definition einer Variation aus den Zeilen

```
VARIATION 0
READCOMMON
```

Damit wird die Variation 0 definiert, die alle Farben enthält, die am Anfang der Datei definiert wurden.

Bei den weiteren Variationen ist es sinnvoll Abweichungen für bestimmte Elemente zu definieren. Dazu ist die gleiche Syntax für ein Element zu verwenden, wie bei den allgemeinen Definitionen. Soll beispielsweise in Variante 1 die Farbe für die Texte in Proportionalschrift auf rot definiert werden, kann das folgendermaßen erreicht werden:

```
VARIATION 1
READCOMMON
PA_TEXT_NORMAL          RED2       GRAY2
```

Die Anweisung READCOMMON wirkt so, als ob die allgemeinen Definitionen an diese Stelle kopiert werden. Die weiteren Farbangaben überschreiben dann die allgemeinen Angaben. Prinzipiell ist es auch möglich auf READCOMMON zu verzichten und alle Angaben einer Variation zu definieren. Eine weitere Vorgehensweise kann darin bestehen, bei den allgemeinen Definitionen nur die aufzuführen, die bei allen Variationen gleich sind und dann die READCOMMON Anweisung nach der Definition der abweichenden Farbangaben aufzurufen. Dabei besteht jedoch die Gefahr, dass die allgemeinen Definitionen die speziellen überschreiben.

Definieren von Farbbezeichnungen

Durch die Angabe

```
DEFINE <name> <farbe>
```

kann für eine Farbe ein logischer Name vergeben werden. <name> ist dabei ein beliebiger Name und <farbe> eine Farbe in der Form RGB_XXXXXX oder ein vordefinierter Farbwert (RED1, BLUE2, ...). Es kann auch der Name einer bereits mit DEFINE festgelegten Farbe für <farbe> verwendet werden.

Beispiel

```
DEFINE HGCOLOR RGB_ECE9D8
```

Als Farbangabe kann danach der Name statt der Farbe verwendet werden. Da die allgemeinen Farbangaben bis zur Definition der ersten Variation überlesen werden, ist es bei diesen nicht notwendig die Farbbezeichnung vorher zu definieren. Sie muss jedoch bei der Definition der Variation vor dem Einlesen der allgemeinen Angaben mit READCOMMON definiert werden.

Beispiel

Hinweis: In diesem Beispiel sind nicht alle Definitionen vorhanden. Es wird nur die Farbe für Texte definiert, um die Funktionsweise zu zeigen.

```
# Allgemeine Angaben - für alle Variationen gültig
TEXT_NORMAL             BLACK      HGCOLOR
TEXT_INVERS             HGCOLOR    RGB_0048F1
TEXT_LOW                GRAY1      HGCOLOR
TEXT_UNDERLINE          BLACK      HGCOLOR UNDERLINE
TEXT_BLINK              BLUE2      HGCOLOR
TEXT_BOLD               WHITE      HGCOLOR
TEXT_GRAYED             GRAY1      HGCOLOR
TEXT_INVISIBLE          GRAY2      HGCOLOR

# Variation 0 - Default
VARIATION 0
DEFINE HGCOLOR RGB_ECE9D8
READCOMMON


# Variation 1 - Hintergrund weiß, schwarz statt rot
VARIATION 1
DEFINE HGCOLOR WHITE
READCOMMON
TEXT_NORMAL             RED1       HGCOLOR
TEXT_UNDERLINE          RED1       HGCOLOR UNDERLINE
```

Der allgemeine Abschnitt definiert alle Farben für Text. Er verwendet für den Hintergrund die Farbbezeichnung HGCOLOR, obwohl sie noch nicht definiert ist. Sie wird erst vor dem Einlesen der allgemeinen Angaben für Variation 0 definiert. Variation 1 legt für diese Farbdefinition die Farbe weiß fest und ließt dann die allgemeinen Angaben ein. Danach werden die Farbdefinitionen für TEXT_NORMAL und TEXT_UNDERLINE neu definiert.

Es wäre sogar ein Fehler, die Farbbezeichnung HGCOLOR direkt am Anfang zu definieren. Denn dadurch zählt sie zu den allgemeinen Angaben von wird von jeder READCOMMON wieder ausgeführt. Die für die Variation festgelegte Farbe für HGCOLOR wird dann wieder überschrieben.

9.2 Verfahren zum Zeichnen von Linien

Beim Zeichnen von Linien von Elementen wird folgende Regel berücksichtigt:

```
Wenn die Angabe für die Farbe bei allen Linien der
Hintergrundfarbe des Elementes entspricht
(<Line1>=<Line2>=..=<Hintergrund>), dann werden nicht nur die
Linien in der Hintergrundfarbe gezeichnet, sondern der
komplette vom Element belegte Platz.

Dieser Platz wird ansonsten in der durch das Element BACKGROUND
```

```
definierten Farbe gezeichnet.
```

Da durch die vier Linien oft kein weiterer Platz mehr zur Verfügung steht, ist eine Auswirkung dieser Regel meistens nur in der größten Bildschirmgröße (Huge) von FIX/Wt zu sehen. Es gibt jedoch auch Elemente, die nur zwei Linien verwenden. In diesem Fall sind trotzdem alle vier Linien zu definieren, damit die Regel zum Einsatz kommt.

Im folgenden wird beschrieben, wie die Linien in den einzelnen Elementen gezeichnet werden. Die Angabe "Standardverfahren mit 4 Linien" bezieht sich auf folgendes Verfahren:

```
<Line1> wird für die obere äußere Linie verwendet.
<Line2> wird für die obere innere Linie verwendet.
<Line3> wird für die untere innere Linie verwendet.
<Line4> wird für die untere äußere Linie verwendet.
```

Damit werden die Linien von oben nach unten in der angegeben Reihenfolge gezeichnet. Wenn eine äußere Linie nicht gezeichnet werden soll, dann kann diese nicht einfach in der Definition weggelassen werden (dann würde als Farbe rot oder grün verwendet). Statt dessen ist die allgemeine Hintergrundfarbe zu verwenden (BACKGROUND). Wenn eine innere Linie nicht gezeichnet werden soll, dann ist sie mit dem gleichen Wert wie <Hintergrund> zu belegen. (Tatsächlich werden die Linien natürlich gezeichnet. Durch die Farbwahl entsprechen sie jedoch dem Hintergrund des Elements.)

Neben diesem Verfahren gibt es ein zweites Standardverfahren, dass nur zwei Linien verwendet:

```
<Line1> wird für die obere Linie verwendet.
<Line2> wird für die untere Linie verwendet.
```

Dieses Verfahren wird im folgenden Text als "Standardverfahren mit 2 Linien" bezeichnet.

Bei der Angabe der folgenden Elemente wurde der Attributanteil des Elementnamens durch * ersetzt (z.B. FIELD_* statt FIELD_NORMAL), weil das jeweilige Verfahren nicht vom Attribut abhängig ist. Das Attribut bestimmt lediglich den Eintrag, der die Farben für die Linien definiert.

*FIELD_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 4 Linien gezeichnet.

*TABLE_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 4 Linien gezeichnet.

*SPECIAL_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 4 Linien gezeichnet.

*MENU_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 2 Linien gezeichnet.

*CHOICE_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 2 Linien

gezeichnet.

*SELO_**

Wenn das Selo keine Tabellendarstellung verwendet, wird das Standardverfahren mit 2 Linien verwendet. Ansonsten wird das Standardverfahren mit 4 Linien verwendet.

*LABEL_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 2 Linien gezeichnet.

*HEAD_**

Die Linien dieser Elemente werden nach dem Standardverfahren mit 2 Linien gezeichnet.

*TEXT_**

Für diese Elemente werden keine Linien gezeichnet.

*PA_TEXT_**

Für diese Elemente werden keine Linien gezeichnet.

*PA_HEADER_**

Hier werden keine Linien verwendet. Statt dessen wird der Hintergrund durch Bitmaps dargestellt (Siehe Abschnitt 12.6 Seite 45)

*PA_BUTTON_**

Hier werden keine Linien verwendet. Statt dessen wird der Hintergrund durch Bitmaps dargestellt (Siehe Abschnitt 12.6 Seite 45)

*PA_MENUITEM_**

Hier werden keine Linien verwendet. Statt dessen wird der Hintergrund durch Bitmaps dargestellt (Siehe Abschnitt 12.6 Seite 45)

10 Einstellen von Schriftarten

Schriftarten werden in FIX/Wt für jedes LookAndFeel in der Datei

```
LookAndFeel/Fixwt-LAF-<style>/lockAndFeel.frc
```

eingestellt (genauso wie in FIX/Win). Im Unterschied zu FIX/Win gibt es jedoch keinen Default für eine Schriftart und die Schrift ist immer explizit einzustellen. Auch die alte Schriftart FIXWIN wird nicht mehr unterstützt. Der große Unterschied im Gegensatz zu FIX/Win besteht jedoch darin, dass die eingestellte Schriftart auf dem Browser verwendet wird und nicht auf dem Rechner, auf dem FIX/Wt installiert wird. Hier kann es vorkommen, dass auf dem System, auf dem der Browser installiert ist, diese Schriftart nicht zur Verfügung steht. In diesem Fall wird eine Ersatzschriftart ausgewählt, die der eingestellten Schriftart nahe kommt. Diese Schriftart kann von Browser zu Browser und von System zu System unterschiedlich sein. Damit sieht eine Anwendung je nach Browser und System immer etwas anderes aus, was die Schriftart betrifft.

