PDHK810 (d.3 hook api (jpl) 1.0)

Inhaltsverzeichnis

1. d.3 hook api (jpl)

Dies ist die Startseite des Bereiches Programmierhandbuch d.3 hook.

Einige populäre Makros werden innerhalb dieser Startseite ausgeführt. Sobald ihr anfangt, neue Seiten anzulegen, oder Blogeinträge anzulegen und zu kommentieren, werdet ihr sehen, wie die Aktivitäten unten aufgeführt werden.

1.1. Einleitung

1.1.1. 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. Zum Beispiel kann beim Import eines Dokumentes eine Plausibilitätsprüfung und Ergänzung der Eigenschaften 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. Der Entwickler der Hook-Funktionen kann durch seine Programmierung für die Internationalisierung des Textes sorgen.

1.1.2. Voraussetzungen

Dieses Handbuch erläutert die d.3 Hook-Programmierung.

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

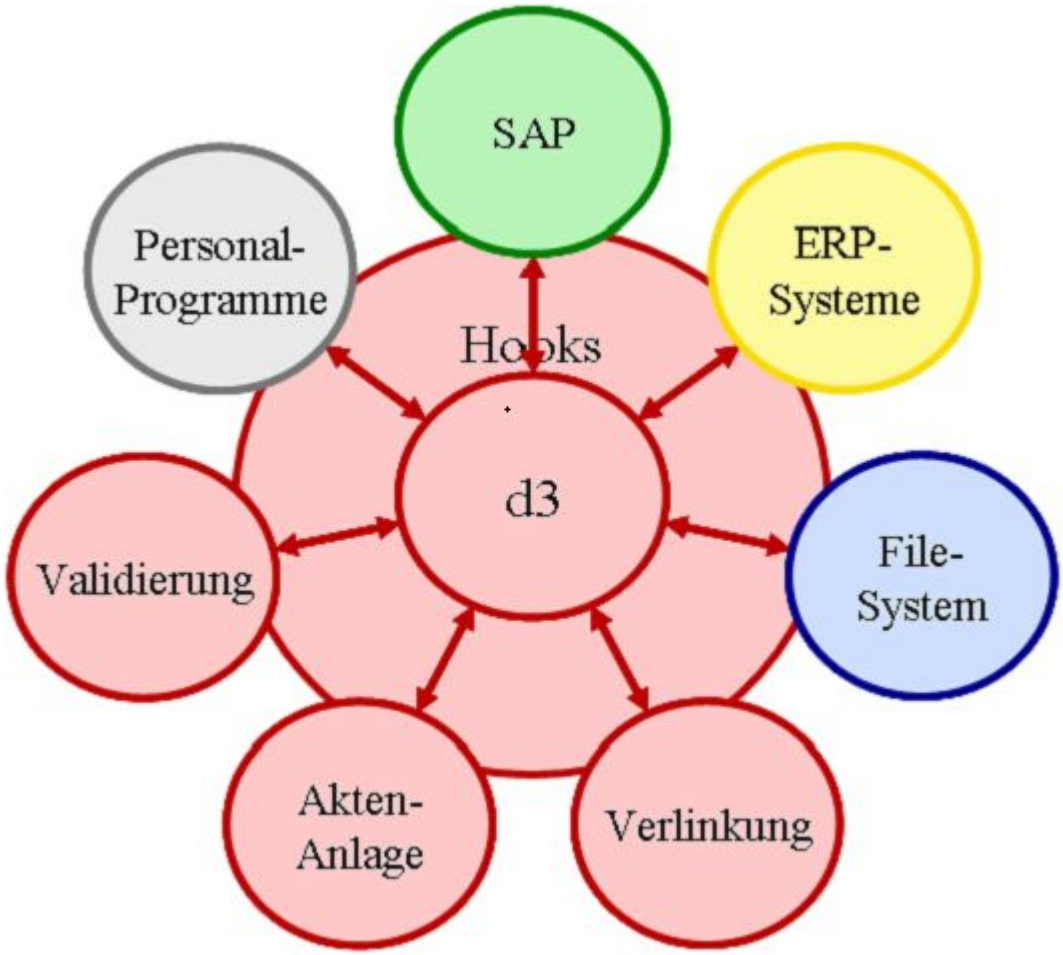
Um erfolgreich Hooks für das d.3-System zu programmieren, benötigen Sie die folgenden Voraussetzungen:

- 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.

1.2. Hook-Funktionen im Einzelnen

1.2.1. Allgemeine Informationen zu Hook-Funktionen

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 Eigenschaften 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 admin folderscheme)
- Dokumentklassen-Hook-Funktionen zur Steuerung der Rechtevergabe

Beispiele aus der Praxis

> **Anmerkung**
> Beim Import eines Dokumentes im Dialog (d.3 import) wird als Eigenschaft 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

Weitere Beispiele

> **Anmerkung**
> Vor der Suche wird die Kontonummer immer auf zehn Stellen mit führenden Nullen aufgeführt.

Vor dem Import einer Rechnung wird geprüft, ob die zugehörige Bestellung im Repository 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.

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

Hinweise zur Hook-Architektur

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

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

> **Anmerkung**
> Eine gute Kommentierung erleichtert die spätere Fehlersuche und Anpassung der Hook-Funktion.

Beschränkungen

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

> **Anmerkung**
> 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)

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 abfragen bzw. ändern zu können:

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

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.

> **Anmerkung**
> Bei der Entwicklung von Hook-Funktionen sollten Sie auf die Funktionen zurückgreifen, denen die d.3 Server API zur Verfügung steht.

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`.

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:

> **Anmerkung**
> `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.

> **Anmerkung**
> `InsertExit_20_siemens (…)`

Hook der Firma Siemens

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

> **Anmerkung**
> 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.

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

Installieren und Aktivieren der Hook-Funktionen

d.3 Hook

- Rufen Sie die d.3 Konfiguration **d.3 Config** (`d3config.ini`) in der d.3 Administration (unter **Systemeinstellungen**) auf.
- Wechseln Sie in die Sektion **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.

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.

Dokumentklassen-Hook

- Öffnen Sie die d.3 Administration mit Hilfe der **d.3 config**-Komponente.
- Bearbeiten Sie die Dokumentklassen in der d.3 Administration.
- Geben Sie per `@D3HOOK` (`<Hook-Funktions-Name>`) für das Eigenschaftsfeld an.
- Starten Sie die d.3 Prozesse neu.

Aktenplan-Hook (d.3 folder scheme)

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

Fehlersuche

Standardfehler zu d.3 finden Sie in der Datei `msglib.dat`, die im Standard 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.

> **Warnung**
> 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.

> **Anmerkung**
> Fehlercode 28005
>
> ```
> 0028005,049,1,"Ist dies ein Fall, der unter FAQ veröffent-
> licht werden soll?", "Bitte Eingabe für das Feld Homepage prü-
> fen.","[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.

Anmerkung
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`

`ReleaseDocument` mit Offset `30000`

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.

1.2.2. Allgemeine Informationen

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]`	Eigenschaftsfeld 1
`dok_dat_feld[2]`	Eigenschaftsfeld 2
`...`	
`dok_dat_feld[59]`	Eigenschaftsfeld 59

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

1.2.3. Abhängige Dateien

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 Dokuments
index	Version des Dokuments
user_o_group	Bearbeiter oder Prüfer-Gruppe des Dokumentes bei Status Bearbeitung, bzw. Prüfung
dep_doc_ext	Dateierweiterung der abhängigen Datei
transfer	1: Aufruf während eines Statustransfers

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
dep_doc_ext	Dateierweiterung der abhängigen Datei
transfer	1: Aufruf während eines Statustransfers

1.2.4. Abwesenheitszeiten bei Benutzern

hook_get_user_absence_time (user_name, user_type, t1, t2)

> **Anmerkung**
> 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 angegebenen Zeitraums `t1` bis `t2`.

