Programmierhandbuch

d.3 hook

d.velop

Impressum

Die in diesem Dokument angeführten Angaben dienen ausschließlich Informationszwecken, können ohne vorherige Ankündigung geändert werden und stellen keinerlei Verpflichtung der d.velop AG dar. Die hier angeführten Waren sowie die hier beschriebene Software unterliegen Lizenzbestimmungen; es liegt keine Eigentumsübertragung vor.

Alle in diesem Dokument enthaltenen Informationen und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt geprüft. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Dokument enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Die d.velop AG übernimmt infolgedessen keine Verantwortung und wird keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Information oder Teilen davon entsteht.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Dokument berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen.

Bitte beachten Sie, dass ab d.3 Version 7.1 der d.3 smart desktop und der d.3 explorer zum d.3 smart explorer verschmolzen sind.

Dieses Dokument wurde zuletzt am 18.03.2013 überarbeitet und bezieht sich auf d.3 hook ab Version 7.2.0.
Name des Dokuments: d3hook.pdf (Buildnummer: 20130318).

Kontakt

d.velop AG
Schildarpstraße 6-8
48712 Gescher

Fon +49 (0) 2542 9307-0
Fax +49 (0) 2542 9307-20

www.d-velop.de
info@d-velop.de

Bei Fragen zu dieser Dokumentation oder zur Software wenden Sie sich bitte an uns.
Fon +49 (0) 2542 9307-6000
support@d-velop.de

Inhalt

Inhalt

Teil I
Einleitung

Teil I Einleitung

1 Über Hook-Programmierung

Hook-Funktionen sind kundenindividuelle Funktionen, die die Funktionaliät des d.3 Serverkerns erweitern bzw. verändern. Sie ermöglichen, dass bei bestimmten d.3 Aktionen benutzerspezifizierte Aktionen ausgelöst werden. Mit Hook-Funktionen können Sie

- individuelle Verarbeitungs-Logik in d.3 einbringen
- auf externe Datenquellen zugreifen
- das Standard d.3 Verhalten ändern

Hook-Funktionen erlauben die Integration in bestehende DV-Prozesse.

Beim Import eines Dokumentes soll eine Plausibilitätsprüfung und Ergänzung der Attribute erfolgen.

Hook-Funktionen müssen in der Interpretersprache JPL (Jam Programming Language) geschrieben werden. Diese Programmiersprache ähnelt der bekannteren Sprache C, ist jedoch wesentlich einfacher zu erlernen, dafür aber auch nicht so leistungsstark.

Ab der d.3 Version 6.2.1 sind die Einsprungspunkte bei Änderungen bzw. Benutzung der Systemdokumentklassen (Klassen die mit "$" anfangen und enden, z.B. "MSK", "MOD", "MIT", "MLI", "ARF") nicht mehr aktiv.

Der Entwickler der Hook-Funktionen hat für die Internationalisierung des Textes zu sorgen. Hierfür kann das dreistellige Kürzel der Client-Sprache ausgelesen werden.

2 Voraussetzungen

Dieses Handbuch erläutert die d.3 Hook-Programmierung.

Für das Verständnis ist es hilfreich, wenn Sie über grundlegende Kenntnisse in Microsoft Windows verfügen.

Um erfolgreich Hooks für das d.3 System zu programmieren, benötigen Sie die folgenden Voraussetzungen:

- Gründliche Kenntnis des d.3 Systems. Dazu zählen insbesondere die Server-Prozesse und die d.3 Administration.
- Grundkenntnisse in mindestens einer prozeduralen Programmiersprache, z.B. Microsoft Visual Basic. Mit Programmelementen wie z.B. Variablendeklarationen, Felder, Schleifen, Verzweigungen, Interpreter, Prozedur sollten Sie vertraut sein.
- Grundkenntnisse in Datenbanken: Die folgenden Grundbegriffe sollten bekannt sein: relationale Datenbank, SQL.

3 Vereinbarungen

Die in diesem Buch eingesetzten Schriften und Symbole haben folgende Bedeutung:

Schrift / Symbol	Bedeutung
Extras | Optionen	Menüpunkte oder Button Untermenüpunkte werden jeweils mit einem vertikalen Balken **|** voneinander getrennt.
Extras | Optionen | [Sonstiges]	Der letzte Untermenüpunkt in eckigen **[Klammern]** steht für eine auszuwählende Registerkarte.
ALT-STRG-X	Taste oder Tastenkombination Bei einer Tastenkombination werden die Tasten jeweils mit einem Bindestrich voneinander getrennt. Die zuerst genannten Tasten müssen gedrückt bleiben, während die letzte betätigt wird.
C:\TEMP	Pfad zu einem Verzeichnis oder Dateinamen werden in dieser `Schriftart` dargestellt.
d.3 hook	Programmnamen erscheinen in dieser Schriftart.

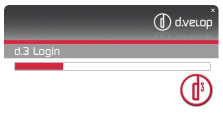
Hinweis:
Bemerkungen mit diesem Zeichen sind besonders zu berücksichtigen, stellen hilfreiche Tipps oder Ausnahmen dar.

Achtung!
Bemerkungen mit diesem Zeichen helfen, Fehler zu vermeiden, die im schlimmsten Fall zu fehlerhaftem Programmverhalten oder dem Verlust von Daten führen können.

Beispiel:
So markierte Abschnitte enthalten Beispiel-Daten zu den erläuterten Funktionen.

Hintergrund-Informationen:
Bemerkungen mit diesem Zeichen enthalten Informationen, die einzelne Punkte näher erläutern und liefern weiter führende Details zum aktuellen Thema.

Abbildung:



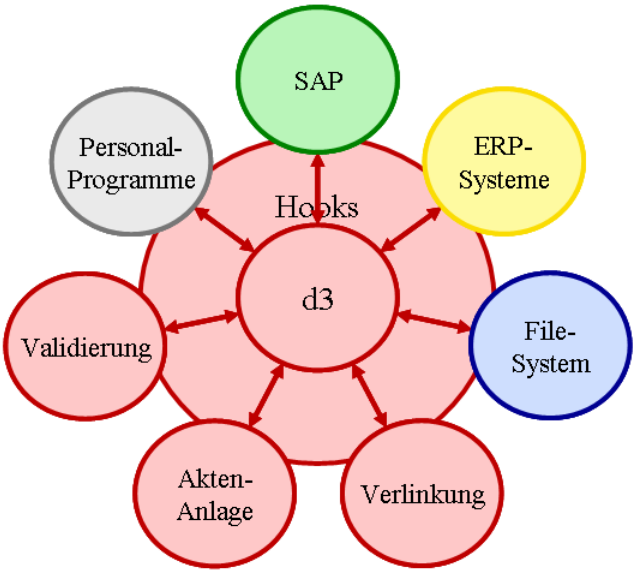
Ausgewählte Abbildungen sind mit sogenannten Hotspots versehen. D. h. durch Anklicken bestimmter Bildbereiche kann direkt zu entsprechenden Hilfethemen gesprungen werden. Einen Hotspot erkennen Sie daran, dass sich der Mauszeiger verändert, wenn Sie über das Bild fahren.

Teil II
Hook-Funktionen im Einzelnen

Teil II Hook-Funktionen im Einzelnen

1 Allgemeine Informationen zu Hook-Funktionen

1.1 Einsatz von Hook-Funktionen



Hookfunktionen greifen bei unterschiedlichen Aktionen des d.3 Servers

Folgende Aktionen können durch eine Hook-Funktion beeinflusst werden:

- Suche von Dokumenten
- Neuanlage von Dokumenten
- Anlage einer neuen Version eines Dokumentes
- Kenndatenaktualisierung eines Dokumentes
- Verknüpfung von Dokumenten
- Dokument in Wiedervorlage stellen (Postkorbfunktionalität)
- Dokument löschen
- Dokument prüfen
- Dokument freigeben
- Validieren der Attribute vor der Suche bzw. Anlage eines Dokumentes
- Statustransfer

Folgende Arten von Hook-Funktionen lassen sich unterscheiden:

- d.3 Hook-Funktionen zur Steuerung der Verarbeitungslogik
- Repository Hook-Funktionen für Wertemengen und Plausibilitätsprüfungen
- Aktenplan-Hook-Funktionen zur Programmierung des Aktenplan-Moduls (d.3

folderscheme)

- Dokumentklassen-Hook-Funktionen zur Steuerung der Rechtevergabe

1.2 Beispiele aus der Praxis

Beim Import eines Dokumentes im Dialog (d.3 import) wird als Attribut nur "Maschinennummer" angegeben. Alle weiteren Daten der referenzierten Maschine sind aus einer anderen Datenbank über die Maschinennummer als Referenz der Maschine zu ermitteln.

- Hook-Prozedur `hook_insert_entry_10`
- Verbindung zur externen Datenbank per ODBC aufmachen (`DECLARE CONNECTION ...`)
- Mit "Maschinennummer" als Referenz an die externe Datenbank gehen und Daten holen
- Daten aus externer Datenbank in die entsprechenden JPL-Felder stellen
- Verbindung zur externen Datenbank schließen (`CLOSE CONNECTION`)

Beim Import eines Dokumentes soll die Dokumentnummer nach einem festen Schema vergeben werden:

Aufbau Dokumentnummer: B-JJJJ-MM-NNNN, mit

B	Bereich (vom Anwender vorgegeben)
JJJJ	Jahreszahl (0000-9999)
MM	Monat (0-12)
NNNN	fortlaufende Nummer innerhalb eines Monats

1.2.1 Weitere Beispiele

Vor der Suche wird die Kontonummer immer auf 10 Stellen mit führenden Nullen aufgeführt.

Vor dem Import einer Rechnung wird geprüft, ob die zugehörige Bestellung im Archiv vorhanden ist.

Vor dem Import eines Dokuments wird zur Kunden-Nummer der Kunden-Name aus der Kunden-Datenbank geholt.

Nach dem Import einer Rechnung wird ein zuständiger Sachbearbeiter per Wiedervorlage informiert.

1.3 Entwicklungsumgebung

Hook-Funktionen werden in der Panther-eigenen Skript-Sprache JPL geschrieben.

- JAM/Panther wird von der Firma Prolifics bereitgestellt (http://www.prolifics.com)
- 4 GL-Entwicklungsumgebung mit Focus auf Datenbanken
- Multi-Plattform Umgebung
- Zwei- und Drei-Schicht-Architektur
- Eigene Skript-Sprache JPL

1.4 Hinweise zur Hook-Architektur

1.4.1 Programmkopf

Der Programmkopf einer Hook-Funktion könnte folgendermaßen aussehen

```
//-------------------------------------------------------
--------//
// NAME
// xxx.jpl
//
// BESCHREIBUNG
// z.B. Hookfunktion zum Zugriff auf das Angebotsdatum
// über die Projektnummer bei Angeboten
//
// AENDERUNGEN
// erstellt: xx.xx.xxxx von xxx
//
// geaendert: xx.xx.xxxx von xxx
//-------------------------------------------------------
--------//
```

Eine gute Kommentierung erleichtert die spätere Fehlersuche und Anpassung der Hook-Funktion!

1.4.2 Beschränkungen

Der String, der an eine Hook-Funktion übergeben wird, darf maximal 255 groß sein.

Die Gesamtzeichenzahl von `doku_art` bis `wert 11` muss kleiner als 255 Zeichen sein, sonst gibt es eine Fehlermeldung `proc Stammdaten_update (doku_art,wert1,wert2,wert3,wert4,wert5,wert6,wert7,wert8 ,wert9,wert10,wert11)`.

1.4.3 Verwendung der d.3 Server API

Langfristig sollen Hook-/Workflow-Programmierer so unterstützt werden, dass kein direkter Zugriff auf die d.3 Datenbank mehr nötig ist.

Dazu stellt der d.3 Server ab Version 5.5.1 Funktionen bereit, um

- dokumentbezogene Informationen (Attribute, Verknüpfungen ...)
- dokumentartbezogene Informationen (Repository ...)
- benutzerbezogene Informationen (Gruppenzugehörigkeit ...)

abfragen bzw. ändern zu können.

Außerdem stehen Helferfunktionen zur Verfügung, die die Konvertierung von Strings, Datumsangaben, numerischen Angaben etc. in das von d.3 bzw. der Datenbank benötigte Format ausführen, sowie Dateizugriffe vereinfachen.

Damit kann eine Entkopplung der einzelnen Hooks von der d.3 Datenstruktur erfolgen, der Programmierer bekommt eine Garantie für Langzeitstabilität, Support etc. und die d.3 Daten werden vor Fehlern besser geschützt und Zugriffe können protokolliert und damit besser gedebugged werden.

Die Funktionen sollen in Bezug auf Namen und Aufrufparameter ähnlich sein und somit eine einheitliche Schnittstelle zwischen Hook und d.3 bilden.

Die Funktionen sollen alle über eine zentrale Codestelle geführt werden, damit hier Standard-Loggings, Kontrollen etc. ausgeführt werden können.

Bei der Entwicklung von Hook-Funktionen sollten Sie auf die Funktionen zurückgreifen, denen die d.3 Server API zur Verfügung steht (diese wird in einem eigenständigen Handbuch beschrieben (`d3server_api.pdf`)). Beispiele zur Verwendung von Hook-Funktionen im Zusammenhang mit der d.3 Server API finden Sie in diesem Handbuch.

1.4.4 Ablage von Hook-Funktionen

Hook-Funktionen sollten unterhalb des d.3 Server-Konfigurationsverzeichnisses, z. B. `D:\d3\d3server.prg\ hook\d3p\xxx.jpl` abgelegt werden.

Der Pfad in der d.3 Konfiguration sollte relativ angegeben werden, z.B. `\hook\ d3p \xxx.jpl`.

1.4.5 Namenskonventionen für Hook-Funktionen

Die Namensgebung einer selbstgeschriebenen Hook-Funktion muss folgendes Schema einhalten:

`proc AaaaBbbb_CC (<Parameterliste>)`

Aaaa	steht für die Aktion, in die die Hook-Funktion eingreift
Bbbb	steht für `entry` bzw. `exit`, je nach dem, ob die Funktion vor der eigentlichen Aktion (`entry`) oder nach der Aktion (`exit`) ausgeführt werden soll
CC	steht für die Einsprungmarke (im allgemeinen 10,20 oder 30) Je höher die Zahl, desto weiter hinten liegt die Einsprungmarke, d.h., die Aktion, in die der Hook eingreift, ist bereits weiter fortgeschritten.

Hier einige Beispiele für Namen selbsterstellter Hook-Funktionen:

`SearchEntry_10 (…)`

Hook der Firma d.velop AG.

Der Hook wird bei der Suche (`search`) und zwar vor dem eigentlichen Suchvorgang (`entry`) durchgeführt.

Er wird an der ersten Stelle (10), an der man Einfluß nehmen kann, aktiviert.

`InsertExit_20_siemens (…)`

Hook der Firma Siemens.

Der Hook wird bei Anlage eines neuen Dokumentes (`insert`) und zwar nach der Anlage (`exit`) durchgeführt.

Beachten Sie die Rückgabewerte bei Hook-Funktionen. Client-Fehlermeldungen sollten in die `msglib.usr` aufgenommen werden (siehe Fehlersuche).

Er wird an der zweiten Stelle (20), an der man Einfluß nehmen kann, aktiviert.

Gespeicherte Hook-Funktionen-Programme müssen immer mit der Endung `.jpl` versehen werden!

1.5 Installieren und Aktivieren der Hook-Funktionen

1.5.1 d.3 Hook

- Rufen Sie die d.3 Config (`d3config.ini`) in der d.3 Administration (unter d.3 Komponenten) auf.
- Wechseln Sie in den Abschnitt **Hook-Funktionen**.
- Tragen Sie Pfad und Name der Hook-Funktion (JPL-Datei!) ein.

- Geben Sie enthaltene Hook-Funktionen an.
- Starten Sie die d.3 Prozesse neu.

1.5.2 Repository-Hook

- Tragen Sie das Hook-Modul in der d.3 Config (`d3config.ini`) ein.
- Bearbeiten Sie das **Repository-Feld** in der d.3 Administration.
- Geben Sie als Wertevorräte bzw. Plausibilität den Funktionsname an.
- Starten Sie die d.3 Prozesse neu.

1.5.3 Dokumentklassen-Hook

- Öffnen Sie die d.3 administration (d.3 admin) mit Hilfe der d.3 config-Komponente (dialog for the `d3config.ini`).
- Bearbeiten Sie die Dokumentklassen in der d.3 Administration.
- Geben Sie per `@D3HOOK (<Hook-Funktions-Name>)` für das Attributfeld an.
- Starten Sie die d.3 Prozesse neu.

1.5.4 Aktenplan-Hook (d.3 folder scheme)

- Öffnen Sie den d.3 Aktenplan (`aktplan.ini`) in der d.3 Administration unter d.3 Komponenten.
- Bearbeiten Sie die Dokument-/ Aktenart.
- Tragen Sie den Hook-Funktions-Name ein.
- Starten Sie die d.3 Prozesse neu.

1.6 Fehlersuche

Standardfehler zu d.3 finden Sie in der Datei `msglib.dat`, die per default im d.3 Client Verzeichnis liegt und mitverteilt wird.

Eigene Fehlermeldungen zu den programmierten Hook-Funktionen sollten Sie in der Datei `msglib.usr` ablegen.

Erzeugen Sie diese Datei ebenfalls im d.3 Client Verzeichnis – als einfache Textdatei - und verteilen Sie sie mit.

Die Datei `msglib.dat` eignet sich nicht für eigene Fehlermeldungen, da sie bei jedem Programmupdate überschrieben wird!

Durch die Verwendung der `msglib.usr` können Sie aussagekräftige Fehlermeldungen Ihren Benutzern liefern, statt Meldungen wie „Fehler in kundenspez. Hook-Funktion …". Die Datei sollte sich im Aufbau an der Datei `msglib.dat` orientieren.

Die erste Spalte der `msglib.dat` enthält den Fehlercode, den der d.3 Server zurückliefert.

Fehlercode 28005

```
0028005,049,1,"Ist dies ein Fall, der unter FAQ
veröffentlicht werden soll?", "Bitte Eingabe für das Feld
Homepage prüfen.","[0028005] Hook"
0028005,001,1,"Is this a FAQ problem?", "Please check
your input for field Homepage.","[0028005] Hook"
```

Dieser Fehlercode ist nicht direkt der Returnwert aus der Hook-Funktion, sondern der Server rechnet einen Offset dazu. Das passiert, damit der Hook-Returnwert nicht mit dem Standard d.3 Fehlercode kollidiert.

Da der Offset nicht immer gleich ist, muss man beim Testen der Hook-Funktion schauen, welcher Fehlercode beim Client ankommt. Dieser wird in der Standard-Meldung „Fehler in kundenspez. Hook-Funktion …" angezeigt!

Zu dem hier angezeigten Fehlercode kann man nun einen Meldungstext in die `msglib.usr` schreiben. Dadurch wird fortan im Fehlerfall diese eigene Meldung von allen d.3 Clientprogrammen angezeigt.

In der Regel wird als Offset `9500` verwendet und der Rückgabewert des Hooks davon abgezogen:

Beispiel:
- Hook liefert `-257`
- Fehlernummer wäre dann `9757`

Ausnahmen sind:
`ImportDocument` mit Offset `10000`
`ImportNewVersionDocument` mit Offset `20000`
`DeleteDocument` mit Offset `4000`

Als Rückgabewerte der Hooks der Funktionen `ImportDocument` und `ImportNewVersionDocument` werden Werte zwischen "-8000" und "-9999" empfohlen, damit sich für die Funktionen Rückgabewerte zwischen "18000" und "19999" (bzw. "28000" und "29999") ergeben. Überschneidungen bei Fehlernummern mit den internen Fehlern aus diesen Funktionen sind damit ausgeschlossen.

Als Rückgabewerte für die Delete-Hooks werden Werte zwischen "-1900" und "-1999" empfohlen, auch hier ist dann eine Überschneidung ausgeschlossen.

Für alle anderen Hooks (die mit Standard-Offset) werden Rückgabewerte zwischen "-1" und "-499" empfohlen, weil der Bereich zwischen "9500" und "9999" dafür reserviert ist.