Ein weiterer markanter Unterschied besteht in der Ermittlung der Größe einer Zeichenzelle. Dieser Wert kann über die Ressourcen CellWidth_<size> und

CellHeight_<size> - wie auch in FIX/Win - fest eingestellt werden. In der Regel werden diese Werte jedoch weg gelassen oder mit -1 belegt. In diesem Fall berechnet FIX/Win die Zellengröße auf der Schriftart. Diese Berechnung ist jedoch unter FIX/Wt nicht möglich, da die Schriftart nur auf dem Browser zur Verfügung steht. Deshalb wird die Zellengröße in FIX/Wt anhand der Bitmaps für Semigrafikzeichen bestimmt. Das erste Bitmap - also das mit dem niedrigsten Code - bestimmt durch seine Ausmaße die Größe einer Zeichenzelle. (Bemerkung: Eigentlich sollten alle Bitmaps für Semigrafikzeichen gleich groß sein. Es ist jedoch aufgefallen, dass bei der Anpassung an andere Größen Bitmaps vergessen wurden oder auf die falsche Größe skaliert wurden. Bei der Ausgabe in FIX/Win und in FIX/Wt fällt dies oft nicht direkt auf, da die Bitmaps bei der Ausgabe immer auf die benötigte Größe skaliert werden.)

Abgesehen von diesen Unterschieden, werden nicht alle Ressourcen von FIX/Wt unterstützt. Andererseits sind aber auch einige neue Ressourcen hinzugekommen. Folgende Ressourcen werden von FIX/Wt in Bezug auf Schriftarten gelesen und ausgewertet:

Ressource	**Einstellung**
Font	Schriftart nichtproportional
FontBoldWeight	Stärke der Schriftart mit Attribut Fett
FontSize_M	Größe der Schriftart für Bildschirmgröße Medium
FontSize_L	Größe der Schriftart für Bildschirmgröße Large
FontSize_H	Größe der Schriftart für Bildschirmgröße Huge
PropFont	Schriftart für Paintareas
PropFontWeight	Stärke der Schriftart für Paintareas
PropFontBoldWeight	Stärke der Schriftart für Paintareas mit Attribut Fett
PropFontSize_M	Größe der Schriftart für Paintareas in Bildschirmgröße Medium
PropFontSize_L	Größe der Schriftart für Paintareas in Bildschirmgröße Large
PropFontSize_H	Größe der Schriftart für Paintareas in Bildschirmgröße Huge
PropFontYOffset_M	Verschiebung der Schrift in Y-Richtung (Anzahl Pixel) bei Medium
PropFontYOffset_L	Verschiebung der Schrift in Y-Richtung (Anzahl Pixel) bei Large
PropFontYOffset_H	Verschiebung der Schrift in Y-Richtung (Anzahl Pixel) bei Huge
FieldFont	Schriftart für Felder
FieldFontWeight	Stärke der Schriftart für Felder
FieldFontBoldWeight	Stärke der Schriftart für Felder mit Attribut Fett
FieldFontSize_M	Größe der Schriftart für Felder in Bildschirmgröße Medium
FieldFontSize_L	Größe der Schriftart für Felder in Bildschirmgröße Large
FieldFontSize_H	Größe der Schriftart für Felder in Bildschirmgröße Huge
FieldFontYOffset_M	Verschiebung der Schrift in Y-Richtung (Anzahl Pixel) bei Medium
FieldFontYOffset_L	Verschiebung der Schrift in Y-Richtung (Anzahl Pixel) bei Large
FieldFontYOffset_H	Verschiebung der Schrift in Y-Richtung (Anzahl Pixel) bei Huge
FieldEditYOffset_M	Verschiebung der Schrift in Y-Richtung beim Editieren (Anzahl Pixel) bei Medium
FieldEditYOffset_L	Verschiebung der Schrift in Y-Richtung beim Editieren (Anzahl Pixel) bei Large
FieldEditYOffset_H	Verschiebung der Schrift in Y-Richtung beim Editieren (Anzahl Pixel) bei Huge
CellWidth_M	Breite einer Zelle bei Medium (bei -1 wird die Größe der ersten Bitmap verwendet)

CellWidth_L	Breite einer Zelle bei Large (bei -1 wird die Größe der ersten Bitmap verwendet)
CellWidth_H	Breite einer Zelle bei Huge (bei -1 wird die Größe der ersten Bitmap verwendet)
CellHeight_M	Höhe einer Zelle bei Medium (bei -1 wird die Größe der ersten Bitmap verwendet)
CellHeight_L	Höhe einer Zelle bei Large (bei -1 wird die Größe der ersten Bitmap verwendet)
CellHeight_H	Höhe einer Zelle bei Huge (bei -1 wird die Größe der ersten Bitmap verwendet)
CellSpace_M	Anzahl Pixel, die über und unter Begrenzungslinien in Größe Medium frei bleiben
CellSpace_L	Anzahl Pixel, die über und unter Begrenzungslinien in Größe Large frei bleiben
CellSpace_H	Anzahl Pixel, die über und unter Begrenzungslinien in Größe Huge frei bleiben
CellLineOffset_M	Position der Begrenzungslinien (z.B. für Felder) in Größe Medium
CellLineOffset_L	Position der Begrenzungslinien (z.B. für Felder) in Größe Large
CellLineOffset_H	Position der Begrenzungslinien (z.B. für Felder) in Größe Huge

10.1 Browserspezifische Anpassungen

FIX/Wt verwendet genau wie FIX/Win ein Verfahren zur Felderfassung, bei dem ein Eingabefeld über das aktuelle FIX/Feld geschoben wird. Bei diesem Verfahren kann es in FIX/Wt je nach Browser zu Verschiebungen kommen, weil die Positionsangaben anderes interpretiert werden. Diese Verschiebungen können über die neuen Ressourcen PropFontYOffset, FieldFontYOffset und FieldEditYOffset ausgeglichen werden. Damit diese Einstellung abhängig vom verwendeten Browser vorgenommen werden können, wird zusätzlich zu der Datei lookAndFeel.frc einer weitere Datei mit dem Namen

```
LookAndFeel/Fixwt-LAF-<style>/browserspec.<browser>.frc
```

eingelesen. Die in dieser Datei angegebenen Ressourcen überschreiben die von lookAndFeel.frc. Der Wert <browser> steht dabei für eine Kennzahl, die von Wt für den Browser vergeben wird.

Folgende Werte werden dabei verwendet:

Wert	Dateiname	Browser
1004	browserspec.1004.frc	Internet Explorer
4205	browserspec.4205.frc	Chrome
6107	browserspec.6107.frc	Firefox
4104	browserspec.4104.frc	iPad

11 Verwenden und Anpassen eines WT-Themes

Das Aussehen der Elemente von Wt wird über Themes gesteuert. In FIX/Wt ist es möglich, je nach LookAndFeel (Stil und Größe) ein anderes Theme zu verwenden. Das Aussehen des Themes selbst kann über CSS eingestellt werden, so dass es möglich ist, es so zu gestalten, dass es zum jeweiligen LookAndFeel passt.

11.1 Zuordnung eines Themes

Die Zuordnung eines Themes zu einem LookAndFeel wird in der Datei

```
config/Fixwt.frc
```

vorgenommen. Dazu sind Ressourcen der Form

```
cssTheme_<style>_<size>: <theme>
```

zu definieren. <style> ist der Stil des LookAndFeels (0-3) und <size> die Bildschirmgröße (M|L|H). Der Wert <theme> ist der Name eines Unterverzeichnisses von

```
ressources/themes
```

In diesem Unterverzeichnis liegen alle Dateien, die das Theme braucht. Wenn für ein LookAndFeel kein Theme angegeben wird, dann wird der Wert "polished" verwendet. Dabei handelt es sich um ein mit Wt ausgeliefertes Defaulttheme, das jedoch ebenfalls in das entsprechende Verzeichnis kopiert werden muss, damit es verwendet werden kann.

Beispiel

```
cssTheme_0_L: gray_l
```

Diese Einstellung definiert das Unterverzeichnis gray_l als Theme, wenn das LookAndFeel auf Bildschirmgröße L (Large) und Stil 0 eingestellt wird.

Zur Erzeugung eines neuen Themes kann eines der Unterverzeichnisse kopiert werden und mit einem anderen Namen versehen werden. Zur Anpassung sind die darin befindlichen Bitmaps und die CSS Datei anzupassen. Leider gibt es keine Beschreibung im Rahmen des Produktes Wt, die die einzelnen Einstellungen erklärt. Deshalb hat es sich als praktikabel erwiesen mittels Firebug die bestehenden Dialogboxen zu untersuchen, um so die für die Darstellung verantwortlichen CSS-Einstellungen und Bitmaps zu finden.

Eine Übersicht über die verschiedenen CSS-Dateien und ihre Aufgaben findet sich in Abschnitt 2 CSS Dateien in FIX/Wt Seite 24."