1.2.5. Aktualisieren der Kenndaten (UpdateAttributes)

> **Anmerkung**
> 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`

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

hook_upd_attrib_entry_10 (doc_id, user_name, doc_type_short, doc_type_short_new)

reserviert

hook_upd_attrib_entry_20 (doc_id, user_name, doc_type_short, doc_type_short_new)

- Verfügbare Felder: Alle beim API-Call übergebenen Felder. Änderbar sind jedoch nur die Datenbankfelder und das Feld `text`.
- Aufrufzeitpunkt: Es wurden lediglich die neuen Eigenschaften empfangen, jedoch noch nicht auf Plausibilität geprüft.
- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, dessen Eigenschaften aktualisiert werden sollen
user_name	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart vor der Eigenschaftenaktualisierung
doc_type_short_new	Kürzel der Dokumentart nach der Eigenschaftenaktualisierung

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

hook_upd_attrib_entry_30 (doc_id, user_name, doc_type_short, doc_type_short_new)

reserviert

hook_upd_attrib_exit_10 (doc_id, error_number, user_name, doc_type_short, doc_type_short_old)

> **Anmerkung**
> 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
`doc_id`	Dokument-ID des Dokuments, dessen Eigenschaften aktualisiert werden sollen
`error_number`	0: Aktualisierung korrekt durchgelaufen <> 0: Fehlernummer; i. a. vom DB-Server geliefert
`user_name`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart nach der Eigenschaftenaktualisierung
`doc_type_short_old`	Kürzel der Dokumentart vor der Eigenschaftenaktualisierung

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

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

> **Anmerkung**
> 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`	Dokument-ID des Dokuments, dessen Eigenschaften 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 Eigenschaftenaktualisierung
`doc_type_short_old`	Kürzel der Dokumentart vor der Eigenschaftenaktualisierung

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

reserviert

1.2.6. Dokumente freigeben (Release Document)

hook_release_entry_10 (doc_id, user_name, doc_type_short, unblock)

> **Anmerkung**
> 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`	Dokument-ID des freizugebenden Dokuments
`user_name`	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

hook_release_exit_10 (doc_id, user_name, error, doc_type_short, unblock)

- Aufrufzeitpunkt: Nach Durchführung der Freigabe, nach Beendigung der DB-Transaktion.
- Eingabeparameter:

Parameter	Beschreibung
`doc_id`	Dokument-ID des freizugebenden Dokuments
`user_name`	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

1.2.7. Dokument prüfen (Verify Document)

hook_verify_entry_10 (doc_id, alteration_number, user_name)

> **Anmerkung**
> Wenn 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`	Dokument-ID des zu prüfenden Dokuments
`alteration_number`	Änderungsnummer der zu prüfenden Dokumentversion
`user_name`	Name des rufenden Benutzers

hook_verify_exit_10 (doc_id, alteration_number, user_name, error)

- Aufrufzeitpunkt: Nach Durchführung der Prüfung. Nach Beendigung der Datenbanktransaktion.
- Eingabeparameter:

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

1.2.8. Dokumentsuche (Search Document)

> **Anmerkung**
> Ab d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter `doc_type_short` für die folgenden Einsprungspunkte:
>
> `hook_search_entry_10`
>
> `hook_search_entry_20`
>
> `hook_search_entry_30`
>
> `hook_search_exit_10`
>
> `hook_search_exit_20`
>
> `hook_search_exit_30`

hook_search_entry_05 (user_name, doc_type_short)

> **Anmerkung**
> 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

hook_search_entry_10 (user_name, 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 Eigenschafts-Felder übernommen. Diese Eigenschaftswerte 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
datum1	untere Grenze des Datumintervalls, in dem die Nutzdatei angelegt bzw. wurde zuletzt geändert wurde Format: TT.MM.JJJJ Bezieht sich auf das Erstellungsdatum der aktuellsten Dokumentversion und stellen eine untere Grenze dar.
datum2	obere Grenze des Datumintervalls, in dem die Nutzdatei angelegt bzw. wurde zuletzt geändert wurde Format: TT.MM.JJJJ Bezieht sich auf das Erstellungsdatum der aktuellsten Dokumentversion und stellt eine obere Grenze dar
dok_dat_feld[1]	Eigenschaftsfeld 1
dok_dat_feld[2]	Eigenschaftsfeld 2
...	
....	
dok_dat_feld[59]	Eigenschaftsfeld 59
dok_dat_feld[70]	Eigenschaftsfeld 70
...	
...	
dok_dat_feld[89]	Eigenschaftsfeld 89
dok_dat_feld_60[1]	Eigenschaftsfeld 60, Zeile 1
dok_dat_feld_61[1]	Eigenschaftsfeld 61, Zeile 1
...	
dok_dat_feld_69[1]	Eigenschaftsfeld 69, Zeile 1
suchtext_ausdruck	Suchstring
color_code	Farbcode des Dokuments (0 = nicht gesetzt; 1-24 = Farbcode)

- Eingabeparameter:

Parameter	Beschreibung
user_name	Name des rufenden Benutzers
doc_type_short	Dokumentartkürzel

hook_search_entry_20 (user_name, 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	Name des rufenden Benutzers
doc_type_short	Dokumentartkürzel

hook_search_entry_30 (user_name, doc_type_short)
reserviert

hook_search_exit_10 (user_name, doc_type_short)
reserviert

hook_search_exit_20 (doc_id, doc_type_short)
reserviert

hook_search_exit_30 (user_name, error, no_results, no_refused, doc_type_short))

> **Warnung**
> 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_name	Benutzer, der die Suche ausführt
error	Fehler, falls aufgetreten, sonst "0"
no_results	Anzahl Treffer
no_refused	Anzahl verweigerter Treffer
doc_type_short	Kürzel der Dokumentart

- Rückgabewert: wird ignoriert

> **Anmerkung**
> 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).

1.2.9. Dokumentanlage (Import Document)

hook_hostimp_entry_10

- 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`	Name des rufenden Benutzers
`doc_type`	Dokumentart des zu importierenden Dokumentes. Dabei kann es sich um das Dokumentart-Kürzel, aber auch um den Langnamen der Dokumentart handeln. Das ist abhängig davon, was in der `default.ini` bzw. JPL-Datei angegeben wurde.

> **Anmerkung**
> In einem `hook_hostimp_entry_10` Hook dürfen Eigenschaften mit Hilfe von Update-Funktionen (wie `attribute_update_single`) nicht geändert werden.
>
> Funktionen wie `attribute_update_single` überschreiben die globalen Datenbankfeld-Variablen, die somit für den folgenden Import nicht mehr mit den ursprünglichen Werten verfügbar sind.

hook_insert_entry_10 (user_name, 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 Eigenschaftswerte wurden in die entsprechenden Eigenschaftsfelder übernommen. Diese Eigenschaftswerte können im Hook geändert werden.

> **Anmerkung**
> 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:

Variable	Beschreibung
`zeich_nr`	Dokument-/Zeichnungsnummer
`dokuart (Kürzel)`	Dokumentart-Kürzel
`var_nr`	Variantennummer
`doku_id`	Dokument-ID

Variable	Beschreibung
`logi_verzeichnis`	Dokumentstatus (= logisches Verzeichnis) zwei signifikante Zeichen: • Be • Pr • Fr • Ar
`bearbeiter`	Bearbeiter des Dokuments
`text`	Kommentartext
`datum1`	untere Grenze des Datumintervalls, in dem die Nutzdatei angelegt bzw. wurde zuletzt geändert wurde Format: TT.MM.JJJJ
`datum2`	obere Grenze des Datumintervalls, in dem die Nutzdatei angelegt bzw. wurde zuletzt geändert wurde Format: TT.MM.JJJJ
`dok_dat_feld[1]`	Eigenschaftsfeld 1
`dok_dat_feld[2]`	Eigenschaftsfeld 2
`...`	
`....`	
`dok_dat_feld[59]`	Eigenschaftsfeld 59
`dok_dat_feld[70]`	Eigenschaftsfeld 70
`...`	
`...`	
`dok_dat_feld[89]`	Eigenschaftsfeld 89
`dok_dat_feld_60[1]`	Eigenschaftsfeld 60, Zeile 1
`dok_dat_feld_61[1]`	Eigenschaftsfeld 61, Zeile 1
`...`	
`dok_dat_feld_69[1]`	Eigenschaftsfeld 69, Zeile 1
`suchtext_ausdruck`	Suchstring
`color_code`	Farbcode des Dokuments (0 = nicht gesetzt; 1-24 = Farbcode)

- Eingabeparameter:

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

> **Anmerkung**
> In einem `hook_insert_entry_10` Hook dürfen Eigenschaften mit Hilfe von Update-Funktionen (wie `attribute_update_single`) nicht geändert werden.
>
> Funktionen wie `attribute_update_single` überschreiben die globalen Datenbankfeld-Variablen, die somit für den folgenden Import nicht mehr mit den ursprünglichen Werten verfügbar sind.

hook_insert_entry_20 (doc_id, doc_type_short, user_name)

- 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	Dokument-ID, die dem Dokument bei erfolgreichem Import gegeben wird
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments
user_name	Name des rufenden Benutzers

hook_insert_entry_30

reserviert

hook_insert_exit_10 (doc_id, file_destination, import_ok, user_name, 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	Dokument-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	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

> **Anmerkung**
> 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.

hook_insert_exit_20 (doc_id, file_destination, import_ok, user_name, 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.

hook_insert_exit_30 (doc_id, file_destination, import_ok, user_name, doc_type_short)

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

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

hook_hostimp_exit_10 (import_dir, import_file, doc_id, doc_type_short, new_or_upd)

- Aufrufzeitpunkt: Nach einem erfolgreichen Import eines Dokuments über den Hostimport-Prozess. Die Import-Dateien wurden bereits aus dem Anlieferungsverzeichnis entfernt.