2 Allgemeine Informationen

d.3 Variablen

Folgende d.3 Variablen können Sie in Hook-Funktionen verwenden:

Variable	Beschreibung
zeich_nr	Dokument-/ Zeichnungsnummer
dokuart (Kürzel)	Dokumentart-Kürzel
var_nr	Variantennummer
doku_id	Dokument-ID
logi_verzeichnis	Dokumentstatus (= logisches Verzeichnis) zwei signifikante Zeichen: • "Be": Bearbeitung • "Pr": Prüfung • "Fr": Freigabe • "Ar": Archiv
bearbeiter	Bearbeiter des Dokuments
text	Kommentartext
datum1	Datum 1
datum2	Datum 2
dok_dat_feld[1]	Attributfeld 1
dok_dat_feld[2]	Attributfeld 2
...	
dok_dat_feld[59]	Attributfeld 59

Variable	Beschreibung
dok_dat_feld [70]	Attributfeld 70
...	
dok_dat_feld [89]	Attributfeld 89
dok_dat_feld_60[1]	Attributfeld 60, Zeile 1
dok_dat_feld_61[1]	Attributfeld 61, Zeile 1
.	
dok_dat_feld_69[1]	Attributfeld 69, Zeile 1
color_code	Farbcode des Dokuments (0 = nicht gesetzt; 1-24 = Farbcode)

3 Abhängige Dateien

3.1 hook_dep_doc_entry_10 (doc_id, status, index, user_o_group, abh_doc_ext, transfer)

Aufrufzeitpunkt:

Vor Eintrag der abhängigen Datei in die Datenbank

doc_id	Dokument-ID des Dokumentes der abhängigen Datei
status	aktueller Status des Dokumentes
index	Version des Dokumentes
user_o_group	Bearbeiter oder Prüfer-Gruppe des Dokumentes bei Status Bearbeitung bzw. Prüfung
abh_doc_ext	Dateierweiterung der abhängigen Datei
transfer	1: Aufruf während eines Statustransfers

3.2 hook_dep_doc_exit_10 (doc_id, status, index, user_o_group, abh_dokument[i], transfer)

Aufrufzeitpunkt:

Nach Eintrag der abhängigen Datei in die Datenbank.

doc_id	Dokument-ID des Dokumentes der abhängigen Datei
status	aktueller Status des Dokumentes
index	Version des Dokumentes
user_o_group	Bearbeiter oder Prüfer-Gruppe des Dokumentes bei Status Bearbeitung bzw. Prüfung
abh_doc_ext	Dateierweiterung der abhängigen Datei

4 Abwesenheitszeiten bei Benutzern

4.1 hook_get_user_absence_time (user_name, user_type, t1, t2)

Ab d.3 Version 6.2

Zur Berücksichtigung benutzer- und standortindividuellen Abwesenheitszeiten wie Urlaub oder Feiertag.

Mit dieser Hook-Funktion kann der Workflow-Entwickler für einen d.3 Benutzer die Abwesenheitszeit in Stunden innerhalb des übergebenen Zeitraums zurückgeben. Diese Zeit wird auf die Eskalationszeit aufgerechnet.

Aufrufzeitpunkt:

Die Hook-Funktion wird aufgerufen, direkt bevor eine Workflow-Wiedervorlage an den Empfänger eines Workflow-Schritt mit benutzerinteraktion gesendet wird, weil dann die Eskalationszeit für diesen Schritt eingetragen wird.

Parameter	Beschreibung
user_name	Benutzer, Gruppe oder Tätigkeitsprofil
user_type	"u" = Benutzer; "g2 = Gruppe; 2p2 = Tätigkeitsprofil
t1	Zeitstempel von ... (Format: "dd.mm.yyyy hh:mi:ss")
t2	Zeitstempel bis ... (Format: "dd.mm.yyyy hh:mi:ss")

Rückgabe:

Abwesenheitszeit in Stunden des Benutzers `user_name` innerhalb des angebenen Zeitraums `t1` bis `t2`.

5 Aktualisieren der Kenndaten (UpdateAttributes)

Ab d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter `doc_type_short` für die Funktionen
`hook_upd_attrib_entry_10`
`hook_upd_attrib_entry_20`
`hook_upd_attrib_exit_10`
`hook_upd_attrib_exit_20`
`hook_upd_attrib_exit_30`

Bei einem evtl. Dokumentartwechsel steht das vorherige Dokumentart-Kürzel im Parameter `doc_type_short`, das neue Dokumentartkürzel steht in der globalen Variablen `dokuart_kurz`.

5.1 hook_upd_attrib_entry_10 (doku_id_ref, user_ref, doc_type_short, doc_type_short_new)

reserviert

5.2 hook_upd_attrib_entry_20 (doku_id_ref, user_ref, doc_type_short, doc_type_short_new)

Verfügbare Felder:

Alle beim API-Call übergebenen Felder. Änderbar sind jedoch nur die `dok_dat_`-Felder und das Feld `text`.

Aufrufzeitpunkt

Es wurden lediglich die neuen Attribute empfangen, jedoch noch nicht auf Plausibilität geprüft.

Eingabeparameter

Parameter	Beschreibung
`doc_id_ref`	Doku-ID des Dokumentes, dessen Attribute aktualisiert werden sollen
`user_ref`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart vor der Attributaktualisierung
`doc_type_short_new`	Kürzel der Dokumentart nach der Attributaktualisierung

Die Werte `doc_type_short` und `doc_type_short_new` unterscheiden sich nur, wenn tatsächlich ein Dokumentartwechsel durchgeführt wurde.

5.3 hook_upd_attrib_entry_30 (doku_id_ref, user_ref, doc_type_short, doc_type_short_new)

reserviert

5.4 hook_upd_attrib_exit_10 (doku_id_ref, fehler_update, user_ref, doc_type_short, doc_type_short_old)

Diese Hook-Funktion wird nur aktiviert, wenn zuvor kein Fehler aufgetreten ist. Liefert diese Funktion einen Wert ungleich "0", kann somit die Aktualisierung noch mal rückgängig gemacht werden.

Aufrufzeitpunkt:

Direkt vor Beendigung der DB-Transaktion.

Eingabeparameter

Parameter	Beschreibung
doku_id_ref	Doku-ID des Dokumentes, dessen Attribute aktualisiert werden sollen
error_number	0: Aktualisierung korrekt durchgelaufen <> 0: Fehlernummer; i. a. vom DB-Server geliefert
user_ref	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart nach der Attributaktualisierung
doc_type_short_old	Kürzel der Dokumentart vor der Attributaktualisierung

Die Werte doc_type_short und doc_type_short_old unterscheiden sich nur, wenn tatsächlich ein Dokumentartwechsel durchgeführt wurde.

5.5 hook_upd_attrib_exit_20 (doc_id, error_number, user, doc_type_short, doc_type_short_old)

Diese Hook-Funktion wird immer aktiviert, auch wenn zuvor kein Fehler aufgetreten ist.

Aufrufzeitpunkt:

Direkt nach Beendigung der DB-Transaktion.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, dessen Attribute aktualisiert werden sollen
error_number	0: Aktualisierung korrekt durchgelaufen <> 0: Fehlernummer; i. a. vom DB-Server geliefert
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart nach der Attributaktualisierung
doc_type_short_old	Kürzel der Dokumentart vor der Attributaktualisierung

Die Werte doc_type_short und doc_type_short_old unterscheiden sich nur, wenn tatsächlich ein Dokumentartwechsel durchgeführt wurde.

5.6 hook_upd_attrib_exit_30 (doc_id, error_number, user, doc_type_short, doc_type_short_old)

reserviert

6 Dokumente freigeben (Release Document)

6.1 hook_release_entry_10 (doc_id_ref, user_ref, doc_type_short, unblock)

Falls diese Hook-Funktion einen Wert ungleich "0" liefert, wird die Freigabe abgebrochen.

Aufrufzeitpunkt:

Direkt vor Start der Datenbank-Transaktion. Es wurde ermittelt, dass der Benutzer das Recht hat, das Dokument freizugeben.

Eingabeparameter

Parameter	Beschreibung
doc_id_ref	Doku-ID des freizugebenden Dokuments
user_ref	Name des rufenden Benutzers
error	gleich "0", wenn Freigabe erfolgreich, sonst Fehlercode
doc_type_short	Dokumentartkürzel
unblock	gleich "1", wenn das Dokument entsperrt wird ungleich "1" bei normalen Freigaben

6.2 hook_release_exit_10 (doc_id_ref, user_ref, error, doc_type_short, unblock)

Aufrufzeitpunkt:

Nach Durchführung der Freigabe, nach Beendigung der DB-Transaktion.

Eingabeparameter

Parameter	Beschreibung
doc_id_ref	Doku-ID des freizugebenden Dokuments
user	Name des rufenden Benutzers
error	0: Freigabe erfolgreich sonst: Fehlercode
doc_type_short	Dokumentartkürzel
unblock	gleich "1", wenn das Dokument entsperrt wird ungleich "1" bei normalen Freigaben

7 Dokument prüfen (VerifyDocument)

7.1 hook_verify_entry_10 (doc_id, alteration_number, user)

Falls diese Hook-Funktion einen Wert ungleich 0 liefert, wird die Prüfung abgebrochen.

Aufrufzeitpunkt:

Direkt vor Start der Datenbank-Transaktion. Es wurde ermittelt, dass der Benutzer das Recht hat, das Dokument zu prüfen.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des zu prüfenden Dokuments
alteration_number	Änderungsnummer der zu prüfenden Dokumentversion
user	Name des rufenden Benutzers

7.2 hook_verify_exit_10 (doc_id, alteration_number, user, error)

Aufrufzeitpunkt:

Nach Durchführung der Prüfung. Nach Beendigung der Datenbanktransaktion.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des zu prüfenden Dokuments
alteration_number	Änderungsnummer der zu prüfenden Dokumentversion
user	Name des rufenden Benutzers
error	0: Prüfung erfolgreich sonst: Datenbank-Fehlernummer beim Eintrag der Prüfung in die Datenbank

8 Dokumentsuche (SearchDocument)

Ab d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter doc_type_short für die folgenden Einsprungpunkte

hook_search_entry_10
hook_search_entry_20
hook_search_entry_30
hook_search_exit_10
hook_search_exit_20
hook_search_exit_30

8.1 hook_search_entry_05 (user, doc_type_short)

Ab d.3 Version 7.0.1

Aufrufzeitpunkt:

Vor dem Aufruf der d.search Suche. Damit wird ermöglicht, mittels Setzen von `suchtext_ausdruck` die Suchbegriffe an d.search anzupassen.

Verfügbare Felder

`suchtext_ausdruck`: Volltext-Suchbegriffe für d.search, ansonsten analog zu hook_search_entry_10

Eingabeparameter: analog zu hook_search_entry_10

8.2 hook_search_entry_10 (user, doc_type_short)

Aufrufzeitpunkt:

Vor der Suche nach Dokumenten: Die übergebenen Suchkriterien sind noch nicht auf Plausibilität geprüft worden. Eine ggf. aktivierte Konvertierung der Suchkriterien nach Klein- bzw. Großschrift (d.3 Konfigurationsparameter KONVERTIERE_SEARCH_CASE) ist noch nicht durchgelaufen.

Bei der Datenvalidierung für eine anschliessende Suche (API `ValidateAttributes` mit Parameter `"function" = "Search"`).
Die Suchbegriffe wurden in die entsprechenden Attribut-Felder übernommen. Diese Attributwerte können im Hook geändert werden.

Verfügbare Felder

Variable	Beschreibung
`zeich_nr`	Dokument-/Zeichnungsnummer
`dokuart (Kürzel)`	Dokumentart-Kürzel
`var_nr`	Variantennummer
`doku_id`	Dokument-ID
`logi_verzeichnis`	Dokumentstatus (= logisches Verzeichnis) zwei signifikante Zeichen: • Be • Pr • Fr • Ar
`bearbeiter`	Bearbeiter des Dokuments
`text`	Kommentartext

Variable	Beschreibung
datum1	Datum 1
datum2	Datum 2
dok_dat_feld[1]	Attributfeld 1
dok_dat_feld[2]	Attributfeld 2
...	
....	
dok_dat_feld[59]	Attributfeld 59
dok_dat_feld[70]	Attributfeld 70
...	
...	
dok_dat_feld[89]	Attributfeld 89
dok_dat_feld_60[1]	Attributfeld 60, Zeile 1
dok_dat_feld_61[1]	Attributfeld 61, Zeile 1
...	
dok_dat_feld_69[1]	Attributfeld 69, Zeile 1
suchtext_ausdruck	Suchstring
color_code	Farbcode des Dokuments (0 = nicht gesetzt; 1-24 = Farbcode)

Eingabeparameter

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Dokumentartkürzel

8.3 hook_search_entry_20 (user, doc_type_short)

Aufrufzeitpunkt:

Vor der Suche nach Dokumenten: Der SELECT-Befehl für die Suche nach den Dokumenten ist entsprechend den Suchkriterien schon zusammengesetzt worden.

Eingabeparameter

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Dokumentartkürzel

8.4 hook_search_entry_30 (user, doc_type_short)

reserviert

8.5 hook_search_exit_10 (user, doc_type_short)

reserviert

8.6 hook_search_exit_20 (doku_id_ref, doc_type_short)

reserviert

8.7 hook_search_exit_30 (user, error, no_results, no_refused, doc_type_short))

Bis d.3 Version 5.5.1.1 reserviert!

danach:

Aufrufzeitpunkt:

Ganz am Ende, direkt bevor die Ergebnisse an den Client geliefert werden

Eingabeparameter

Parameter	Beschreibung
user	Benutzer, der die Suche ausführt
error	Fehler, falls aufgetreten, sont "0"
no_results	Anzahl Treffer
no_refused	Anzahl verweigerter Treffer
doc_type_short	Kürzel der Dokumentart

Rückgabewert:

wird ignoriert

Die globale Variable `no_results_refused` wird vor dem Aufruf auf den entsprechenden Wert gesetzt (= `no_refused`). Diese globale Variable wird nach dem Hook wieder eingelesen, so dass sie hier verändert werden kann. Allerdings werden Werte != 0 ignoriert, d. h. der Hook hat nur die Möglichkeit die Anzahl der verweigerten Treffer zu verbergen (=0 setzen).

9 Dokumentanlage (ImportDocument)

9.1 hook_insert_entry_05

Aufrufzeitpunkt:

Wird nur beim Hostimport aufgerufen!

Direkt nach dem Einlesen der `default.ini` und der JPL-Datei.

Die BFC (best fitting Codepage)-Konvertierung wurde an dieser Stelle noch nicht durchgeführt. Auch die Werte der übersetzbare Wertemengen wurden noch nicht konvertiert.

Hier wäre noch eine Änderung der übergebenen Kenndaten möglich.

Verfügbare Felder:

analog zu hook_insert_entry_10

zusätzlich:

```
logi_verzeichnis
bearbeiter
dokuart
as400_erlaube_ueberspielen
```

Eingabeparameter

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

9.2 hook_insert_entry_10 (user, doc_type_short)

Aufrufzeitpunkt:

Vor dem Import. Es wurde lediglich getestet, ob die Verbindung zur DB noch in Ordnung ist. Hier wäre noch eine Änderung der übergebenen Kenndaten möglich.

Bei der Datenvalidierung für einen anschliessenden Dokumentimport (API `ValidateAttributes` mit Parameter `"function" = "Insert"`)
Die Attributwerte wurden in die entsprechenden Attributfelder übernommen. Diese Attributwerte können im Hook geändert werden.

Ab d.3 Version 6.0.1 wird vor einem `ImportNewVersionDocument` nicht mehr der `hook_insert_entry_10` aufgerufen wird, sondern der `hook_new_version_entry_10`!

Verfügbare Felder

```
logi_verzeichnis
bearbeiter
zeich_nr
var_nr
text
dateiname
color_code

dok_dat_feld[1]
dok_dat_feld[2]
   .
   .
dok_dat_feld[59]
dok_dat_feld[70]
   .
   .
dok_dat_feld[89]
dok_dat_feld_60[1...100]
dok_dat_feld_61[1...100]
   .
   .
dok_dat_feld_69[1...100]
```

Eingabeparameter

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

9.3 hook_insert_entry_20 (doc_id, doc_type_short, user)

Aufrufzeitpunkt:

Vor dem Import. Die Nutzdatei wurde bereits in das Zielverzeichnis übertragen. Der SQL-Befehl für das Anlegen des Dokumentes wurde zusammengestellt. Die übergebenen Kenndaten dürfen/können hier nicht mehr geändert werden. Die Kenndaten sind noch nicht auf Gültigkeit (Wertebereich, reg. Expression, Min.-Max.-Bereich, ...) geprüft worden.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID, die dem Dokument bei erfolgreichem Import gegeben wird
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments
user	Name des rufenden Benutzers

9.4 hook_insert_entry_30

reserviert

9.5 hook_insert_exit_10 (doc_id, file_destination, import_ok, user, doc_type_short)

Aufrufzeitpunkt:

Nach dem Import. Die Datenbanktransaktion wurde noch nicht comittet. Somit kann man hier noch ein letztes Rollback erzwingen und damit den Import rückgängig machen

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokuments von dem ein TIFF- oder PDF-Abbild erstellt werden soll
file_destination	Pfad und Name der Zieldatei (Angabe, wohin die Zieldatei geschrieben wurde)
import_ok	1: bisher kein Fehler ausgetreten 0: Es trat ein Fehler beim Importieren des Dokuments auf
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

Wenn ein Import in die Freigabe erfolgt, steht der Freigabeschritt vom Server aus. Wenn man im Hook nun voraussetzt, dass sich das Dokument bereits in Freigabe befindet, muss die Hook-Funktion hook_release_exit_10 verwendet werden.

9.6 hook_insert_exit_20 (doc_id, File_destination, import_ok, user, doc_type_short)

Wie hook_insert_exit_10, jedoch wurde die DB-Transaktion geschlossen (COMMIT oder ROLLBACK). Somit kann ein erfolgreicher Import nicht mehr rückgängig gemacht werden.

Wenn ein Import in die Freigabe erfolgt, steht der Freigabeschritt vom Server aus. Wenn man im Hook nun voraussetzt, dass sich das Dokument bereits in Freigabe befindet, muss die Hook-Funktion hook_release_exit_10 verwendet werden.

9.7 hook_insert_exit_30 (doc_id, file_destination, import_ok, user, doc_type_short)

Wie hook_insert_exit_20, die Funktion wird direkt danach aufgerufen, also nach der DB-Transaktion.

Die Funktion kann (zur Zeit) genauso verwendet werden wie hook_insert_exit_20.

Wenn ein Import in die Freigabe erfolgt, steht der Freigabeschritt vom Server aus. Wenn man im Hook nun voraussetzt, dass sich das Dokument bereits in Freigabe befindet, muss die Hook-Funktion hook_release_exit_10 verwendet werden.

10 Einspielen einer neuen Version (ImportNewVersionDocument)

Ab d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter `doc_type_short` für die Funktionen
`hook_new_version_entry_10`
`hook_new_version_entry_20`
`hook_new_version_entry_30`
`hook_new_version_exit_10`
`hook_new_version_exit_20`
`hook_new_version_exit_30`

Ab d.3 Version 6.0.1 wird vor einem `ImportNewVersionDocument` nicht mehr der hook_insert_entry_10 aufgerufen wird, sondern der hook_new_version_entry_10!

10.1 hook_new_version_entry_10 (doc_id, file_source, file_destination, user, doc_type_short)

Die Hook-Funktion `hook_new_version_entry_10` wird beim Einspielen einer neuen Dokumentversion auch vom HOSTIMP aufgerufen.

Wird beim `ValidateAttributes` `nextcall=ImportNewVersionDocument` übergeben, wird die Funktion auch aufgerufen. In diesem Fall wird die Funktion ohne `doku_id`, `quell_datei` und `ziel_datei` aufgerufen, die ersten drei Parameter sind also Leerstrings.

Verfügbare Felder:

Alle beim API-Call übergebenen Felder. Änderbar sind jedoch nur die `dok_dat_`-Felder und das Feld `text`.

Aufrufzeitpunkt:

Es wurde geprüft, ob das Dokument bereits in d.3 existiert. Existenz der neuen Quelldatei wurde noch nicht geprüft.

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Doku-ID des Dokumentes, zu dem eine neue Nutzdatei eingespielt werden soll
`file_source`	Pfad und Name der Quelldatei
`file_destination`	Pfad und Name der Zieldatei (Angabe, wohin die Nutzdatei geschrieben werden soll)
`user`	Name des rufenden Benutzers

Parameter	Beschreibung
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

10.2 hook_new_version_entry_20 (doc_id, file_source, file_destination, user, doc_type_short)

Kann zur Zeit nicht über HOSTIMP verwendet werden (nur bei API-Funktion `ImportNewVersionDocument`).

Verfügbare Felder:

siehe hook_new_version_entry_10

Aufrufzeitpunkt:

Es wurde erfolgreich geprüft, ob die neue Nutzdatei existiert. Die Quelldatei wurde noch nicht eingespielt. Die Datenbanktransaktion wurde noch nicht gestartet.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, zu dem eine neue Nutzdatei eingespielt werden soll
file_source	Pfad und Name der Quelldatei
file_destination	Pfad und Name der Zieldatei (Angabe, wohin die Nutzdatei geschrieben werden soll)
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

10.3 hook_new_version_entry_30 (doc_id, file_source, file_destination, user, doc_type_short))

Aufrufzeitpunkt:

Wird sofort nach hook_new_version_entry_20 ausgeführt.

Alle Angaben analog zu hook_new_version_entry_20.

10.4 hook_new_version_exit_10 (doc_id, error_update_attributes, user, doc_type_short)

Kann zur Zeit nicht über HOSTIMP verwendet werden (nur bei API-Funktion `ImportNewVersionDocument`).