Bei der Einstellung des Themes besteht ein grundsätzliches Problem: Das Theme kann erst festgelegt werden, wenn das LookAndFeel festgelegt wurde. Das LookAndFeel wiederum ergibt sich aus den Einstellungen, die ein Benutzer festgelegt hat. Um das zu einem LookAndFeel passende Theme zu laden ist es deshalb notwendig, den Namen des Benutzers als URL Parameter (user=<name> - siehe Abschnit 5 Starten von FIX/Wt auf Seite 16) anzugeben. Ansonsten wird das zu Stil 0 und Größe 0 zugehörige Theme geladen. Genauso verhält sich FIX/Wt, wenn der Benutzer sich das erste Mal anmeldet und noch kein LookAndFeel definiert und abgespeichert hat.

12 Weitere Konfigurationen

12.1 Fenstertitel, Start- und Endelogos

Der Text der in der Titelzeile des Browsers erscheinen soll, kann durch einen Eintrag in config/fixwt.frc eingestellt werden:

```
appTitle: ALCIB 2012.1
```

Über den Eintrag

```
logostart: resources/logostart.png
```

kann eine Grafik festgelegt werden, die als Hintergrund für den Dialog zum Verbindungsaufbau verwendet wird.

Wenn es zu einem Absturz der Verbindung durch eine Exception kommt, die von FIX/Wt abgefangen wird, wird ebenfalls eine Grafik angezeigt. Diese kann über

```
logokill: resources/logokill.png
```

festgelegt werden.

12.2 Konfiguration des LoadingIndicator

Wenn FIX/Wt auf die Antwort von FIX wartet, wird statt eines Wartecursor ein LoadingIndicator angezeigt. Dieser besteht aus einer GIF-Datei, die eine Animation enthält. Zur Erzeugung von animierten GIF-Dateien für einen LoadingIndicator gibt es spezielle Generatoren im Internet. Z.B.: http://www.chimply.com/Generator#misc-spinner.

Die GIF-Datei besitzt den Basisname loading.gif. Damit auch stil- und größenabhängige Loadingindikatoren definiert werden können, wird der Basisname um eine Größenangabe ergänzt und in den LookAndFeel Verzeichnissen gesucht. Dabei wird folgende Suchreihenfolge angewendet:

```
LookAndFeel\Fixwt-LAF-<style>\loading-<size>.gif
LookAndFeel\Fixwt-LAF-<style>\loading.gif
LookAndFeel\Fixwt-LAF\loading-<size>.gif
LookAndFeel\Fixwt-LAF\loading.gif
```

Der Wert <style> wird dabei durch die Nummer des aktuellen Stils (0-3) und der Wert <size> durch die Kennung der aktuellen Größe (M|L|H) ersetzt.

Das Verhalten des LoadingIndicators lässt sich über Ressourcen in den anwendungsspezifischen Einstellungen anpassen (config/<anwendung>.frc):

```
LoadingIndicatorTimeout: 500
LoadingIndicatorOffOnSync: FALSE
LoadingIndicatorOffsetX: 0
LoadingIndicatorOffsetY: 5
```

LoadingIndicatorTimeout gibt die Zeit in Millisekunden an, die FIX/Wt auf FIX wartet, bevor der Loadingindicator aktiviert wird.

Durch die Angabe von LoadingIndicatorOffOnSync kann bestimmt werden, ob der LoadingIndicator nur durch eine Tastenabfrage von FIX beendet wird (Wert FALSE) oder auch durch das Synchronisieren des Bildschirms durch FIX (Wert TRUE). Das Synchronisieren wird oft verwendet, um ähnliche Anzeigen (z.B. eine Progessbar) zu realisieren, die vom Backend gesteuert werden. Wenn in diesem Fall der Wert auf FALSE steht, dann wird zusätzlich der LoadingIndicator angezeigt.

Mit Hilfe der Ressourcen LoadingIndicatorOffsetX und LoadingIndicatorOffsetY ist es möglich die Position zu bestimmen. Normalerweise wird das animierte GIF zentriert. Durch Angabe eines Offsets kann es außer mittig platziert werden.

12.3 Internationalisierung

Im Unterschied zu FIX/Win wird FIX/Wt als ein einziger Prozess gestartet, zu dem mehrere Sessions von unterschiedlichen Benutzern aufgebaut werden können. Das bedeutet, das die Sprache für FIX/Wt nicht festgelegt werden kann, weil sie sich von Session zu Session unterscheiden kann. Die Sprache wird deshalb aufgrund der Sprache des Browsers für jede Session eingestellt. Damit FIX/Wt zur Sprache passende Meldungen und Dialoge erzeugen kann, wurden alle Texte in die Dateien

```
fixwt_<sprachkuerzel>.xml
wt_<sprachkuerzel>.xml
```

ausgelagert. <sprachkuerzel> ist dabei das zur Sprache passende Kürzel (de für Deutsch, en für Englisch, ...). In fixwt_<sprachkuerzel>.xml stehen alle Texte, die speziell FIX/Wt betreffen, wie z.B. die Bezeichner des Dialogs zum Verbindungsaufbau. In wt_<sprachkuerzel>.xml stehen alle Texte, die das Toolkit Wt definiert, wie z.B. die Monatsnamen des Datepickers. Wenn zu einer Sprache keine Datei gefunden wird, dann werden die Defaultdateien

```
fixwt.xml
wt.xml
```

verwendet, die die Meldungen in Englisch enthalten. Wenn neue Sprachen hinzugefügt werden, dann können diese Dateien als Vorlage verwendet werden. Um ein Mischen von Sprachen zu verhindern sollten immer beide Dateien für eine neue Sprache ergänzt werden.

FIX-Anwendung

Zur Erstellung von FIX Anwendungen stehen entsprechende Verfahren zur Internationalisierung bereit. Aufgabe von FIX/Wt ist es an dieser Stelle nur, die Anwendung in der richtigen Sprache zu starten. Dazu wird von FIX/Wt eine zur Sprache passende Programmtabelle verwendet. Das Format des Dateinamens ist:

```
<basisname>_<sprachkuerzel>.fix
```

<basisname> ist entweder der über URL-Parameter prgtab angegebene Basisname oder wenn der Parameter fehlt, der Wert "program". <sprachkuerzel> ist das vom Browser definierte Sprachkürzel. program_de.fix ist beispielsweise die Programmtabelle für FIX/Anwendungen in deutsch. Existiert die Datei nicht, dann lädt FIX/Wt die Programmtabelle ohne Sprachkürzel und Unterstrich - also program.fix, wenn kein anderer Basisname angegeben wurde.

In der Programmtabelle können die Namen der Programme in der jeweiligen Sprache angegeben werden und beim Kommando können entsprechende Parameter definiert werden, die die FIX-Anwendung in der richtigen Sprache starten.

12.4 Konfiguration der Eingabefelder

FIX/Wt verwendet zur Felderfassung zwei Eingabefelder, die an die richtige Stelle in der Maske zur Erfassung positioniert werden. Eines der Felder wird für rechtsbündige Eingaben verwendet, das andere für linksbündige. Das Aussehen dieser Felder wird über CSS-Klassen gesteuert. Typischerweise werden diese Klassen in

```
LookAndFeel/Fixwt-LAF-<style>/lookAndFeel<size>.css
```

definiert. Folgende Klassen werden verwendet:

inputleft - Feld mit linksbündiger Eingabe

inputright - Feld mit rechtsbündiger Eingabe

Wt-invalid - Eigenschaften für Feld bei Eingabefehlern

12.5 Konfiguration der Dummyfelder

Damit die Felderfassung richtig funktioniert benötigt FIX/Wt zwei Dummyfelder. Diese Felder bekommen beispielsweise den Fokus, wenn die Erfassung nicht aktiv ist und sammeln Zeichen auf, die der Benutzer während dieser Zeit tippt. Die Eigenschaften dieser Felder lassen sich über CSS definieren. Sie werden typischerweise in

```
resources/common.css
```

definiert und sind somit in allen Stilen und Größen gleich. Als CSS-Klassen werden

dummy1 - Eigenschaften von Feld 1

dummy2 - Eigenschaften von Feld 2

verwendet. Die Eigenschaften sollten so gewählt werden, dass die Felder unsichtbar sind. Für Debugzwecke kann es jedoch hilfreich sein, den Inhalt der Felder sichtbar zu machen.

12.6 Standardbitmaps zur Darstellung von Paintareas

Wenn die Darstellung von Paintareas von FIX/Wt übernommen wird, dann werden dazu (anderes als bei FIX/Win) Bitmaps geladen, die den Rahmen der Paintareas darstellen. Diese Bitmaps müssen die Größe von drei Zeichenzellen belegen. Beim Zeichnen der Paintarea werden diese auf folgende Weise gestreckt: Der Teil, der der ersten Zeichenzelle entspricht wird für den linken Rand der Paintarea verwendet. Der Teil, der der dritten Zeichenzelle entspricht wird für den rechten Rand verwendet. Der Raum dazwischen wird durch das wiederholte Zeichnen des Teils gefüllt, der der zweiten Zeichenzelle entspricht.

Folgende Bitmaps im png-Format werden geladen:

Dateiname	**Verwendung**
paMenuitem_<attr>-<size>.png	Menüpunkt
paVarButton_<attr>-<size>.png	Varbutton (Reiter)
paTableheader_<attr>-<size>.png	Tabellenkopf
paButton_<attr>-<size>.png	Button

<attr> steht dabei für das Bildschirmattribut, mit dem die Paintarea dargestellt wird:

- 0 - normal (NORMAL)
- 1 - invertiert (INVERS)
- 2 - halbhell (LOW)
- 3 - unterstrichen (UNDERLINE)
- 4 - blinkend (BLINK)
- 5 - fett (BOLD)
- 6 - gegraut (GRAYED)
- 7 - unsichtbar (NOTVISIBLE)

<size> steht für den Code der Bildschirmgröße M,L oder H.