- Eingabeparameter:

Parameter	Beschreibung
import_dir	Anlieferungsverzeichnis der Hostimport-Daten für das Dokument
import_file	Dateiname der Nutzdatei
doc_id	Dokument-ID des erfolgreich importierten Dokuments
doc_type_short	Dokumentart-Kürzel des Dokuments
new_or_upd	0: Aktualisierung eines Dokumentes; 1: Neuimport eines Dokuments

1.2.10. Einspielen einer neuen Version (ImportNewVersionDocument)

> **Anmerkung**
> 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`

> **Anmerkung**
> 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.

hook_new_version_entry_10 (doc_id, file_source, file_destination, user_name, doc_type_short)

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

> **Anmerkung**
> 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	Dokument-ID des Dokuments, 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	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

hook_new_version_entry_20 (doc_id, file_source, file_destination, user_name, doc_type_short)

> **Anmerkung**
> Kann zur Zeit nicht über HOSTIMP verwendet werden (nur bei API-Funktion `Import-NewVersionDocument`).

- 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	Dokument-ID des Dokuments, 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	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

hook_new_version_entry_30 (doc_id, file_source, file_destination, user_name, doc_type_short))

- Aufrufzeitpunkt: Wird sofort nach hook_new_version_entry_20 ausgeführt.

Alle Angaben analog zu hook_new_version_entry_20.

hook_new_version_exit_10 (doc_id, error_update_attributes, user_name, doc_type_short)

- Verfügbare Felder: siehe hook_new_version_entry_10
- Aufrufzeitpunkt: Die Datenbanktransaktion wurde gestartet. Alle Kenndaten, auch die Mehrfacheigenschaftsfelder (60er-Felder) wurden aktualisiert.
- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, 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	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

hook_new_version_exit_20 (doc_id, file_destination, import_ok, user_name, doc_type_short)

- Aufrufzeitpunkt: Auch die Mehrfacheigenschaftsfelder (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`	Dokument-ID des Dokuments, 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 OK 0: Einspielung der neuen Version mit Fehler abgebrochen
`user_name`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart des zu importierenden Dokuments

hook_new_version_exit_30 (doc_id, import_ok, error_nr_api, user_name, doc_type_short)

siehe `hook_new_version_exit_20`

- Eingabeparameter:

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

1.2.11. Erstellen/Bearbeiten von TIFF- oder PDF-Dokumenten

hook_rendition_entry_10 (doc_id, user_name)

> **Anmerkung**
> 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.

> **Anmerkung**
> Sie können den Aufruf über einen Return-Wert ungleich 0 abbrechen.

- Eingabeparameter:

Parameter	Beschreibung
`doc_id`	Doku-ID des Dokuments von dem ein TIFF- oder PDF-Abbild erstellt werden soll

Parameter	Beschreibung
user_name	Name des d.3-Benutzers, der Erstellung angefordert hat

hook_rendition_entry_20 (doc_id, doc_type_short, source_path, source_filename, file_destination)

- Aufrufzeitpunkt: Direkt vor dem Senden des Erstellungsjobs an den d.ecs rendition service.

> **Anmerkung**
> 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	Dokument-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
file_destination	Zielverzeichnis für die fertige Abbilddatei

hook_rendition_exit_30 (doc_id, source_logi, tiff_file_with_path, error, file_type)

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

Parameter	Beschreibung
doc_id	Dokument-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)

1.2.12. Login

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	Sprach-ID, die von der Anwendung übergeben wurde, z.B. 049=deutsch, 001=englisch

Parameter	Beschreibung
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.

hook_val_passwd_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_passwd_entry_10).

1.2.13. Löschen eines Dokuments (Delete Document)

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

hook_delete_entry_10 (doc_id, user_name, 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	Dokument-ID des zu löschenden Dokuments
user_name	Name des Benutzers (maximal zehn Zeichen), der das Dokument löschen möchte
doc_type_short	Kürzel der Dokumentart des zu importierenden Dokuments

hook_delete_exit_10 (doc_id, user_name, error, doc_type_short)

- Aufrufzeitpunkt: Nach dem Löschen des Dokuments
- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des zu löschenden Dokuments

Parameter	Beschreibung
`user_name`	Name des Benutzers (maximal zehn 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

1.2.14. Löschen von Verknüpfungen (Unlink)

hook_unlink_entry_30 (doc_id_father, doc_id_child)

> **Anmerkung**
> ab d.3 Version 6.0.0 Hotfix 1

- Aufrufzeitpunkt: direkt vor dem Datenbank-Befehl zur Lösung der Verknüpfung
- Eingabeparameter:

Parameter	Beschreibung
`doc_id_father`	Dokument-ID des übergeordneten Dokuments
`doc_id_child`	Dokument-ID des untergeordneten Dokuments

hook_unlink_exit_10 (doc_id_father, doc_id_child, unlink_error_code, error_number)

- Aufrufzeitpunkt: Nach der Verknüpfungslösung zweier Dokumente
- Eingabeparameter:

Parameter	Beschreibung
`doc_id_father`	Dokument-ID des übergeordneten Dokuments
`doc_id_child`	Dokument-ID des untergeordneten Dokuments
`unlink_error_code`	0: Verknüpfungslösung war erfolgreich -1: übergeordnetes und untergeordnetes Dokument 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

1.2.15. Postkorb

hook_ack_holdfile_exit_10 (user_name, doc_id, holdfile_id)

Quittieren einer Wiedervorlage

> **Anmerkung**
> 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	Dokument-ID des Dokuments, welches quittiert wurde
holdfile_id	eindeutige ID des Postkorbeintrages

1.2.16. Redlining (redline)

hook_write_redline_entry_10

Die Funktion wird vor dem Schreiben der Redlining-Datei ausgeführt. Sie hat folgenden Aufbau:

```
hook_write_redline_entry_10 (doc_id, user_name, doc_type_short)
```

doc_id	Dokument-ID des Dokuments zu dem eine Redlining-Datei abgelegt wird
user_name	Name des d.3 Benutzers
doc_type_short	Dokumentart Kürzel

hook_write_redline_exit_30 (doc_id, user_name, doc_type_short)

> **Anmerkung**
> ab d.3 Version 6.0.0 Hotfix 3

- Aufrufzeitpunkt: Nach dem Schreiben einer Redlining-Datei (per d.3-API-Call `WriteRedline`).
- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments zu dem eine Redlining-Datei abgelegt wird
user_name	Name des d.3 Benutzers
doc_type_short	Dokumentart Kürzel

1.2.17. Senden einer Wiedervorlage (Send Holdfile)

hook_holdfile_entry_10 (doc_id, recipient, sender, chain_id, subject, wv_type)

- Aufrufzeitpunkt: Wird aufgerufen, bevor die Übergabeparameter geprüft werden. Empfänger und Sender können 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]	siehe Parameter recipient
d3server_sender_wv[1]	siehe Parameter sender
d3server_kette_id	siehe Parameter chain_id

Diese Werte können verändert werden.

- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, welches in die Wiedervorlage gestellt werden soll
recipient	Benutzername des Empfängers (maximal zehn Zeichen)
sender	Benutzername des Sender (maximal zehn Zeichen)
chain_id	Ketten-ID, die für diesen Wiedervorlageeintrag verwendet werden soll
subject	Betrefftext der Postkorbbenachrichtigung
wv_typ	Typ-ID der Postkorbbenachrichtigung

Mögliche Werte:

"" = normale Postkorbbenachrichtigung

"W" = Workflow-Benachrichtigung

... = sonstige (ggf. selbst definierte Werte)

hook_holdfile_entry_20 (doc_id, recipient, sender, chain_id, subject, wv_type)

- Aufrufzeitpunkt: 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 zehn Zeichen lang.

Die Werte sind hier nicht mehr änderbar.

- Eingabeparameter: siehe hook_holdfile_entry_10.

hook_holdfile_entry_30 (doc_id, recipient, sender, chain_id, subject, wv_type)

- Aufrufzeitpunkt: 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.

> **Anmerkung**
> 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 kompatibel.

- Eingabeparameter: siehe hook_holdfile_entry_10.

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	Dokument-ID des Dokuments, welches in die Wiedervorlage gestellt werden soll
recipient	Benutzername des Empfängers (maximal zehn Zeichen)
sender	Benutzername des Sender (maximal zehn 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

1.2.18. Senden von E-Mails bei Wiedervorlage (send_email)

hook_send_email_entry_10 (doc_id, recipient, sender, subject, holdfile)

Ab d.3 Version 6.1.1

- Aufrufzeitpunkt: Vor dem Versenden einer E-Mail.
- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des d.3 Dokuments bei E-Mail wegen Wiedervorlage

Parameter	Beschreibung
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
url_link	HTTP-Link für die Ansicht in d.3 One Hinweis: Der Parameter ist nur gefüllt wenn d.3 One installiert ist.

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

Eigenschaft	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 (Standard)
api_email_attach	1 = d.3-Dokument als Anhang 0 = Dokument nicht anhängen (Standard)

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

hook_send_email_entry_20 (doc_id, recipient, sender, subject, holdfile)

Ab d.3 Version 6.1.1

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

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

- Eingabeparameter:

Parameter	Beschreibung
doc_id	Dokument-ID des d.3 Dokuments 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
url_link	HTTP-Link für die Ansicht in d.3 One Hinweis: Der Parameter ist nur gefüllt wenn d.3 One installiert ist.