Verfügbare Felder:

siehe hook_new_version_entry_10

Aufrufzeitpunkt:

Die Datenbanktransaktion wurde gestartet. Alle Kenndaten, auch die Mehrfachattributfelder (60er-Felder) wurden aktualisiert.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, zu dem eine neue Nutzdatei eingespielt werden soll
error_update_attributes	1: Beim Aktualisieren der Kenndaten ist ein Fehler aufgetreten 0: Aktualisieren der Kenndaten O.K.
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

10.5 hook_new_version_exit_20 (doc_id, file_destination, import_ok, user, doc_type_short)

Aufrufzeitpunkt:

Auch die Mehrfachattributfelder (60er-Felder) wurden aktualisiert. Die neue Nutzdatei mit ggf. abhängigen Dokumenten wurde eingespielt. Die Datenbanktransaktion wurde beendet (COMMIT oder ROLLBACK).

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, zu dem eine neue Nutzdatei eingespielt werden soll
file_destination	Pfad und Name der Zieldatei (Angabe, wohin die Nutzdatei geschrieben worden ist)
import_ok	1: Einspielung der neuen Version O.K. 0: Einspielung der neuen Version mit Fehler abgebrochen.

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

10.6 hook_new_version_exit_30 (doc_id, import_ok, error_nr_api, user, doc_type_short)

Siehe hook_new_version_exit_20.

Eingabeparameter

Parameter	Beschreibung
doc_id	Dokumenten_ID des Dokuments, zu dem eine neue Nutzdatei eingespielt werden soll
import_ok	1: Einspielung der neuen Version OK 2: Einspielung der neuen Version mit Fehler abgebrochen
error_nr_api	Im Fehlerfall (import_ok=0): fehlercode des zuvor aufgetretenen Fehlers
user	Name des aufrufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

11 Erzeugen/ Bearbeiten von TIFF- oder PDF-Dokumenten

11.1 hook_rendition_entry_10 (doc_id, user)

Ab d.3 Version 6.0.0 Hotfix 2

Aufrufzeitpunkt:

Vor dem Start der Abbildungserstellung, wenn diese über einen d.3 Benutzer aufgerufen wurde (über d.3 API oder Server API). Wird nicht aufgerufen bei automatischem Aufruf über hinterlegte Regeln.

Der Aufruf kann durch Returnwert ungleich 0 abgebrochen werden.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokuments von dem ein TIFF- oder PDF-Abbild erstellt werden soll
user	Name des d.3-Benutzers, der Erstellung angefordert hat

11.2 hook_rendition_entry_20 (doc_id, doc_type_short, source_path, source_filename, dest_file)

Ab d.3 Version 6.0.0 Hotfix 2

Aufrufzeitpunkt:

Direkt vor dem Senden des Erstellungsjobs an den d.3 rendition service.

In dieser Hook-Funktion können Rendition-Optionen über die globalen Arrays `rendition_parameter_name` und `rendition_parameter_value` ausgelesen und gesetzt werden.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokuments von dem ein TIFF- oder PDF-Abbild erstellt werden soll
doc_type_short	Dokumentart-Kürzel
source_path	Quellpfad der Stammdatei
source_filename	Dateiname der Stammdatei
dest_file	Zielverzeichnis für die fertige Abbilddatei

11.3 hook_rendition_exit_30 (doc_id_ref, source_logi, tiff_file_with_path, error, file_type)

Ab d.3 Version 6.0.0 Hotfix 2

Aufrufzeitpunkt:

Nach dem Abholen der fertigen TIFF-/PDF-Datei vom Rendition Server.

Eingabeparameter

Parameter	Beschreibung
doc_id_ref	Doku-ID des Dokuments von dem ein TIFF- oder PDF-Abbild erstellt wurde
source_logi	Zielstatus des Dokumentes (B, P, F, A)
tiff_file_with_path	Zielpfad + Dateiname der Abbild-Datei
error	0 = ok -1 = Fehler beim Abholen der Datei vom Rendition Service -> siehe d.3-Logdatei
file_type	Dateityp, der gerendert wurde (P1, T1, TXT)

12 Login

12.1 hook_val_passwd_entry_10 (user_name, app_language, app_version)

Aufrufzeitpunkt:

Hook-Funktion vor der Prüfung von Benutzername und Passwort durch API-Funktion `ValidatePasswordForUser`.

Ein langer Benutzername ist bereits gegen den internen Namen getauscht worden.

Anmeldedaten können nicht verändert werden.

Ein Rückgabewert `!= 0` führt zum Abbruch

Eingabeparameter

Parameter	Beschreibung
user_name	anzumeldender Benutzer (d.3 Kurz- oder Langname; LDAP-Benutzername)
app_language	SprachID, die von der Anwendung übergeben wurde, z.B. 049=deutsch, 001=englisch
app_version	Versionsstring, der von der Anwendung übergeben wurde
Zeichen 1..3	Modulkennung z.B. 200 für d.xplorer
Zeichen 4..6:	Version des Moduls z.B. 620 für Version 6.2.0
Zeichen 7..8	Loglevel, z.B. 09 für DEBUG

Rückgabe:

Ein Wert != 0 führt zur Änderung des Rückgabewertes von API-Funktion `ValidatePasswordForUser` und somit zum Abbruch des Login.

Der Rückgabewert des Hook wird von 9500 abgezogen, d.h. 1 => 9500(-1) = 9501.

Diese Zahl wird an den Client zurückgegeben und ist somit in der `msglib.usr` zu hinterlegen.

12.2 hook_val_passwrd_exit_10 (error, user_name, app_language, app_version)

Aufrufzeitpunkt:

Test von Benutzername und Passwort gegen d.3-Benutzerstamm oder auch ggf. gegen einen Directory Server (per LDAP/Kerberos) sind gelaufen.

Das Ergebnis steht fest und wird als Parameter `error` übergeben.

Eingabeparameter

Parameter	Beschreibung
`error`	Fehlernummer der Benutzername/Passwort Prüfung; z.B. 0002 = falsche Benutzername/Passwort ; 0 = Erfolg
`user_name`	anzumeldender d.3-Benutzername (d.3-Benutzerkurzname)
`app_language`	Sprach-ID, die von der Anwendung übergeben wurde, z.B. 049=deutsch, 001=englisch
`app_version`	Versionsstring, der von der Anwendung übergeben wurde

Rückgabe:

Ein Rückgabewert != 0 führt zum Abbruch (siehe hook_val_passwrd_entry_10).

13 Löschen eines Dokuments (DeleteDocument)

Ab d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter `doc_type_short` für die Funktionen `hook_delete_entry_10`.

13.1 hook_delete_entry_10 (doc_id, user, doc_type_short)

Aufrufzeitpunkt:

Vor dem Löschen des Dokumentes. Es wurde erfolgreich geprüft, ob der Benutzer das Dokument löschen darf.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des zu löschenden Dokuments
user	Name des Benutzers (max. 10 Zeichen), der das Dokument löschen möchte
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

13.2 hook_delete_exit_10 (doc_id, user, error, doc_type_short)

Aufrufzeitpunkt:

Nach dem Löschen des Dokumentes.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des zu löschenden Dokuments
user	Name des Benutzers (max. 10 Zeichen), der das Dokument löschen möchte
error	0: Dokument wurde erfolgreich gelöscht sonst: Löschen fehlgeschlagen; Fehlercode
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

14 Löschen von Verknüpfungen (Unlink)

14.1 hook_unlink_entry_30 (doc_id_father, doc_id_child)

Ab d.3 Version 6.0.0 Hotfix 1

Aufruf:

Direkt vor dem Datenbank-Befehl zur Lösung der Verknüpfung

Eingabeparameter

Parameter	Beschreibung
doc_id_father	Doku-ID des übergeordneten Dokuments
doc_id_child	Doku-ID des untergeordneten Dokuments

14.2 hook_unlink_exit_10 (doc_id_father, doc_id_child, unlink_error_code, error_number)

Ab d.3 Version 6.0.0 Hotfix 1

Aufruf:

Nach der Verknüpfungslösung zweier Dokumente.

Eingabeparameter

Parameter	Beschreibung
doc_id_father	Doku-ID des übergeordneten Dokuments
doc_id_child	Doku-ID des untergeordneten Dokuments
unlink_error_code	0: Verknüpfungslösung war erfolgreich -1: Vater und Kind sind identisch bzw. einer der beiden existiert gar nicht -2: Die beiden Dokumente sind nicht verknüpft -4: Beim Entfernen der Verknüpfung trat ein Datenbankfehler auf (s. dazu error_number)
error_number	0 = Ok sonst Datenbank- oder Hook-Fehler

15 Postkorb

15.1 hook_ack_holdfile_exit_10 (user, doc_id)

Quittieren einer Wiedervorlage

ab d.3 Version 6.2.1

Aufrufzeitpunkt:

Nach dem Quittieren eines Postkorb-Eintrages durch Aufruf der API-Funktion AcknowledgeReceivedHoldFile. Verhindern lässt sich ein Quittieren nicht, da der Aufruf nach dem Quittieren stattfindet.

Eingabeparameter:

`user_name`: Benutzer, der die Quitterung ausgelöst hat

`doc_id`: Doku-ID des Dokumentes, welches quittiert wurde

16 Redlining (redline)

16.1 hook_write_redline_exit_30 (doc_id, user, doc_type_short)

Ab d.3 Version 6.0.0 Hotfix 3

Aufruf:

Nach dem Schreiben einer Redliningdatei (per d.3-API-Call `WriteRedline`).

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Doku-ID des Dokuments zu dem eine Redliningdatei abgelegt wird
`user`	Name des d.3 Benutzers
`doc_type_short`	Dokumentart Kürzel

17 Senden einer Wiedervorlage (SendHoldfile)

17.1 hook_holdfile_entry_10 (doc_id, recipient, sender, chain_id)

Aufrufzeitpunkt:

Wird aufgerufen, bevor die Übergabeparameter geprüft werden (geändert!). Empfänger und Sender könnenn somit auch noch länger als zehn Zeichen sein. Die Werte der Übergabeparameter sind auch noch in den folgenden globalen Feldern verfügbar:

```
d3server_empfaenger_wv[1]
d3server_sender_wv[1]
d3server_kette_id
```

Diese Werte können verändert werden!

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Doku-ID des Dokuments, welches in die Wiedervorlage gestellt werden soll

Parameter	Beschreibung
recipient	Benutzername des Empfängers (max. 10 Zeichen)
sender	Benutzername des Sender (max. 10 Zeichen)
chain_id	Ketten-ID, die für diesen Wiedervorlageeintrag verwendet werden soll
betreff	Betrefftext der Postkorbbenachrichtigung
wv_typ	Typ-ID der Postkorbbenachrichtigung

Mögliche Werte:

"" = normale Postkorbbenachrichtigung
"W" = Workflow-Benachrichtigung
... = sonstige (ggf. selbst definierte Werte)

17.2 hook_holdfile_entry_20 (doc_id, recipient, sender, chain)

Wird aufgerufen, wenn Datum etc. bereits auf Plausibilität geprüft worden sind. Es sind aber noch die Rechte des Empfängers auf das Dokument geprüft. Hier sind der Sender und der Empfänger maximal 10 Zeichen alng. Die Werte sind hier nicht mehr änderbar.

Eingabeparameter siehe hook_holdfile_entry_10.

17.3 hook_holdfile_entry_30 (doc_id, recipient, sender, chain)

Wird aufgerufen, direkt vor dem Eintrag in die Datenbank, wenn auch schon die Rechte des Empfängers auf das Dokument geprüft wurden. Die Werte sind hier nicht mehr änderbar.

Dies ist die Stelle, wo bisher hook_holdfile_entry_10 aufgerufen wurde. Alter und neuer hook_holdfile_entry_10 sind von der Übergabe und den verfügbaren Werten her kompatibel.

Eingabeparameter siehe hook_holdfile_entry_10.

17.4 hook_holdfile_exit_10 (doc_id, recipient, sender, chain_id, error_number_db)

Aufrufzeitpunkt:

Direkt nach dem Datenbank-Befehl, der die Wiedervorlage aktiviert.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des Dokuments, welches in die Wiedervorlage gestellt werden soll
recipient	Benutzername des Empfängers (max. 10 Zeichen)
sender	Benutzername des Sender (max. 10 Zeichen)
chain_id	Ketten-ID, die für diesen Wiedervorlageeintrag verwendet werden soll
error_number_db	0: alles OK sonst: Datenbank-Fehlernummer beim Eintrag der Wiedervorlage in die Datenbank

18 Senden von E-Mails bei Wiedervorlage (send_email)

18.1 hook_send_email_entry_10 (doc_id, recipient, semder. subject, holdfile)

Ab d.3 Version 6.1.1

Aufruf:

Vor dem Versenden einer E-Mail.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des d.3 Dokumentes bei E-Mail wegen Wiedervorlage
recipient	Empfänger der E-Mail (d.3 Benutzername oder E-Mail-Adresse)
sender	Absender der E-Mail (d.3 Benutzername)
subject	Betrefftext
holdfile	0 = keine Wiedervorlage-E-Mail 1 = E-Mail für Wiedervorlage 2 = E-Mail für Workflow-Wiedervorlage

In dieser Hook-Funktion können folgenden E-Mail-Attribute gesetzt werden:

Attribut	Beschreibung
api_email_body_file	Name und Pfad einer Datei, die den Body-Text enthält
api_email_mail_format	"html": HTML-Format sonst: Text-Format (Default)
api_email_attach	1 = d.3-Dokument als Anhang 0 = Dokument nicht anhängen (Default)

Die Attribute werden jeweils nach erfolgtem E-Mail-Versand zurückgesetzt. Die E-Mail-Funktion kann durch Returnwert ungleich 0 abgebrochen werden.

18.2 hook_send_email_entry_20 (doc_id, recipient, semder. subject, holdfile)

Ab d.3 Version 6.1.1

Aufruf:

Vor dem Versenden einer E-Mail. E-Mailadresse wurde ermittelt, Gruppenauflösung wurde durchgeführt.

Die Hook-Funktion wird nur einmal aufgerufen und nicht für jede versendete Mail verschickt.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des d.3 Dokumentes bei E-Mail wegen Wiedervorlage
recipient	Empfänger der E-Mail (d.3 Benutzername oder E-Mail-Adresse)
sender	Absender der E-Mail (d.3 Benutzername)
subject	Betrefftext
holdfile	0 = keine Wiedervorlage-E-Mail 1 = E-Mail für Wiedervorlage 2 = E-Mail für Workflow-Wiedervorlage

In dieser Hook-Funktion können folgende E-Mail-Attribute gesetzt werden:

Attribut	Beschreibung
api_email_body_fil	Name und Pfad einer Datei, die den Body-Text enthält

Attribut	Beschreibung
e	
api_email_mail_format	"html": HTML-Format sonst: Text-Format (Default)
api_email_attach	1 = d.3 Dokument als Anhang 0 = Dokument nicht anhängen (Default)

Die Attribute werden jeweils nach erfolgtem E-Mailversand zurückgesetzt. Die E-Mail-Funktion kann durch Returnwert ungleich 0 abgebrochen werden.

18.3 hook_send_email_exit_10 (doc_id, recipient, sender, subject, success)

Ab d.3 Version 6.1.1

Aufruf:

Nach dem Versenden einer E-Mail.

Eingabeparameter

Parameter	Beschreibung
doc_id	Doku-ID des d.3 Dokumentes bei E-Mail wegen Wiedervorlage
recipient	Empfänger der E-Mail (d.3 Benutzername oder E-Mail-Adresse)
sender	Absender der E-Mail (d.3 Benutzername)
subject	Betrefftext
success	1 = Nachricht wurde gesendet 0 = Nachricht konnte nicht gesendet werden

19 Sperren eines Dokuments

Aufrufzeitpunkt:

Vor bzw. nach dem Sperren eines Dokuments in den Status "Freigabe".

19.1 hook_block_entry_10 (doc_id, user)

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, zu blockieren / freizugeben ist
user	Name des rufenden Benutzers

19.2 hook_block_exit_10 (doc_id, user)

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, zu blockieren / freizugeben ist
user	Name des rufenden Benutzers

20 Statustransfer

Aufrufzeitpunkt:

Vor bzw. nach dem Statustransfer eines Dokuments in einen anderen Status.

20.1 hook_transfer_entry_30 (user, doc_id, archiv_index, source_logi, desti_logi, desti_user_o_group)

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_id	Doku-ID des Dokumentes, das transferiert werden soll
archive_index	Index der Version im Archiv (> 0), falls eine Version aus dem Archiv zurückgeholt werden soll
source_logi	Quellstatus des Dokumentes (B, P, F, A)
desti_logi	Zielstatus des Dokumentes (B, P, A)
destination_user_o_group	Zielstatus „Bearbeitung“: Benutzer- oder Gruppenname Zielstatus „Prüfung“: Gruppenname Sonst leer.

Wird erst ab der d.3 Version 5.5.1 Hotfix 06 auch beim asynchronem Statustransfer (vom d.3 async) aufgerufen.

20.2 hook_transfer_exit_30 (user, doc_id, archiv_index, source_logi, desti_logi, desti_user_o_group, error)

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_id	Doku-ID des Dokumentes, das transferiert werden soll
archive_index	Index der Version im Archiv (> 0), falls eine Version aus dem Archiv zurückgeholt werden soll.
source_logi	Quellstatus des Dokumentes (B, P, F, A)
desti_logi	Zielstatus des Dokumentes (B, P, A)
destination_user_o_group	Zielstatus „Bearbeitung“: Benutzer- oder Gruppenname Zielstatus „Prüfung“: Gruppenname Sonst leer.
error	0 = Statustransfer erfolgreich Fehlercode sonst

21 Validieren der Attribute vor der Kenndatenaktualisierung

Aufrufzeitpunkt:

Bei der Validierung vor der Kenndatenaktualisierung

Seit d.3 Version 5.5.0.

21.1 hook_validate_update_entry_10 (user, doc_type_short)

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Dokumentart Kürzel

22 Validieren der Attribute vor Suchen bzw. Anlegen eines Dokumentes (ValidateAttributes)

Ab der d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter `doc_type_short` für die Funktionen
hook_validate_search_entry_10
hook_validate_search_exit_30

22.1 hook_validate_search_entry_10 (user, doc_type_short)

Falls diese Hook-Funktion einen Wert ungleich 0 liefert, wird die Validierung der Suchbegriffe abgebrochen. Die API-Funktion `ValidateAttributes` liefert dann den Wert 9500-(X) zurück (allgemeiner Fehlercode in kundenspezifischer Hook-Funktion). "X" ist hier der Rückgabewert des Hooks. Es wird empfohlen, im Hook einen negativen Return-Wert zu verwenden, damit die Ausgabe des API-Calls >9500 ist, da dieser Bereich freigehalten wurde. Es ist dann möglich, auf den Client-Rechnern über die Client-Verteilung eine `msglib.usr` mit einem beliebigen Text für den Returncode (z.B. 9542) zu hinterlegen.

Verfügbare Felder:

Siehe hook_search_entry_10

Aufrufzeitpunkt:

Es wurden lediglich die Suchbegriffe in die entsprechenden JAM-Felder transportiert.

Eingabeparameter

Parameter	Beschreibung
`user`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart des zu importierenden Dokuments

22.2 hook_validate_import_entry_10 (user, doc_type_short)

Falls diese Hook-Funktion einen Wert ungleich 0 liefert, wird die Validierung der Kenndaten für den Import abgebrochen. Die API-Funktion `ValidateAttributes` liefert dann den Wert 9500-(X) zurück (allgemeiner Fehlercode in kundenspezifischer Hook-Funktion). "X" ist hier der Rückgabewert des Hooks. Es wird empfohlen, im Hook einen negativen Return-Wert zu verwenden, damit die Ausgabe des API-Calls >9500 ist, da dieser Bereich freigehalten wurde. Es ist dann möglich, auf den Client-Rechnern über die Client-Verteilung eine `msglib.usr` mit einem beliebigen Text für den Returncode (z.B. 9542) zu hinterlegen.

Verfügbare Felder:

Siehe hook_insert_entry_10

Aufrufzeitpunkt:

Es wurden lediglich die Kenndaten des neu zu importierenden Dokumentes in die entsprechenden JAM-Felder transportiert.

Eingabeparameter

Parameter	Beschreibung
user	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart

Diese Funktion wird erst nach einem `ValidateAttributes` ausgeführt. Das bedeutet, dass die Funktion nicht ausgeführt wird, wenn ein Dokument über den Hostimport importiert wird. Sie wird ausgeführt, wenn man einen Import über den Import Client ausführt, da vor dem `ImportDocument` ein `ValidateAttributes` ausgeführt wird.