Die Dateien müssen für jedes Attribut, jede Größe und für jeden Stil, den der Benutzer einstellen kann, im passenden Unterverzeichnis vorhanden sein:

```
LookAndFeel/Fixwin-LAF-<style>
```

wobei style die Nummer des Stils im Bereich von 0 bis 3 ist.

Wenn die Paintareas durch die Benutzerbibliothek gezeichnet werden, dann lädt FIX/Wt trotzdem diese Bitmaps, da nicht analysiert werden kann, welche Arten von Paintareas die Zeichenroutine darstellt. Dabei kommt es zu Fehlermeldungen, wenn ein Bitmap nicht geladen werden kann. Wenn jedoch sichergestellt ist, dass die Benutzerbibliothek alle Paintareas zeichnet und die Bitmaps nicht benötigt werden, kann durch Setzen des Schalters

```
NoPaintareaBitmaps: TRUE
```

in

```
LookAndFeel/lookAndFeel.frc
```

ein Laden der Bitmaps unterbunden werden. Die Bitmaps müssen in diesem Fall nicht (für jeden Stil und jede Größe) bereitgestellt werden.

V Erstellen einer Benutzerbibliothek

Wie FIX/Win kann auch FIX/Wt über eine eigene Benutzerbibliothek um Funktionalitäten erweitert werden. In der ersten Version wurden nicht alle Funktionen von FIX/Win portiert. Die in FIX/Wt vorhanden Funktionen werden im Folgenden beschrieben.

1 Allgemeine Hinweise

1.1 Laden der Benutzerbibliothek

Um beim Start von FIX/Wt eine Bibliothek als Benutzerbibliothek zu laden, ist sie über die Ressource

```
fwown: <Pfad>
```

in der Datei

```
config/fixwt.frc
```

zu definieren. <Pfad> ist der Pfad zu der Bibliothek. Er kann absolut oder relativ zum Installationsverzeichnis von FIX/Wt angegeben werden.

1.2 Include Dateien

Alle Datentypen, Definitionen, Makros und Prototypen zur Erstellung einer Benutzerbibliothek sind in den beiden Dateien

```
include/fwownWt.h
include/fwownStdWt.h
```

definiert. Obwohl es in FIX/Wt keine wirkliche Unterteilung in Kernklassen und Framework gibt, gibt es zur Zeit noch zwei Include-Dateien, da die Struktur im Quellcode noch vorhanden ist. Diese wird in einer der nächsten Versionen bereinigt und damit auf eine Datei reduziert.

1.3 Arten von Funktionen

Bei der Verwendung einer Benutzerbibliothek sind zwei Arten von Funktionen relevant:

- Funktionen, deren Code in der Benutzerbibliothek liegt. Diese Funktionen werden im kommenden Text FWown-Funktionen genannt. Sie werden von FIX/Wt zu bestimmten Zeitpunkten aufgerufen. Damit dies funktioniert, muss die Benutzerbibliothek diese Funktionen (über eine Moduldefinitionsdatei – def-Datei) exportieren.
- Funktionen, deren Code in FIX/Wt liegt und die der Benutzerbibliothek zur Verfügung gestellt werden. Diese Funktionen werden im folgenden Text FIX/Wt-Funktionen genannt.

Da die Definition der FWown-Funktionen sowohl FIX/Wt als auch der Benutzerbibliothek bekannt sein müssen, werden die Dateien fwownWt.h und fwownStdWt.h von beiden verwendet. Bei der Verwendung in dem Code der Benutzerbibliothek ist es wichtig, dass vor dem Einschließen dieser Dateien das Makro

```
#define FWOWN_EXPORTS
```

gesetzt wird, damit die Funktionen richtig definiert werden.

1.4 Parameter CallBack

Alle FWown-Funktionen bekommen den Parameter CallBack. Er ermöglicht den Zugriff auf FIX/Wt-Funktionen. Dazu enthält er einen Zeiger auf eine Funktion innerhalb des FIX/Wt Codes. Durch Aufruf dieser Funktion mit einer bestimmten Konstante als Parameter können Zeiger auf weitere Funktionen von FIX/Wt ermittelt werden. Die Konstanten, die als Parameter verwendet werden können, sind in den Dateien fwownWt.h und fwownStdWt.h als Makros definiert. Der Rückgabewert der Callback-Funktion ist in einen Funktionszeiger des entsprechenden Typs zu casten.

Um die Programmierung zu erleichtern wird in fwownWt.h für jede Funktion ein Makro definiert, das als Parameter den Callback und die Variable, die den Funktionszeiger aufnehmen soll, bekommt. Das Makro prüft, ob die Variable bereits mit einem Funktionszeiger belegt ist (!= NULL) und ruft ansonsten die Callback Funktion zur Ermittlung des Funktionszeigers auf. Bei der Zuweisung an die Variable wird dann in den richtigen Typ gecastet. In den folgenden Beschreibungen der FIX/Wt-Funktionen werden neben dem Prototyp das Makro für die Konstante der Callback-Funktion und das Makro zum Aufruf der Callback-Funktion angegeben.

1.5 Parameter p_callID

Der Parameter p_callID ist genauso wie der Parameter CallBack, bei allen FWown-Funktionen vorhanden. Außerdem besitzt jede FIX/Wt-Funktion diesen Parameter als ersten Parameter. Beim Aufruf einer FIX/Wt-Funktion ist der Wert, den die FWown-Funktion von FIX/Wt bekommen hat, an die FIX/ Wt-Funktion weiterzugeben. Ansonsten muss sich der Entwickler der Benutzerbibliothek nicht um diesen Parameter kümmern.

2 Multisessionfähigkeit

Da die Bibliothek - genau wie der FIX/Wt Prozess - nicht für jede Session geladen wird, muss die Bibliothek multisessionfähig sein. Das bedeutet, dass die Funktionen der Bibliothek von mehreren Threads, die verschiedenen Sessions zugeordnet sind, gleichzeitig aufgerufen werden können.

Ein Beispiel verdeutlicht die Problematik:

```
static _TCHAR errorbuf[ERRORBUFLEN];

FWOWN_API _TCHAR* CALLBACK FwownExec(CallBackFct CallBack,
void* p_callID, _TCHAR* str)
{
     ...

    switch (iCommandSwitch) {
          ...

         default:
              _stprintf(errorbuf, _T("Unbekanntes Kommando:
%s"),cmd);
             return errorbuf;
     }
}
```

FwownExec kann von zwei Threads, die zwei verschiedenen Sessions zugeordnet sind, zur gleichen Zeit aufgerufen werden. Dabei kann das Ergebnis von errorbuf von dem einen Thread wieder überschrieben werden, bevor der andere zurückkehrt. Die Funktion liefert somit das falsche Ergebnis.

Wenn eine bestehende Bibliothek, die zusammen mit FIX/Win verwendet wurde, für FIX/Wt portiert werden soll, sind folgende Dinge zu klären:

- Bei statischen Variablen muss überlegt werden, ob sie pro Session vorhanden sein müssen, oder ob sie gemeinsam genutzt werden können.
- Wenn eine Variable pro Session genutzt wird, dann muss sie für jede Session angelegt werden.
- Wenn eine Variable gemeinsam genutzt wird, dann muss überlegt werden, ob ein synchronisierter Zugriff notwendig ist.

Grundsätzlich ist es notwendig, über einen Schlüssel die Session zu eindeutig zu identifizieren und über diesen Schlüssel auf die Variablen zuzugreifen. Da FIX/Wt (und FIX/Win) multisessionfähig ist, besteht dort das gleiche Problem. Dazu wurde der Parameter p_callID eingeführt, der beispielsweise beim Aufruf von SendEvent() dafür sorgt, bei mehreren Verbindungen (wenn FIX/Win nicht mit /single gestartet wird) das Event an die richtige FIX-Anwendung zu senden. Dieser Parameter ist also eine eindeutige ID für eine Session und kann deshalb als Schlüssel verwendet werden. Man könnte also alle Variablen in eine Struktur packen, bei einer neuen Session ein neues Element der Struktur allokieren und den Zeiger auf die Struktur in einer Hashtable ablegen, die die p_callID als Schlüssel besitzt. Beim Zugriff auf die Variable ließt man dann über die p_callID den Zeiger aus der Hashtable.

Da die p_callID nichts anderes ist, als ein Zeiger auf ein FIX/Wt internes Objekt, bietet sich statt der Hashtable ein effizienteres Verfahren an. Der Zeiger auf die Struktur mit Variablen wird als Element dieses Objektes gespeichert. Zum Zugriff gibt es zwei neue Funktionen:

```
void SetSessionPtr(void* p_callID, void* ptr);
void* GetSessionPtr(void* p_callID);
```

Mit SetSessionPtr() kann zu einer p_callID ein Zeiger gespeichert werden, der dann mit GetSessionPtr() wieder gelesen werden kann. Der Zugriff ist wesentlich einfacher und schneller als über eine Hashtable.