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

Eigenschaft	Beschreibung
api_email_body_file	Name und Pfad einer Datei, die den Body-Text enthält

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

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

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

Ab d.3 Version 6.1.1

- Aufrufzeitpunkt: Nach dem Versenden einer E-Mail.
- Eingabeparameter:

Parameter	Beschreibung
`doc_id`	Dokument-ID des d.3 Dokuments 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

1.2.19. Stammdaten

hook_on_user_change_exit_10 (actionUser, newUserData, oldUserData)

ab d.3 Version 7.2.1

In diesem Einsprungspunkt können Anpassungen an einem gerade geänderten oder neu anzulegenden Benutzerobjekt vorgenommen werden.

Weder das Anlegen des Benutzer noch ein Ändern des Benutzers kann in diesem Hook verhindert werden. D.h., macht dieser Hook Änderungen am Benutzerobjekt, die nicht in die Datenbank geschrieben werden können (beispielsweise, weil die maximale Spaltenlängen überschritten wurden), wird das Objekt so angelegt als wäre der Hook nicht ausgeführt worden.

Wird der Hook mit einem Returncode ungleich "0" beendet, werden die Änderungen des Hooks am Benutzerobjekt ignoriert.

> **Anmerkung**
> Bei den Übergabe-Parametern handelt es sich um Objekt-Handles und nicht um einfache Zeichenfolgen.
>
> Diese Handles referenzieren Objekte, welche die eigentlichen Daten enthalten, welche dann mit den im folgenden beschriebenen Funktionen gelesen und geschrieben werden können.

`d3obj_property (object, property_name)`	Liefert den Wert des entsprechenden Properties zum übergebenen Objekt.
`d3obj_set_property (object, property_name, new_value)`	Setzt den Wert des angegebenen Properties für das übergebene Objekt auf den neuen Wert.

`d3obj_is_null (object)`	Liefert '0', wenn das Objekt NULL ist und somit nicht mit den zuvor genannten Funktionen genutzt werden kann. Falls das Objekt korrekt initialisiert ist, gibt die Funktion '1' zurück.

Name der Eigenschaft	Beschreibung	Lesen/ Schreiben
`id`	interne Kennung des d.3 Benutzers	nur Lesen
`long_name`	Anmeldekennung des Benutzers (Langname)	nur Lesen
`name`	tatsächlicher Name des Benutzers	nur Lesen
`email`	E-Mail Adresse des Benutzers	Lesen und Schreiben
`phone`	Telefonnummer des Benutzers	Lesen und Schreiben
`plant`	Werk des Benutzers	Lesen und Schreiben
`department`	Abteilung des Benutzers	Lesen und Schreiben
`is_checked_out`	'1', wenn der Benutzer als abwesend markiert ist, sonst '0'	nur Lesen
`checkout_text`	Abwesenheits-Benachrichtigung des Benutzers	nur Lesen
`is_sys_user`	gibt an, ob es sich um einen Systembenutzer handelt	nur Lesen
`opt_field[1-10]`	Optionale Felder zum Benutzer (Array-Indizes 1-10)	Lesen und Schreiben
`ldap_dn`	LDAP-DN des Benutzers (GUID bei `Active Directory`)	nur Lesen
`ldap_original_dn`	`distinguish Name` bei `Active Directory`	nur Lesen

Anmerkung

Sichern des Wertes aus Opt-Feld 5 in Opt-Feld 6, falls der Wert in Feld 5 geändert wurde:

```
{code}

proc hook_on_user_change_exit_10_test( actionUser, newUserData,
oldUserData )

{

// Get the id of the user that performs this change (most times
'd3_admin')

vars actionUserId = d3obj_property(actionUser, "id")

// Get the id of the user that has been changed

vars newUserId = d3obj_property(newUserData, "id")



// The old object is NULL, if the user was newly created

if( d3obj_is_null(oldUserData) )

call api_log_info("The user " ## actionUserId ## " has CREATED
the user " ## newUserId)

else

{

call api_log_info("The user " ## actionUserId ## " has CHANGED
the user " ## newUserId)

if( d3obj_property(newUserData, "opt_field[5]") != d3obj_proper-
ty(oldUserData, "opt_field[5]") )

{

call api_log_info("Opt field 5 was changed. Writing old value to
opt field 6")

vars oldOpt5 = d3obj_property(oldUserData, "opt_field[5]")

// Only the 'newUserData' is a mutable object ('actionUser' and
'oldUserData' are constant)

call d3obj_set_property(newUserData, "opt_field[6]", oldOpt5)



// Tell d.3 that our changes should be persisted

return 0

}

}
```

```
// Changes made in hook will be ignored, if we return something
other than '0'

return 1

}

{code}
```

1.2.20. Sperren eines Dokuments

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

hook_block_entry_10 (doc_id, user_name)

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, das zu blockieren / freizugeben ist
user_name	Name des rufenden Benutzers

hook_block_exit_10 (doc_id, user_name)

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, das zu blockieren / freizugeben ist
user_name	Name des rufenden Benutzers
error	Fehler, falls aufgetreten, sonst "0"

1.2.21. Statustransfer

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

hook_transfer_entry_30 (user_name, doc_id, archiv_index, source_logi, desti_logi, desti_user_o_group)

Parameter	Beschreibung
user_name	Name des rufenden Benutzers
doc_id	Dokument-ID des Dokuments, das transferiert werden soll
archive_index	Index der Version im Status "Archiv" (> 0), falls eine Version aus dem Status "Archiv" zurückgeholt werden soll
source_logi	Quellstatus des Dokuments (B, P, F, A)
desti_logi	Zielstatus des Dokuments (B, P, A)
destination_user_o_group	Zielstatus "Bearbeitung": Benutzer- oder Gruppenname Zielstatus "Prüfung": Gruppenname sonst leer

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

hook_transfer_exit_30 (user_name, doc_id, archiv_index, source_logi, desti_logi, desti_user_o_group, error)

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

1.2.22. Validieren der Eigenschaften vor der Kenndatenaktualisierung

- Aufrufzeitpunkt: Bei der Validierung vor der Kenndatenaktualisierung

> **Anmerkung**
> Seit d.3 Version 5.5.0.

hook_validate_update_entry_10 (user_name, doc_type_short)

Parameter	Beschreibung
`user_name`	Name des rufenden Benutzers
`doc_type_short`	Dokumentart Kürzel

1.2.23. Validieren der Eigenschaften vor Suchen bzw. Anlegen eines Dokumentes (ValidateAttributes)

> **Anmerkung**
> Ab der d.3 Version 5.5.1 Hotfix 2 gibt es den zusätzlichen Parameter `doc_type_short` für die Funktion
>
> `hook_validate_search_entry_10`

hook_validate_search_entry_10 (user_name, doc_type_short)

> **Anmerkung**
> 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`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart des zu importierenden Dokuments

hook_validate_import_entry_10 (user_name, doc_type_short)

> **Anmerkung**
> 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`	Name des rufenden Benutzers
`doc_type_short`	Kürzel der Dokumentart

> **Anmerkung**
> 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.

hook_validate_update_entry_10 (user_name, doc_type_short, doc_id)

Parameter	Beschreibung
user_name	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart
doc_id	Dokument-ID des d.3 Dokuments, dessen Kenndaten aktualisiert werden sollen

Wird eine Eigenschaft bei mehreren Dokumenten gleichzeitig mit Hilfe des `changeatt.dxp` geändert, greift der Einsprungpunkt `hook_validate_update_entry_10` nicht, da keine Validierung (ValidateAttributes) stattfindet.

Diese Validierung findet statt, wenn eine Eigenschaft bei einem Dokument anpasst wird.

hook_val_passwd_entry_10 (user_name, language, version)

Ausführung bei Validierung des Passwortes

Parameter	Beschreibung
user_name	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart
doc_id	Dokument-ID des d.3 Dokuments, dessen Kenndaten aktualisiert werden sollen

Wird eine Eigenschaft bei mehreren Dokumenten gleichzeitig mit Hilfe des `changeatt.dxp` geändert, greift der Einsprungpunkt `hook_validate_update_entry_10` nicht, da keine Validierung (ValidateAttributes) stattfindet.

Diese Validierung findet statt, wenn eine Eigenschaft bei einem Dokument anpasst wird.

hook_val_passwd_exit_10 (error, user_name, language, version)

Parameter	Beschreibung
user_name	Name des rufenden Benutzers
doc_type_short	Kürzel der Dokumentart
doc_id	Dokument-ID des d.3 Dokuments, dessen Kenndaten aktualisiert werden sollen

Wird eine Eigenschaft bei mehreren Dokumenten gleichzeitig mit Hilfe des `changeatt.dxp` geändert, greift der Einsprungpunkt `hook_validate_update_entry_10` nicht, da keine Validierung (`ValidateAttributes`) stattfindet.

Diese Validierung findet statt, wenn eine Eigenschaft bei einem Dokument anpasst wird.