22.3 hook_validate_update_entry_10 (user_ref, dokuart_kurz, doc_id)

Parameter	Beschreibung
user_Ref	Name des rufenden Benutzers
dokuart_kurz	Kürzel der Dokumentart
doc_id	Doku-ID des d.3 Dokumentes, dessen Kenndaten aktualisiert werden sollen

22.4 hook_val_passwd_entry_10 (user, language, version)

Ausführung bei Validierung des Passwortes

22.5 hook_val_passwd_exit_10 (error, user, language, version)

23 Verknüpfen zweier Dokumente (z.B. Akte mit Dokument)

23.1 hook_link_entry_10

reserviert

23.2 hook_link_entry_20

reserviert

23.3 hook_link_entry_30 (doc_id_father, doc_id_child)

Diese Hook-Funktion wird nur aktiviert, wenn zuvor kein Fehler aufgetreten ist. Liefert diese Funktion einen Wert ungleich 0, so kann somit die Verknüpfung gestoppt werden.

Aufrufzeitpunkt:

Alle Verknüpfungsdaten sind korrekt. Direkt vor dem Datenbank-Befehl, der die Verknüpfung durchführt.

Eingabeparameter

Parameter	Beschreibung
doc_id_father	Doku-ID des übergeordneten Dokuments (i. a. Akte)
doc_id_child	Doku-ID des untergeordneten Dokuments

23.4 hook_link_exit_10 (doc_id_father, doc_id_child, error_code, error_number)

Aufrufzeitpunkt:

Direkt nach dem Datenbank-Befehl, der die Verknüpfung durchführen sollte.

Eingabeparameter

Parameter	Beschreibung
doc_id_father	Doku-ID des übergeordneten Dokuments (i.a. Akte)
doc_id_child	Doku-ID des untergeordneten Dokuments
error_code	0: Verknüpfung war erfolgreich -1: Vater und Kind sind identisch bzw. einer der beiden existiert gar nicht -2: Die beiden Dokumente sind bereits verknüpft -3: Die beiden Dokumente sind bereits in umgekehrter Hierarchie miteinander verknüpft -4: Beim Eintrag der Verknüpfung in die Datenbank trat ein Datenbankfehler auf (s. dazu „error_number“) error_number <> 0: - 91: Die beiden Dokumente sind bereits in umgekehrter Hierarchie miteinander verknüpft sonst Datenbank-Fehlernummer beim Eintrag der Verknüpfung in die Datenbank

23.5 hook_link_exit_20

reserviert

23.6 hook_link_exit_30

reserviert

24 Web-Veröffentlichung

Aufrufzeitpunkt:

Vor bzw. nach dem Veröffentlichen eines Dokuments für das Web.

24.1 hook_webpublish_entry_10 (doc_id, user)

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, das veröffentlicht / zurückgezogen werden soll
user	Name des rufenden Benutzers

24.2 hook_webpublish_entry_20 (doc_id, user, doc_type_short)

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, das veröffentlicht / zurückgezogen werden soll
user	Name des rufenden Benutzers
doc_type_short	Dokumentartkürzel

24.3 hook_webpublish_entry_30 (doc_id, user, doc_type_short)

siehe hook_webpublish_entry_20

24.4 hook_webpublish_exit_10 (doc_id, user, error, doc_type_short)

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, das veröffentlicht / zurückgezogen werden soll
user	Name des rufenden Benutzers
error	0= Aktion erfolgreich Fehlercode sonst
doc_type_short	Dokumentartkürzel

24.5 hook_webpublish_exit_20 (doc_id, user, error, doc_type_short)

Parameter	Beschreibung
doc_id	Doku-ID des Dokumentes, das veröffentlicht / zurückgezogen werden soll
user	Name des rufenden Benutzers
doc_type_short	Dokumenartkürzel
error	0 = Aktion erfolgreich Fehlercode sonst

24.6 hook_webpublish_exit_30 (doc_id, user, error, doc_type_short)

siehe hook_webpublish_exit_20

25 Workflow

25.1 hook_workflow_cancel_exit_20 (doc_id, wfl_id, step_id, user)

Ab d.3 Version 7.2.0

Diese Hookfunktion wird nur aktiviert, wenn zuvor kein Fehler aufgetreten ist.
Der Abbruch des Workflows kann hier nicht gestoppt werden.

Aufrufzeitpunkt:
Nachdem der Workflow für ein Dokument abgebrochen wurde.

Parameter	Beschreibung
doc_id	d.3 Dokument ID
wfl_id	ID des Workflows
step_id	ID des Workflow-Schrittes
user	ID des d.3 Benutzers

Teil II Hook-Funktionen im Einzelnen

26 Aktivieren der Hook-Funktionen

Zu allen umleitbaren Hook-Funktionen besitzt d.3 standardmäßig sogenannte „Stubs". Dies sind leere Funktionen, die als Dummies fungieren. Man kann diese auf die selbstgeschriebenen Funktionen umleiten und damit eine individuelle Programmfunktionalität implementieren.

Nachdem man seine selbstgeschriebene Hook-Funktion erstellt hat, muss diese dem d.3 System bekannt gegeben werden. Dies geschieht mittels der d.3 Administration unter d.3 Config in zwei Schritten:

- Öffnen Sie die Sektion **Hook-Funktionen**.
- Öffnen Sie die Sektion **JPL-Datei für kundenspezifische Programmanpassungen**.
- Tragen Sie hier die JPL-Datei ein, die Ihre Hook-Funktion beinhaltet. Diese wird dann beim Start der d.3 Prozesse geladen.
- Öffnen Sie die Sektion **Hook-Funktionen ausführen**.
- Geben Sie hier an der gewünschten Stelle (Eintrittspunkt) den Namen Ihrer Hook-Funktion an, die die Standardfunktion ersetzen soll.

Teil III
Besondere Hook-Funktionen

Teil III Besondere Hook-Funktionen

1 Repository-Hooks

Diese Hook-Funktionen dienen zur Erzeugung dynamischer Wertemengen oder zur individuellen Eingabe-Validierung.

Die Sortierung der Wertemengen durch den Server kann durch den Parameter `hook_dataset_sort=0` abgeschaltet werden, ansonsten findet im Standard eine alphabetische Sortierung statt.

1.1 Zur Erzeugung dynamischer Wertemengen (Wertemengenhooks)

Maximal können über einen Wertemengen-Hook 10.000 verschiedene Werte ermittelt werden.

Implizite Parameter

`repos_id`	Repository-ID (ID des Attributes, für das der Hook gerufen wird)
`user`	aufrufender Benutzer
`doc_type_short`	Kürzel der Dokumentart
`row_no`	In Ergänzung zur `repos_id` die Zeilenbei 60er-Feldern, für die die Werte abgefragt werden sollen.

Rückgabewert

ohne

d.3 Variablen

`d3server_value_char_allowed`	Array für die Zusammenstellung von alphanumerischen Wertemengen
`d3server_value_date_allowed`	Array für die Zusammenstellung für Datums-Wertemenge
`d3server_repos_id_allowed`	Array mit Angabe der zugehörigen Repository-IDs

d.3 Konfiguration (d.3 Config in der d.3 Administration – `d3config.ini`)

DATASET_NO_CACHING aktivieren

Die übrigen `dok_dat` Felder sind je nach Kontext gesetzt.

Suche: die übrigen Suchkriterien

Import: die übrigen bereits gefüllten Attribute

Der Schalter HOOK_DATASET_SORT kann in Hook-Funktionen für dynamische Wertemengen gesetzt werden.

Per Default werden die Werte aus dynamischen Wertemengen von d.3 sortiert.

Durch Setzen von HOOK_DATASET_SORT = 0 in der Hook-Funktion kann man erreichen, dass d.3 die Werte nicht sortiert.

Die durch die Hook-Funktion vorgegebene Reihenfolge der Werte wird dadurch beibehalten.

1.2 Hinzufügen von Werten für dynamische Wertemengen zu einer Benutzer-Auswahlliste

1.2.1 Funktion user_dataset_add_value

Diese Funktion kann in Hook-Funktionen für dynamische Wertemengen benutzt werden, um beliebig viele Werte zu einer Benutzer-Auswahlliste hinzuzufügen.

Ist die Wertemenge eine Pflichtwertemenge (Default = Ja), muss zur Wert-Validierung auch eine Hook-Funktion für die Plausibilitäts-Kontrolle des Repositoryfeldes hinterlegt werden.

Parameter	Beschreibung
doc_type_short	Kürzel der Dokumentart
repos_id	ID des Repository-Feldes
value	hinzuzufügender Wert

```
// Repository-Hook-Funktionen für dynamische Wertemenge

proc DW_FunktionsName( h_ReposID, h_User, h_DocTypeShort,
h_RowNo )
  {
    call api_log_debug( "++++++++START DW_FunktionsName
( :h_ReposID, :h_User, :h_DocTypeShort, :h_RowNo ) ++++++
++" )
    if(WERTEMENGEN_NICHT_CACHEN !=1)
  {
    call api_log_error( "Wertemengen werden aktuell
gecached und werden daher nur einmalig berechnet!" )
       return ReturnValue //Sollte durch einen sinnvollen
Returnwert ersetzt werden
  }
//Datenzuweisung:
vars i[10]={1,2,3,4,5,6,7,8,9,10}
for i = 1 while(i<=10) step 1
{
//Beide Optionen sind möglich!

//Option 1:
call user_dataset_add_value( h_DocTypeShort,
h_ReposID,"Wert :i")

//Option2:
```

```
d3server_value_char_allowed[i] = "Wert :i"
d3server_repos_id_allowed[i] 0 h_ReposID


}

call api_log_debug( "++++++++ENDE DW_FunktionsName
( :h_ReposID, :h_User, :h_DocTypeShort, :h_RowNo ) ++++++
++" )
}// end of DW_FunktionsName
```

1.3 Zur individuellen Eingabe-Validierung (Plausibilitätshooks)

Impliziter Parameter

`Feld-Inhalt`: Wert, der getestet werden soll

Rückgabewert

0 bei Erfolg

< > 0 bei Fehler

Hier kann man sich nicht darauf verlassen, dass die bekannten globalen Variablen wie `doku_id`, `dok_dat_feld[x]` etc. sinnvolle Werte enthalten, ja nachdem, in welchem Zusammenhang die Funktion gerufen wird. Die anderen `dok_dat_feld`-Felder stehen nicht zur Verfügung.

Beispiel für einen Repository-Hook zur Plausibilitätsprüfung

```
proc check_projekt_nr (projekt_nr)
  {
   //-----------------------------------------------------
-
   // Hier wird die eingegeben Projekt-Nr geprüft.
   // Falls diese OK ist, werden die korrespondierenden
   // Attributfelder aufgefüllt.
   //-----------------------------------------------------
-
    vars  gp_nr, gp_name, projekt_bez, fehler, anz
    if (projekt_nr == "")
      {
        dok_dat_feld_60[1] = ""
        return 0
      }
    DBMS ALIAS  gp_nr, gp_name, projekt_bez
    DBMS SQL SELECT f.dok_dat_feld_44, f.dok_dat_feld_28,
value_char \
                  FROM firmen_spezifisch  f,
firm_spez_mult_val m       \
               WHERE f.doku_id = m.doku_id
             \
                 AND f.kue_dokuart = 'PROJA'
             \
                 AND f.dok_dat_feld_8 = :+ projekt_nr
             \
                 AND m.field_no = 60
    anz = @dmrowcount
    DBMS ALIAS
    if ( anz > 0 )
      {
        dok_dat_feld[44] = gp_nr
        dok_dat_feld[28] = gp_name
        dok_dat_feld_60[1] = projekt_bez
        fehler = 0
      }
    else
        fehler = -1
    return fehler
  }  // proc check_projekt_nr
```

2 Dokumentklassen-Hooks

Dokumentklassen-Hooks dienen zur Bestimmung von Berechtigungen.

`@D3HOOK (<procedure_name>)`

Import Parameter

Parameter	Beschreibung
attrib_value	Wert des Dokumentattributes, für das die Hook-Funktion aufgerufen wurde
user	Name des ausführenden Benutzers
doc_type_short	Kürzel der Dokumentart

Rückgabewert

1: Berechtigt

0: kein Zugriff

3 Aktenplan-Hooks (d.3 folder scheme)

Aktenplan-Hooks dienen zur individuellen Verknüpfung von Dokumenten und Akten.

Implizite Parameter

1. Nummer der Dokument - Akte Zuordnung im Aktenplan

2. Doku-ID des aktuellen (zu verknüpfenden) Dokuments oder Akte

4 Lastverteiler

4.1 hook_holdfile_balance_entry_10 (doc_id, object_id, ignore_checkout)

Ermöglicht das Tätigkeitsprofil vor Beginn des Lastverteilers zu ändern. Über die globale Variable `d3server_lv_profile` lässt sich das neue Profil setzen.

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Dokument-ID des Dokumentes, welches per Lastverteiler versendet werden soll
`object_id`	ID des Tätigkeitsprofils (aus den Tabellen `user_functions` und `object_types`)
`ignore_checkout`	ignorieren, ob Benutzer sich abwesend gemeldet haben

globale JPL-Variablen:

`d3server_lv_profile`: Enthält bei Rückgabewert = 0 die Bezeichnung des neuen Tätigkeitsprofils (kann im Hook gesetzt werden)

4.2 hook_holdfile_balance_exit_10 (doc_id, object_id, ignore_checkout, username, altuser)

Wenn im Tätigkeitsprofil nur ein Benutzer Mitglied ist, und sowohl dieser Benutzer als auch sein Vertreter abwesend gemeldet sein, wird dieser Hook aufgerufen. Damit kann man in diesem „Notfall" eingreifen, und den Benutzer oder eine Gruppe selber setzen.

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Dokument-ID des Dokumentes, welches per Lastverteiler versendet werden soll
`object_id`	ID des Tätigkeitsprofils (Tabellen `user_functions` und `object_types`)
`ignore_checkout`	ignorieren, ob Benutzer sich abwesend gemeldet haben
`username`	Vom Lastverteiler gewählter Benutzer
`altuser`	Vertreter von „username"

globale JPL-Variablen:

`d3server_lv_user`: Enthält bei Rückgabewert = 0 den Benutzer- oder Gruppenname für die Postkorb-Versendung (kann im Hook gesetzt werden)

4.3 hook_holdfile_balance_exit_20 (doc_id, object_id, ignore_checkout)

Wenn der Lastverteiler keinen Benutzer bestimmen konnte, weil z.B. alle Benutzer sich „abwesend" gemeldet haben, kann man in diesem „Notfall" über diesen Hook den Benutzer oder eine Gruppe selber setzen.

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Dokument-ID des Dokumentes, welches per Lastverteiler versendet werden soll
`object_id`	ID des Tätigkeitsprofils (Tabellen `user_functions` und `object_types`)
`ignore_checkout`	ignorieren, ob Benutzer sich abwesend gemeldet haben

globale JPL-Variablen

`d3server_lv_user`: Enthält bei Rückgabewert = 0 den Benutzer- oder Gruppenname für die Postkorb-Versendung (kann im Hook gesetzt werden)

4.4 hook_holdfile_balance_exit_30 (doc_id, object_id, ignore_checkout, username)

Mit diesem Hook kann der vom Lastverteiler ausgewählte Benutzer noch einmal geändert werden. Dabei kann dann ein Benutzer oder eine Gruppe gesetzt werden.

Eingabeparameter

Parameter	Beschreibung
`doc_id`	Dokument-ID des Dokumentes, welches per Lastverteiler versendet werden soll
`object_id`	ID des Tätigkeitsprofils (Tabellen `user_functions` und `object_types`)
`ignore_checkout`	ignorieren, ob Benutzer sich abwesend gemeldet haben
`username`	Enthält den vom Lastverteiler ausgewählten Benutzer

globale JPL-Variablen

`d3server_lv_user`: Enthält bei Rückgabewert = 0 den Benutzer- oder Gruppenname für die Postkorb-Versendung (kann im Hook gesetzt werden)

5 Dynamische Rückmeldungen

Die globale Variable `additional_info_text` kann aus jedem Hook gesetzt werden. Wird der Hook von einem d.3 server-Prozess (nicht hostimp oder async) aufgerufen, wird dieser Text als zusätzlicher Exportparameter bei dem aktuellen API-Call an an den Client übergeben.

Dieser Text wird dem Benutzer nicht durchgängig bei allen Clients angezeigt.

Gilt für die folgenden Einsprungspunkte:

- hook_search_entry_10
- hook_upd_attrib_entry_20
- hook_holdfile_entry_10
- hook_validate_import_entry_10
- hook_transfer_entry_30
- hook_rendition_entry_10

Teil IV
Grundzüge der JPL-Programmierung

Teil IV Grundzüge der JPL-Programmierung

1 Allgemein

Im nachfolgenden Kapitel erfolgt eine erste Beschreibung der Grundzüge der JPL-Programmierung. Hierzu wird aber auch auf ergänzende JPL-Dokumentation verwiesen!

2 Prozeduraufbau

Man unterscheidet benannte und unbenannte Prozeduren.

Beispiel einer benannten Prozedur

```
proc benannte_prozedur
{
     vars  h_Count,  h_Zaehler

     for h_Count=1 while h_Count<=100 step 1
     msg emsg "Der Inhalt der Variablen h_Count ist jetzt
:h_Count"
}// end of benannt_prozedur
```

Beispiel einer unbenannten Prozedur

```
vars  h_Count  // zaehler

for h_Count=1 while( h_Count<=100) step 1
     msg emsg "Der Inhalt der Variablen h_Count ist
jetzt :h_Count"
```

In jeder JPL-Sourcecodedatei kann maximal eine unbenannte und beliebig viele benannte Prozeduren enthalten sein.

Falls eine unbenannte Prozedur verwendet wird, muss sie vor allen benannten Prozeduren stehen.

Eine unbenannte Prozedur wird implizit beim Laden der JPL-Sourcedatei (z.B. mit „public“) ausgeführt. Eine benannte Prozedur hingegen muss explizit aktiviert werden (z.B. mit „call“). Hook-Funktionen müssen immer in benannte Prozeduren gestellt werden, da sie über einen Prozedurnamen aktiviert werden.

Eine Prozedur wird entweder bis zu ihrem Ende durchlaufen oder bis das erste „return“ erreicht wird.

Zur Gruppierung von Anweisungen in Schleifen oder anderen Kontrollstrukturen dienen die geschweiften Klammern (analog zu **BEGIN** und **END** in Pascal). Dabei ist darauf zu achten, dass sowohl „{“ , als auch „}“ in einer eigenen Zeile stehen.

```
for h_Count = 1 while( h_Count<=100) step 1
{
      h_Zaehler_1 = h_Zaehler_1 * h_Count
     h_Zaehler_2 = h_Zaehler_2 + h_Count
}
```

Kommentarzeilen beginnen mit "//" oder "#" ab der ersten Stelle einer Kodezeile.

Falls eine JPL-Anweisung nicht in eine einzelne Zeile paßt, muss man mit Fortsetzungszeilen arbeiten. Jede Fortsetzungszeile ist in der vorausgehenden Zeile mit einem angehängten „\" zu kennzeichnen.

```
H_Zaehler  =     h_Zaehler1 + h_Zaehler2 + \
           +     h_Zaehler3 + h_Zaehler4 + \
             +     h_Zzaehler5 \
\                +     h_Zaehler 6
```

3 Funktionsparameter

Die formalen Parameter einer Funktion werden kommasepariert in Klammern nach dem Prozedurnamen angegeben.

```
proc hook_prozedur (doc_id, doc_number, inv_number)"
```

JPL übergibt Parameter `call by value`, d.h. das Ändern des Wertes eines Parameter in der gerufenen Prozedur hat keine Auswirkung auf die rufende Prozedur. Für jeden Parameter wird in der Prozedur ein eigener Speicherbereich angelegt.

Alle d.3 Hook-Funktionen liefern einen Integerwert (Ganzzahl) an die rufende Programmeinheit zurück. Dieser Wert gibt, sofern ungleich 0, i. a. einen Fehlerkode an.

4 Prozeduraufruf

Allgemeiner Aufruf einer JPL-Prozedur:

```
call  <prozedurname>  (<Parameterliste>)
```

```
call hook_prozedur(doc_id, doc_number, inv_number)
```

Damit eine JPL-Prozedur mit `call` aufgerufen werden kann, muss das JPL-Quellfile, welches die Prozedur enthält, zuvor mit `public` in den Speicher geladen worden sein. Das Quellfile kann mittels `unload` wieder aus dem Speicher entfernt werden.

Rückgabewert einer Prozedur:

Der (ganzzahlige) Rückgabewert einer mit `call` gerufenen Prozedur landet in einer mit `retvar` definierten Empfangsvariablen.

Beispiel

```
vars        i  j
retvar      empf_variable(10)
```

Alternativ kann eine Prozedur auch folgendermaßen aufgerufen werden:

```
<Empfangsvariable> = <Prozedurname> ()

vars i  j  wert
wert = ermittle_wert (i)
```

5 Variablendeklaration

Variablen werden am Anfang einer benannten bzw. unbenannten Prozedur deklariert.

Deklaration in einer benannten Prozedur: Die Variablen sind in der gesamten Prozedur verfügbar.

Deklaration in einer unbenannten Prozedur: Die Variablen sind in der unbenannten und allen benannten Prozeduren des JPL-Quellfiles verfügbar.