Damit der Speicher zum richtigen Zeitpunkt angelegt und gelöscht werden kann, werden folgende Hooks von FIX/Wt aufgerufen:

```
    HOOK_SESSION_START
    HOOK_SESSION_END
```

2.1 Vorschlag zur Umstellung

Alle statischen Variablen werden in die neue Klasse CSessionData (Name frei wählbar) gepackt. Sie dürfen dann allerdings nicht mehr statisch sein.

Außerdem wird ein Konstruktor für die Klasse definiert, in dem alle Variablen initialisiert werden. Im Destruktor können ggf Aufräumarbeiten vorgenommen werden.

```
class CSessionData {
public:

    _TCHAR errorbuf[ERRORBUFLEN];
    ...
    _TCHAR* var_x;
    ...
    short m_nY;;
    ...
    CSessionData();
    ~CSessionData();
};

CSessionData::CSessionData() {
    var_x      = NULL;
    m_nY = 0;
}
```

Bei jedem Aufruf des Hooks HOOK_SESSION_START wird ein Objekt der Klasse angelegt und dessen Zeiger gespeichert. Beim Hook HOOK_SESSION_END wird das Element gelesen und dann wieder gelöscht.

```
GetSessionPtrFct    GetSessionPtr   = NULL;
SetSessionPtrFct    SetSessionPtr   = NULL;

...

FWOWN_API _TCHAR* CALLBACK FwownHook(CallBackFct CallBack,
void* p_callID, int id)
{
    CSessionData* sdta;

    switch ( id ) {
    case HOOK_SESSION_START:
```

```
            FWFCT_GETSESSIONPTR(CallBack, GetSessionPtr);
            FWFCT_SETSESSIONPTR(CallBack, SetSessionPtr);
            sdta = new CSessionData();
            SetSessionPtr(p_callID, (void*)sdta);
            break;
      case HOOK_SESSION_END:
            sdta = (CSessionData*)GetSessionPtr(p_callID);
            delete sdta;
            SetSessionPtr(p_callID, (void*)0);
            break;
            ...
      }
}
```

Wenn der Code jetzt kompiliert wird, dann werden Fehler gemeldet, weil nicht mehr auf die ursprünglichen statischen Variablen zugegriffen werden kann. Diese Fehler werden behoben, indem vor jeden Zugriff der Zeiger "sdta->" geschrieben wird. Die Fehlerliste von Visual Studio unterstützt beim Auffinden der betroffenen Stellen.

Jetzt muss noch dafür gesorgt werden, dass der Zeiger sdta an der entsprechenden Stelle auch zur Verfügung steht. Hier gibt es zwei Möglichkeiten:

- Die Funktion, in der der Zeiger benötigt wird, bekommt als Parameter p_callID.
- Die Funktion bekommt den Parameter nicht.

Im ersten Fall kann der Zeiger durch Einfügen der Zeile

```
CSessionData* sdta = (CSessionData*)GetSessionPtr(p_callID);
```

am Anfang der Funktion ermittelt werden.

Im zweiten Fall muss überlegt werden, ob die Funktion um den Parameter p_callID oder sdta erweitert wird. Das hängt davon ab, welche Parameter die aufrufende Funktion kennt. Letztendlich werden alle Funktionen indirekt von FIX/Wt aufgerufen und besitzen somit den Parameter p_callID.

Wird die Funktion um p_callID erweitert, dann kann sdta nach dem obigen Verfahren ermittelt werden.

Beispiel

Die Funktion bekommt p_callID:

```
FWOWN_API _TCHAR* CALLBACK FwownExec(CallBackFct CallBack,
void* p_callID, _TCHAR* str)
{
     CSessionData* sdta =
(CSessionData*)GetSessionPtr(p_callID);
     ...
     switch (iCommandSwitch) {
            ...
     default:
            _stprintf(sdta->errorbuf, _T("Unbekanntes Kommando:
%s"),cmd);
            return sdta->errorbuf;
     }
}
```

Die Funktion bekommt p_callID nicht:

```
static BOOL ExecAdditionalCmd(CSessionData* sdta, _TCHAR *
addcmd, _TCHAR * errbuf )
{

     ...
}
```

Hier wurde der Parameter sdta ergänzt, da der Aufrufer (FwownExec) diesen Wert bereits kennt.

3 Binden mit dem Wt Tookit

Um zum Zeichnen die Funktionen von Wt zu nutzen, muss die Bibliothek mit den Bibliotheken von Wt (Version 3.2.1) gebunden werden. Da Wt die Boost Bibliothek (Version 1.46.1) und GraphicsMagick (Version 1.3.13) benutzt, sind auch diese einzubinden. In den Einstellungen des Visual Studio Projektes müssen die im folgenden beschriebenen Einstellungen gesetzt werden. Dabei wird davon ausgegangen, dass im Verzeichnis <pfad>\WtlLibs die benötigten Dateien von Wt, Boost und GraphicsMagick installiert wurden. Die Werte in eckigen Klammern geben die Einstellungen zum Erstellen einer Release-Version an. Ansonsten wird von einer Debug Version ausgegangen.

Konfigurationseigenschaften – C/C++ - Allgemein – Zusätzlich Includeverzeichnisse:

<pfad>\WtLibs\boost-1.46.1
<pfad>\WtLibs\boost-1.46.1\boost\tr1
<pfad>\WtLibs\wt-3.2.1-rc2\include

Konfigurationseigenschaften – C/C++ - Codegenerierung – Laufzeitbibliothek:

Multithreded-Debug (/Mtd)
[Multithreded (/MT)]

Konfigurationseigenschaften – Linker – Eingabe – Zusätzliche Abhängigkeiten:

wtd.lib [wtd.lib]
CORE_DB_bzlib_.lib [CORE_RL_bzlib_.lib]
CORE_DB_coders_.lib [CORE_RL_coders_.lib]
CORE_DB_filters_.lib [CORE_RL_filters_.lib]
CORE_DB_jbig_.lib [CORE_RL_jbig_.lib]
CORE_DB_jp2_.lib [CORE_RL_jp2_.lib]
CORE_DB_jpeg_.lib [CORE_RL_jpeg_.lib]
CORE_DB_lcms_.lib [CORE_RL_lcms_.lib]
CORE_DB_libxml_.lib [CORE_RL_libxml_.lib]
CORE_DB_png_.lib [CORE_RL_png_.lib]
CORE_DB_tiff_.lib [CORE_RL_tiff_.lib]
CORE_DB_ttf_.lib [CORE_RL_ttf_.lib]
CORE_DB_wand_.lib [CORE_RL_wand_.lib]
CORE_DB_wmf_.lib [CORE_RL_wmf_.lib]
CORE_DB_zlib_.lib [CORE_RL_zlib_.lib]
CORE_DB_magick_.lib [CORE_RL_magick_.lib]
CORE_DB_Magick++_.lib [CORE_RL_Magick++_.lib]

Konfigurationseigenschaften – Linker – Allgemein – Zusätzliche Bibliotheksverzeichnisse:

<pfad>\WtLibs\boost-1.46.1\lib
<pfad>\WtLibs\wt-3.2.1-rc2\lib
<pfad>\WtLibs\GraphicsMagick-1.3.13\VisualMagick\lib

Abgesehen davon sind die entsprechenden .h Dateien einzuschließen, wenn eine entsprechende Klasse von Wt benutzt wird. Dabei ist es hilfreich, den namespace wt global bekannt zu machen:

```
#include <Wt/WPainter>
#include <Wt/WException>

using namespace Wt;
```

4 Beschreibung der FWown-Funktionen

FwownGetVersionInfo - Versionsinfo ermitteln

Prototyp

```
extern FWOWN_API char* CALLBACK FwownGetVersionInfo(
    CallBackFkt p_callBack, void* p_callID, int* version);
```

Aufrufzeitpunkt

Beim Verbindungsaufbau.

Beschreibung

Die Funktion wird zur Ermittlung einer Versionsangabe aufgerufen. Die Funktion liefert als Ergebnis einen (selbst zu definierenden) Versionsstring und einen (selbst zu definierenden) Featurelevel, der in der Variablen version abgelegt werden muss. Auf der FIX Seite können diese Werte mittels

```
char *fx_fwown_get_version_info(int *feature_level);
```

abgefragt und für eigene Auswertungen genutzt werden. Dazu wird die Funktion unmittelbar nach dem Verbindungsaufbau aufgerufen. Besitzt die Benutzerbibliothek nicht die Funktion FwownGetVersionInfo, dann wird eine leere Zeichenkette("") als Rückgabewert geliefert und 0 nach feature_level geschrieben.

FwownExec - Backend Request

Prototyp

```
FWOWN_API char* CALLBACK FwownExec(CallBackFkt p_callBack,
    void* p_callID, char* str);
```

Aufrufzeitpunkt

Wenn die Funktion fx_exec_frontend(char* str) von der FIX-Anwendung aufgerufen wird.

Beschreibung

Die von der FIX-Anwendung übergebene Zeichenkette wird an die Funktion FwownExec() als Parameter str übergeben. Diese kann den String analysieren und aufgrund des Inhaltes bestimmte Aktionen ausführen. Als Rückgabewert muss sie entweder NULL im Erfolgsfall oder einen Zeiger auf eine Zeichenkette mit einer Fehlermeldung liefern. Die Fehlermeldung wird von FIX/Wt im Meldungsfenster ausgegeben. Die Größe der Zeichenkette in str ist durch den Protokollpuffer von FIX/Wt auf ca. 8000 Bytes begrenzt.

