i.s.h.med Parametrierbare Medizinische Dokumentation (PMD): Band 2

Fortgeschrittene Themen

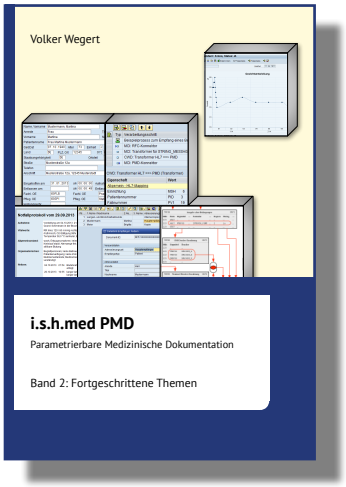
von
Volker Wegert

1. Auflage

tredition 2014

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 7323 0101 0

Zu Inhaltsverzeichnis



Aus dem Inhalt:

Wiederverwendbare Standard-Bausteine
Linkbausteine, Fremddatenbausteine

Fortgeschrittene User Exit-Programmierung
Objektorientierung, Alias-Konstanten, Generische Programmierung

Fortgeschrittene Dialogprogrammierung
Standarddialoge und eigene Popup-Fenster, Listenauswahl, Kontextmenüs, Rechtschreibprüfung, Volltextsuche, Docking-Container und amodale Dialoge, ...

Dokumentanzeige und -ausgabe
Standard-Kontextdaten, Orgmittelsteuerung, Versandsteuerung, HTML-Ausgabe mit XSLT

Dokumenttyp flexibler gestalten
Steuertabellen, Personalisierung, Text vorbelegen, Inhaltsvorlagen, Berechtigungsprüfung, Protokollierung, ...

Arbeit mit Dokumenten
Dokumentverwaltungsdaten, Zugriff auf Dokumentinhalte, Dokumente anlegen und aufrufen, HL7-Kommunikation mit MCI, ...

Technische Informationen
Namensvergabe, Umstellung von Generatorversion 1

Volker Wegert

i.s.h.med PMD
Band 2: Fortgeschrittene Themen

Dieses Buch setzt auf den in Band 1 vermittelten Grundlagen auf und vertieft eine Reihe inhaltlich und technisch anspruchsvollerer Themen. Sie lernen wiederverwendbare Standard-Bausteine und nützliche Komponenten der Entwicklungsumgebung kennen. Die fortgeschrittene User Exit- und Dialogprogrammierung wird ebenso vertieft wie verschiedene Erweiterungen der Druckausgabe. Dazu zählt insbesondere auch der Einsatz der Versandsteuerung und der Orgmittelsteuerung. Das Buch geht auch auf den externen Zugriff auf Dokumentdaten — etwa aus anderen Dokumenten oder eigenen Anwendungen — sowie auf den Umgang mit Dokumentverwaltungsdaten ein. Ein Kapitel beschreibt ausführlich die Entwicklung eigener Linkbausteine. Das Buch schließt mit technischen Hintergrundinformationen zur Namensvergabe der generierten Objekte und zur Umstellung von Dokumenttypen, die noch mit Generatorversion 1 erstellt wurden.

354 Seiten, Hardcover, 2014, 69,90 €
ISBN 978-3-7323-0101-0
http://www.pmd-buch.de

1 Wiederverwendbare Standard-Bausteine

In diesem Abschnitt werden einige Linkbausteine und Fremddatenbausteine vorgestellt, deren genaue Beschreibung den Umfang des ersten Bands gesprengt hätte. Sie können diese Bausteine gemäß der hier angegebenen Vorgehensweisen in Ihren Dokumenttypen einsetzen.

1.1 Linkbaustein DOCVIEWER

Linkbaustein / Linkelement (Konzept)
1 / 2.3.3.1 / S. 72

Der Linkbaustein `DOCVIEWER` kann eingesetzt werden, um innerhalb eines Dokuments andere Dokumente anzuzeigen. Damit ist es zum Beispiel möglich, verwandte Daten anzuzeigen, einen Vergleich mit historischen Dokumenten anzubieten oder Quelldokumente zur Befundübernahme bereitzustellen. Die Grundfunktion des Linkbausteins `DOCVIEWER` ist auch mit der Drucktaste Daten holen in der generierten Dialoganwendung verfügbar; mit dem Linkbaustein ist allerdings eine feinere Steuerung der Anzeige möglich. Abbildung 1.1 zeigt den Linkbaustein in einer möglichen Konfiguration.

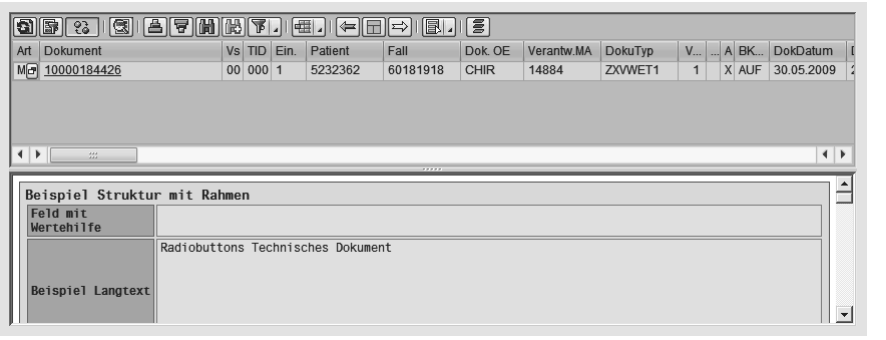


Abbildung 1.1: Linkbaustein `DOCVIEWER`

Linkbaustein DOCVIEWER

Der Linkbaustein bietet verschiedene Einstellungsmöglichkeiten, die mit der Drucktaste Werkzeuge in der Dokumenttypdefinition aufgerufen werden können. Diese erscheint erst, wenn der Linkbaustein `DOCVIEWER` im Dokelement eingetragen wurde. Abbildung 1.2 zeigt den Einstellungsdialog. Aus diesem Dialog erreichen Sie mit der Drucktaste i die Online-Dokumentation des Linkbausteins.

Mit der Einstellung *Steuerelement vertikal teilen* können Sie steuern, ob die Liste der Dokumente links neben dem Inhalt des ausgewählten Dokuments oder – wie in

⇒ [PMD-DV]

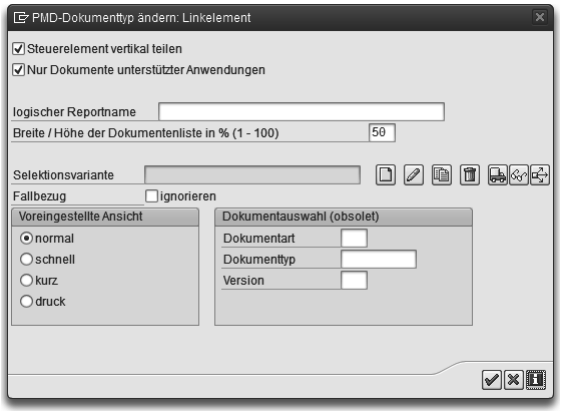


Abbildung 1.2: Einstellungen des Linkbausteins DOCVIEWER

Abbildung 1.1 gezeigt – oberhalb des Dokumentinhalts angezeigt wird. Weiterhin können Sie die *Breite / Höhe der Dokumentenliste* prozentual einstellen.