Variablennamen:

- Bestehen aus bis zu 31 Zeichen (Ziffern, Buchstaben, „_“).
- Das erste Zeichen darf keine Ziffer sein.
- Groß- und Kleinschreibung wird unterschieden.
- Per Default kann eine Variable bis zu 255 Zeichen aufnehmen.

Variablen wird kein Datentyp zugeordnet. Alle Variablen werden als Zeichenketten behandelt. Bei Bedarf werden sie z.B. automatisch in numerische Werte verwandelt.

Man kann auch eindimensionale Felder definieren:

```
vars variablen_name[Anzahl Feldelemente](Länge)
```

```
vars zaehler(10),  zeile_1,  feld[100](20)
```

Die Variable `zaehler` kann bis zu 10 Zeichen aufnehmen. Die Variable `zeile_1` kann bis zu 255 Zeichen aufnehmen. `feld` ist ein Feld mit bis zu 100 Elementen zu je max. 20 Zeichen Länge.

6 Zeichenkettenverarbeitung

Beispiel zur Zeichenkettenverarbeitung

```
vars  kette_1  kette_2(5)  kette_3  laenge
kette_1 =     "ABCDEFG"
kette_2 =     "ZYXWVUTSR“      // kette_2 nimmt nur "ZYXWV"
                               // auf
kette_3 =     kette_1(3,2) ## kette_2(3,5)
laenge  =     @length (kette_3)

msg emsg "kette_3: :kette_3  Länge von kette_3: :laenge"
```

Ausgabe:

```
kette_3: CDXWV   Länge von kette_3: 5
```

7 Operatoren

Operatortyp	Operatoren
numerisch	+, -, *, /
relational	<, <=, >, >=, ==, !=
logisch	!, &&, ||, (NOT, AND, OR)
bitweise	~, &, |

8 Schleifen

```
for <zaehler> = <Anfangswert> while <logischer Ausdruck>
step <Schrittweite>
```

```
for   i=1 while i<=100 step 1
{
…..
}
```

```
while   <logischer Ausdruck>
```

```
while fehler == 0
{
…..
}
```

`next`: bewirkt, dass der nächste Schleifendurchlauf vorzeitig initiiert wird

`break`: springt aus einer Schleife

9 Verzweigungen

```
if (<logischer Ausdruck>)
{
}
else if (<logischer Ausdruck>)
{
}
else
{
}
```

Beispiel

```
if (zaehler == 100)
{
}
else if (zaehler < 100)
{
}
else
{
}
```

10 Aufruf eigener Programme (.exe) aus einer Hook-Funktion

Zum Aufruf selbstgeschriebener `.exe`-Programme gibt es die Funktion

`commando ("<Programmaufruf>" )`

Hierbei steht `<Programmaufruf >` für die Kommandozeile, mit der das Programm gestartet wird. Dieser Aufruf kann also auch Parameter beinhalten.

```
vars  datei1  datei2
datei1 = ...
datei2 = ...
call commando ("C\:\\PROGS\\KONVERT  :datei1  :datei2")
```

11 Testen selbstgeschriebener Hook-Funktionen

Hook-Funktionen greifen in elementare d.3 Operationen ein. Daher sollten sie sorgfältig getestet werden, bevor sie in einem Produktivsystem eingesetzt werden.

Um den Durchlauf einer Hook-Funktion zu tracen, kann man Ausgaben erzeugen und dabei z.B. Variableninhalte ausgeben o.ä.:

`msg emsg "...."`

Alle mit `msg emsg` erzeugten Meldungen können als "critical error" im Logviewer eingeschaut werden

`call d3_server_api_error_log ("<Text>", 1, <Log-Level>)`

Gibt `<Text>` gezielt im Logviewer aus. Dabei hängt es vom Schweregrad der Meldung ab, ob sie wirklich mitprotokolliert wird.

Den Schweregrad kann man mit <Log-Level> angeben. Zulässige Log-Level sind 0, 3, 6, 7 und 9.

Nur wenn der d.3 Serverprozess mit einer Log-Levelstufe (D3_SERVER_LOG_LEVEL) größer oder gleich der hier angegebenen läuft, gibt er die Meldung aus.

Kritische Meldungen sind normalerweise mit Log-Level 0 auszugeben, Debug-Trace-Informationen hingegegen mit Log-Level 9.

Die msg-Meldungen sollten Sie nur beim Testen der Hook-Funktionen benutzen. Im Betrieb sollten Sie diese nicht verwenden!

Teil V
Datenbankzugang mittels JPL

Teil V Datenbankzugang mittels JPL

1 DBMS SQL

Man kann in JPL einen Datenbankbefehl prinzipiell genauso schreiben wie man es von SQLPlus, SQLScope oder ISQL gewohnt ist.

Einem SQL-Befehl sind die Schlüssewörter **DBMS SQL** voranzustellen.

```
DBMS SQL SELECT zeich_nr FROM … WHERE …
```

2 DBMS ALIAS

DBMS ALIAS leitet die Daten eines SQL-SELECT in JPL-Variablen um.

```
vars  h_var_1, h_var_2, h_var_3
DBMS ALIAS h_var_1, h_var_2, h_var_3
DBMS SQL SELECT spalte_1, spalte_2, spalte_3 \
      FROM tabelle \
      WHERE …
DBMS ALIAS
```

Die Inhalte der drei selektierten Spalten werden, sofern der Datenbank-Server einen Datensatz liefert, in die JPL-Variablen `var_1`, `var_2` und `var_3` geladen.

Nicht vergessen, einen gesetzten **ALIAS** durch **DBMS ALIAS** wieder zu deaktivieren.

3 DB-Statusvariablen

@dmretcode

Numerischer Returncode des JAM/Panther-Datenbankinterfaces.

Diese Returncodes sind unabhängig vom DB-Server (Oracle, Microsoft SQL, Pervasive SQL).

Es folgt ein kleiner Auszug der Liste typischer DBI-Fehlernummern. Eine detailliertere Liste liegt im d.3 Serverprogrammverzeichnis in der Datei **JAM_DBI_ERR.TXT**.

32878 // Variablenname nicht vorhanden
32897 // Syntax error
53250 // Nicht an der DB angemeldet
53251 // Bereits an der DB angemeldet
53253 // DB-Login verweigert
53254 // falsche Parameter gesetzt
53255 // keine weiteren Datensätze vorhanden
53256 // wegen Datenbankfehler abgebrochen
53258 // Cursor existiert nicht
53264 // SQL parse error
53271 // keine Verbindung zur DB

`@dmengerrcode`

Numerischer Returncode des DB-Servers (siehe Dokumentation Statuscodes von Oracle, Microsoft SQL oder Pervasive SQL)

`@dmrowcount`

Anzahl der vom letzten SQL-Befehl betroffenen Datensätze

`SELECT`

Anzahl der an d.3 gelieferten Datensätze

`INSERT, UPDATE, DELETE`

Anzahl der eingefügten, aktualisierten bzw. gelöschten Datensätze

4 Colon preprocessing

Durch das Voranstellen des „:" vor den Namen einer JPL-Variablen wird an dieser Stelle der aktuelle Wert der Variablen eingesetzt. Man kann somit den Wert einer JPL-Variablen an Stellen referenzieren, wo es sonst nicht möglich wäre, z.B. bei Ausgabe des Wertes der Variablen auf dem Bildschirm (oder d.3 logviewer).

```
vars  h_Zaehler
h_Zaehler = 123
msg emsg „Versuch 1: Der Inhalt des Zaehler ist
h_Zaehler“
msg emsg „Versuch 2: Der Inhalt des Zaehler ist
:h_Zaehler“
```

Ausgabe:

Versuch 1	Der Inhalt des Zaehler ist h_Zaehler
Versuch 2	Der Inhalt des Zaehler ist 123

5 Colon-plus processing

„Colon-plus processing" wird bei Verwendung des Wertes von JPL-Variablen bei Datenbankbefehlen verwendet. Hier gibt es zahlreiche Regeln. Der einfachste Unterschied zwischen „:" und „:+" sei an folgendem Beispiel verdeutlicht:

```
vars  h_DocIdTesten(8)
h_DocIdTesten = “A0001234”

// Versuch1:
DBMS SQL SELECT dok_dat_feld_10              \
FROM  :D3_TABELLE_FIRMEN_SPEZIFISCH    \
WHERE doku_id =  h_DocIdTesten

// Versuch2:
DBMS SQL SELECT dok_dat_feld_10              \
FROM  :D3_TABELLE_FIRMEN_SPEZIFISCH    \
WHERE doku_id = : h_DocIdTesten

// Versuch3:
DBMS SQL SELECT dok_dat_feld_10              \
FROM  :D3_TABELLE_FIRMEN_SPEZIFISCH    \
WHERE doku_id = :+ h_DocIdTesten
```

Ausgabe:

Versuch 1	`WHERE doku_id = h_DocIdTesten`
	liefert einen SQL-Fehler, da `h_DocIdTesten` unbekannt
Versuch 2	`WHERE doku_id = A0001234`
	liefert einen SQL-Fehler, da `A0001234` keine Zeichenkette
Versuch 3	`WHERE doku_id = 'A0001234'`
	ist korrekt

Die Zeichen ":+" expandieren also im einfachsten Fall den Wert einer JPL-Variablen in eine durch " begrenzte Zeichenkette – genau so, wie der SQL den Syntax benötigt, sofern eine Spalte des Datentyps CHAR referenziert wird.

Bei Referenz eines numerischen Wertes verwendet man dagegen nur „:“.

```
vars    zahl_testen(8)

zahl_testen = 4711

// Versuch1:
DBMS SQL SELECT doku_id             \
FROM    :D3_TABELLE_FIRMEN_SPEZIFISCH   \"
WHERE   dok_dat_feld_80 = zahl_testen

// Versuch2:
DBMS SQL SELECT dok_dat_feld_80         \
FROM    :D3_TABELLE_FIRMEN_SPEZIFISCH   \
WHERE   dok_dat_feld_80 = :zahl_testen

// Versuch3:
DBMS SQL SELECT dok_dat_feld_80         \
FROM    :D3_TABELLE_FIRMEN_SPEZIFISCH   \
WHERE   dok_dat_feld_80 = :+zahl_testen
```

Versuch 1	`WHERE dok_dat_feld_80 = zahl_testen`
	liefert einen SQL-Fehler, da `zahl_testen` unbekannt
Versuch 2	`WHERE dok_dat_feld_80 = 4711`
	ist korrekt
Versuch 3	`WHERE dok_dat_feld_80 = ‘4711’`
	liefert einen SQL-Fehler, da `’4711’` kein numerischer Wert ist

Teil VI
Verbindungen zu anderen Datenquellen einrichten

Teil VI Verbindungen zu anderen Datenquellen einrichten

1 Verbindung zu externen Datenquellen

Von einer d.3 Hook-Funktion aus kann man eine Verbindung zu jeder anderen Datenbank aufmachen, sofern diese per ODBC erreichbar ist. Man muss also zuerst per ODBC-Administrator eine lauffähige Datenquelle (datasource) eingerichtet haben.

- Installieren Sie die Datenbank-Client-Software.
- Richten Sie die ODBC-Treiber ein.
- Legen Sie die ODBC-Datasource an.
- Bauen Sie die Verbindung auf (`DECLARE CONNECTION`).
- Greifen Sie auf die Daten zu (`WITH CONNECTION`).
- Schließen Sie die Verbindung zu externen Datenbank wieder (`CLOSE CONNECTION` – siehe unten).

Beachten Sie die Performance der Datenbank!

2 Verbindungsaufbau

```
DBMS DECLARE <session_name> CONNECTION FOR    \
USER                '<Benutzername>'          \
PASSWORD            '<Passwort>'              \
DATASOURCE    '<Datasource>'              [\]
[CONN_STRING  '<Connection-String>']
```

`session_name:` Name für die Datenbankverbindung

frei vergebbar, jedoch heißt die d.3 Sitzung `odbc_session`, das heißt, dieser Name ist bereits vergeben.

3 Verbindungsabbau

```
DBMS CLOSE CONNECTION <session_name>
```

```
DBMS CLOSE CONNECTION sap_connection
```

Hat man in einer Hook-Funktion über JAM/ Panther eine eigene Datenbanksitzung eröffnet, dann muss man allen Datenbankbefehlen, die auf die eigene DB-Sitzung ausgerichtet sind, die `WITH CONNECTION`-Klausel voranstellen.

```
DBMS DECLARE sap_db CONNECTION FOR \
USER              'sap_user'     \
PASSWORD          'geheim'       \
DATASOURCE        '<Datasource>'   [\]
[CONN_STRING      '<Connection-String>']

DBMS WITH CONNECTION sap_db ALIAS var_1
DBMS WITH CONNECTION sap_db SQL SELECT spalte_1 FROM
tabelle …
DBMS WITH CONNECTION sap_db ALIAS
```

Teil VII
Massenverarbeitung von Dokument-Metadaten

Teil VII Massenverarbeitung von Dokument-Metadaten

Administrative Operationen und Funktionen zur Massenverarbeitung von Dokument-Metadaten.

Die im Folgenden beschriebenen Funktionen und Techniken berücksichtigen keinerlei höhere Logik von d.3.

Insbesondere bei den schreibenden Operationen sollte beachtet werden, dass keine Rechte geprüft werden, keine Validierungen vorgenommen werden, keine Hooks ausgeführt werden, keine Datei-Lokalisationen im Dok-Baum angepasst werden und keine Änderungshistorie gepflegt wird.
Die hier aufgelisteten Funktionen verstehen sich als Ersatz zu direkten SQL-Operationen auf den d.3 Dokument-Tabellen und sind keine Alternative zu den offiziellen d.3 APIs (siehe `d3api.pdf` bzw. `d3serverapi.pdf`).

Im Gegensatz zu direktem SQL-Zugriff wird bei diesen Funktionen der Cache genutzt und konsistent gehalten (Prozess-interner Cache und Membase).

1 Eigenschaftsfelder

Die im Folgenden beschriebenen Funktionen arbeiten mit Namen von Eigenschaften (Propertys).

In dieser Liste sind die Eigenschaftsfelder aufgelistet, die zur Verfügung stehen.

Name	Datenbank-Spalte	Inhalt/ Zweck
doc_type_short	firmen_spezifisch.kue_dokuart	Dokumentart-Id (Kürzel)
doc_number	phys_datei.zeich_nr	Dokumentennummer
doc_status	phys_datei.logi_verzeichnis	Dokumentenstatus ('Be', 'Pr', 'Fr' und 'Ar')
user	phys_datei.bearbeiter	Bearbeiter
owner	phys_datei.besitzer	Besitzer
doc_id	firmen_spezifisch.doku_id	Documenten-Id
doc_extenstion	phys_datei.datei_erw	Dateierweiterung der aktuellen Nutzdatei
doc_size	phys_datei.size_in_byte	Dateigröße der aktuellen Nutzdatei
doc_text	phys_datei.text	Bemerkung (alle Zeilen des Bemerkugsfeldes)
date_create	phys_datei.datum_einbring	Erstelldatum
date_access	phys_datei.dat_letzter_zugr	Datum des letzten Zugriffs
date_upd_file	phys_Datei.last_update_file	Datum der letzten Änderung
date_upd_attrib	firmen_spezifisch.last_update_attr	Datum der letzten Attributänderung
doc_field[X]	firmen_spezifisch.dok_dat_feld_X	Attribut-Feld Nr. X (1-89)

2 Dokument-Metadaten

2.1 get_doc_property(docID, name)

Funktion `get_doc_property(docId, name)`

Gibt den Wert einer einzelnen Eigenschaft des jeweiligen Dokumentes zurück.

Da diese Funktion den d.3 internen Cache benutzt, führt nicht jeder Aufruf zu einem DB-Zugriff. Diese Funktion ist nicht für mehrzeilige Eigenschaften wie das Bemerkungsfeld oder Mehrfachattribute nutzbar (siehe `get_doc_text()` und `get_doc_mult_attrib()`).

Parameter:

`docId`	d.3 Dokumenten-ID
`name`	Name der Eigenschaft (s.o.)

```
hName = get_doc_property(hDocId, "doc_field[1]")
```

Äquivalente SQL-Abfrage:

```
DBMS ALIAS hName
DBMS SQL SELECT dok_dat_feld_1 FROM firmen_spezifisch
WHERE doku_id = :+hDocId
DBMS ALIAS
```

2.2 get_doc_text(docID, index)

Funktion `get_doc_text(docId, index)`

Gibt den Wert einer Zeile des Bemerkungsfeldes zurück.

Parameter:

`docId`	d.3 Dokumenten-ID
`index`	Index des Zeile (Gültige Werte sind 0 bis 3)

```
hText[1] = get_doc_text(hDocId, 0)
hText[2] = get_doc_text(hDocId, 1)
hText[3] = get_doc_text(hDocId, 2)
hText[4] = get_doc_text(hDocId, 3)
```

2.3 get_doc_mult_attrib(docID, fieldNo, rowNo, jplDestArray)

Funktion `get_doc_mult_attrib(docId, fieldNo, rowNo, jplDestArray)`

Gibt den Wert für eine Zeile eines Mehrfachattributes zurück oder schreibt alle Zeilen in ein globales JPL-Array.

Parameter:

`docId`	d.3 Dokumenten-ID
`fieldNo`	Nummer des Attribut-Feldes (Gültige Werte sind 60-69)
`rowNo`	Zeile des Attribut-Feldes. Ist dieser Wert gleich -1 werden alle Zeilen des Mehrfach-Attributes in das Ziel-Array (`jplDestArray`) geschrieben.
`jplDestArray`	Name eines globalen JPL-Arrays, in welches die Zeilen des Attributfeldes geschrieben werden sollen. Das Array muss in einer Datei definiert sein, die beim Prozess-Start geladen wird, z.B. eine der Dateien in denen die Hooks definiert sind (HOOK_JPL_FILES_CUSTOMER).

```
hMultVal = get_doc_mult_attrib(hDocId, 60, 1)
```

Äquivalente SQL-Abfrage:

```
DBMS ALIAS hMultVal
DBMS SQL SELECT value_char \
FROM firm_spez_mult_val \
WHERE doku_id = :+hDocId \
AND field_no = 60 AND row_number = 1
DBMS ALIAS
```

2.4 set_doc_property(docId, name, value)

Funktion `set_doc_property(docId, name, value)`

Setzt den Wert einer einzelnen Eigenschaft des jeweiligen Dokumentes.

Die entsprechende Änderung wird nicht direkt geschrieben, sondern erst beim Aufruf von `upd_doc_db_data()`.

Diese Funktion ist nicht für mehrzeilige Eigenschaften wie das Bemerkungsfeld oder Mehrfachattribute nutzbar (siehe `set_doc_text()` und `set_doc_mult_attrib()`).

Parameter:

`docId`	d.3 Dokumenten-ID

name	Name der Eigenschaft (s.o.)
value	Neuer Wert für die jeweilige Eigenschaft

```
call set_doc_property(hDocId, "doc_field[1]", hFirstName)
call set_doc_property(hDocId, "doc_field[2]", hLastName)
if( !upd_doc_db_data(hDocId) )
call api_log_error("Could not write changes on document
:hDocId to db!")
```

Äquivalente SQL-Abfrage:

```
DBMS SQL UPDATE firmen_spezifisch \
SET dok_dat_feld_1 = :+hFirstName, dok_dat_feld_2 =
:+hLastName \
WHERE doku_id = :+hDocId
```

2.5 set_doc_text(docId, index, value)

Funktion `set_doc_text(docId, index, value)`

Setzt den Wert einer Zeile des Bemerkungsfeldes.

Die entsprechende Änderung wird nicht direkt geschrieben, sondern erst beim Aufruf von `upd_doc_db_data()`.

Parameter:

docId	d.3 Dokumenten-ID
index	Index der Zeile (Gültige Werte sind 0 bis 3)
value	Neuer Wert für die jeweilige Eigenschaft

```
call set_doc_text(hDocId, 0, hText[1])
call set_doc_text(hDocId, 1, hText[2])
call set_doc_text(hDocId, 2, hText[3])
call set_doc_text(hDocId, 3, hText[4])
if( !upd_doc_db_data(hDocId) )
...
```

2.6 set_doc_mult_val(docId, fieldNo, rowNo, value)

Funktion `set_doc_mult_val(docId, fieldNo, rowNo, value)`

Setzt den Wert einer einzelnen Eigenschaft des jeweiligen Dokumentes.

Die entsprechende Änderung wird nicht direkt geschrieben, sondern erst beim Aufruf von `upd_doc_db_data()`.

Diese Funktion ist nicht für mehrzeilige Eigenschaften wie das Bemerkungsfeld oder Mehrfachattribute nutzbar (siehe `set_doc_text()` und `set_doc_mult_attrib()`).