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

hook_link_entry_10

reserviert

hook_link_entry_20

reserviert

hook_link_entry_30 (doc_id_father, doc_id_child)

> **Anmerkung**
> 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	Dokument-ID des übergeordneten Dokuments (i. a. Akte)
doc_id_child	Dokument-ID des untergeordneten Dokuments

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	Dokument-ID des übergeordneten Dokuments (i.a. Akte)
doc_id_child	Dokument-ID des untergeordneten Dokuments
error_code	0: Verknüpfung war erfolgreich -1: übergeordnetes und untergeordnetes Dokument 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 (siehe 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
error_number	0 = Ok sonst Datenbank- oder Hook-Fehler

hook_link_exit_20
reserviert

hook_link_exit_30
reserviert

1.2.25. Web-Veröffentlichung

- Aufrufzeitpunkt: Vor bzw. nach dem Veröffentlichen eines Dokuments für das Web.

hook_webpublish_entry_10 (doc_id, user_name, publish)

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, das veröffentlicht / zurückgezogen werden soll
user_name	Name des rufenden Benutzers
publish	0 = Dokument zurückziehen 1 = Dokument veröffentlichen

hook_webpublish_entry_20 (doc_id, user_name, doc_type_short, publish)

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, das veröffentlicht / zurückgezogen werden soll

Parameter	Beschreibung
user_name	Name des rufenden Benutzers
doc_type_short	Dokumentartkürzel
publish	0 = Dokument zurückziehen 1 = Dokument veröffentlichen

hook_webpublish_entry_30 (doc_id, user_name, doc_type_short, publish)

siehe hook_webpublish_entry_20

hook_webpublish_exit_10 (doc_id, user_name, error, doc_type_short, publish)

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, das veröffentlicht / zurückgezogen werden soll
user_name	Name des rufenden Benutzers
error	0= Aktion erfolgreich Fehlercode sonst
doc_type_short	Dokumentartkürzel
publish	0 = Dokument zurückziehen 1 = Dokument veröffentlichen

hook_webpublish_exit_20 (doc_id, user_name, error, doc_type_short, publish)

Parameter	Beschreibung
doc_id	Dokument-ID des Dokuments, das veröffentlicht / zurückgezogen werden soll
user_name	Name des rufenden Benutzers
doc_type_short	Dokumenartkürzel
error	0 = Aktion erfolgreich sonst Fehlercode
publish	0 = Dokument zurückziehen 1 = Dokument veröffentlichen

hook_webpublish_exit_30 (doc_id, user_name, error, doc_type_short, publish)

siehe hook_webpublish_exit_20

1.2.26. Workflow

hook_workflow_cancel_exit_20 (doc_id, wfl_id, step_id, user_name)

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	Dokument-ID
wfl_id	ID des Workflows
step_id	ID des Workflow-Schrittes
user_name	ID des d.3 Benutzers

1.2.27. 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.

1.3. Besondere Hook-Funktionen

1.3.1. 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.

Zur Erzeugung dynamischer Wertemengen (Wertemengenhooks)

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

- Implizite Parameter:

`repos_id`	Eigenschaften-ID (ID der Eigenschaft, für die der Hook gerufen wird)
`user`	aufrufender Benutzer
`doc_type_short`	Kürzel der Dokumentart
`row_no`	In Ergänzung zur `repos_id` die Zeilen bei Mehrfacheigenschaften (60er-Felder), 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 Eigenschaften-IDs

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

> **Anmerkung**
> Die übrigen `dok_dat`–Felder sind je nach Kontext gesetzt.

Suche: die übrigen Suchkriterien

Import: die übrigen bereits gefüllten Eigenschaften

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

Im Standard 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.

Hinzufügen von Werten für dynamische Wertemengen zu einer Benutzer-Auswahlliste

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.

Handelt es sich bei der Wertemenge nicht um eine Vorschlagswertemenge (Standard = 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 Eigenschaften-Feldes
`value`	hinzuzufügender Wert

Anmerkung

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

proc DW_FunktionsName( h_ReposID, h_User, h_DocTypeShort, h_Row-
No )

{

call api_log_debug( "++++++++START DW_FunktionsName( :h_Repo-
sID, :h_User, :h_DocTypeShort, :h_RowNo ) ++++++++" )

if(WERTEMENGEN_NICHT_CACHEN !=1)

{

call api_log_error( "Wertemengen werden aktuell gecached und wer-
den 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_Repo-
sID, :h_User, :h_DocTypeShort, :h_RowNo ) ++++++++" )

}// end of DW_FunktionsName
```

Zur individuellen Eingabe-Validierung (Plausibilitätshooks)

- Impliziter Parameter: `Feld-Inhalt`: Wert, der getestet werden soll
- Rückgabewert:

0 bei Erfolg

< > 0 bei Fehler

Anmerkung

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 Datenbankfelder stehen nicht zur Verfügung.

Anmerkung

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

// Eigenschaftenfelder 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
```

1.3.2. Dokumentklassen-Hooks

Dokumentklassen-Hooks dienen zur Bestimmung von Berechtigungen.

`@D3HOOK (<procedure_name>)`

- Import Parameter:

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

- Rückgabewert:

1: Berechtigt

0: kein Zugriff

> **Anmerkung**
> Die Verwendung von Dokumentklassen-Hooks kann zu Performance-Problemen bei der Rechteprüfung führen und somit die Dokumenten-Suche verlangsamen, insbesondere dann, wenn im Dokumentklassen-Hook SQL-Kommandos abgesetzt werden. Noch größere Auswirkungen auf die Performance bestehen, wenn Dokumentklassen-Hooks für Mehrfach-Eigenschaftsfelder (60er Felder) eingesetzt werden, da dieser Hook für jede mit einem Wert belegte Zeile des Eigenschaftsfeldes ausgeführt wird.

1.3.3. 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. Dokument-ID des aktuellen (zu verknüpfenden) Dokuments oder Akte

1.3.4. Lastverteiler

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 Dokuments, welches per Lastverteiler versendet werden soll
`object_id`	ID des Tätigkeitsprofils (aus den Tabellen `user_functions` und `object_types`)

Parameter	Beschreibung
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)

hook_holdfile_balance_entry_20 (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 Dokuments, 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)

hook_holdfile_balance_exit_10 (doc_id, object_id, ignore_checkout, user_name, alt_user_name)

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 Dokuments, 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
user_name	vom Lastverteiler gewählter Benutzer
alt_user_name	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)

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 Dokuments, welches per Lastverteiler versendet werden soll

Parameter	Beschreibung
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)

hook_holdfile_balance_exit_30 (doc_id, object_id, ignore_checkout, user_name)

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 Dokuments, 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
user_name	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)

1.3.5. 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.

> **Anmerkung**
> Der Text wird 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

1.4. Grundzüge der JPL-Programmierung

1.4.1. Allgemein

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

1.4.2. Prozeduraufbau

Man unterscheidet benannte und unbenannte Prozeduren.

Anmerkung
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
```

Anmerkung
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.

Warnung
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.

Anmerkung

```
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 Codezeile.

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.

Anmerkung

```
H_Zaehler=h_Zaehler1 + h_Zaehler2 + \

+h_Zaehler3 + h_Zaehler4 + \

+ h_Zzaehler5 \

\+h_Zaehler 6
```

1.4.3. Funktionsparameter

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

Anmerkung

```
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 Fehlercode an.

1.4.4. Prozeduraufruf

Allgemeiner Aufruf einer `JPL`-Prozedur:

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

Anmerkung

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

Damit eine `JPL`-Prozedur mit `call` aufgerufen werden kann, muss die `JPL`-Quelldatei, welches die Prozedur enthält, zuvor mit `public` in den Speicher geladen worden sein. Die Quelldatei 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.

Anmerkung

Beispiel

```
varsi j

retvarempf_variable(10)
```

Alternativ kann eine Prozedur auch folgendermaßen aufgerufen werden:

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

varsi j wert

wert = ermittle_wert (i)
```

1.4.5. 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 der `JPL`-Quelldatei verfügbar.

Variablennamen:

- Bestehen aus bis zu 31 Zeichen (Ziffern, Buchstaben, „_“).
- Das erste Zeichen darf keine Ziffer sein.
- Groß- und Kleinschreibung wird unterschieden.
- Im Standard 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)
```

> **Anmerkung**
> `varszaehler(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 maximal zwanzig Zeichen Länge.

1.4.6. Zeichenkettenverarbeitung

> **Anmerkung**
> 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
```

1.4.7. Operatoren

Operatortyp	Operatoren
numerisch	+, -, *, /
relational	<, <=, >, >=, ==, !=

Operatortyp	Operatoren
logisch	!, &&, ||, (NOT, AND, OR)
bitweise	~, &, |

1.4.8. Schleifen

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

> **Anmerkung**
>
> ```
> fori=1 while i<=100 step 1
> {
> …..
> }
> ```

```
while<logischer Ausdruck>
```

> **Anmerkung**
>
> ```
> while fehler == 0
> {
> …..
> }
> ```

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

`break`: springt aus einer Schleife

1.4.9. Verzweigungen

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

Anmerkung
Beispiel

```
if (zaehler == 100)

{

}

else if (zaehler < 100)

{

}

else

{

}
```

1.4.10. 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.