Mit dem Kennzeichen *Nur Dokumente unterstützter Anwendungen*, das standardmäßig gesetzt ist, werden in der Liste nur PMD- und Wordcontainer-Dokumente sowie DTA-Dokumente mit ausgewählten Datei-Endungen[1], die im Linkbaustein angezeigt werden können, aufgeführt. Dieses Kennzeichen sollten Sie normalerweise nicht entfernen. Die eigentliche Auswahl der anzuzeigenden Dokumente wird über eine sogenannte *Selektionsvariante* festgelegt, die Sie mit den Drucktasten im Dialog anlegen, bearbeiten, kopieren, löschen und transportieren können.

Sie müssen die Selektionsvariante manuell transportieren – beim Transport des Dokumenttyps wird die Selektionsvariante nicht automatisch transportiert. Verwenden Sie dazu die Drucktaste im Dialog oder den Report RSTRANSP unter Angabe des Programmnamens RN2DOCVIEWER_SVAR.

Abbildung 1.3 zeigt die zur Verfügung stehenden Selektionsoptionen. Es werden immer nur die Dokumente zum aktuellen Patienten angezeigt; mit der Option *Fallbezug ignorieren* können Sie steuern, ob auch der Fallbezug berücksichtigt werden soll. Die Einstellungen des Bereichs *Dokumentauswahl* sind obsolet und sollten nicht mehr verwendet werden.

Das Listen-Layout, das zur Anzeige der auswählbaren Dokumente verwendet wird, ist mithilfe von Layoutvarianten einstellbar. Da ALV-Listenlayouts immer nur mit Bezug zu einem Rahmenprogramm gespeichert werden können, müssen Sie dazu im Feld *logischer Reportname* ein Programm angeben. Dazu können Sie prinzipiell den Namen der generierten Funktionsgruppe inklusive des Präfix SAPL verwenden;

[1] siehe Hinweis 967474

Abbildung 1.3: Selektionsvariante des Linkbausteins DOCVIEWER

Sie können aber auch ein funktionsloses Programm anlegen, das lediglich als Träger des Layouts dient.

Wenn Sie den Linkbaustein innerhalb eines Dokumenttyps mehrfach verwenden und dabei unterschiedliche Listenlayouts nutzen möchten, müssen Sie verschiedene logische Reportnamen angeben.

Das Listen-Layout können Sie nur innerhalb des Dokuments mit den Standard-Funktionen der Listenanzeige bearbeiten. Zur Pflege systemweiter Standardlayouts benötigen Sie die Berechtigung S_ADMI_FCD mit der Funktion TCTR.

Auch die Layoutvarianten werden nicht automatisch transportiert; verwenden Sie dazu die Funktion Layout » Transportieren der Variantenverwaltung bzw. des Reports RKKBALVI.

Mit der Option *Voreingestellte Ansicht* können Sie steuern, welche Aufbereitung im Anzeigebereich des Linkbausteins verwendet werden soll. Diese Einstellung ist insbesondere dann relevant, wenn Sie XSLT-Programme verwenden, um eine spezialisierte Aufbereitung zu realisieren. In diesem Fall steuert die voreingestellte Ansicht, welches XSLT-Programm zur Aufbereitung verwendet wird.

XSLT-Aufbereitung 4.4 / S. 139

1.2 Linkbaustein HTMLVIEWER

Mit dem Linkbaustein HTMLVIEWER können Sie einen Browser zur Anzeige von HTML-Dateien, Internet- oder Intranet-Seiten in die Dokumentationsanwendung integrieren. Damit können Sie Bedienungshinweise, medizinische Informationen oder Leitlinien bereitstellen oder – in gewissen Grenzen – auch web-basierte Anwendungen integrieren. Abbildung 1.4 zeigt eine mögliche Anwendung des Linkbausteins.

2.2 Alias-Konstanten

2.2.1 Zielsetzung

Bei der Vorstellung der verschiedenen API-Methoden haben Sie erfahren, dass die verschiedenen Elemente eines Dokumenttyps durch ihren Alias adressiert werden. Bisher wurden dazu in den Quelltextbeispielen immer Zeichenkettenliterale verwendet:

```
* ...
  <pub__service>->api__get_value(
    EXPORTING
      i_alias = 'MEINALIAS'
    IMPORTING
      e_value = l_wert ).
  l_wert = l_wert + 42.
  <pub__service>->api__set_value( i_alias = 'MEINAILAS'
                                  i_value = l_wert ).
* ...
```

Diese Vorgehensweise ist leicht zu vermitteln und einfach umzusetzen, birgt allerdings ein Problem: Das System kann während der statischen Prüfungen nicht zwischen inhaltlich korrekten und fehlerhaften Aliasangaben unterscheiden. Für den ABAP-Compiler sind alle gültigen Zeichenkettenliterale als Wert akzeptabel, auch wenn es kein Dokelement mit dem angegebenen Alias gibt.

Im obigen Beispiel hat sich ein Schreibfehler in den Aufruf von `api__set_value` eingeschlichen. Dieser Fehler kann während der Übersetzung nicht gefunden werden, sondern erst bei der Ausführung des User Exits. Dies ist für die Entwicklung relativ unglücklich, weil Fehler erst sehr spät gefunden werden und dadurch aufwändige Tests erforderlich werden. In diesem Kapitel wird eine Möglichkeit vorgestellt, dieses Problem zu vermeiden.

2.2.2 Voraussetzungen

Zur Anwendung der hier beschriebenen Technik sind keine besonderen Vorkenntnisse erforderlich. Der Dokumenttyp muss prinzipiell in keiner Weise vorbereitet werden; auch vorhandene User Exits müssen nicht umgestellt werden. Es ist allerdings hilfreich, wenn Sie nach der Anlage des Dokumenttyps zuerst die Dokelement-Alias nach Ihren Wünschen anpassen, bevor Sie mit der Umsetzung dieser Technik beginnen.

2.2.3 Vorgehensweise

Die Grundidee ist relativ simpel: Definieren Sie einmal zentral eine Konstante mit dem Alias jedes Dokelements und verwenden Sie im Programmtext ausschließlich diese Konstanten. Auf diese Weise können Sie maximal einen Schreibfehler erzeugen, nämlich bei der Konstantendefinition – und selbst dieser Fehler lässt sich durch geschickten Werkzeugeinsatz vermeiden. Jeder weitere Schreibfehler bei der Angabe einer Konstante führt zu einem Syntaxfehler, weil das entsprechende – falsch geschriebene – Symbol nicht definiert wurde.

Grundsätzlich können Sie die Konstantendefinitionen manuell erstellen und in das Daten-Include aufnehmen:

```
1     CONSTANTS zca_master  TYPE N2_ALIAS1 VALUE 'MASTER'.
2     CONSTANTS zca_patnr   TYPE N2_ALIAS1 VALUE 'ZPATNR'.
3   * ...
```

Dies ist aber unnötig viel Schreibarbeit. In den folgenden Kapiteln werden daher zwei Verfahren vorgestellt, mit denen Sie sich diese Arbeit erleichtern können.