Parameter:

`docId`	d.3 Dokumenten-ID
`fieldNo`	Nummer des Attribut-Feldes (Gültige Werte sind 60 bis 69)
`rowNo`	Zeile des Attribut-Feldes. Ist dieser Wert gleich "-1", werden alle Zeilen des Mehrfach-Attributes geleert. Ist dieser Wert gleich "0", wird der neue Wert in einer neuen Zeile hinter der bisher letzten Zeile eingefügt.
`value`	Neuer Wert für die jeweilige Zeile des Attributes

```
call set_doc_mult_val(hDocId, 60, 1, hMultVal)
if( !upd_doc_db_data(hDocId) )
...
```

2.7 upd_doc_db_data(docId)

Funktion upd_doc_db_data(docId)

Führt die eigentliche Update-Operation für die zuvor geänderten Eigenschaften durch.

Parameter:

`docId`	d.3 Dokumenten-ID

Rückgabe:

"0" im Fehlerfall

"1" im Erfolgsfall

Tritt ein SQL-Fehler auf, wird der entsprechende Fehlercode in die globale Variable `dbi_retcode` eingetragen.

Außerdem wird `dbi_rowcount` im Erfolgsfall auf "1" gesetzt und im Fehlerfall auf "0".

```
call set_doc_mult_val(hDocId, 60, 1, hMultVal)
if( !upd_doc_db_data(hDocId) )
...
```

3 Massenupdate von Dokument-Metadaten

3.1 queue_upd_doc_db_data(docId)

Funktion `queue_upd_doc_db_data(docId)`

Diese Funktion merkt ein Dokument für ein Update vor und ist ein Ersatz für `upd_doc_db_data()`.

Wird diese Funktion aufgerufen, ist ein späterer Aufruf von `finalize_upd_doc_db_data()` nötig.

Ist die intern festgelegte maximale Puffergröße erreicht, werden die Änderungen automatisch in die Datenbank geschrieben; der letzte Block an Dokumenten wird beim Aufruf von `finalize_upd_doc_db_data()` geschrieben.

Wenn auf mehreren Dokumenten gleichartige Änderungen durchgeführt werden (dieselben Spalten geändert werden), ist diese Funktion in der Regel performanter als `upd_doc_db_data()`.
Sind die neuen Werte bei einer Änderung fix (müssen nicht noch pro Dokument programmatisch bestimmt werden), kann u.U. einfacher mit `docSearchAddReplacementParam()` gearbeitet werden.

Parameter:

`docId`	d.3 Dokumenten-ID

Rückgabe:

"0" im Fehlerfall

"1" im Erfolgsfall

Tritt ein SQL-Fehler auf, wird der entsprechende Fehlercode in die globale Variable `dbi_retcode` eingetragen.

Außerdem wird `dbi_rowcount` im Erfolgsfall auf "1" gesetzt und im Fehlerfall auf "0".

```
call set_doc_mult_val(hDocId, 60, 1, hMultVal)
if( !upd_doc_db_data(hDocId) )
...
```

3.2 finalize_upd_doc_db_data(docId)

Funktion `finalize_upd_doc_db_data(docId)`

Diese Funktion muss am Ende eines Massenupdate-Vorgangs einmalig aufgerufen werden, um dessen Ende zu kennzeichnen (siehe queue_upd_doc_db_data).

4 Dokument-Suche und Massenverarbeitung

4.1 docSearchCreate (user)

Funktion `docSearchCreate (user)`

Erstellt einen internen Such-Handle.

Diese Funktion muss vor jeder der folgenden Funktionen aufgerufen werden.

Auf jeden Aufruf dieser Funktion muss später genau ein Aufruf von `docSearchDestroy()` folgen.

Parameter:

`user`	Kennung des Benutzers, für den die Suche ausgeführt werden soll. Ist dieser Wert leer oder auf "Master" gesetzt, wird später keine Rechteprüfung durchgeführt.

4.2 docSearchSetSearchtextExpression(searchtext_expression)

Funktion `docSearchSetSearchtextExpression (searchtext_expression)`

Anfrage für die Volltextsuche (dieser Wert wird an d.search übergeben).

Die Nutzung dieser Funktion für eine Suche schließt die Nutzung von `docSearchSetSourcePool()` bei derselben Suche aus.

4.3 docSearchSetIdsFromArray(globalJplArray)

Funktion `docSearchSetIdsFromArray(globalJplArray)`

Übernimmt die DocIds in dem angegebenen globalen JPL-Array als Quelle für die folgende Suche (ID-List-Suche).

Parameter:

globalJplArray	Der Name eines JPL-Arrays, welches gültige d.3 Dokumenten-IDs enthält. Dieses Array muss in einer Datei deklariert sein, die beim Start des Prozesses geladen wird; z.B. eine der Dateien in denen die Hooks definiert sind (**HOOK_JPL_FILES_CUSTOMER**).

4.4 docSearchAddSearchParam(name, filterOperator, value)

Funktion `docSearchAddSearchParam(name, filterOperator, value)`

Gibt eine Vergleichsoperation an, die bei der Suche als Teil der Filter/der Where-Klausel dienen soll.

Die Angaben für verschiedene Eigenschaftsfelder werden mit logischem "UND" verknüpft. Mehrfachnennungen eines Feldes werden mit logischem "ODER" verknüpft.

Parameter:

name	Name der Dokument-Eigenschaft welche überprüft werden soll (s.o.).
filterOperator	Operator für die Abfrage (in C- oder SQL-Notation); z.B. `"=="`, `"<>"`, `"LIKE"`, `"NOT LIKE"`, `"<"`, `">="`
value	Vergleichswert

4.5 docSearchAddSortParam(name, direction)

Funktion `docSearchAddSortParam(name, direction)`

Parameter:

name	Name der Dokument-Eigenschaft, nach welcher die Treffer sortiert werden sollen.
direction	"1" für aufsteigende Sortierung, sonst absteigende Sortierung

4.6 docSearchAddReplacementParam(name, pattern, value)

Funktion `docSearchAddReplacementParam(name, pattern, value)`

Definiert eine bedingte Ersetzungsregel für alle Treffer der jeweiligen Suche. Die Ersetzungen werden dann mit einem Massenupdate vorgenommen.

Parameter:

`name`	Name der Dokument-Eigenschaft, in der eine Ersetzung vorgenommen werden soll.
`pattern`	Muster, dem der alte Wert entsprechen muss, damit die Ersetzung vorgenommen wird (SQL-Wildcards sind möglich)
`value`	Der neue Wert, welcher für die Eigenschaft gesetzt werden soll.

4.7 docSearchNext()

Funktion `docSearchNext()`

Gibt "1" zurück, wenn es ein weiteres Dokument gibt, andernfalls "0".

Außerdem wird die interne Referenz auf den nächsten Treffer gesetzt (siehe `docSearchCurrentId`).

Diese Funktion ist nicht in Kombination mit `docSearchExecute()` nutzbar.

4.8 docSearchCurrentId()

Funktion `docSearchCurrentId()`

Gibt die Dokumenten-Id des aktuellen Treffers zurück.

Die aktuelle DocId kann dann für Aufrufe von Server-API-Funktionen oder der oben genannten (Massen-)Updatefunktionen genutzt werden.

Die Nutzung dieser Funktion ist nur erlaubt, wenn der vorherige Aufruf von `docSearchNext()` "1" zurückgegeben hat.

4.9 docSearchExecute()

Funktion `docSearchExecute()`

Führt die Suche mit den vorher gesetzten Optionen aus.

Diese Funktion kann genutzt werden, wenn keine weitere Interaktion mit den Suchtreffern gewünscht ist (z.B. weil die Suche zum Update mit Hilfe von `docSearchAddReplacementParam()` dient oder zum Cache-Warming genutzt wird).

Diese Funktion ist nicht in Kombination mit `docSearchNext()` nutzbar.

4.10 docSearchDestroy()

Funktion `docSearchDestroy()`

Bereinigt den internen Such-Handle.

Zu jedem Aufruf von `docSearchCreate()` muss es einen Aufruf dieser Funktion geben.

5 Beispiele

5.1 Szenario 1

Änderung des Lieferanten-Namens zu "Meier AG" (`dok_dat_feld_1`) bei allen Dokumenten in den Dokumenten-Art "DRECH" und "DLSCH" bei denen die Lieferanten-Nummer gleich "L123456" ist (`dok_dat_feld_2`)

Bisheriges SQL-Statement das ersetzt werden soll:

```
DBMS SQL UPDATE firmen_spezifisch \
          SET dok_dat_feld_1 = 'Meier AG' \
          WHERE (kue_dokuart = 'DRECH' OR kue_dokuart = 'DLSCH')
\
          AND dok_dat_feld_2 = 'L123456'
```

Äquivalenter neuer Code:

```
 call docSearchCreate()
 call docSearchAddSearchParam("doc_type_short", "==", "DRECH")
 call docSearchAddSearchParam("doc_type_short", "==", "DLSCH")
 call docSearchAddSearchParam("doc_field[2]", "==", "L123456")
 call docSearchAddReplacementParam("doc_field[1]", "*", "Meier
GmbH")
 call docSearchExecute()
 call docSearchDestroy()
```

5.2 Szenario 2

Erhöhen des Wertes in Feld 80 um 1, wenn der Wert in Feld 1 mit "TEST " anfängt

Aufruf der Ersetzung:

```
call docSearchCreate()
call docSearchAddSearchParam("doc_field[1]", "like",
"TEST *")
while( docSearchNext() )
{
  hDocId = docSearchCurrentId()
  oldValue = get_doc_property(hDocId, "doc_field[80]")
  call set_doc_property(hDocId, "doc_field[80]",
oldValue+1)
  call queue_upd_doc_db_data(hDocId)
}
call finalize_upd_doc_db_data()
call docSearchDestroy()
```

Teil VIII
Hook-Funktionen in Java

Teil VIII Hook-Funktionen in Java

1 Einbindung von Java-Code in Hook-Funktionen

In d.3 Hook-Funktionen, erstellt in der Skriptsprache JPL, ist es möglich,

- eigenen, nativen Code zu integrieren, durch die Möglichkeit DLL's / Shared Libs zu laden und die enthaltenen Funktionen direkt auszuführen,
- über die Microsoft Windows COM-Schnittstelle auf andere Prozesse zu zuzugreifen, um Daten auszutauschen.

Die Integration eigener Java-Module war bisher nicht möglich, weil der Zugriff aus der Java-Welt auf d.3 Objekte wie z.B. die d.3 Attributfelder nicht möglich war.

Mit der d.3 Version 6.3 ist diese Möglichkeit nun dadurch geschaffen worden, dass man eine eigene Java-Klasse von d.3 laden lassen und deren Methoden direkt als Hook-Funktionen in der d.3 Konfiguration eintragen kann.

Es werden nur die Java-Hooks unterstützt, die im Konfigurationsmodul des d.3 admins eingestellt werden können. Insbesondere Werteauswahl-, Wertevalidierungs- und Dokumentklassenfilter-Hooks können nicht in der Programmiersprache Java umgesetzt werden.

2 Konfiguration für den Aufruf von Java-Hook-Funktionen

Folgende Angaben müssen in der d.3 Konfiguration gemacht werden, damit der Aufruf einer Java-Hook-Funktionen funktioniert:

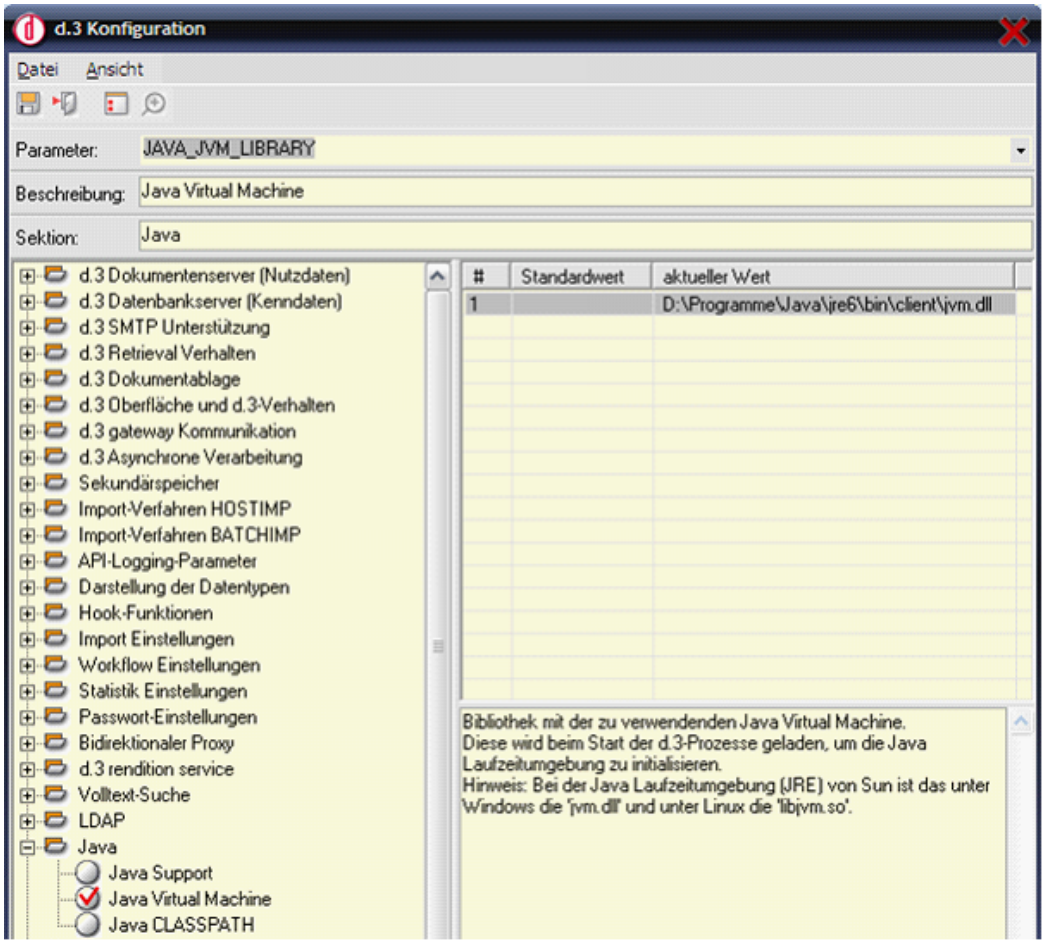
1. Der d.3 Java Support muss aktiviert sein.

   Siehe dazu Abschnitt **Java** im Konfigurationsmodul d.3 Config der d.3 Administration.

2. Der Pfad zum Startmodul der zu verwendenden Java Laufzeitumgebung (JRE) muss im Abschnitt **Java** unter **Java Virtual Machine** eingetragen werden.

   Bei der JRE von Sun ist das unter Microsoft Windows die `jvm.dll` und unter Linux die `libjvm.so` im Unterverzeichnis `jre\bin\client` der Java Installation.

Es wird empfohlen, die JRE von Sun ab Version 1.6 (Java 6) zu verwenden.

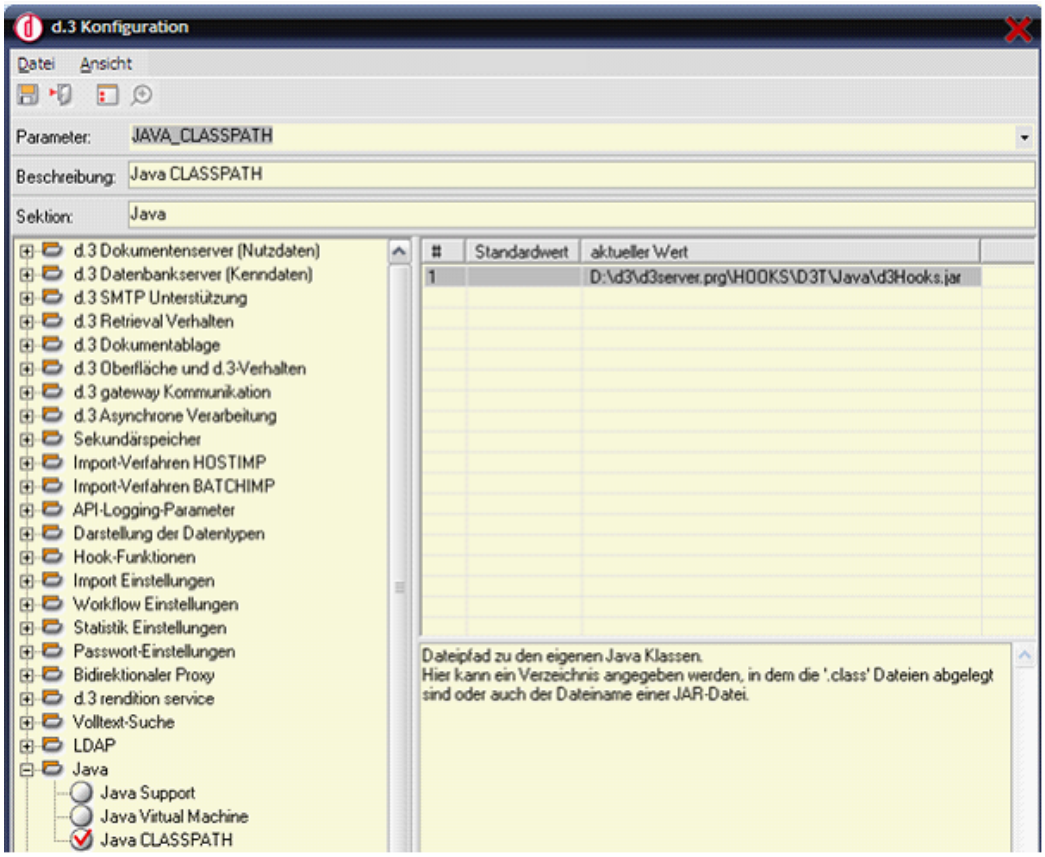


d.3 Konfiguration Java-Zweig JVM

Der d.3 presentation server installiert eine JRE. Diese ist unterhalb des Presentation Server Basisverzeichnisses zu finden.

3. Der Pfad zu den eigenen Java Klassen muss unter **Java CLASSPATH** angegeben werden.

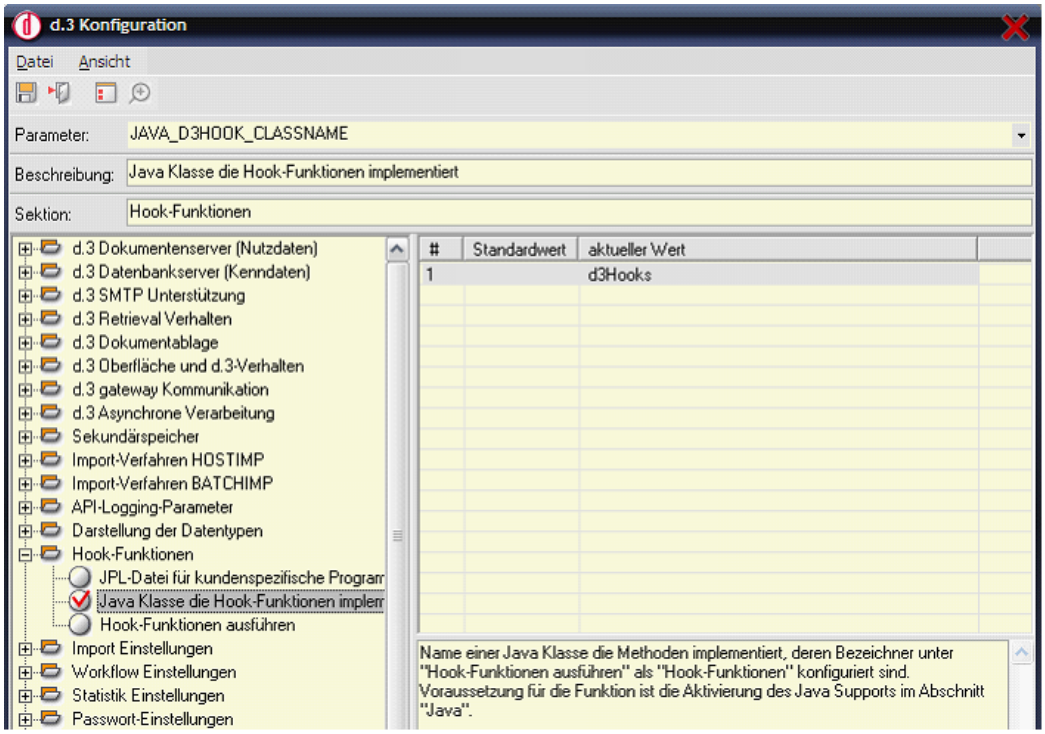
Hier kann ein Verzeichnis angegeben werden, in dem die `.class` Dateien abgelegt sind oder Pfad+Dateiname einer `JAR`-Datei, die eigene Java Klassen enthält. Auch mehrere Pfadangaben sind semikolongetrennt möglich.



d.3 Konfiguration Java-Zweig Classpath

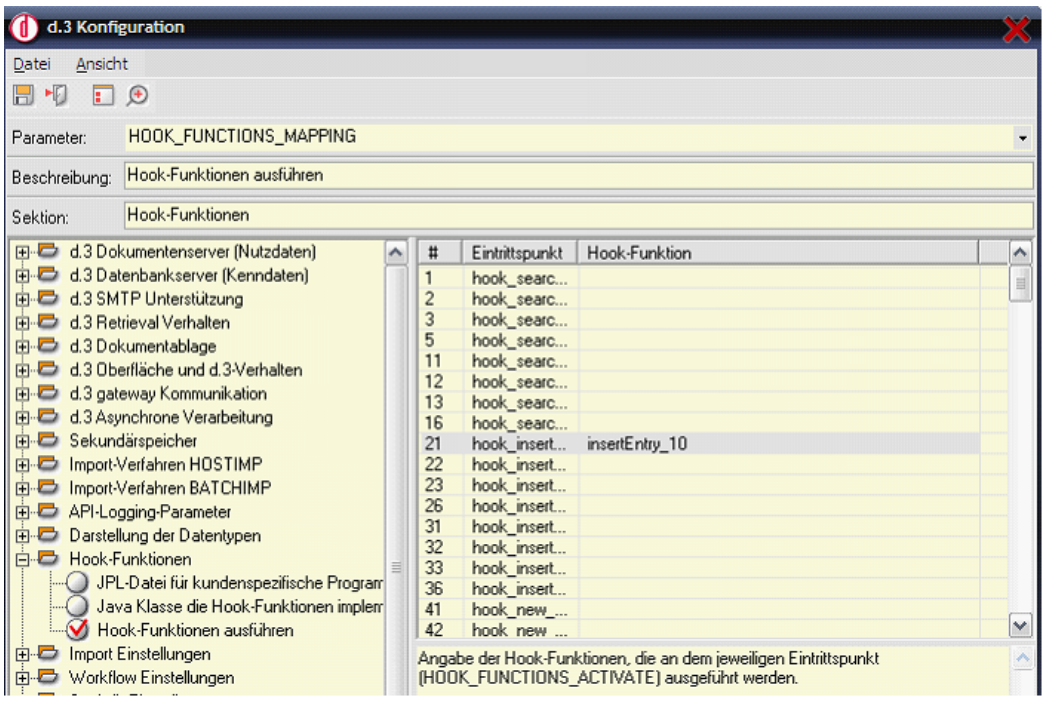
4. Im Abschnitt **Hook-Funktionen** der Konfiguration ist der Name einer eigenen Java Klasse einzutragen unter **Java Klasse die Hook-Funktionen implementiert**.