FwownProcessData - Binäre Daten verarbeiten

Prototyp

```
FWOWN_API char* CALLBACK FwownProcessData(
    CallBackFkt p_callBack, void* p_callID,
    long* p_bytes, char** p_data);
```

Aufrufzeitpunkt

Wenn bei dem Anfordern einer Taste Daten von FIX übertragen wurden.

Beschreibung

Im Gegensatz zu FwownExec() können von dieser Funktion binäre Daten beliebiger Länge empfangen werden. Weiterhin ist es möglich, die Daten nach einer Veränderung zurück an die FIX-Anwendung zu übermitteln. Aufgerufen wird diese Funktion als Reaktion auf die FIX Funktion fx_transfer_to_frontend(int bytes, char* data). Da die Daten stückchenweise über den Protokollpuffer gesendet werden müssen, wird die Funktion erst aufgerufen, wenn FIX eine Taste anfordert. Denn nur in diesem Zustand

ist es möglich, die veränderten Daten als Antwort an das FIX Programm zurückzusenden.

Der Parameter bytes enthält einen Zeiger auf die Länge der Daten. Dieser Wert kann von der Funktion geändert werden. Ist er nach Ablauf der Funktion > 0, dann wird diese Menge an Daten an das FIX Programm zurückgesendet. Der Parameter data enthält einen Zeiger auf die Adresse der Daten. Auch dieser Wert kann geändert werden, was sinnvoll ist, wenn ein größerer Datenbereich benötigt wird. Die Adresse muss jedoch im Speicherbereich von FIX/Wt liegen. Deshalb ist ein neuer Puffer mit der Funktion ResizeBuffer() (kann durch Aufruf von FWFKT_RESIZEBUFFER ermittelt werden) zu allokieren. Sie bekommt als Parameter den alten Speicherbereich und die neue Größe und liefert die Adresse des neuen Puffers.

Da das FIX/Programm zum Abarbeitungszeitpunkt der Funktion FwownProcessData() sich in der Tastaturabfrage befindet, kann der veränderte Puffer nicht als Rückgabewert der Funktion fx_transfer_to_frontend() zurückgeliefert werden. Statt dessen bekommt die FIX Anwendung das Event K_SB. Sie kann bei der Abarbeitung dieses Events mittels getFrontendData(char ** data) den Zeiger auf die Daten (Parameter data) und die Länge (Rückgabewert) ermitteln. Diese Daten stehen jedoch nach der Abarbeitung des Events nicht mehr zur Verfügung und sollten deshalb, wenn notwendig, in einen eigenen Puffer kopiert werden

FwownDrawPaintAreaWt - PaintArea zeichnen

Prototyp

```
FWOWN_API _TCHAR* CALLBACK FwownDrawPaintAreaWt(
    CallBackFct CallBack, void* p_callID,
    WPainter* p_painter,
    int p_x, int p_y, int width, int height,
    unsigned long p_pa_id, short p_pa_type, _TCHAR* strLabel,
    unsigned short attr, short p_pa_align,
    long p_longval1, long p_longval2,
    long p_longval3, long p_longval4,
    _TCHAR* ms_name, short p_objclass, int aktSize,
    int aktStyle, short p_variation);
```

Aufrufzeitpunkt

Wenn eine Paintarea oder ein Teil einer Paintarea sich geändert hat und neu gezeichnet werden muss.

Beschreibung

Die Funktion hat die Aufgabe eine Paintarea zu zeichnen. Die Parameter haben folgende Bedeutung:

- p_callBack und p_callID - Dies sind die üblichen Parameter zum Aufruf von FIX/Wt-Funktionen, wie sie an alle Funktionen der Benutzerbibliothek übergeben werden.
- p_painter - Mit Hilfe dieser Klasse kann die Paintarea durch Zeichenoperationen

gefüllt werden. Die zur Verfügung stehenden Methoden sind in der Wt-Dokumentation beschrieben.

- p_x, p_y, width, height - Rechteck, das die Lage der Paintarea definiert. In diesem Rechteck müssen die Zeichenoperationen ausgeführt werden. **Hier besteht ein wesentlicher Unterschied zu FIX/Win. In FIX/Win beginnen alle Zeichenoperation ab Position 0,0. In FIX/Wt ist unbedingt die Position p_x, p_y zu verwenden, sonst wird außerhalb der Paintearea gezeichnet**.
- p_pa_id - ist eine eindeutige Nummer für die Paintarea. Mit Hilfe dieser Nummer ist es möglich, Daten zu einer Paintarea in der Benutzerbibliothek zu verwalten und zum Darstellen zu nutzen.
- p_pa_type - beinhaltet den Typ der Paintarea. Es kann eine der folgenden Konstanten aus fwownWt.h sein:
  - PA_TEXTLABEL - ein Textlabel.
  - PA_TABLEHEADER - ein Text als Kopf einer Tabellenspalte.
  - PA_BITMAP - eine Bitmap.
  - PA_BUTTON - ein Button.
  - PA_VARBUTTON - ein Button zur Umschaltung von Varianten.
  - PA_USERDEFINED - ein benutzerdefinierter Typ.
- p_text, p_pa_attr - Der Parameter p_text enthält den übersetzten Text der Paintarea. In p_pa_attr steht das Textattribut dazu. Abgesehen von dem Typ PA_BUTTON enthält diese Variable den durch pa_declare() oder pa_put() definierten Wert.
- p_align - Die Ausrichtung des Textes wird durch den Wert in p_align bestimmt. Folgende Werte sind möglich:
  - PA_ALGN_LEFT - Der Text soll linksbündig ausgerichtet werden.
  - PA_ALGN_CENTER - Der Text soll zentriert werden.
  - PA_ALGN_RIGHT - Der Text soll rechtsbündig ausgerichtet werden.
- p_longval1, p_longval2, p_longval3 und p_longval4 - Die Werte in p_longval1, p_longval2, p_longval3 und p_longval4 stammen aus den gleichnamigen Parametern, die bei pa_put() oder pa_declare() anzugeben sind. Zusammen mit p_ms_name und p_objclass, die den Namen und die Objektklasse des Objekts beinhalten, dem die Paintarea zugeordnet ist, können diese Werte verwendet werden, um davon gewisse Abweichungen beim Zeichnen abzuleiten. Einige der Werte werden von FIX bereits mit einer festen Bedeutung versehen (siehe unten bei den Besonderheiten der verschiedenen Typen).
- p_size, p_style Um Paintareas je nach Look&Feel anders darstellen zu können, wird in p_size die aktuelle Größe (0-2) und in p_style der aktuelle Stil (0-3) übergeben.

Beim Aufruf der Funktion FwownDrawPaintArea() kann davon ausgegangen werden, dass vorher:

- Die zum Attribut passende Zeichenfarbe und Hintergrundfarbe eingestellt wurde.
- Der Hintergrund durch Zeichnen eines Rechtecks in dieser Farbe gelöscht wurde.
- Die von FIX/Wt definierte Schrift für das LookAndFeel eingestellt wurde.

Für den Rückgabewert der Funktion gibt es drei verschiedene Möglichkeiten:

- Es ist ein Fehler aufgetreten: Der Rückgabewert muss auf eine Fehlermeldung zeigen, die von FIX/Wt ausgegeben wird.
- Der Bereich konnte von der Funktion gefüllt werden: Als Rückgabewert muss NULL verwendet werden.

- Die Paintarea wurde nicht von der Funktion der Benutzerbibliothek gefüllt: Als Rückgabewert muss "" (leere Zeichenkette) verwendet werden. FIX/Wt übernimmt in diesem Fall die Darstellung durch eine Standardmethode.

Besonderheiten des Typs PA_BUTTON

Bei dem Typ PA_BUTTON bestimmt FIX/Wt das Attribut in p_pa_attr, wenn es nicht von pa_declare() oder pa_put() als GRAYED definiert wurde. Dazu wird der Zustand des Buttons ausgewertet. Die möglichen Zustände und die zugeordneten Attribute sind wie folgt definiert:

- "Normal" - wie von pa_declare() oder pa_put() definiert.
- "Mauszeiger über Paintarea" - A_BOLD
- "Button gedrückt" - A_INVERS

Besonderheiten des Typs PA_TABLEHEADER

Für den Typ PA_TABLEHEADER definiert FIX die folgenden Werte für p_longval1:

- PA_TABLEHEADER_FIRST markiert den Kopf der ersten und der folgenden Tabellenspalten.
- PA_TABLEHEADER_LAST markiert den Kopf der letzten Tabellenspalte, der anders gezeichnet werden muss als die der ersten Spalten.

Besonderheiten des Typs PA_VARBUTTON

Für den Typ PA_VARBUTTON enthält longval1 in den Bits 8-9 die Ausrichtung und in den übrigen Bits die Art des Buttons. Für die Art kommen folgende Werte in Betracht:

- 0L - normaler Varbutton
- PA_VBTN_ACTIVE - aktiver Varbutton
- PA_VBTN_SCRL_FWRD - Varbutton, um nach vorne zu scrollen.
- PA_VBTN_SCRL_BWRD - Varbutton, um nach hinten zu scrollen.