Diese Vorgehensweise bietet zwei weitere Vorteile:

- Wenn Sie den neuen ABAP-Editor verwenden und in den Einstellungen im Bereich *Code-Vervollständigung* die Option *Nicht-Schlüsselwörter aus dem Text vorschlagen* (Abbildung 2.3) aktiviert haben, erhalten Sie die Konstantennamen als Vorschlag zur Code-Vervollständigung. Das kann bei sinnvoller Wahl der Konstantennamen Tipparbeit sparen.

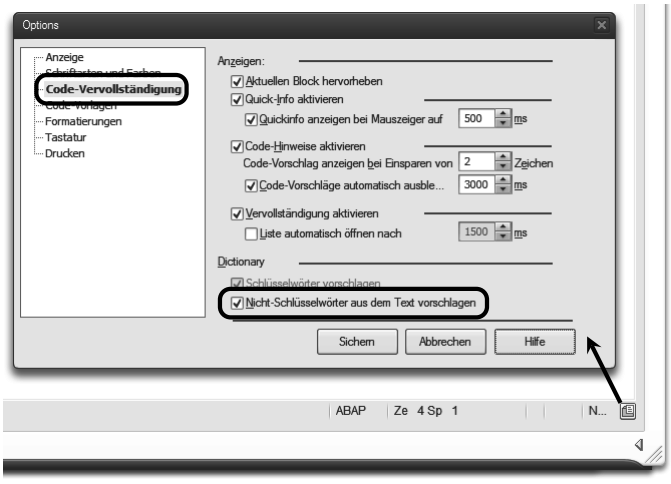


Abbildung 2.3: Editor-Einstellung zum Vorschlag der Konstantennamen

4 Dokumentanzeige und -ausgabe

In diesem Abschnitt erfahren Sie, welche weiteren Möglichkeiten Ihnen zur Gestaltung der (Druck-)Ausgabe und der Anzeige von Dokumenten im System zur Verfügung stehen.

4.1 Standard-Kontextdaten bereitstellen

4.1.1 Zielsetzung

Während es beim Entwurf der Dokumentationsanwendung sinnvoll ist, bereits im System vorhandene Daten nicht noch einmal im Dokumenttyp redundant vorzuhalten, ist es doch bei der Ausgabe oft erforderlich, bestimmte Standarddaten mit auszugeben. Wenn Sie etwa den Fallkopf verwenden, um Patienten- und Falldaten im Dialog darzustellen, werden die Daten nicht noch einmal im Dokument gespeichert. Es gibt also keine Dokelemente, die Sie zum Ausdruck dieser Angaben verwenden könnten. Deshalb ist es notwendig, die Angaben anderweitig zum Druck aufzubereiten. Für viele grundlegende Entitäten in IS-H gibt es dafür vorbereitete Datenstrukturen und Funktionen zur Aufbereitung der Daten. In diesem Kapitel erfahren Sie, wie Sie diese Strukturen nutzen können.

Fallkopf
1 / 2.2.3 / S. 40

4.1.2 Voraussetzungen

Sie sollten über grundlegende Programmierkenntnisse verfügen und sich sicher in der ABAP-Entwicklungsumgebung und dem Data Dictionary bewegen können. Der Dokumenttyp muss nicht besonders vorbereitet werden; es ist allerdings sinnvoll, wenn die Entscheidung über die einzusetzende Ausgabetechnologie bereits getroffen wurde.

4.1.3 Vorgehensweise

Im Folgenden werden die Datenstrukturen und Versorgungsfunktionen kurz dargestellt; anschließend wird die Verwendung in SAPscript- und SIFbA-basierten Druckanwendungen erläutert.

4.1.3.1 Strukturen und Funktionen

Die hier vorgestellten Strukturen stammen aus dem IS-H-Formulardruck und sind speziell zur Ausgabe von Daten vorbereitet worden. Nicht alle der Strukturen sind zur Nutzung in Dokumentationsanwendungen von Interesse – mit Kostenübernahme- und Rechnungsdaten wird man sich im Rahmen einer medizinischen Dokumentation selten beschäftigen. Deshalb ist in Tabelle 4.1 nur eine Auswahl der verfügbaren Strukturen wiedergegeben. Die Tabelle enthält auch die Angabe, mit welchen Funktionsbausteinen die angegebenen Strukturen gefüllt werden können; diese Funktionsbausteine befinden sich fast ausschließlich in der Funktionsgruppe NPR2. Einige Strukturen können mit mehreren Bausteinen gefüllt werden; hier sind nur die jeweils gebräuchlichsten Bausteine aufgeführt.

4.1.3.2 Verwendung in SAPscript-Umgebungen

Zusatzdaten in SAPscript
1 / 14.3.5 / S. 489

Die Verwendung der hier vorgestellten RNF-Strukturen in SAPscript-Formularen folgt der in Band 1 beschriebenen allgemeinen Vorgehensweise zur Ausgabe zusätzlicher Daten. Sie können die RNF-Strukturen sowohl in eine – eventuell bereits bestehende – Druckdaten-Struktur einbetten als auch direkt verwenden; letzteres ist bei der späteren Eingabe im Formular handlicher. Im Daten-Include könnten Sie also z. B. folgende Zeilen eintragen:

```
DATA: zs_rnf01 TYPE rnf01, " Patientendaten
      zs_rnf05 TYPE rnf05, " Falldaten
      zs_rnf06 TYPE rnf06. " Aufnahmedaten
```

Verwenden Sie anschließend den User Exit *Druck-Anfang*, um diese Datenstrukturen zu befüllen. Mit Hilfe der in Tabelle 4.1 angegebenen Funktionsbausteine könnte dies z. B. mit der folgenden Routine erfolgen:

```
FORM z_fill_rnf_structures.

  DATA: ls_ndoc TYPE ndoc,
        ls_nfal TYPE nfal.

* falls irgendetwas schiefgeht, dürfen keine alten Daten in den Variablen stehen
  CLEAR: zs_rnf01, zs_rnf05, zs_rnf06.

* Bezugsdaten besorgen
  ls_ndoc = <pub__service>->api__get_ndoc( ).
  ls_nfal = <pub__service>->api__get_nfal( ).

* Patientendaten ermitteln
  CALL FUNCTION 'ISH_FILL_RNF01'
    EXPORTING
      i_einri = ls_ndoc-einri
      i_patnr = ls_ndoc-patnr
    IMPORTING
      e_rnf01 = zs_rnf01.
```