Diese Klasse muss über den bei 3. angegebenen **Java CLASSPATH** für den Java Interpreter zu finden sein.



d.3 Konfiguration Java-Hook-Klasse

5. Nun können im gleichen Abschnitt unter **Hook-Funktionen ausführen** ein oder mehrere Methoden aus der Java Klasse als Hook-Funktionen eingetragen werden.



Einsprung-Punkt-Zuweisung

Beim Start der d.3 Prozesse wird die konfigurierte Java Klasse (Punkt 4) geladen und die enthaltenden Methoden werden auf die beiden folgenden Bedingungen überprüft:

a) besitzt die Methode die von d.3 erwarteten Parameter `FieldInterface` und `String-Array` (s.u.)

b) ist der Methodenname als Hook-Funktion konfiguriert

Wenn beide Bedingungen zutreffen, dann wird die Java-Methode für den Aufruf registiert.

Nach erledigter Konfiguration müssen also die d.3 Prozesse einmal durchgestartet werden.

Wenn der Java-Support aktiviert ist, aber für eine konfigurierte Hook-Funktion keine Java-Methode existiert, wird die Hook-Funktion wie zuvor als JPL-Funktionsaufruf registiert.

JPL- und Java-Hook-Funktionen können also gemischt benutzt werden.

Existiert für eine Hook-Funktion eine gleichnamige Java-Methode und auch eine entsprechende JPL-Funktion, wird die Java-Methode aufgerufen und die JPL-Funktion nicht. Aufruf von Java- und JPL-Funktion für den gleichen Hook-Einsprungpunkt ist nicht vorgesehen.

3 Erstellen von Java-Hook-Funktionen

d.3 erwartet folgende Signatur für eine Java-Methode die als Hook-Funktion aufgerufen werden soll:

```
public int hook_funktions_name (FieldInterface f, String params[])
```

Über den ersten Parameter `FieldInterface` ist der Zugriff auf d.3 Objekte möglich.

Im zweiten Parameter String-Array `params` werden die d.3 Parameterwerte der Hook-Funktion übergeben und zwar in der Reihenfolge, wie sie für jede Funktion im Hook-Programmierhandbuch im Kapitel Hook-Funktionen im Einzelnen beschrieben sind.

Die Hook-Funktion `hook_insert_entry_10` zum Beispiel wird folgendermaßen beschrieben:

```
hook_insert_entry_10 (user, doctype_short)
```

Die Eingabeparameter sind:

`user`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart des zu importierenden Dokuments

Im Sting-Array `params` wird somit übergeben:

params[0]	Wert von user
params[1]	Wert von doc_type_short

params[0] = Wert von user

params[1] = Wert von doc_type_short

Das FieldInterface ist implementiert in der JAR-Datei pro5.jar.

Diese befindet sich im d.3 Server-Programmverzeichnis.

Die hier enthaltenen Klassen werden im Java-Modul importiert per Aufruf:

```
import com.prolifics.jni.*;
```

Bei der Kompilierung muss die JAR-Datei pro5.jar im Classpath angegeben sein.

Beispiel einer d.3 Hook Java-Klasse.
Datei d3Hooks.java:

```
import com.prolifics.jni.*;

public class d3Hooks
{
   public int insertEntry_10( FieldInterface f,String params[] )
   {
      CFunctionsInterface c  = f.getCFunctions();
      DMFunctionsInterface d = f.getDMFunctions();

      // d.3-Log Meldung per JPL-Server-API-Funktion
      c.sm_jplcall( "api_log_info('in java insertEntry_10')" );

      // d.3 Attributwert zuweisen
      c.sm_i_putfield( "dok_dat_feld", 1, "wert aus insertEntry_10 " );

      // Datenbankzugriff über d.3-Datenbankschnittstelle
      d.dm_dbms( "SQL INSERT INTO tabelle VALUES('wert')" );
      return 0;
   }// end of insertEntry_10
}// end of d3Hooks
```

Kommandozeilenaufruf zur Kompilierung der Java-Klasse:

```
..\Java\jdk\bin\javac -classpath ..\pro5.jar d3Hooks.java
```

Das resultierende Klassenmodul d3Hooks.class muss nun über den in d.3 konfigurierten Java Classpath zu finden sein, damit es geladen werden kann.

Die Java-Hook-Klassen müssen mit derselben Java-Version der verwendeten Java-Runtime kompiliert worden sein.

3.1 Zugriff auf d.3 Variablen und Funktionen

Über das `FieldInterface` wird per Aufruf `getCFunctions()` eine Referenz auf das `CFunctionsInterface` ermittelt. Darüber kann auf die internen, nativen Funktionen der von d.3 verwendeten Panther Funktionsbibliothek zugegriffen werden.

```
CFunctionsInterface  c = f.getCFunctions();
```

Das `CFunctionsInterface` stellt u.a. folgende Methoden zur Verfügung:

Lesen von Variablenwerten	
`a1 = Name einer Variablen;`	
`a2 = Arrayindex;`	z.B. `dok_dat_feld`, falls es sich um eine Array-Variable handelt wird die `sm_i_` Variante der Funktion benutzt
Zeichenketten	
`String sm_n_fptr (String a1);`	
`String sm_i_fptr (String a1, int a2);`	
Zahlen	
`int sm_n_intval(String a1);`	
`int sm_i_intval(String a1, int a2);`	
Kommazahlen	
`double sm_n_dblval (String a1);`	
`double sm_i_dblval (String a1, int a2);`	
Schreiben von Variablenwerten	
`a3 = der zu schreibende Wert`	
Zeichenketten	
`int sm_n_putfield (String a1, String a2);`	
`int sm_i_putfield (String a1, int a2, String a3);`	
Zahlen	

Lesen von Variablenwerten	
`int sm_n_itofield (String a1, int a2);`	
`int sm_i_itofield (String a1, int a2, int a3);`	
JPL-Funktion ausführen	

a1 = JPL-Funktionsaufruf inkl. der Parameter;	z.B. `api_log_info('Parameterwert')`
	mit Rückgabewert Zeichenkette:
	`String sm_sjplcall(String a1);`
	mit Rückgabewert Zahl:
	`int sm_jplcall(String a1);`
	mit Rückgabewert Kommazahl:
	`double sm_djplcall(String a1);`

3.2 Zugriff auf die Datenbank

Über das `FieldInterface` wird per Aufruf `getDMFunctions()` eine Referenz auf das `DMFunctionsInterface` ermittelt. Darüber kann auf die d.3 Datenbank-Schnittstelle zugegriffen werden.

```
DMFunctionsInterface d = f.getDMFunctions();
```

Das `DMFunctionsInterface` stellt u.a. die nachfolgenden Methoden zur Verfügung.

3.2.1 Ausführen eines Datenbank-Kommandos

```
int dm_dbms(String a1);
int dm_dbms_noexp(String a1);    // ohne Auswertung des Panther Doppelpunkt-Operators
```

3.2.2 Kontrolle, ob ein DB-Cursor existiert, bzw. noch gültig ist

```
int dm_cursor_consistent(String a1);
```

3.2.3 Kontrolle, ob eine Datenbankverbindung existiert bzw. noch gültig ist

```
int dm_is_connection(String a1);
```

4 Aufruf von Java-Funktionen aus JPL

Nur die Standard d.3 Hook-Funktionen können durch Java-Hooks ersetzt werden. Dies umfasst insbesondere folgende Bereiche nicht:

- Wertemengen- und Plausbilitätshooks
- Eigene JPL-Skripte

Um auch in diesen Bereichen die Entwicklung in Java zu ermöglichen, kann aus JPL Java-Code aufgerufen werden.

Dies erfolgt über die d.3 Server API Funktionen `java_add_param`, `java_clear_param` und `java_call_static` (siehe d.3 Server API Dokumentation).

Diese Funktionen erlauben es, aus JPL eine statische Methode einer Java-Klasse aufzurufen.

Als Beispiel wird hier ein Wertemengen-Hook in Java geschrieben.

1. Schritt

Zunächst muss aus dem JPL-Code heraus in den Java-Code gesprungen werden.

```
proc test_value_set_hook(repos_id, user, doc_type_short,
row_no)
{
    // Relevante Parameter an Java-Funktion übergeben
    call api_function("java_add_param", repos_id)
    call api_function("java_add_param", user)
    call api_function("java_add_param", doc_type_short)
    call api_function("java_add_param", row_no)

    // Java-Funktion aufrufen
    call api_function("java_call_static", "com/dvelop/
test/JavaHooks", "valueSetHook")
}
```

2. Schritt

Danach kann im Java-Code die Logik hinterlegt werden.

```
public static int valueSetHook(FieldInterface field,
String[] params)
{
    // Parameter auslesen (Reihenfolge im JPL-Code
beachten)
    String reposID = params[0];
    String user = params[1];
    String docTypeShort = params[2];
    String rowNo = params[3];

    // Parameter auswerten (hier nur beispielhaft loggen)
    CFunctionsInterface c = field.getCFunctions();
    c.sm_jplcall("api_log_info('valueSetHook(" + reposID
+ ", "
                + user + ", " + docTypeShort + ", " +
rowNo + ")')");

    // Wertemenge füllen (hier nur beispielhaft mit
festen Werten)
    c.sm_i_putfield("d3server_repos_id_allowed", 1,
reposID);
    c.sm_i_putfield("d3server_value_char_allowed", 1,
"Max Schmitt");
    c.sm_i_putfield("d3server_repos_id_allowed", 2,
reposID);
    c.sm_i_putfield("d3server_value_char_allowed", 2,
"Heike Mustermann");

    return 0;
}
```

Teil IX
Beispiele

1 Beispiel 1: Automatische Vergabe der zeich_nr

```
//
// Beim Import eines Dokumentes soll die Dokumentnummer
nach einem festen
// Schema vergeben werden:
//    Aufbau Dokumentnummer:
//       B-JJJJ-MM-NNNN, mit
//          B:          Bereich    (vom Anwender
vorgegeben)
//                            Bsp.:      I: Installation,
//                            P: Ingenieurbüro,
//                            Z: Zentrale Dienste
//          JJJJ:       Jahreszahl (0000-9999)
//          MM:    Monat       (01-12)
//          NNNN:       fortlaufende Nummer innerhalb des
Monats
//
// Die automatische Nummernvergabe soll aber nur
erfolgen, wenn der
// Anwender den Vorspann der <zeich_nr> entsprechend
eingegeben hat
// (Positionen 2, 7 und 10 sind "-").
//

proc  hook_insert_entry_10_jag (user_ref, dokuart_kurz)
  {
   vars  fehler  header(30)
   vars  z_bereich(1)  z_jahr(4)  z_monat(2)  z_lfd_nr(4)
 upper_lfd_nr(4)
   vars  hilf_zeich_nr(30)  letzte_besetzte_zahl(4)

//
// Zerlege die <zeich_nr> in ihre Bestandteile
//
   z_bereich = zeich_nr (1,  1)
   z_jahr    = zeich_nr (3,  4)
   z_monat   = zeich_nr (8,  2)
   z_lfd_nr  = zeich_nr (11, 4)

//
// Wenn die <zeich_nr> den Regeln entspricht (an
Positionen 2, 7 und 10
// ein "-"), dann greift ggf. die Regel
//
   if (      zeich_nr(2,1)  == "-"  &&  zeich_nr(7,1) ==
"-"       \
 &&     zeich_nr(10,1) == "-"
   \
)
     {
//
// Konvertiere den Bereich NNNN (fortlaufende Nummer) der
<zeich_nr>
// in Großbuchstaben
//
      upper_lfd_nr = muster_to_upper_huel (z_lfd_nr)
```

```
      if (upper_lfd_nr == "XXXX")
        {
         header = zeich_nr(1,10) ## "%"

//
// Suche höchste bisher vergebene Nummer
//
         DBMS ALIAS hilf_zeich_nr
         DBMS SQL SELECT zeich_nr
         FROM :D3_TABELLE_PHYS_DATEI   \

         WHERE zeich_nr LIKE :+header   \

         AND UPPER ( :SUBSTR (zeich_nr,11,1) ) <> 'X'  \
                   ORDER BY zeich_nr DESC
         DBMS ALIAS

         if (hilf_zeich_nr == "")
             {
//
// Falls es zu gegebenem Vorspann noch keine laufende
Nummer gibt, ist
// "0000" die erste zu besetzende Nummer (wird noch um 1
erhöht)
//
         hilf_zeich_nr        = zeich_nr(1,10)
         letzte_besetzte_zahl = "0000"
        }
         else
              letzte_besetzte_zahl = hilf_zeich_nr (11,4)

//
// Addiere zur höchsten bisher vergebenen Nummer eins
hinzu
//
         letzte_besetzte_zahl = letzte_besetzte_zahl + 1

//
// Ergänze die laufende Nummer ggf. mit führenden Nullen
(-> vierstellig)
//
         if ( @length(letzte_besetzte_zahl) == 1 )
         letzte_besetzte_zahl = "000" ##
letzte_besetzte_zahl

         else if ( @length(letzte_besetzte_zahl) == 2 )
         letzte_besetzte_zahl = "00" ##
letzte_besetzte_zahl

         else if ( @length(letzte_besetzte_zahl) == 3 )
         letzte_besetzte_zahl = "0" ##
letzte_besetzte_zahl

      zeich_nr = hilf_zeich_nr(1,10) ##
letzte_besetzte_zahl
         }
     }
  }
```

2 Beispiel 2: Anschluß an eine externe Datenbank

```
//-------------------------------------------------------
--------------
//
// NAME
//    hook_xx1.jpl
//
// BESCHREIBUNG
//    Hook-Funktionen für Firma xx1
//
// AENDERUNGEN
//    29.09.00   tguk  erstellt
//    19.05.00   mben  insert_hook erweitert für
Einkaufsmappen
//    23.05.00   mben  insert_hook erweitert
//                     lieferdatum wird nicht mehr
benötigt
//    22.08.00   mben  Routine für NKL-Dokumente geändert
(online)
//                     von Herrn Mustermann
//
//-------------------------------------------------------
--------------



proc  hook_insert_entry_10_xx1 (user_ref, dokuart_kurz)
  {
   vars  auftragsart  auftragslaufnr maschinentyp
   vars  kennwort     auftragsdatum  auftragsstatus

   vars  werk         bestelldatum  lieferantennr
   vars  lieferant

   vars  fehler

   fehler = 0


//-------------------------------------------------------
----------
// Mechanikband kompl.
//-------------------------------------------------------
----------

   if (dokuart_kurz == "DMECH")
     {
      // Für die Übernahme der Altdaten ist der
Urpsrungsdateiname der TIFF-Datei
      // identisch mit der Auftragsnummer. Diese
Übernahme wird über den HOSTIMP
      // gemacht

      if (user_ref == "hostimp"  ||  user_ref ==
"d3_batch")
         if (dok_dat_feld[83] == "")
            dok_dat_feld[83] = dateiname
```

```
      if (dok_dat_feld[83] == "")
         fehler = -8001
      else
        {
         DBMS SQL SELECT auftragsart, auftragslaufnr,
maschinentyp,   \
                        kennwort,    auftragsdatum,
auftragsstatus  \
                  FROM mcpruef
             \
                 WHERE auftragsnr = :dok_dat_feld[83]

         if (@dmrowcount == 0)
            fehler = -8001
         else
           {
            zeich_nr         = dok_dat_feld[83]
            dok_dat_feld[1]  = maschinentyp
            dok_dat_feld[3]  = kennwort
            dok_dat_feld[51] = auftragsdatum
           }
        }
     }


//-------------------------------------------------------
----------------
// Konstruktionsband
//-------------------------------------------------------
----------------
   else if (dokuart_kurz == "AKOBA")
     {

      if (dok_dat_feld[83] == "")
         fehler = -8001
      else
        {
         DBMS SQL SELECT auftragsart, auftragslaufnr,
maschinentyp,   \
                        kennwort,    auftragsdatum,
auftragsstatus  \
                  FROM mcpruef
             \
                 WHERE auftragsnr = :dok_dat_feld[83]

         if (@dmrowcount == 0)
            fehler = -8001
         else
           {
            zeich_nr         = dok_dat_feld[83]
            dok_dat_feld[1]  = maschinentyp
            dok_dat_feld[3]  = kennwort
           }
        }
     }

//-------------------------------------------------------
----------------
```

```
// Maschinenband
//-------------------------------------------------------------
---------------
   else if (dokuart_kurz == "AMAS")
     {

      if (dok_dat_feld[83] == "")
         fehler = -8001
      else
        {
         DBMS SQL SELECT auftragsart, auftragslaufnr,
maschinentyp,   \
                         kennwort,    auftragsdatum,
auftragsstatus  \
                  FROM mcpruef
              \
                 WHERE auftragsnr = :dok_dat_feld[83]

         if (@dmrowcount == 0)
            fehler = -8001
         else
           {
            zeich_nr          = dok_dat_feld[83]
            dok_dat_feld[81] = auftragsart
            dok_dat_feld[82] = auftragslaufnr
            dok_dat_feld[1]  = maschinentyp
            dok_dat_feld[3]  = kennwort
            dok_dat_feld[4]  = auftragsstatus
           }
        }
     }

//-------------------------------------------------------------
---------------
// Nachkalkulation
//-------------------------------------------------------------
---------------
   else if (dokuart_kurz == "DNKL")
     {

      if (dok_dat_feld[83] != "")
        {
         DBMS SQL SELECT auftragsart, auftragslaufnr,
maschinentyp,   \
                         kennwort,    auftragsdatum,
auftragsstatus  \
                  FROM mcpruef
              \
                 WHERE auftragsnr = :dok_dat_feld[83]

         if (@dmrowcount == 0)
            fehler = -8001
         else
           {
            dok_dat_feld[1]  = maschinentyp
            dok_dat_feld[3]  = kennwort
           }
        }
     }
```

```
//-------------------------------------------------------
---------------
// Nachkalkulationsband
//-------------------------------------------------------
---------------
   else if (dokuart_kurz == "ANKBA")
     {

      if (dok_dat_feld[83] == "")
         fehler = -8001
      else
        {
         DBMS SQL SELECT auftragsart, auftragslaufnr,
maschinentyp,   \
                         kennwort,    auftragsdatum,
auftragsstatus  \
                  FROM mcpruef
              \
                WHERE auftragsnr = :dok_dat_feld[83]

         if (@dmrowcount == 0)
            fehler = -8001
         else
           {
            zeich_nr         = dok_dat_feld[83]
            dok_dat_feld[1]  = maschinentyp
            dok_dat_feld[3]  = kennwort
           }
        }
     }


//-------------------------------------------------------
---------------
// Einkaufsband kompl.
//-------------------------------------------------------
---------------

   else if (dokuart_kurz == "DEKBK")
     {
      // Für die Übernahme der Altdaten ist der
Urpsrungsdateiname der TIFF-Datei
      // identisch mit der Bestellnummer. Diese Übernahme
wird über den HOSTIMP
      // gemacht

      if (user_ref == "hostimp"  ||  user_ref ==
"d3_batch")
         if (dok_dat_feld[80] == "")
            dok_dat_feld[80] = dateiname

      if (dok_dat_feld[80] == "")
         fehler = -8002
      else
        {
         DBMS SQL SELECT werk,      bestelldatum,
lieferantennr, \
```

```
                                lieferant
          \
                           FROM ekband
          \
                          WHERE bestellnr = :dok_dat_feld[80]


           if (@dmrowcount == 0)
              fehler = -8002

           else
             {
              zeich_nr          = dok_dat_feld[80]
              dok_dat_feld[44] = werk
              dok_dat_feld[50] = bestelldatum
              dok_dat_feld[42] = lieferantennr
              dok_dat_feld[41] = lieferant
             }
          }
      }

//-------------------------------------------------------------
----------------
// Service- und Reiseberichte
//-------------------------------------------------------------
----------------

   else if (dokuart_kurz == "DMOSE")
     {
      vars i(5)

      DBMS SQL SELECT kennwort                                \
                 FROM mcpruef                                 \
                WHERE auftragsnr = :dok_dat_feld[83]


      if (@dmrowcount == 0)
         fehler = -8001
      else
         dok_dat_feld[3] = kennwort



      i=1
      while dok_dat_feld_61[i] != ""
       {
         DBMS SQL SELECT maschinentyp
     \
                      FROM mcpruef
     \
                     WHERE auftragsnr = :dok_dat_feld_61[i]

         if (@dmrowcount == 0)
           {
            fehler = -8001
            break
           }
         else
           {
```

```
                dok_dat_feld_62[i] = maschinentyp
               }

             i = i + 1
            }  // while ...

          }  // else if ...


    return fehler
   }


//-----------------------------------------------------------
----------
// eof
//-----------------------------------------------------------
----------
```