Anmerkung

```
vars datei1 datei2

datei1 = ...

datei2 = ...

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

1.4.11. 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 hingegen mit Log-Level 9.

> **Warnung**
> Die `msg`-Meldungen sollten Sie nur beim Testen der Hook-Funktionen benutzen. Im laufenden Betrieb sollten Sie diese nicht verwenden.

1.5. Datenbankzugang mittels JPL

1.5.1. 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 …
```

1.5.2. 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.

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

1.5.3. 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

1.5.4. 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).

> **Anmerkung**
>
> ```
> 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

1.5.5. 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:

Anmerkung

```
varsh_DocIdTesten(8)

h_DocIdTesten = "A0001234"

// Versuch1:

DBMS SQL SELECT dok_dat_feld_10\

FROM:D3_TABELLE_FIRMEN_SPEZIFISCH\

WHEREdoku_id = h_DocIdTesten

// Versuch2:

DBMS SQL SELECT dok_dat_feld_10\

FROM:D3_TABELLE_FIRMEN_SPEZIFISCH\

WHEREdoku_id = : h_DocIdTesten

// Versuch3:

DBMS SQL SELECT dok_dat_feld_10\

FROM:D3_TABELLE_FIRMEN_SPEZIFISCH\

WHEREdoku_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 „:“.

```
varszahl_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

1.6. Verbindungen zu anderen Datenquellen einrichten

1.6.1. 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).

> **Anmerkung**
> Beachten Sie die Performance der Datenbank.

1.6.2. 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.

1.6.3. Verbindungsabbau

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

Anmerkung

```
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.

Anmerkung

```
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
```

1.7. 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.

Warnung

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 Dokumenten-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.

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

1.7.1. Eigenschaftsfelder

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

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	Dokumenten-ID

Name	Datenbank-Spalte	Inhalt/ Zweck
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 Bemerkungsfeldes)
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 Eigenschaftsänderung
doc_field[X]	firmen_spezifisch.dok_dat_feld_X	Eigenschafts-Feld Nr. X (1-89)

1.7.2. Dokument-Metadaten

get_doc_property(docID, name)

Funktion `get_doc_property(docId, name)`

Gibt den Wert einer einzelnen Eigenschaft des jeweiligen Dokuments 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 Mehrfacheigenschaften nutzbar (siehe `get_doc_text()` und `get_doc_mult_attrib()`).

Parameter:

docId	Dokument-ID
name	Name der Eigenschaft (s.o.)

> **Anmerkung**
> `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 = :+hDo-
cId

DBMS ALIAS
```

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)

Anmerkung

```
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)
```

get_doc_mult_attrib(docID, fieldNo, rowNo, jplDestArray)

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

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

Parameter:

`docId`	Dokument-ID
`fieldNo`	Nummer des Eigenschaftsfelds (gültige Werte sind 60-69)
`rowNo`	Zeile des Eigenschaftsfelds . Ist dieser Wert gleich -1 werden alle Zeilen der Mehrfacheigenschaft in das Ziel-Array (`jplDestArray`) geschrieben.
`jplDestArray`	Name eines globalen JPL-Arrays, in welches die Zeilen des Eigenschaftenfeldes 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).

Anmerkung

```
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
```

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 Mehrfacheigenschaften nutzbar (siehe `set_doc_text()` und `set_doc_mult_val()`).

Parameter:

docId	Dokument-ID
name	Name der Eigenschaft (s.o.)
value	neuer Wert für die jeweilige Eigenschaft

Anmerkung

```
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
```

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 Dokument-ID
index	Index der Zeile (Gültige Werte sind 0 bis 3)
value	neuer Wert für die jeweilige Eigenschaft

Anmerkung

```
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) )

...
```

set_doc_mult_val(docId, fieldNo, rowNo, value)

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

Setzt den Wert einer Mehrfacheigenschaft des jeweiligen Dokumentes.

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

Parameter:

`docId`	Dokument-ID
`fieldNo`	Nummer des Eigenschaftenfeld (gültige Werte sind 60 bis 69)
`rowNo`	Zeile des Eigenschaftsfelds. Ist dieser Wert gleich "-1", werden alle Zeilen der Mehrfacheigenschaft 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 der Eigenschaft

Anmerkung

```
call set_doc_mult_val(hDocId, 60, 1, hMultVal)

if( !upd_doc_db_data(hDocId) )

...
```

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 Dokument-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".

1.7.3. Massenupdate von Dokument-Metadaten

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.

Anmerkung
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 Dokument-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".

finalize_upd_doc_db_data()
Funktion `finalize_upd_doc_db_data()`

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

1.7.4. Dokument-Suche und Massenverarbeitung

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.

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.

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`).

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

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

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.

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.

docSearchCurrentId()

Funktion `docSearchCurrentId()`

Gibt die Dokument-ID des aktuellen Treffers zurück.

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

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

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.

docSearchDestroy()

Funktion `docSearchDestroy()`

Bereinigt den internen Such-Handle.

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

1.7.5. Beispiele

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()
```

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()
```

1.8. Hook-Funktionen in Java

1.8.1. 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 Eigenschaftsfelder 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.

> **Anmerkung**
> Es werden nur die Java-Hooks unterstützt, die im Konfigurationsmodul der d.3 Administration ingestellt werden können. Insbesondere Werteauswahl-, Wertevalidierungs- und Dokumentklassenfilter-Hooks können nicht in der Programmiersprache Java umgesetzt werden.

1.8.2. 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:

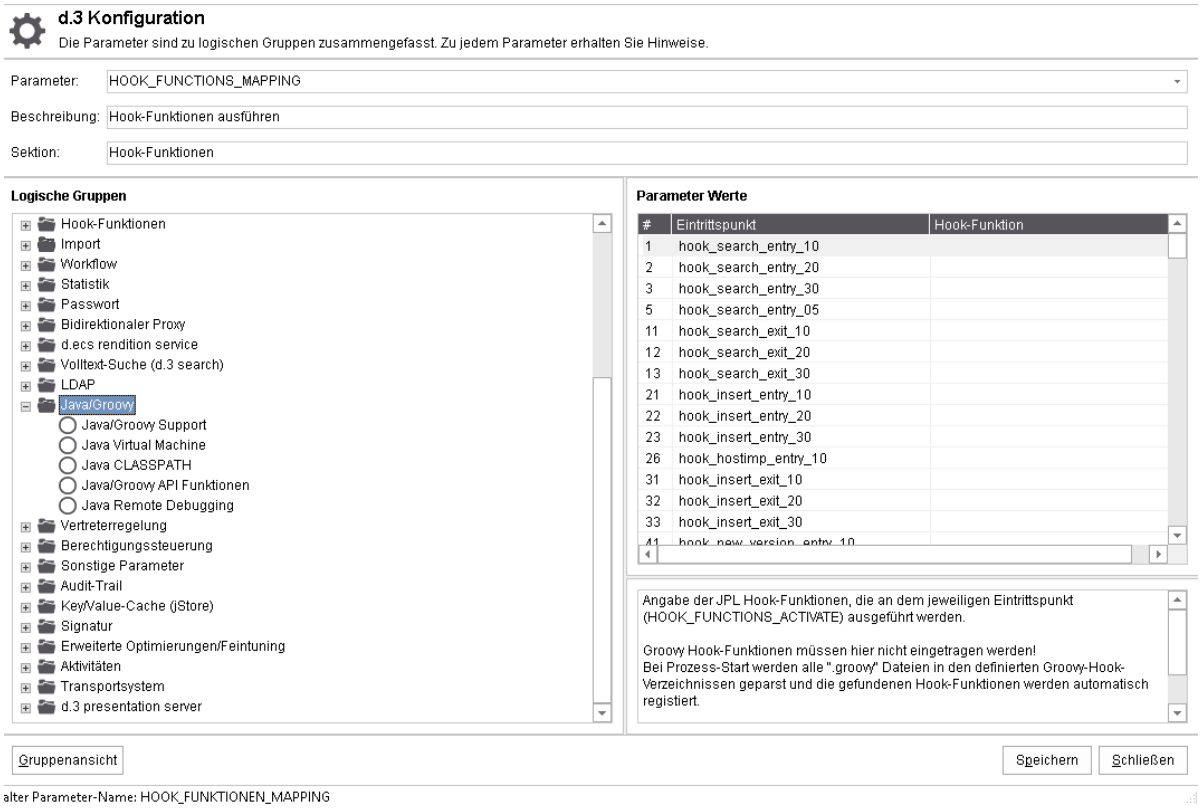
- Der d.3 Java-Support muss aktiviert sein.

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

- 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.