Struktur	Inhalt	Versorgungsbaustein	Eingabedaten
RNF01	Patientendaten	ISH_FILL_RNF01	EINRI, PATNR
RNF02	Arbeitgeber Patient	ISH_FILL_RNF02	EINRI, PATNR
RNF03	Angehörige Patient	ISH_FILL_RNF03	EINRI, PATNR
RNF031	Angehöriger 1	ISH_FILL_RNF03	EINRI, PATNR
RNF032	Angehöriger 2	ISH_FILL_RNF03	EINRI, PATNR
RNF04	weitere Patientendaten	ISH_FILL_RNF04	EINRI, PATNR
RNF05	Falldaten	ISH_FILL_RNF05	EINRI, FALNR
RNF06	Aufnahme	ISH_FILL_RNF06 ISH_FILL_RNF06_AMB	EINRI, FALNR, (LFDBEW) EINRI, FALNR, LFDBEW
RNF07	Einweisungsdaten	ISH_FILL_RNF07 ISH_FILL_RNF07_AMB	EINRI, PATNR, FALNR EINRI, PATNR, FALNR, LFDBEW
RNF071	ein-/überweisender Arzt	ISH_FILL_RNF07 ISH_FILL_RNF07_AMB	EINRI, PATNR, FALNR EINRI, PATNR, FALNR, LFDBEW
RNF072	einweisendes Krankenhaus	ISH_FILL_RNF07 ISH_FILL_RNF07_AMB	EINRI, PATNR, FALNR EINRI, PATNR, FALNR, LFDBEW
RNF073	Hausarzt	ISH_FILL_RNF07 ISH_FILL_RNF07_AMB	EINRI, PATNR, FALNR EINRI, PATNR, FALNR, LFDBEW
RNF08	Entlassung	ISH_FILL_RNF08	EINRI, PATNR, FALNR
RNF081	weiterbehandelndes Krankenhaus	ISH_FILL_RNF08	EINRI, PATNR, FALNR
RNF082	weiterbehandelnder Arzt	ISH_FILL_RNF08	EINRI, PATNR, FALNR
RNF09	Begleitperson	ISH_FILL_RNF09	EINRI, PATNR, FALNR
RNF10	Geburts- und Entbindungsdaten	ISH_FILL_RNF10	EINRI, PATNR, FALNR, NFAL, NGEB
RNF11	Kostenträgerdaten (Versicherung)	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF111	Name, Anschrift Kostenträger	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF112	Hauptversicherter	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF113	Arbeitgeber Hauptversicherter	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF114	Kostenträgerdaten (Zentrale)	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF115	abweichende Rechnungsadresse Selbstzahler	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF116	FI-Daten Regulierer	ISH_FILL_RNF11 ISH_FILL_RNF11_TAB	EINRI, FALNR, NCIR EINRI, FALNR, NCIR[], NGPA[], ...
RNF15	Aufnahme- und Einweisungsdiagnose	ISH_FILL_RNF15	NDIA[]
RNF16	Entlassungs- und Hauptdiagnose	ISH_FILL_RNF16	NDIA[]
RNF17	Behandlungsdiagnose	ISH_FILL_RNF17	NDIA
RNF17L	Langtext Behandlungsdiagnose	ISH_FILL_RNF17	NDIA
RNF29	Abrechnungsdaten Operationen/Prozeduren	ISH_FILL_RNF30	EINRI, FALNR
RNF30	Operationen (Prozeduren, max. 15)	ISH_FILL_RNF30	EINRI, FALNR
RNF44	Risikofaktoren (max. 10 → RNF61)	ISH_FILL_RNF44	NRSF[]
RNF57	Termindaten	ISH_FILL_RNF57	NAPP
RNF61	Risikofaktor	ISH_FILL_RNF61 ISH_FILL_RNF61_TAB	NRSF NRSF[]
RNF62	Prozedur zum Fall	ISH_FILL_RNF30 ISH_FILL_RNF62	EINRI, FALNR EINRI, FALNR
RNF63	Prozedur mit Leistungen	ISH_FILL_RNF62	EINRI, FALNR
RNF64	Prozedur mit Diagnose	ISH_FILL_RNF30 ISH_FILL_RNF62	EINRI, FALNR EINRI, FALNR
RNF68	Diagnose	ISH_FILL_RNF68	NDIA[]
RNF69	DRG-Daten	ISH_FILL_RNF69	EINRI, NDRG

Tabelle 4.1: RNF-Strukturen und Versorgungsbausteine

```
20
21  * Falldaten ermitteln
22    CALL FUNCTION 'ISH_FILL_RNF05'
23      EXPORTING
24        i_einri          = ls_nfal-einri
25        i_falnr          = ls_nfal-falnr
26        i_read_nfal_flag = abap_false " nicht erneut lesen
27        i_nfal           = ls_nfal
28      IMPORTING
29        e_rnf05          = zs_rnf05.
30
31  * je nach Fallart Aufnahme- oder Behandlungsdaten ermitteln
32    IF ls_nfal-falar = '2'. " ambulant
33      CALL FUNCTION 'ISH_FILL_RNF06_AMB'
34        EXPORTING
35          i_lfdnr          = ls_ndoc-lfdbew " Achtung: Erfordert Bewegungsbezug!
36          i_einri          = ls_nfal-einri
37          i_falnr          = ls_nfal-falnr
38          i_read_nfal_flag = abap_false " nicht erneut lesen
39          i_nfal           = ls_nfal
40        IMPORTING
41          e_rnf06          = zs_rnf06.
42    ELSE. "stationär oder teilstationär
43      CALL FUNCTION 'ISH_FILL_RNF06'
44        EXPORTING
45          i_einri          = ls_nfal-einri
46          i_falnr          = ls_nfal-falnr
47          i_read_nfal_flag = abap_false " nicht erneut lesen
48          i_nfal           = ls_nfal
49        IMPORTING
50          e_rnf06          = zs_rnf06.
51    ENDIF.
52
53  ENDFORM.
```

Verwenden Sie die so bereitgestellten Daten als normale strukturierte Symbole im SAPscript-Formular:

```
1  *  Fallnummer:,,&ZS_RNF05-CASENO&
```

4.1.3.3 Verwendung in SIFbA-Umgebungen

SIFbA-Ausgabe
1 / 14.4 / S. 499

Wenn Sie zusätzliche Daten zum Druck in einem SIFbA-Formular bereitstellen möchten, müssen Sie die in Band 1 beschriebene Datenstruktur verwenden. Sie können die RNF-Strukturen dabei wie in Abbildung 4.1 gezeigt als Teilstrukturen in Ihre Struktur einfügen oder inkludieren.

Die Versorgung dieser Strukturen erfolgt nach dem gleichen Verfahren, das bereits für SAPscript-Formulare vorgestellt wurde. Die Formularschnittstelle ändert sich durch diese Erweiterungen nicht. Im Formular selbst können Sie die RNF-Strukturen, die Sie jetzt unterhalb des Knotens ADDITIONAL_DATA finden, direkt in den Formularkontext ziehen und verwenden (Abbildung 4.2). Beachten Sie, dass Sie auch hier nur die Felder aktivieren sollten, die auch tatsächlich verwendet werden.

5.3 Personalisierung

5.3.1 Zielsetzung

In Abschnitt 5.1 auf Seite 157 haben Sie ein Verfahren kennengelernt, mit dem Sie Dokumenttypen mit beinahe beliebigen Steuerungsmöglichkeiten ausstatten können. Dabei haben Sie aus technischer Sicht völlig freie Wahlmöglichkeit was die Schlüssel der Steuertabellen angeht; Sie könnten also prinzipiell auch eine Tabelle erstellen, in der Sie benutzerbezogene Einstellungen hinterlegen. Es ist allerdings aus mehreren Gründen keine gute Idee, so vorzugehen:

- Die zusätzlichen Einstellungen sind der Benutzerverwaltung nicht bekannt und können daher bei keinerlei Benutzeränderungen berücksichtigt werden. Wenn ein Benutzer kopiert wird, werden diese Einstellungen nicht mit kopiert; wenn ein Benutzer gelöscht wird, bleibt ein verwaister Eintrag in der Steuertabelle zurück.
- Die Einstellungen in der Tabelle sind für den Benutzer nur dann pflegbar, wenn Sie eine Zugriffsmöglichkeit dafür programmieren – die normalen Tabellenpflegeanwendungen zeigen die gesamte Tabelle und sind damit benutzerübergreifend, was in diesem Fall in der Regel nicht gewünscht ist.
- Bei intensiver Verwendung dieses Verfahrens sind die Einstellungen zu einem Benutzer über eine Vielzahl von Orten verstreut und damit für den Supportbetrieb schwierig zu handhaben.