Hierbei ist zu beachten, dass in den Bits 8-9 von longval1 die Ausrichtung kodiert ist. Daher ist vor einem Vergleich der Wert von longval1 mit dem Komplementärwert von PA_VBTN_PLMT_BITS zu verknüpfen (longval1 & ~PA_VBTN_PLMT_BITS). Als Wert für die Ausrichtung kommen folgende Werte in Betracht:

- PA_VBTN_PLMT_TOP - oben liegender Reiter.
- PA_VBTN_PLMT_BOTTOM - unten liegender Reiter.
- PA_VBTN_PLMT_LEFT - links liegender Reiter.
- PA_VBTN_PLMT_RIGHT - rechts liegender Reiter.

Um hier nur die Bits 8-9 für einen Vergleich zu erhalten ist mit dem Wert PA_VBTN_PLMT_BITS zu verknüpfen (longval1 & PA_VBTN_PLMT_BITS). Das Zeichnen von links und rechts liegenden Reitern ist nicht notwendig, da diese von FIX z.Z. noch nicht unterstützt werden.

Der folgende Code zeigt ein Grundgerüst, das den Wert in longval1 auswertet:

```
long placement;
long subtype;

subtype = longval1 & ~PA_VBTN_PLMT_BITS;
placement = longval1 & PA_VBTN_PLMT_BITS;

switch (placement) {
case PA_VBTN_PLMT_BOTTOM:
     switch (subtype) {
     case PA_VBTN_ACTIVE:
          /* Zeichne aktiven Varbutton mit Ausrichtung unten */
          break;
     case PA_VBTN_SCRL_FWRD:
          /* Zeichne Varbutton zum Vorwärtsscrollen mit
Ausrichtung unten */
          break;
     case PA_VBTN_SCRL_BWRD:
          /* Zeichne Varbutton zum Rückwärtsscrollen mit
Ausrichtung unten */
          break;
     default:
          /* Zeichne normalen Varbutton mit Ausrichtung unten
*/
     break;
     }
     break;
case PA_VBTN_PLMT_TOP:
     switch (subtype) {
     case PA_VBTN_ACTIVE:
          /* Zeichne aktiven Varbutton mit Ausrichtung oben */
          break;
     case PA_VBTN_SCRL_FWRD:
          /* Zeichne Varbutton zum Vorwärtsscrollen mit
Ausrichtung oben */
          break;
     case PA_VBTN_SCRL_BWRD:
          /* Zeichne Varbutton zum Rückwärtsscrollen mit
Ausrichtung oben */
          break;
     default:
          /* Zeichne normalen Varbutton mit Ausrichtung oben */
          break;
     }
     break;
}
```

Neben dem Zeichnen der Varbuttons ist es notwendig, den Rahmen so darzustellen, dass er zur Art der Varbuttons passt. Dazu ist mit Semigrafikzeichen ein Rahmen um den Bereich zu zeichnen, der auch die Leiste mit den Varbuttons in der oberen Zeile (unteren Zeile bei PA_VBTN_PLMT_BOTTOM) enthält. Für die Semigrafikzeichen sind typischerweise Codes zu verwenden, die bisher noch nicht in der Anwendung genutzt wurden. Für diese Codes sind Bitmaps in FIX/Wt zu erstellen, die zu den Varbuttons in Ausrichtung und Farbe passen. Die Varbuttons werden auf diesen Rahmen platziert. Die Funktionen von FIX sorgen dafür, dass die Zeichen unten den Varbuttons gerettet werden und beim Entfernen des Varbuttons oder beim Scrollen der Leiste wieder dargestellt werden.

FwownHook - FIX/Wt hat bestimmte Codestelle erreicht

Prototyp

```
FWOWN_API char* CALLBACK FwownHook(CallBackFkt p_callBack,
    void* p_callID, int p_id)
```

Aufrufzeitpunkt

Bei verschiedenen Ereignissen in FIX/Wt.

Beschreibung

Die Funktion wird von FIX/Wt aufgerufen, um die Benutzerbibliothek über bestimmte Ereignisse zu informieren. Die Art des Ereignisses wird in Form des Integerwertes p_id mitgegeben. Als Rückgabewert kann die Funktionen einen Text zurückgeben (Fehlermeldung), der im Meldungsfenster angezeigt wird. Ansonsten ist NULL zurückzugeben. Die möglichen Werte für p_id werden in fwownWt.h und fwownStdWt.h als Makros definiert. Für folgende Ereignisse erfolgt ein Aufruf von FwownHook():

- **HOOK_SESSION_START** - Eine neue Session wird gestartet
- **HOOK_SESSION_END** - Eine Session wurde beendet
- **HOOK_START** - FIX/Wt wurde gestartet
- **HOOK_END** - FIX/Wt wird beendet
- **HOOK_PROTOINIT** - Protokoll initialisiert.
- **HOOK_EXEC_PROGRAM** - Weiteres FIX Programm wurde mittels fxExecProgram() gestartet.
- **HOOK_EXEC_END** - Mit fxExecProgram() gestartetes Programm wurde beendet.

5 Beschreibung der FIX/Wt-Funktionen

Zu Funktionen, die in FIX/Wt liegen, muss erst ein Zeiger ermittelt werden, der dann einer Variablen zugewiesen wird. Diese Zuweisung findet typischerweise über ein Makro statt, das in fwownWt.h definiert wird. Dieses Makro prüft, ob der Zeiger bereits aus einem früheren Aufruf zur Verfügung steht und optimiert so die Anzahl der Aufrufe. Der Code zum Aufruf einer Funktion in FIX/Wt erfolgt immer nach dem gleichen Schema. In diesem Beispiel wird die fiktive Funktion mit dem Namen function verwendet:

```
/* Funktionszeiger deklarieren */
FunctionFct function = NULL;

...

/* Funktionszeiger ermitteln, wenn noch nicht bekannt */
FWFCT_FUNCTION(callback, function);

/* Funktion aufrufen */
function(p_callID, p1, p2, p3);
```

In der folgenden Beschreibung der Funktionen werden zu jeder Funktion der Typ des Funktionszeigers, der Prototyp mit den Parametern und das Macro zur Ermittlung beschrieben.

GetUsername - Benutzername ermitteln

Prototyp

```
char* GetUsername(void* p_callID)
```

Typ des Funktionszeigers, Makro

```
StringInfoFct, FWFKT_GETUSER(callback, varname)
```

Beschreibung

Die Funktion liefert den Namen des Benutzers, der sich an der FIX-Anwendung angemeldet hat.

GetPassword - Passwort ermitteln

Prototyp

```
char* GetPassword(void* p_callID)
```

Typ des Funktionszeigers, Makro

```
StringInfoFct, FWFKT_GETPASS(callback, varname)
```

Beschreibung

Die Funktion liefert das Passwort des Benutzers, der sich an der FIX-Anwendung angemeldet hat.

GetHostname - Hostname ermitteln

Prototyp

```
char* GetHostname(void* p_callID)
```

Typ des Funktionszeigers, Makro

```
StringInfoFct, FWFKT_GETHOSTNAME(callback, varname)
```

Beschreibung

Die Funktion liefert den Hostnamen des Rechners, an den sich FIX/Wt angemeldet hat. Das Format des Hostnamens entspricht dem Format, das zum Verbindungsaufbau verwendet wurde. Wird statt des Namens eine IP-Adresse zum Verbindungsaufbau verwendet, dann liefert die Funktion ebenfalls eine IP-Adresse. Beinhaltet die Angabe den Port, dann liefert die Funktion auch die durch ":" getrennte Portangabe.

GetScreenInfo - Informationen zu Bildeinstellung ermitteln

Prototyp

```
void GetScreenInfo(void* p_callID, int* style, int* size,
    int* sx, int* sy)
```

Typ des Funktionszeigers, Makro

```
GetScreenInfoFct, FWFKT_GETSCREENINFO(callback, varname)
```

Beschreibung

Mit dieser Funktion können die Parameter der aktuellen Bildschirmeinstellung ermittelt werden. Sie bekommt als Parameter Zeiger auf Variablen, in denen die Werte abgelegt werden. style ist der aktuelle Stil des Bildschirms. size ist die Nummer der Größe (0=medium, 1=large, 2=huge). In sx und sy werden die Ausmaße einer Zelle dieser Größe in Pixeln abgelegt.

GetColor - Farbinformation aus Farbzuordnungstabelle lesen

Prototyp

```
RGBVAL GetColor(void* p_callID, ColtabIdx idx, ColtabPart part,
    short variation)
```

Typ des Funktionszeigers, Makro

```
GetColorFct, FWFKT_GETCOLOR(callback, varname)
```

Beschreibung

Mit Hilfe dieser Funktion kann auf die Farbzuordnungstabelle zugegriffen werden, die aus der Datei coltab.fix gelesen wird. Der Parameter idx bestimmt das Element, dessen Farbe ermittelt werden soll. Die möglichen Werte werden in enum ColtabIdx definiert (fwownWt.h). Der Parameter part bestimmt, welcher Farbanteil (Vordergrund, Hintergrund, Linien) ermittelt werden soll. Die möglichen Werte sind in der enum ColtabPart definiert. Der Parameter variation definiert die Farbvariation. Da FIX/Wt in der aktuellen Version keine Farbvariationen unterstützt sollte für diesen Wert 0 angegeben werden. Als Ergebnis liefert die Funktion die Farbe als RGBVAL zurück. Mittels der Macros RedVal, GreenVal und BlueVal können die einzelnen Bestandteile der Farbe ermittelt werden, um in anderen Funktionen verwendet zu werden.