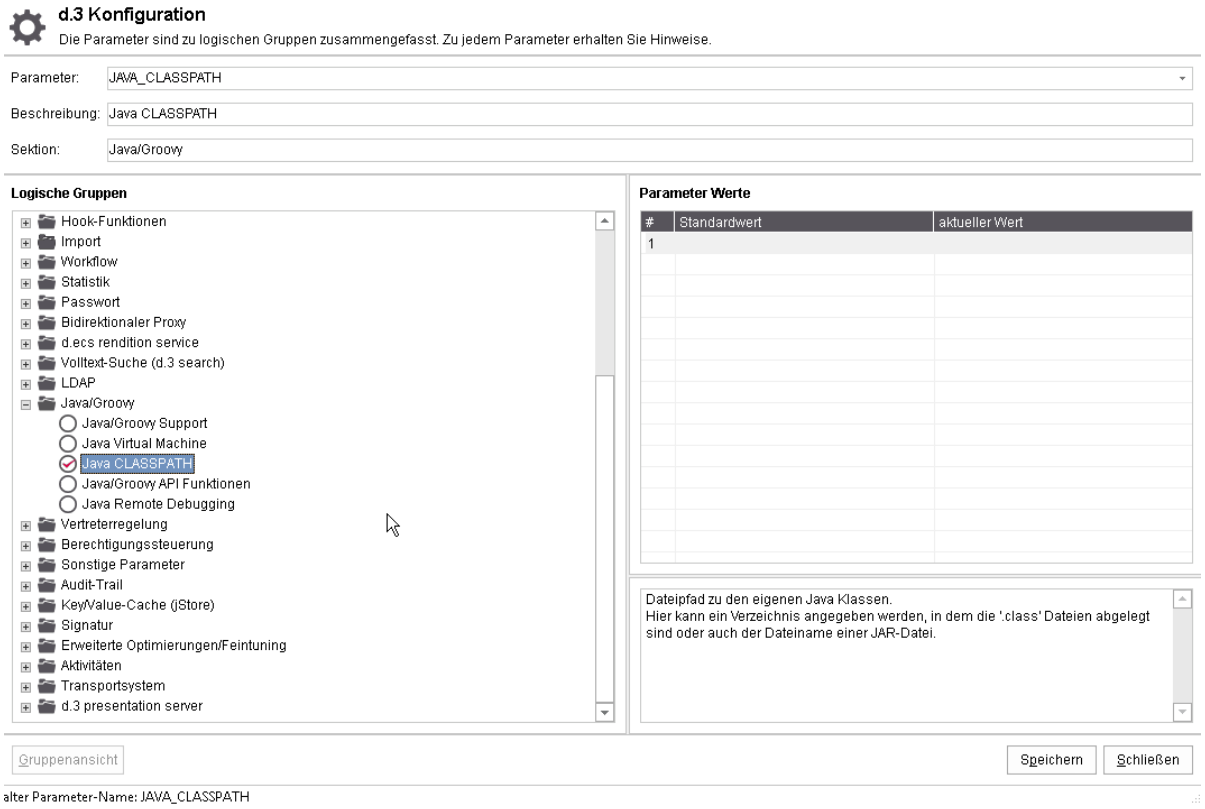
> **Anmerkung**
> Es wird empfohlen, die JRE von Sun ab Version 1.6 (Java 6) zu verwenden.



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

- 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.



- 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.

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

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.

> **Anmerkung**
> 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-Einsprungspunkt ist nicht vorgesehen.

1.8.3. 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.*;
```

> **Anmerkung**
> Bei der Kompilierung muss die `JAR`-Datei `pro5.jar` im Classpath angegeben sein.

Anmerkung

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.

Anmerkung

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

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);`	

Lesen von Variablenwerten	
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	
`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);`

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.

Ausführen eines Datenbank-Kommandos

```
int dm_dbms(String a1);
```

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

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

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

Kontrolle, ob eine Datenbankverbindung existiert bzw. noch gültig ist

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

1.8.4. 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 Plausibilitätshooks
- 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", "valueSe-
tHook")

}
```

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;
}
```

1.9. Beispiele

1.9.1. 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
}
}
}
```

1.9.2. 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

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

1.9.3. 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 Eigenschaften 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 \

WHEREA.doku_id = B.doku_id \

ANDA.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 Eigenschaften des neu importierten Dokuments in die korrespon-
dierenden
// 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 Erkennungseigenschaften für die Akte
//
ernie_job_attr_name[1] = "dokuart"
ernie_job_attr_value[1] = "akar"
laenge = @length(zeich_nr_hilf)
fori=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 Vererbungsseigenschaften vom untergeordneten Dokument 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, "")

}

}
```

1.10. Häufige Probleme und Fragen

1.10.1. Oracle Datenbankzugriff

Probleme beim Datumsformat

Problembeschreibung

Probleme beim Datumsformat unter Oracle Datenbanken.

Lösung

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

Anmerkung

```
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).

Anmerkung

```
DBMS ALIAS ccmindat,ccmaxdat DBMS SQL SELECT TO_CHAR(sys-
date -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_feld_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_...)
```

1.10.2. Aktenplan

Anlage untergeordneter Akten

Problembeschreibung

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

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.

Verknüpfungen unabhängig vom Aktenplan erstellen

Problembeschreibung

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

Lösung

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

Anmerkung

```
//

// Lösche die Inhalte der globalen Felder für die Verknüpfungsre-
geln //

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-/Verknüpfungseigenschaften angeben

//

// hier Einzelwerte der betr. 60er-Felder ermitteln ..

ernie_job_attr_name[1] = "dokuart"

ernie_job_attr_value[1] = "XXX" // Kürzel der Dokumentart eintra-
gen

ernie_job_attr_name[2] = "dok_dat_feld_1" // Bsp

ernie_job_attr_value[2] = "Verknüpfungswert aus 60er Feld eintra-
gen"

// ...



ernie_job_attr_name[3] = "folder_definition" // autom. Aktenanla-
ge

ernie_job_attr_value[3] = "1" // ist so eingeschaltet

//

// ggf. Vererbungsseigenschaften vom untergeordneten Dokument auf
die Akte angeben, wenn Eigenschaft in Akte noch nicht besetzt ist

//

inherit_field_name_1[1] = "dok_dat_feld_2"

inherit_field_value_1[1] = Variable mit Wert angeben

// ...

//
```

```
// ggf. Vererbungsseigenschaften vom untergeordneten Dokument auf
die Akte angeben, wobei Eigenschaften in Akte überschrieben wird

//

inherit_field_name_2[1] = "dok_dat_feld_3"

inherit_field_value_2[1] = Variable mit Wert angeben

// ...

//

// ggf. Vererbungseigenschaften von der Akte auf das untergeord-
nete Dokument angeben, wenn Eigenschaft in untergeordnetem Doku-
ment noch nicht besetzt ist

//

inherit_field_name_3[1] = "dok_dat_feld_4"

inherit_field_value_3[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 ver-
knüpft werden soll
```

Weitere Informationen

Die Variablen `inherit_field_XXX_1` und `inherit_field_XXX_2` sind globale Arrays. Sie vererben von Dokument auf Akte:

ohne Überschreiben	`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_XXX_3` oder `inherit_field_XXX_4` verwendet werden:

ohne Überschreiben	`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`.

Löschen aller Verknüpfungen eines Dokuments

Problembeschreibung
Wie kann ich aus einem Hook via `JPL` die Verknüpfungen eines Dokuments löschen?

Lösung
Die Server-API Funktion `link_delete(doc_id_parent, doc_id_child, user_name)` kann zum Entfernen von Verknüpfungen genutzt werden.

Innerhalb von `AttributeUpdate()` wird die Hook-Funktion `hook_upd_attrib_exit_10()` aufgerufen, welche als Einsprungpunkt dienen kann.

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

Anmerkung

```
proc hook_upd_attrib_exit_10_x(doc_id, error_number, user)

{

vars i

// Hier Kontrolle, ob Verknüpfungseigenschaft sich geändert hat

// Wenn ja, dann folgender Code:


// Alle übergeordneten Dateien ermitteln.

// Analog dazu kann auch "link_get_children" genutzt werden,

// um die untergeordneten Dateien zu ermitteln

call api_function("link_get_parents", doc_id, user)


//"api_single_info" enthält die Menge der gefunden Verkn.

for i=1 while i <= api_single_info step 1

{

call api_function("link_delete", api_links[i], doc_id, user)

}

return 0

}
```

1.10.3. Hinweise

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

Lösung

Nicht direkt. Bei `UpdateAttributes()` werden die im Client geänderten Eigenschaften 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`

1.10.4. d.3 Archivierung

Zu archivierende Dateitypen in der d.3 Archivierung festlegen

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.

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.