Um benutzerbezogene Einstellungen zu hinterlegen, bietet sich das Verfahren der **Personalisierung** an, das in diesem Abschnitt vorgestellt wird.

Personalisierung

⇒ [HP-PER]

5.3.2 Voraussetzungen

Zum Einsatz der hier beschriebenen Techniken sollten Sie mit den Grundlagen der ABAP-Entwicklung vertraut sein; insbesondere müssen Sie in der Dialogentwicklung relativ sicher sein. Der Dokumenttyp muss nicht besonders vorbereitet werden; Sie sollten allerdings relativ genau wissen, welche Einstellungen Sie erlauben möchten.

5.3.3 Vorgehensweise

Um die Personalisierung einzusetzen, müssen Sie im einfachsten Fall eine Ablagestruktur und einen Pflegedialog bereitstellen, das Personalisierungsobjekt registrieren und die Lesezugriffe sowie optional den Aufruf des Pflegedialogs in den

Dokumenttyp integrieren. Auch hier wird wieder die Protokollierung des Benutzernamens in der Notiz des Notfallprotokolls realisiert – ein Beispiel, das bereits in Abschnitt 5.1 auf Seite 157 verwendet wurde.

In den ausgelieferten Beispiel befindet sich das BC-Set `/PMDBUCH/PER_DEMOPER_V01` mit dem Dokumenttyp `DEMOPER`, der genau die hier vorgestellte Lösung abbildet. Die Entwicklungsumgebungsobjekte dazu befinden sich im Paket `/PMDBUCH/PER`.

5.3.3.1 Ablagestruktur bereitstellen

Die Ablagestruktur ist eine einfache Dictionary-Struktur, die die zu speichernden Daten aufnimmt. Es gibt gewisse Restriktionen hinsichtlich der möglichen Datentypen, die aber für diesen Anwendungszweck irrelevant sind; Details dazu finden Sie in der Online-Dokumentation. Für unser Beispiel kann die Struktur sehr einfach gehalten werden; Abbildung 5.14 zeigt ein Beispiel.

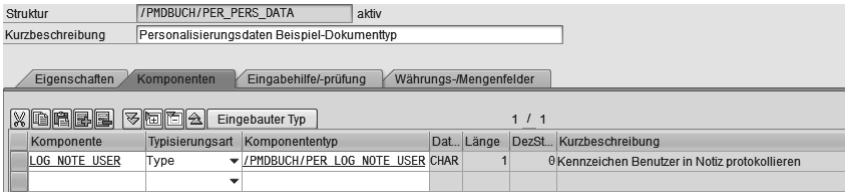


Abbildung 5.14: Ablagestruktur für Personalisierungsdaten

5.3.3.2 Pflegedialog erstellen

Eigene Popup-Fenster
3.2 / S. 69

Die Erstellung des Pflegedialogs ist der Erstellung eines eigenen Popup-Fensters sehr ähnlich, deshalb werden hier nicht noch einmal alle Schritte aufgeführt. Die Funktionsgruppe samt Anzeigestruktur, Dynpro, Ablauflogik, GUI-Status und anderen Elementen sind weitestgehend identisch aufgebaut; Sie müssen allerdings zusätzlich eine Bildmodifikation vorsehen, mit der Sie den Dialog in den Anzeigemodus versetzen können. Der Funktionsbaustein hat einen anderen Inhalt, da Sie die Daten aus dem übergebenen Personalisierungsdatenobjekt lesen und in dieses zurückschreiben müssen:

```
1  FUNCTION /pmdbuch/per_dialog_edit.
2  *"----------------------------------------------------------------------
3  *"*"Lokale Schnittstelle:
4  *"  IMPORTING
5  *"     REFERENCE(P_PERS_OBJECT) TYPE REF TO  CL_PERS_OBJECT_DATA
6  *"     REFERENCE(P_VIEW_MODE) TYPE  CHAR1
7  *"  EXCEPTIONS
8  *"      DIALOG_CANCELED
9  *"      PERS_OBJECT_ERROR
```

```
*"----------------------------------------------------------------------

  DATA: ls_pers_data TYPE /pmdbuch/per_pers_data.

* derzeitige Daten aus dem Personalisierungsobjekt lesen und
* in die Dynpro-Struktur übertragen
  CALL METHOD p_pers_object->get_data
    EXPORTING
      p_reload_data   = abap_false
    IMPORTING
      p_pers_data     = ls_pers_data
    EXCEPTIONS
      no_data_found   = 1
      data_type_error = 2
      internal_error  = 3
      OTHERS          = 4.
  CASE sy-subrc.
    WHEN 0.
*     alles OK
    WHEN 1.
*     keine Daten vorhanden - Standard vorbelegen
      ls_pers_data-log_note_user = abap_false.
    WHEN OTHERS.
      RAISE pers_object_error.
  ENDCASE.
  CLEAR /pmdbuch/per_pers_data.
  /pmdbuch/per_pers_data-log_note_user = ls_pers_data-log_note_user.

* Anzeigemodus?
  IF p_view_mode IS NOT INITIAL.
    g_0100_display = abap_true.
  ENDIF.

* Popup-Fenster anzeigen
  CALL SCREEN 0100 STARTING AT 5 5.

* Rückgabewert (letztes Kommando) prüfen.
  IF g_0100_function = 'OKAY'.
*   Daten zurückübertragen
    CLEAR ls_pers_data.
    ls_pers_data-log_note_user = /pmdbuch/per_pers_data-log_note_user.
    CALL METHOD p_pers_object->set_data
      EXPORTING
        p_pers_data     = ls_pers_data
        p_write_through = abap_false
      EXCEPTIONS
        OTHERS          = 1.
    IF sy-subrc <> 0.
      RAISE pers_object_error.
    ENDIF.
  ELSE.
    RAISE dialog_canceled.
  ENDIF.

ENDFUNCTION.
```

5.3.3.3 Personalisierungsobjekt registrieren

Um das Personalisierungsobjekt im System bekannt zu machen, starten Sie die Transaktion PERSREG, geben Sie einen neuen Objektnamen im Kundennamensbereich ein und wählen Sie [ ]. Geben Sie auf dem folgenden Bild eine Beschreibung ein, wählen Sie *generische Ablage* und geben Sie den Namen des *Funktionsbausteins für den Dialog* sowie im unteren Teil den *Typnamen* ein. Da es sich um eine benutzerbezogene Einstellung handelt, aktivieren Sie *bei Änderungen am Benutzer berücksichtigen* sowie – falls für den konkreten Anwendungsfall gewünscht – die beiden Unterpunkte *mit Benutzer kopieren* und *mit Benutzer verteilen*. Es ist außerdem sinnvoll, das Kennzeichen *sichtbar in der Benutzerpflege* zu setzen, da dadurch die Einstellungen in den Benutzerdaten auf der Registerkarte *Personalisierung* sichtbar werden. Abbildung 5.15 zeigt ein Beispiel der Registrierung. Sichern Sie die Einstellungen.