GetColorByName - Farbinformation über Namen lesen

Prototyp

```
bool GetColorByName(void* p_callID, _TCHAR* name, RGBVAL* col,
     short variation)
```

Typ des Funktionszeigers, Makro

```
GetColorByNameFct, FWFKT_GETCOLORBYNAME(callback, varname)
```

Beschreibung

Die Funktion liefert zu dem Namen name einen Farbwert und legt ihn in col ab. Wenn kein Farbwert gefunden wurde, liefert die Funktion false, sonst true. name muss entweder ein definierter Farbname sein (RED1, WHITE, BLACK ... gleiche Schreibweise wie in der Farbzuordnungstabelle), ein mit RGB_ definierter Farbwert oder ein über DEFINE in der Farbzuordnungstabelle definierter Name zu einer Farbvariation sein. Mittels der Macros RedVal, GreenVal und BlueVal können die einzelnen Bestandteile der Farbe ermittelt werden, um in anderen Funktionen verwendet zu werden.

GetLFRes - Ressource aus lookAndFeel.frc lesen

Prototyp

```
void* GetLFRes(void* p_callID, _TCHAR* resname)
```

Typ des Funktionszeigers, Makro

```
GetResFct, FWFCT_GETLFRES(callback, varname)
```

Beschreibung

Die Ressource wird nicht unmittelbar aus der Datei lookAndFeel.frc gelesen. Statt dessen wird die Datei beim Laden eines Look&Feels gelesen und im Speicher abgelegt. Der Aufruf von GetLFRes() greift dann auf den Wert im Speicher zu.

Als Parameter resname kann eines der Makros aus fwownWt.h angegeben werden, die mit LFRES_ beginnen. Für diese Werte ist ein Defaultwert definiert, der zurückgegeben wird, wenn die Ressource nicht definiert ist. FIX/Wt verwendet den gleichen Defaultwert,

bei nicht definierter Ressource. Je nach Art der Ressource unterscheidet sich der Rückgabewert. Er kann entweder vom Typ long sein oder vom Typ _TCHAR*. Das Ergebnis ist in den entsprechenden Typ zu casten. Die Typen der Rückgabewerte und die Defaultwerte sind in fwownWt.h zur jeder Ressource hinter dem Makro dokumentiert.

Zusätzlich ist es möglich, selbst definierte Ressourcen in lookAndFeel.frc zu hinterlegen und mittels GetLFRes() auszulesen. Als resname ist der Name der Ressource als _TCHAR* anzugeben. Bei den Ressourcewerten findet keine Umwandlung statt. Das Ergebnis ist immer vom Typ _TCHAR*. Wenn Zahlenwerte oder boolesche Werte gewünscht werden, dann ist eine Umwandlung in den entsprechenden Datentyp in der Benutzerbibliothek vorzunehmen.

Obwohl die Ressource beim Zugriff bereits im Speicher steht, kostet der Aufruf von GetLFRes() relativ viel Performance. Der Grund liegt darin, dass der Schlüssel (resname) eine Zeichenkette ist, die über binäre Suche und Stringvergleich gefunden werden muss. Bei häufig aufgerufenen Funktionen (z.B. FwownDrawPaintAreaWt()) lohnt es sich deshalb beim ersten Zugriff den Wert in eine Variable zu lesen und von da an diese zu verwenden.

GetObjInfo - Informationen zu einem Objekt ermitteln

Prototyp

```
int GetObjInfo(void* p_callID, unsigned long p_ob_id,
     short* p_ob_class,
     int* p_x, int* p_y, _TCHAR* p_ms_name)
```

Typ des Funktionszeigers, Makro

```
GetResFct, FWFCT_GETOBJINFO(callback, varname)
```

Beschreibung

Mit Hilfe dieser Funktion können Informationen zu einem Objekt ermittelt werden. Schlüssel zum Zugriff ist der Parameter p_ob_id, der entweder die ID eines Objektes, die einer Paintarea oder einer Fieldarea beinhalten muss. Im Falle von Paintarea und Fieldarea wird die ID des Objektes intern von der Funktion bestimmt. Die Funktion schreibt in die Variablen p_ob_class, p_x, p_y und p_ms_name die Klasse des Objektes, die Position und den Namen.

Als Rückgabewert liefert die Funktion im Fehlerfall -1 und ansonsten einen anderen Wert.

SendEvent- Event an FIX-Anwendung senden

Prototyp

```
void SendEvent(void* p_callID, int ev)
```

Typ des Funktionszeigers, Makro

```
SendEventFct, FWFKT_SENDEVENT(callback, varname)
```

Beschreibung

Die Funktion kann zum Senden eines Events an die FIX-Anwendung aufgerufen werden. Das Event ist als Parameter ev zu übergeben.

ResizeBuffer - Puffergröße ändern

Prototyp

```
_TCHAR* ResizeBuffer(void* p_callID, _TCHAR* buffer, long size)
```

Typ des Funktionszeigers, Makro

```
ResizeBufferFct, FWFKT_RESIZEBUFFER(callback, varname)
```

Beschreibung

Die Funktion ResizeBuffer() kann verwendet werden, um einen Speicherbereich, der von FIX/Win übergeben wurde, zu vergrößern. Die Funktion bekommt den Zeiger auf den Speicherbereich (buffer) und die gewünschte Größe (size). Sie liefert einen Zeiger auf den neuen Speicherbereich zurück. Die Funktion wird typischerweise innerhalb der Funktion FwownProcessData() aufgerufen.

SetPainterFont - Schriftart für Textausgabe festlegen

Prototyp

```
void SetPainterFont(void* callID, WFont* p_font)
```

Typ des Funktionszeigers, Makro

```
SetPainterFontFct, FWFCT_SETPAINTERFONT(callback, varname)
```

Beschreibung

Die Funktion dient dazu, den Font des aktuellen Painters (Parameter WPainter* p_painter von FwownDrawPaintAreaWt) festzulegen. Prinzipiell ginge das auch mit der entsprechenden Funktion des Wt Toolkits. Das Problem liegt jedoch darin, dass ein Zugriff auf die Session bestehen muss. Die Session wird von Wt aus threadlokalem Speicher gelesen. Da dieser jedoch in der Benutzerbibliothek nicht zur Verfügung steht, kommt es zu einer Exception.

GetWtConnector - Art des Webservers ermitteln

Prototyp

```
int GetWtConnector(void* callID, WFont* p_font)
```

Typ des Funktionszeigers, Makro

```
IntInfoFct, FWFCT_GETWTCONNECTOR(callback, varname)
```

Beschreibung

Liefert die Art, wie FIX/Wt gestartet wurde. Folgende Werte sind möglich:

- CONNECTOR_HTTP - als eigener Webserver
- CONNECTOR_ISAPI - Als ISAPI-Extension

Diese Information ist beim Zugriff auf Dateien wichtig. Im Falle von CONNECTOR_ISAPI ist das aktuelle Verzeichnis das Unterverzeichnis bin, in dem das Binary von FIX/Wt liegt. Im Falle von CONNECTOR_HTTP ist das aktuelle Verzeichnis das Installationsverzeichnis von FIX/Wt.

SetSessionPtr - Speicherbereich mit Variablen zu einer Session definieren

Prototyp

```
void SetSessionPtr(void* callID, void* ptr)
```

Typ des Funktionszeigers, Makro

```
SetSessionPtrFct, FWFCT_SETSESSIONPTR(callback, varname)
```

Beschreibung

Die Funktion merkt sich den als Parameter ptr übergebenen Zeiger als Zeiger auf einen Speicherbereich zu einer Session. Mit Hilfe der Funktion GetSessionPtr(), kann der Zeiger an anderer Stelle zu der Session ermittelt werden.

GetSessionPtr - Speicherbereich mit Variablen zu einer Session ermitteln

Prototyp

```
void* GetSessionPtr(void* callID)
```

Typ des Funktionszeigers, Makro

```
GetSessionPtrFct, FWFCT_GETSESSIONPTR(callback, varname)
```

Beschreibung

Die Funktion ließt den mit SetSessionPtr() gemerkten Zeiger zu einer Session.

GetWtSessionID - Session ID zu einer Session ermitteln

Prototyp

```
char* GetSessionPtr(void* callID)
```

Typ des Funktionszeigers, Makro

```
GetWtSessionIdFct, FWFCT_GETWTSESSIONID(callback, varname)
```

Beschreibung

Die Funktion ermittelt die Session ID, die von Wt für eine Session vergeben wurde. Dieser Wert kann als eindeutige Zeichenkette, die eine Session kennzeichnet, verwendet werden.

OpenBrowserWindow - neues Fenster im Browser öffnen

Prototyp

```
OpenBrowserWindow(void* callID, _TCHAR* url, _TCHAR* name,
_TCHAR* props);
```

Typ des Funktionszeigers, Makro

```
OpenBrowserWindowFct, FWFCT_OPENBROWSERWINDOW(callback,
    varname)
```

Beschreibung

Die Funktion erzeugt Java-Script Code, der im Browser ausgeführt wird und mit Hilfe der Funktion

```
window.open(url, name, props)
```

ein neues Fenster öffnet. Über die Parameter, die von FIX/Wt an die Funktion durchgereicht werden, können Inhalt und Aussehen des Fensters bestimmt werden. Die Parameter sind in der Dokumentation zu Java-Script zu finden (z.B. auf: http://www.w3schools.com/jsref/met_win_open.asp).

Stichwortverzeichnis