Anmerkung
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
```

Serverseitiges Verschlagworten von bereits importierten Dokumenten

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.

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

> **Anmerkung**
> --
>
> ```
> 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.

> **Anmerkung**
> `..\d3odbc32.exe haupt "" Master password d3o ext_jpl\ueberneh-me_ocr.jpl`

1.10.5. Sonstige Probleme

Sonderzeichen in einem Hook

Problembeschreibung

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

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;

Anmerkung
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 kleine Textdatei schreiben und daraus in `JPL`-Variable einlesen

1.10.6. Problematik bei der Verwendung von Datenbankfeld-Kontext und api_doc_field-Kontext Feldern

Die globalen Kontexte bringen einige Probleme, Risiken und Schwierigkeiten mit sich - daher sollte man sehr genau wissen was man tut.

Grundsätzlich wird der `api_doc_field`-Kontext nur von `JPL` Server-API Funktionen verwendet, dafür aber auch konsequent.

Eine Eigenschaftenaktualisierung über `attribute_update_*` interessiert die Werte im Datenbankfeld (`dok_dat_feld`) - Kontext überhaupt nicht.

Umgekehrt wird für alle anderen Vorgänge im d.3 Server der Datenbankfeld-Kontext verwendet (d.3 API Funktionen, hostimp und. auch für Async-Jobs aller Arten (auch UPD-Jobs)).

Sollte es im Hook erforderlich sein, einen Kontext in den anderen Kontext 1 zu 1 zu übernehmen, kann man die Funktionen `api_set_api_doc_field` (überträgt die Werte aus dem Datenbankfeld-Kontext in den `api_doc_field`-Kontext) und `api_set_dok_dat_feld` (überträgt die Werte aus dem `api_doc_field`-Kontext in den Datenbankfeld-Kontext) verwenden.

Aber insbesondere mit der zweiten Funktion muss man sehr vorsichtig umgehen.

Probleme und Risiken

- beide Kontexte sind letztlich globale Variablen und Arrays
- der api_doc_field-Kontext steht komplett in der Verantwortung des Hookentwicklers und wird nicht automatisch geleert; es ist also wichtig, diesen Kontext zum richtigen Zeitpunkt zu leeren
- der Datenbankfeld-Kontext wird vor neuen Vorgängen geleert; als Vorgang könnte man z.B. Import eines Dokuments (hostimp oder API) oder Aktualisierung von Dokumenteigenschaften (API, aber nicht Eigenschaftenvererbung) nennen
- besonders schwerwiegend ist der falsche Umgang mit dem Datenbankfeld-Kontext (genauere Beschreibung in den Beispielen unten)

Weitere Schwierigkeiten

- Bisher ist es für den Hookentwickler nicht immer ersichtlich, ob ein Hook von einer d.3 API Funktion oder von einer `JPL` Server-API Funktion aufgerufen wurde. Es gibt ein paar Hinweise in diversen globalen Variablen, aber es ist nicht immer eindeutig. Es erschwert dem Hookentwickler aber immer die Entscheidung, welcher der beiden Kontexte im Hook verwendet werden muss (und insbesondere, welcher Kontext nicht verändert werden darf).

- "Rekursive" Aufrufe von update-Hooks: Damit nicht zu viele Eigenschaftenaktualisierungen gleichzeitig ausgeführt werden können, erlaubt der Server intern nur noch zwei von diesen gleichzeitig und meldet ansonsten Fehler. Dies verhindert insbesondere ein falsches Hookdesign, indem unendlich viele Eigenschaftenaktualisierungen rekursiv aufgerufen werden (Eigenschaftsaktualisierung ruft Update-Hook ruft Eigenschaftsaktualisierung ruft Update-Hook ruft Eigenschaftenaktualisierung ruft ...) und einen Prozessabsturz vermuten lassen würden. Normalerweise genügen diese beiden geschachtelten Updates, da man in solcher Konstellation schon immer `SERVER_API_NO_HOOKS` verwenden musste. Möglicherweise gibt es sinnvolle Szenarien, in denen drei verschachtelte Updates erlaubt werden sollten, weil in dem Szenario der rekurive Aufruf der update-Hooks (also ohne `SERVER_API_NO_HOOK`S) keine Rolle spielt.

Anhand konstruierter Beispielszenarien sollen im Folgenden die Zustände der Kontexte beschrieben und dadurch die Probleme genauer erläutert werden.

Beispiel 1: Neuimport eines Dokuments (Client oder hostimp)

- `insert_entry`-Hooks: vor Ausführung der `insert_entry`-Hooks wird der Datenbankfeld-Kontext mit den mitgegebenen Eigenschaftswerten gefüllt (bei `ImportDocument` intern vom Servercode, beim hostimp durch Einlesen der `default.ini` und der JPL-Datei). Diese Eigenschaftswerte sind zu diesem Zeitpunkt an keiner anderen Stelle gesichert, daher muss vorsichtig mit ihnen umgegangen werden (der `api_doc_field`-Kontext ist zunächst leer). Was ist zu diesem Zeitpunkt möglich, was sollte man unterlassen oder aber zumindest mit sehr viel Vorsicht angehen?

  1. Durch Direktzuweisung eines Wertes können die Eigenschaftswerte verändert werden. Für das Dokument, das gerade importiert wird, ist eine Eigenschaftebaktualisierung über `attribute_update_*` zu diesem Zeitpunkt des Imports unmöglich (keine Dokument-ID bekannt), aber auch unnötig (Beispiel für Änderung einer Eigenschaft: `dok_dat_feld[1] = "ich bin ein geänderter Eigenschaftswert"`).
  2. Möchte man eine Eigenschaft (oder auch mehrere Eigenschaften) eines anderen Dokuments ändern, können dafür die `attribute_update_*`-Funktionen genutzt werden. Intern entsteht dadurch zunächst kein weiteres Problem, da die Eigenschaftswerte des anderen Dokuments in den `api_doc_field`-Kontext geladen werden. Die Gefahr ist aber, dass bei Aufruf eines update-Hooks der Datenbankfeld-Kontext verändert werden könnte, falls einer der update-Hooks nicht unterscheidet, ob er nun aus einer JPL Server-API Funktion heraus oder direkt von einer d.3 API Funktion aufgerufen wurde und einfach den Datenbankfeld-Kontext verändert. Sollte eine solche Eigenschaftsaktualisierung im `insert_entry`-Hook notwendig sein, ist die Verwendung von `SERVER_API_NO_HOOKS` zu empfehlen. In der Praxis wird es wahrscheinlich sowieso sinnvoller sein, eine solche Aktualisierung in einem insert_exit-Hook nach erfolgreichem Import vorzunehmen.
- Ablegen der Metadaten des neues Dokuments in der Datenbank: zu diesem Zeitpunkt werden die Eigenschaftswerte aus dem Datenbankfeld-Kontext gelesen und in der DB abgelegt
- `insert_exit`-Hooks: nachdem das Dokument in der Datenbank abgelegt ist, bleibt der Datenbankfeld-Kontext weiterhin erhalten, damit `insert_exit`-Hooks auf die jeweiligen Werte reagieren können. Eine Direktzuweisung auf ein Datenbankfeld wird im Fortlauf aber nicht mehr ausgewertet, kann hier also nicht mehr genutzt werden, um Eigenschaftswerte zu verändern. Allerdings können die Werte weiterhin relevant sein für diverse implizite Vorgänge, z.B. Aktenverknüpfungen und Eigenschaftsvererbung. Um Eigenschaftswerte des soeben importierten Dokuments in den `insert_exit`-Hooks zu verändern, müssen also `attribute_update_*` Funktionen verwendet werden (aus Sicht der Performance sollte man möglichst die Direktzuweisung in den `insert`-Hooks nutzen).

Beispiel 2: Aktualisierung von Dokumenteigenschaften(Client)

- Grundsätzlich gilt das Gleiche wie in Beispiel 1 für die `update_entry`- und `update_exit`-Hooks
- die Schwierigkeit mit den rekursiven Updates muss zwangsläufig über `SERVER_API_NO_HOOKS` gelöst werden

- werden `attribute_update_*`-Funktionen in den Hooks für andere Dokument ausgeführt als dem aktiven Dokument, werden zum Ende der Funktion die Dokumenteigenschaften des aktiven Dokuments wieder in den Datenbankfeld-Kontext geladen. Dies korrigiert ein möglicherweise stattgefundenes Überschreiben des Kontexts, kann aber nicht alle Probleme beseitigen. Befindet man sich zu dem Zeitpunkt in den insert-Hooks und hat dort über Direktzuweisung ein Datenbankfeld verändert, wird diese Veränderung nach der internen Korrektur (erneutes Laden der bisherigen Eigenschaftswerte) wieder verschwunden sein. Man muss also darauf achten, dass in dem Fall die Reihenfolge eingehalten wird (zuerst `attribute_update_*` auf andere Dokument-ID, dann Direktzuweisung von neuen Eigenschaftswerten auf dem aktiven Dokument.

1.10.7. Direkte Änderungen an bestimmten bzw. Systemtabellen nicht erlaubt

gültig ab d.3 Version 7.1.0 (Membase, bzw. Couchbase-Integration):

Alle Informationen zu einem Dokument, die für die Suche relevant sind, werden in einem Blob gespeichert.

Dies umfasst die vollständigen Informationen aus den Datenbank-Tabellen `PHYS_DATEI`, `FIRMEN_SPEZIFISCH`, `DOCUMENT_FL AGS` und `FIRM_SPEZ_MULT_VAL` sowie ausgewählte Informationen aus den Tabellen `DEPENDENT_DOCUMENTS`, `DOKUMENTEN_VERKNUEPF` und `DOKUMENTEN_ARCHIV`.

Direkte Änderungen an diesen Tabellen, welche nicht die d.3 APIs nutzen, führen also zu Inkonsistenzen zwischen Cache und DB!

Schreibende SQL-Operationen auf Dokument-Tabellen werden abgeblockt, sofern der Membase-Cache aktiviert ist. Alle d.3-eigenen Schreiboperationen auf diesen Tabellen kümmern sich um die Konsistenz des Caches.