Bevor Sie mit der Entwicklung fortfahren, sollten Sie prüfen, ob der Einstellungsbaustein funktioniert, indem Sie ihn über die Benutzerpflege aufrufen. Abbildung 5.16 zeigt die Einstellungen in der Benutzerpflege.

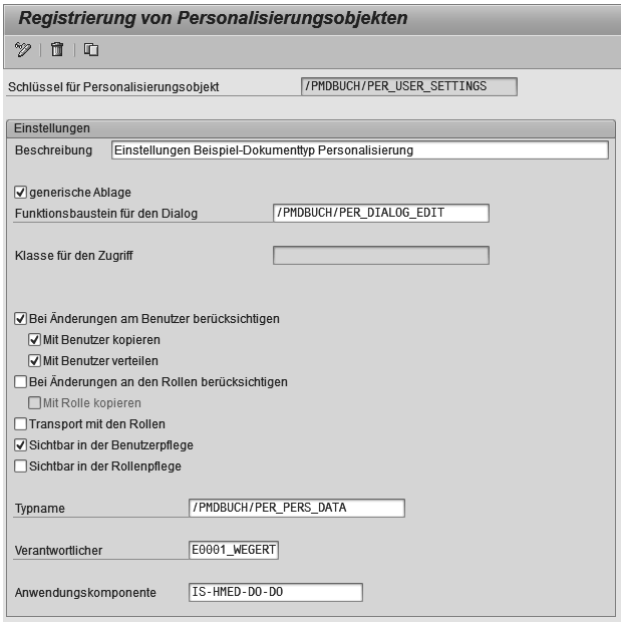


Abbildung 5.15: Registrierung des Personalisierungsobjekts

5.7 Inhaltliche Protokollierung mit Änderungsbelegen

5.7.1 Zielsetzung

Die generierte Dokumentationsanwendung bietet die Möglichkeit, das komplette Dokument zu versionieren und so eine gewisse Kontrolle über Bearbeitungsstände und Änderungen zwischen diesen Versionen zu ermöglichen. Diese Versionierung muss allerdings manuell veranlasst werden und ermöglicht keine exakte Nachverfolgung von Änderungen. In bestimmten Anwendungsfällen ist aber genau das gewünscht.

Änderungsbelege

⇒ [OT-CDO]

Das SAP NetWeaver-System verfügt mit **Änderungsbelegen** über eine Technologie, um auf Datensatz- und Feldebene Datenänderungen zu protokollieren. Die generierten PMD-Dokumentationsanwendungen sind im Normalfall nicht an die Änderungsbelegschreibung angeschlossen. In diesem Abschnitt erfahren Sie, welche Schritte notwendig sind, um diese Verbindung manuell herzustellen.

5.7.2 Voraussetzungen

Um die hier beschriebenen Schritte auszuführen, müssen Sie sicher im Umgang mit dem ABAP Dictionary sein, da verschiedene Objekte manuell angelegt und bearbeitet werden müssen. Auch ein fundiertes Grundwissen bezüglich grundlegender Datenmanipulation in ABAP ist notwendig.

Der Dokumenttyp muss unter Umständen vorbereitet werden: Falls der Dokumenttyp tabellarische Strukturen enthält, ist es in der Regel notwendig, ein zusätzliches technisches Feld einzufügen. Da dieses Feld nicht an der Oberfläche erscheinen muss, hat dies keine Auswirkungen auf die Dialoggestaltung oder die Druckausgabe. Wenn Sie einen bereits bestehenden Dokumenttyp um die Änderungsbelegschreibung erweitern möchten, sollten Sie aber vor Beginn der Arbeiten sicherstellen, dass Sie die Elementliste gefahrlos aktualisieren können. Es ist hilfreich, wenn Sie durchgängig mit eigenen Datenelementen gearbeitet haben, weil die Datenelemente unter Umständen angepasst oder kopiert werden müssen.

Objektorientierte User-Exits
2.1 / S. 27

Im folgenden Abschnitt wird gezeigt, wie die Änderungsbelegschreibung in objektorientierten User Exits implementieren können. Das Verfahren lässt sich analog auch in herkömmlichen prozeduralen User Exits anwenden.

5.7.3 Hintergrundinformationen

Änderungsbelege werden vom System in einer zentralen generischen Ablage gespeichert. Zur Identifikation und Strukturierung dienen sogenannte **Änderungsbelegob-**

jekte. Diese Objekte beziehen sich jeweils auf eine oder mehrere Datenbanktabellen, deren inhaltliche Änderungen protokolliert werden sollen. An dieser Stelle sind echte transparente Tabellen erforderlich, da zur Speicherung der Änderungsbelege zwischen den Schlüsselfeldern und den Attributfeldern unterschieden werden muss. Es ist also nicht ausreichend, reine Datenstrukturen zu definieren. Die Ermittlung der durchgeführten Änderungen erfolgt auf Basis des jeweiligen Tabellenschlüssels. Durch den Schlüssel werden – wie in RDBMS üblich – der alte und neue Stand eines Datensatzes einander zugeordnet sowie neu hinzugefügte oder gelöschte Datensätze identifiziert.

Es ist nicht sinnvoll, die generierten Datenbanktabellen der Dokumentationsanwendung für die Änderungsbelegschreibung zu verwenden. Dafür gibt es im Wesentlichen zwei Gründe:

- Die generierten Tabellen enthalten für erweiterte Textfelder und SAPscript-Texte nicht den eigentlichen Text, sondern nur einen technischen Schlüssel. Auf diesem Wege können keine Textänderungen aufgezeichnet werden.
- In Sekundärtabellen wird die Zeilennummer im Dokument als Schlüsselbestandteil der Datenbanktabelle verwendet. Das führt dazu, dass bei einer Löschung einer Zeile alle nachfolgenden Zeilen nachrücken und sich in ihrer Zeilennummer ändern. Da Änderungsbelege die Datensätze anhand des Tabellenschlüssels identifizieren, würde daher immer eine Löschung der letzten Zeile und eine komplette inhaltliche Änderung jeder vorherigen Zeile bis zur gelöschten Zeile protokolliert werden. Dadurch wird das Änderungsprotokoll von einer Vielzahl von Phantom-Änderungen überschwemmt und unleserlich.

Die Identifikation der Zeilen anhand ihrer Position ist auch der Grund für die in den Voraussetzungen bereits erwähnte Strukturanpassung: Zur Lösung dieses Problems erhalten die Zeilen einer Sekundärtabelle einen eindeutigen Schlüssel, der auch bei einer Löschung oder Verschiebung einer Tabellenzeile stabil bleibt.

Weiterhin haben Änderungsbelege zwei Beschränkungen, die beim Einsatz in PMD-Anwendungen in der Regel umgangen werden müssen:

- Eine Detaildokumentation von Langtexten ist nicht vorgesehen; es kann lediglich dokumentiert werden, dass ein Langtext geändert wurde. Diese Beschränkung ist allerdings relativ einfach zu umgehen, seit Änderungsbelege auch `STRING`-Felder erfassen können.
- Bei der Anlage eines Datensatzes werden die Feldinhalte nicht erfasst. Das in der Dokumentation noch beschriebene Kennzeichen *Doku. der einzelnen Felder bei Einfügung* wurde niemals realisiert. Auch diese Beschränkung ist

aber relativ einfach zu umgehen, wenn nach einer Datensatzanlage sofort ein Änderungssatz geschrieben wird.