3 Beispiel 3: Komplexe Aktenbildung beim Import

```
############################################################
###############
#
# Hookfunktion zur Aktenbildung bei Firma xx2.
#
# Zur Erkennung der Akte wird die Dokumentnummer
(zeich_nr) aber bis max.
# zum ersten "-" herangezogen.
#
# Die "dok_dat"-Felder 1 bis 5 werden vererbt.
#
############################################################
###############

global  dokuart_in_akte[1000](5)

proc hook_insert_exit_30_xx2 (doku_id_ref, ziel_datei,
import_ok, user_ref)
  {
    vars       zeich_nr_hilf(30)     zeich_nr_akte(30)
    vars  doc_type_short(5)          doc_type_short_folder
(5)
    vars  in_akte_tun(3)             i(5)
               laenge(5)
    vars  dok_dat_1(35)              dok_dat_2(35)
          dok_dat_3(35)
    vars  dok_dat_4(35)              dok_dat_5(35)
    vars  text_hilf

//
// Lade die Liste der Dokumentarten, deren Dokumente ggf.
in eine Akte
// verlinkt werden sollen
//
```

```
   public  dokuart_in_akte_liste.lst
   unload  dokuart_in_akte_liste.lst

//
// Ermittle die Attribute der Felder 1-5 des gerade
importierten Dokumentes
//
   DBMS ALIAS zeich_nr_hilf, doc_type_short,
                                \
               dok_dat_1, dok_dat_2, dok_dat_3, dok_dat_4,
dok_dat_5

   DBMS SQL SELECT zeich_nr, dokuart,
   \
        B.dok_dat_feld_1, B.dok_dat_feld_2,
B.dok_dat_feld_3,                \
   B.dok_dat_feld_4, B.dok_dat_feld_5
              \
           FROM    :D3_TABELLE_PHYS_DATEI A,
         \
 :D3_TABELLE_FIRMEN_SPEZIFISCH B         \
         WHERE     A.doku_id = B.doku_id
                                \
           AND     A.doku_id = :+doku_id_ref

   DBMS ALIAS


//
// Prüfe, ob das gerade importierte Dokument zu einer der
in "dokuart_in_akte_liste.lst"
// aufgelisteten Dokumentarten gehört
//
   in_akte_tun = 0
   for i=1 while i<=1000 step 1
       if (dokuart_in_akte[i] == doc_type_short)
{
    in_akte_tun = 1
    break
   }

   if (in_akte_tun == 1)
     {
//
// Das gerade importierte Dokument muss in eine Akte
//

//
// Lade die Attribute des neu importierten Dokumentes in
die korrespondierenden
// JAM/Panther-Felder
//
      call  uebernimm_attribute_in_widgets (doku_id_ref)

//
// Lösche die Inhalte der globalen Felder für die
Verknüpfungsregeln
//
      call  sm_n_clear_array ("ernie_job_attr_name")
```

```
        call  sm_n_clear_array ("ernie_job_attr_value")
        call  sm_n_clear_array ("inherit_field_name_1")
        call  sm_n_clear_array ("inherit_field_value_1")
        call  sm_n_clear_array ("inherit_field_name_2")
        call  sm_n_clear_array ("inherit_field_value_2")

//
// Bestimme die Erkennungsattribute für die Akte
//
        ernie_job_attr_name[1]  = "dokuart"
        ernie_job_attr_value[1] = "akar"

        laenge = @length(zeich_nr_hilf)
        for     i=1 while i<=laenge step 1
                if (zeich_nr_hilf(i,1) != "-")
                        zeich_nr_akte(i,1) = zeich_nr_hilf(i,1)
                else
                break


        ernie_job_attr_name[2]  = "zeich_nr"
        ernie_job_attr_value[2] = zeich_nr_akte

        ernie_job_attr_name[3]  = "folder_definition"
        ernie_job_attr_value[3] = "1"


//
// Hole die Vererbungssattribute vom Kind auf die Akte.
// Aber nur, wenn das Feld noch nicht besetzt ist.
//
        inherit_field_name_1[1]  = "dok_dat_feld_1"
        inherit_field_value_1[1] = dok_dat_1

        inherit_field_name_1[2]  = "dok_dat_feld_2"
        inherit_field_value_1[2] = dok_dat_2

        inherit_field_name_1[3]  = "dok_dat_feld_3"
        inherit_field_value_1[3] = dok_dat_3

        inherit_field_name_1[4]  = "dok_dat_feld_4"
        inherit_field_value_1[4] = dok_dat_4

        inherit_field_name_1[5]  = "dok_dat_feld_5"
        inherit_field_value_1[5] = dok_dat_5

//
// Erzeuge einen Job für den Prozeß d3_async.
// Jobtype: LIN002 (Verknüpfung gemäß Aktenplan
//
        call  erzeuge_ernie_job ("LIN002", 0, doku_id_ref,
"", "", "", "", 0, "")
     }
  }
```

Teil X
Häufige Probleme und Fragen

Teil X Häufige Probleme und Fragen

1 Oracle Datenbankzugriff

1.1 Probleme beim Datumsformat

1.1.1 Problembeschreibung

Probleme beim Datumsformat unter Oracle Datenbanken.

1.1.2 Lösung

Das Datumsformat (unter Oracle) ist sprachabhängig! Man kann sich nie darauf verlassen, dass es auf allen Systemen das gleiche Format gibt.

```
20.05.2003
05/20/03
2003-05-20
20. Mai 2003
```
usw.

Damit man volle Kontrolle über das Datumsformat bekommt, muss man das Datumsformat immer explizit angeben.

Bei Oracle geht das über TO_CHAR(datum, format) und TO_DATE(char-datum, format).

```
DBMS ALIAS ccmindat,ccmaxdat
   DBMS SQL SELECT TO_CHAR(sysdate -30, 'DD.MM.YYYY'),
TO_CHAR(sysdate,'DD.MM.YYYY')  from dual
   ...
   ...

   DBMS ALIAS ccKundennr, ccBelegnr, ccBelegdat,
ccLiefdat
   DBMS SQL SELECT distinct
dok_dat_feld_2,dok_dat_feld_3,dok_dat_feld_50,dok_dat_fel
d_51 \
                from firmen_spezifisch \
                WHERE ((    dok_dat_feld_50 > TO_DATE
(:+ccmindat,'DD.MM.YYYY') \
                          AND dok_dat_feld_50 < TO_DATE
(:+ccmaxdat,'DD.MM.YYYY') ) \
                    AND (dok_dat_feld_...)
```

2 Aktenplan

Weitere Informationen zum Aktenplan sowie ein Hook-Beispiel finden Sie auch im Aktenplan-Handbuch (d3adminfolderscheme.pdf).

2.1 Anlage untergeordneter Akten

2.1.1 Problembeschreibung

Kann von einer übergeordneten Akte eine untergeordnete Akte automatisch angelegt werden?

2.1.2 Lösung

Nicht mit den Mitteln des Aktenplan-Moduls in der d.3 Administration. Sie haben jedoch die Möglichkeit diese Aktenanlage über einen Hook zu gestalten.

2.2 Verknüpfungen unabhängig vom Aktenplan erstellen

2.2.1 Problembeschreibung

Kann man Verknüpfungen (mit 60er-Feldern) unabhängig vom Aktenplan in einem Hook erstellen?

2.2.2 Lösung

Um selber, unabhängig vom Aktenplan Verknüpfungen zu erstellen, müssen verschiedene globale Arrays mit den Verknüpfungs-/Vererbungs-Daten gefüllt werden.

```
//
// Lösche die Inhalte der globalen Felder für die
Verknüpfungsregeln //
      call  sm_n_clear_array ("ernie_job_attr_name")
      call  sm_n_clear_array ("ernie_job_attr_value")
      call  sm_n_clear_array ("inherit_field_name_1")
      call  sm_n_clear_array ("inherit_field_value_1")
      call  sm_n_clear_array ("inherit_field_name_2")
      call  sm_n_clear_array ("inherit_field_value_2")

//
// Erkennungs-/Verknüpfungsattribute angeben
//

      // hier Einzelwerte der betr. 60ger-Feldes
ermitteln ..


      ernie_job_attr_name[1]  = "dokuart"
      ernie_job_attr_value[1] = "XXX"     // Kürzel der
Dokumentart eintragen

      ernie_job_attr_name[2]  = "dok_dat_feld_1"   // Bsp
      ernie_job_attr_value[2] = "Verknüpfungswert aus
```

```
60ger Feld eintragen"

      // ...

      ernie_job_attr_name[3]  = "folder_definition"  //
autom. Aktenanlage
      ernie_job_attr_value[3] = "1"                  //
ist so eingeschaltet


//
// ggf. Vererbungssattribute vom Kind auf die Akte
angeben
//
      inherit_field_name_1[1]  = "dok_dat_feld_2"
      inherit_field_value_1[1] = Variable mit Wert
angeben

      // ...

//
// ggf. Vererbungssattribute vom Akte auf Kind angeben
//
      inherit_field_name_2[1]  = "dok_dat_feld_3"
      inherit_field_value_2[1] = Variable mit Wert
angeben

      // ...

//
// Erzeuge einen Job für den Prozeß d3_async.
// Jobtype: LIN002 (Verknüpfung gemäß Aktenplan)
//
      call  erzeuge_ernie_job ("LIN002", 0, doku_id_ref,
"", "", "", "", 0, "")  // doku_id_ref = Doku-ID des
aktu. Dokumentes, das verknüpft werden soll
```

2.2.3 Weitere Informationen

Die Variablen `inherit_field_name_1` und `inherit_field_value_2` sind globale Arrays. Sie vererben von Dokument auf Akte:

ohne Überschriften	`inherit_field_name_1` `inherit_field_value_1`
mit Überschreiben	`inherit_field_name_2` `inherit_field_value_2`

Für die Vererbung von Akte auf Dokumente muss `inherit_field_name_3` / `inherit_field_value_3` oder `inherit_field_name_4` / `inherit_field_value_4` verwendet werden:

ohne Überschriften	`inherit_field_name_3` `inherit_field_value_3`
mit Überschreiben	`inherit_field_name_4` `inherit_field_value_4`

Die Funktion `sm_n_clear_array()` dient dazu alle Array-Elemente auf einen Schlag zu leeren. Weitere Informationen finden Sie im Panther Programming Guide.

`ernie_job_attr_name` ist genauso ein Array wie `inherit_field_name_1` oder `inherit_field_name_2`.

2.3 Entfernen einer Verknüpfung

2.3.1 Problembeschreibung

Gibt es eine Möglichkeit eine Verlinkung per Hook-Funktion zu entfernen?

2.3.2 Lösung

Leider gibt es z.Zt. keinen d.3 Hook-Einsprungpunkt für das Entfernen einer Verlinkung. Eine mögliche Alternative wäre vielleicht auf DB-Ebene einen `ON-DELETE`-Trigger für Tabelle `dokumenten_verknuepf` zu implementieren.

2.4 Löschen aller Verknüpfungen eines Dokuments

2.4.1 Problembeschreibung

Wie kann ich aus einem Hook via JPL alle Verknüpfungen eines Dokuments löschen?

2.4.2 Lösung

Die Funktion `proc hierarch_loesen_huel (doku_id_ueber, doku_id_unter)` kann zum Entfernen einer Verknüpfung benutzt werden.

Bei `Updateattributes()` wird die Hook--Funktion `hook_upd_attrib_exit_10()` vor der Erzeugung des Async-Jobs für die neue Verknüpfung aufgerufen.

Eine mögliche Lösung könnte also ähnlich wie folgt aussehen (etwas vereinfacht und ohne Fehlerbehandlung):

```
proc hook_upd_attrib_exit_10_x(doc_id, error_number,
user)
{
  vars i, akten_arr[100]

  // hier Kontrolle, ob Verknüpfungsattribut sich
geändert hat
  // wenn ja, dann folgender Code:
  // alle übergeordneten Dokumente (Akten) ermitteln

  dbms alias akten_arr
  dbms sql select doku_id_ueber from dokumenten_verknuepf
where doku_id_unter = :+doc_id
  dbms alias
  for i=1 while akten_arr[i] != ""
    call hierarch_loesen_huel (akten_arr[i], doc_id)
```

3 Hinweise

3.1 Problembeschreibung

Gibt eine Möglichkeit im Hook festzustellen, ob sich ein Feldinhalt geändert hat?

3.2 Lösung

Nicht direkt. Bei `UpdateAttributes()` werden die im Client geänderten Attribute übermittelt. Die alten Werte stehen nicht zur Verfügung.

Eine Möglichkeit wäre, in `hook_upd_entry()` die vorhandenen Werte aus der Datenbank auszulesen und diese dann mit den übermittelten neuen Werten zu vergleichen.

Wenn sich eine Änderung ergeben hat, dann diese in globaler Variable merken, um danach in `hook_upd_exit()` die Info auslesen zu können.

JPL-Deklaration glob. Variable: `global glob_var`

4 d.3 Archivierung

4.1 Zu archivierende Dateitypen in der d.3 Archivierung festlegen

4.1.1 Problembeschreibung

Über die d.3 Archivierung sollen nur definierte Dateitypen archiviert werden können.

Gibt es eine Möglichkeit dokumentartabhängig festzulegen, welche Dateitypen (`*.doc`, `*.xls`, `*.tif` usw) manuell über den Importclient archiviert werden können?

Wenn versucht wird, andere Dateitypen als vorgegeben zu archivieren, dann sollen diese abgelehnt werden.

4.1.2 Lösung

Im d.3 System kann man einer Dokumentart keine bestimmten für die Dialog-Archivierung erlaubten Dateitypen zuordnen. Die Auswertung des Dateityps bzw. der Dateiendung bei der Dialog-Archivierung ist aber über einen Hook möglich.

Möglicher Einsprungpunkt: `hook_insert_entry_10`

Aufruf-Parameter: `user_ref`, `dokuart_kurz`

Die Dateiendung ist in der globalen Variablen `datei_erw` hinterlegt.

Beispiel-Hook für die Dialog-Archivierung:

```
proc hook_insert_entry_10_beispiel (user_ref,
dokuart_kurz)
{
vars erw
if (user_ref != "hostimp")
{
erw = muster_to_upper_c (datei_erw)
if (dokuart_kurz == "BSP1")
{
if ( (erw != "DOC") && (erw != "XLS") && ... && ... )
{
fehler = -8001
}
}
else if (dokuart_kurz = "BSP2")
{
}
...
} // if (user_ref != hostimp)

} // proc hook_insert_entry_10_beispiel
```

4.2 Serverseitiges Verschlagworten von bereits importierten Dokumenten

4.2.1 Problembeschreibung

Bereits importierte Dokumente sollen serverseitig verschlagwortet werden in einem eigenen Prozess. Eine Schlagwortdatei wird dabei generiert und soll nachträglich d.3 zur Verfügung gestellt werden.

4.2.2 Lösung

Es ist nicht möglich, per Hostimp oder Async nur eine OCR-Datei zu vorhandenen Dokumenten nachzuliefern.

Wenn die API nicht benutzt werden soll, gibt es serverseitig nur die Möglichkeit ein JPL-Skript zu schreiben und darin die Funktion `uebernehme_ocr_begriffe()` für jedes Dokument aufzurufen, das nachträglich verschlagwortet werden soll.

```
proc uebernehme_ocr_begriffe (doc_id, ocr_datei,
attrib_datei, neues_dokument)
```

Die letzten beiden Parameter müssen in diesem Fall leer bleiben.

Rückgabe: 0 = Erfolg

```
--
call uebernehme_ocr_begriffe ("P0001799", "G\:\\temp\
\OCR_P0001799.txt", "", 0)
call uebernehme_ocr_begriffe ("P0001800", "G\:\\temp\
\OCR_P0001800.txt", "", 0)
...
--
```

So ein JPL-Skript kann generiert und im Unterverzeichnis `d3server.prg\ext_jpl` abgelegt werden. Anschliessend kann das d.3 Hauptprogramm zur Ausführung des Skriptes ausgerufen werden. Dazu muss es als sechster Aufruf-Parameter angegeben sein. Der Aufruf kann über den d.3 process manager zeitgesteuert automatisiert werden.

```
..\d3odbc32.exe haupt "" Master password d3o ext_jpl
\uebernehme_ocr.jpl
```

5 Sonstige Probleme

5.1 Sonderzeichen in einem Hook

5.1.1 Problembeschreibung

Gibt es eine Liste, in der man erfahren kann, wie man in HOOK Sonderzeichen abfragen kann, wie z.B. CRLF , etc.?

5.1.2 Lösung

Man kann in JPL Sonderzeichen nicht direkt angeben. Eine Liste gibt es also nicht.

Es gibt aber Workaround-Möglichkeiten:

- eine C-Funktion schreiben (z.B. in DLL), die das erledigt

- oder es kann die Datenbank dazu "missbraucht" werden;

z.B. NL aus DB holen:

```
vars NL
 // New Line aus DB besorgen
 dbms alias NL
 dbms sql select chr(10) NL from dual  // Oracle-
spezifisch
 dbms alias
```

Variable NL kann jetzt im JPL-Code verwendet werden.

- Zeichen in kl. Textdatei schreiben und daraus in JPL-Variable einlesen

Teil XI
Anhang

1 Die Panther Dokumentation

Zum besseren Verständnis der Hookprogrammierung sollten Sie sich folgende Teile der Panther Dokumentation besonders anschauen.

Die Dokumentation finden Sie auch im Internet unter http://www.prolifics.com/docs/panther.

1.1 Panther Programming Guide

Aus dem Panther Programming Guide:

(http://www.prolifics.com/docs/panther/html/prg_html/index.htm)

Kapitel JPL Command Overview, Befehle:

Control Flow
Procedure Structure
Variable Declaration
Command/Function Execution
Module Access and Availability
Text Display

Kapitel DBMS Statements and Commands, Unterkapitel:

DBMS Command Summary mit den Befehlen

Using Connections	
	DBMS CONNECTION
	DBMS CLOSE CONNECTION
	DBMS DECLARE CONNECTION
	DBMS WITH CONNECTION
Using Cursors	
	DBMS CLOSE CURSOR
	DBMS DECLARE CURSOR
	DBMS DECLARE CURSOR
	DBMS EXECUTE
	DBMS WITH CURSOR
Executing SQL Statements	
	DBMS SQL
	DBMS Query
	DBMS RUN
Changing SELECT Behaviour	
	DBMS CATQUERY
Paging through Multiple Rows	
	DBMS CONTINUE

Weitere Informationen finden Sie im Panther Developer's Guide (Application Development Guide)!

1.2 Panther Developer's Guide

Aus dem Panther Developer's Guide (Application Development Guide:http://www.prolifics.com/docs/panther/html/dev_html/index.htm)

Kapitel Data Types, Operators and Expressions, Unterkapitel:

Operators mit den Teilen	
	Operator Precedence
	Conversion of Operands
	Concatenation
	Substring Specifiers
	@date
	@length
Expressions mit den Teilen	
	String
	Numeric
	Logical

Weitere Informationen finden Sie im Panther Developer's Guide (Application Development Guide).

Index