5.7.4 Vorgehensweise

Falls der Dokumenttyp tabellarische Strukturen enthält, ist zunächst die beschriebene Strukturerweiterung erforderlich. Um Änderungsbelege aus einem Dokumenttyp heraus zu erzeugen, müssen Sie dann die Datenbanktabellen anlegen, die den Protokollumfang festlegen. Anschließend können Sie ein Änderungsbelegobjekt anlegen und die Verbuchungsbausteine dazu generieren. Diese Bausteine können Sie dann in die User Exits des Dokumenttyps einbauen.

In den ausgelieferten Beispielen finden Sie das BC-Set `/PMDBUCH/CDO_DEMOCDO_V01`, das ein Installationspaket für den Dokumenttyp `DEMOCDO` enthält. Die zugehörigen Entwicklungsumgebungsobjekte finden Sie im Paket `/PMDBUCH/CDO`.

5.7.4.1 Strukturanpassung

Fügen Sie in jede tabellarische Struktur, deren Änderungen protokolliert werden sollen, ein Feld mit dem Datentyp `SYSUUID_C` oder einer Kopie dieses Datenelements ein. Dieses Feld nimmt später einen GUID auf, der die Zeile stabil identifiziert. Sie brauchen dieses Feld weder in den Dialog noch in die Druckausgabe zu integrieren. Vergeben Sie einen Alias und passen Sie die Tabellenstrukturen an.

5.7.4.2 Tabellen und Datentypen anlegen

Datenstruktur-Generierung
1 / 11.6.4 / S. 318

Generieren Sie zunächst die Druck-Strukturen zum Dokumenttyp. Diese Strukturen werden zwar für die spätere Änderungsbelegschreibung nicht benötigt, erleichtern aber die Erstellung der Tabellen enorm.

Denken Sie daran, dass Sie nach der Generierung der Strukturen auch die Anwendung neu generieren müssen!

Legen Sie dann die zur Primärtabelle des Dokuments korrespondierende Protokolltabelle an. Dazu können Sie wahlweise die ABAP Workbench (Transaktion `SE80`) oder das ABAP Dictionary (Transaktion `SE11`) verwenden. Beachten Sie bei der Eingabe der Beschreibung, dass dieser Text später in den Änderungsbelegen angezeigt wird. Übernehmen Sie zunächst den Primärschlüssel aus der Primärtabelle und alle Felder aus der Druckstruktur des Dokuments. Dazu bietet es sich an, den Menüpunkt Bearbeiten » Felder übernehmen in der Tabellenpflege zu verwenden. Passen Sie jetzt die Tabellenstruktur wie folgt an:

- Entfernen Sie Felder, die nicht protokolliert werden sollen – insbesondere etwaige Anzeigefelder oder Zusatzdaten zum Druck.
- Entfernen Sie eingebettete Tabellen, die bei der Verwendung tabellarischer Strukturen im Dokumenttyp entstehen. Für diese Tabellen werden im nächsten Schritt eigene Protokolltabellen angelegt.
- Für Langtextfelder werden jeweils zwei Felder in die Struktur eingebettet – ein tabellarisches Feld und ein `STRING`-Feld. Entfernen Sie die tabellarischen Felder.
- Stellen Sie sicher, dass das Kennzeichen *Änderungsbeleg* auf der Registerkarte *Zusatzeigenschaften* jedes verwendeten Datenelements gesetzt ist (vgl. Abbildung 5.27). Falls Sie fremde Datenelemente wiederverwendet haben, bei denen dieses Kennzeichen nicht gesetzt ist, müssen Sie hier ggf. eigene Datenelemente als Kopien anlegen und in die Struktur einfügen.

Abbildung 5.27: Kennzeichen *Änderungsbeleg* im Datenelement

Sie können die Tabelle mit der Auslieferungsklasse `L` anlegen, da sie niemals Daten enthalten wird – einzig ihre Struktur ist von Bedeutung. Dementsprechend sind auch die technischen Einstellungen und die Erweiterungskategorie der Tabelle irrelevant.

In den ausgelieferten Beispielen finden Sie die Tabelle `/PMDBUCH/CDO_PRI`, die nach den hier beschriebenen Regeln aus der Druckstruktur des Dokumenttyps `DEMOCDO` erstellt wurde.

Legen Sie jetzt für jede Sekundärtabelle eine Protokolltabelle an. Übernehmen Sie auch hier zunächst den Primärschlüssel der Sekundärtabelle und die Inhalte der generierten Druck-Struktur der Tabelle. Passen Sie jetzt die Tabellenstruktur wie folgt an:

6.6 HL7-Kommunikation mit der MCI

6.6.1 Zielsetzung

Nachrichtenverarbeiter (MCI)

⇒ [HP-MCI]

Wenn Sie strukturierte Dokumente aus externen Systemen entgegennehmen und in i.s.h.med ablegen möchten, können Sie dazu in aktuellen Releases die Message Communication Infrastructure (MCI) verwenden. Dabei handelt es sich um ein flexibles, komponentenorientiertes System zur Nachrichtenverarbeitung. Die MCI ersetzt keinen Kommunikationsserver, denn sie ist nicht darauf ausgelegt, beliebige Nachrichtenumsetzungen mit der Flexibilität und dem Komfort anzubieten, wie dies gängige Kommunikationsserver tun. Darüber hinaus sind die externen Schnittstellentechnologien durch die Möglichkeiten der SAP NetWeaver-Basis limitiert; so ist zwar eine synchrone RFC- oder HTTP-Kommunikation möglich, das im HL7-Umfeld gebräuchliche Minimal Lower Layer Protocol (MLLP) wird hingegen nicht unterstützt. Dennoch ist die MCI ein sehr wertvolles Werkzeug bei der Abbildung von Kommunikationsprozessen im i.s.h.med-Umfeld.

Im Rahmen dieses Buchs ist es nicht möglich, einen vollständigen Überblick über die Möglichkeiten der MCI zu geben. Hier soll exemplarisch auf den Aspekt des Dokumentimports eingegangen werden. Dazu wird ein sehr spartanischer Dokumenttyp bereitgestellt, der einen vorstrukturierten, aus Freitexten bestehenden Pathologiebefund aufnehmen soll. Um den Gesamtprozess testen zu können, wird ein kleines Programm verwendet, das eine HL7-Nachricht aus einem Vorlagetext erzeugt und an die MCI übergibt. So ist ein Probelauf ohne Konfiguration von Dateiablagen oder anderer externer Software möglich.

6.6.2 Voraussetzungen

Um das hier gezeigte Beispiel nachvollziehen zu können, benötigen Sie ein System mit Release 6 EhP 5 oder höher. In diesem Beispiel werden Komponenten verwendet, für die Sie die Lizenz zur Komponente *i.s.h.med Befunddokumentation mit Clinic WinData* benötigen.

Zum Verständnis dieses Kapitels benötigen Sie ausnahmsweise kaum ABAP-Kenntnisse – für einfache[1] Abläufe ist die MCI ohne Zusatzprogrammierung einsetzbar. Sie müssen aber mit den Grundkonzepten von HL7 Version 2.6 vertraut sein, da auf die genaue Nachrichtenstruktur im Rahmen dieses Buchs nur oberflächlich eingegangen werden kann.

[1] oder vielmehr von den Standardkomponenten bereits unterstützte

6.6.3 Vorgehensweise

Wie bereits erwähnt wird in diesem Beispiel ein Dokumenttyp zur Aufnahme des Befunds verwendet. Dieser Dokumenttyp besteht lediglich aus drei erweiterten Textfeldern ohne jegliche Zusatzprogrammierung und ist hier deshalb nicht abgebildet.

Wenn Sie das Beispiel in Ihrem System nachvollziehen möchten, finden Sie im BC-Set `/PMDBUCH/MCI_DEMOMCI_V01` ein Installationspaket zum Dokumenttyp `DEMOMCI`.

In diesem Beispiel soll eine relativ einfache `ORU`-Nachricht verarbeitet werden, in deren `OBX`-Segmenten die drei zu übernehmenden Texte enthalten sind. Die Nachricht sieht etwa wie folgt aus, wobei die Umbrüche und Einrückungen lediglich der Wiedergabe im Druck geschuldet sind:

```
MSH|^~\&|PATHOLOGIE|HOSPITAL_A|HD2|101^0104|20140827205803||ORU^R01^ORU_R01|
  20140827205803000001|P|2.6|||AL|NE|||
PID|1|0001238996|0001238996||Gallier^Asterix||19591029|||||||||||||||||||||
PV1|1|I|||||||||||||||||0001173783|
OBX|1|ST|MACRO^Makroskopie||Mehrere, zusammen 2,8 x 2,3 x 1,1 cm messende,
  gelb-graue Gewebsstücke.||||||F||||||
OBX|2|ST|MICRO^Mikroskopie||Fragmente eines teils knotigen, zellreichen,
  fibromatösen Gewebes, bestehend aus spindelförmigen Zellen, eingebettet in
  eine kollagenfaserreiche interzelluläre Matrix.\.br\Angrenzend von
  Bindegewebssepten durchzogenes Fettgewebe mit einigen Vater-Pacini-Körpern
  und einzelnen kleinen Fremdkörpergranulomen. \.br\Polarisationsoptisch kein
  Nachweis doppelbrechenden Fremdmaterials.||||||F||||||
OBX|3|ST|DIAG^Diagnose||Proliferationsaktive Faszienfibromatose in einem
  Exzidat von der linken Fußsohle.||||||F||||||
```

In den folgenden Abschnitten wird erst das Werkzeug zur Erzeugung der Nachrichten und anschließend der korrespondierende MCI-Prozess beschrieben.

6.6.3.1 Nachrichtenerzeugung

Es gibt eine ganze Reihe von Möglichkeiten, Nachrichten an die MCI zu übergeben – eine Dialogeingabe innerhalb der SAP GUI gehört im Standardumfang leider nicht dazu, was den Test etwas erschwert. Prinzipiell können Sie die hier beschriebene Vorgehensweise auch ausprobieren, indem Sie die HL7-Nachricht in eine Datei schreiben und den Kommunikationsprozess mit dem Standardprogramm `RN1MCI_MESSAGE_RECEIVER` starten. In diesem Abschnitt wird allerdings ein Programm verwendet, das den Start direkt aus dem i.s.h.med-System erlaubt.

Das Beispielprogramm `/PMDBUCH/MCI_START_PROCESS` samt der verwendeten Komponenten ist im Paket `/PMDBUCH/MCI` enthalten.

Das Programm verfügt über das in Abbildung 6.5 gezeigte Selektionsbild. Nach dem Start liest es den in den Parametern angegebenen Vorlage-Text, der als *Allgemeiner Text* in der Transaktion `SE61` gepflegt sein muss. Der in der Vorbelegung angegebene Text ist im Beispielpaket enthalten und kann nach Belieben kopiert und angepasst werden.

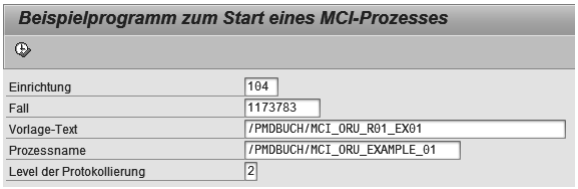


Abbildung 6.5: Selektionsbild des Nachrichtenerzeugers

Anschließend findet eine Ersetzung der in der Vorlage verwendeten Textsymbole statt. Dabei können neben den Systemvariablen die Einrichtungsdaten aus der Tabelle `TN01` und die Patienten- und Falldaten aus den Druckstrukturen `RNF01` und `RNF05` verwendet werden. Dadurch ist es möglich, Nachrichten zu erzeugen, die zu den vorhandenen Fall- und Patientendaten passen. Die Aufbereitung des Texts erfolgt in der Funktionsgruppe `/PMDBUCH/MCI_GEN_HL7_TEXT`.

RNF-Strukturen 4.1 / S. 113

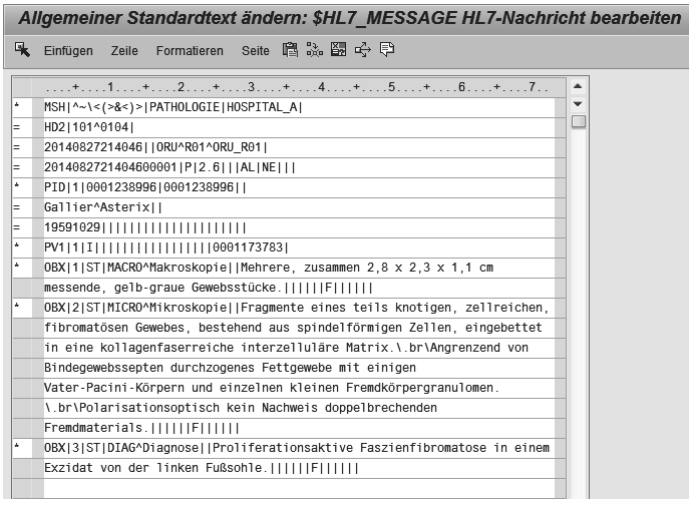


Abbildung 6.6: Editor mit der generierten HL7-Nachricht

Nach erfolgter Aufbereitung wird der Text in einem einfachen Standard-Editor angezeigt (Abbildung 6.6). Damit ist es einfach möglich, die Testdaten nochmals anzupassen. Wenn der Editor verlassen wird, wird nach einer Sicherheitsabfrage der auf dem Selektionsbild spezifizierte MCI-Prozess über den RFC-Funktionsbaustein gestartet. Falls dabei Nachrichten auftreten, was in der Regel der Fall ist, werden diese in einer Liste angezeigt.

6.6.3.2 MCI-Konfiguration

Die MCI arbeitet mit Prozessen, die der besseren Übersicht halber in Prozessgruppen geordnet werden können. Eine Prozessdefinition der MCI besteht immer aus genau einem Start- und einem End-Konnektor, zwischen denen beliebig viele Transformer angeordnet werden. Der Start-Konnektor ist dafür verantwortlich, eine Nachricht aus einer beliebigen Quelle in die MCI zu überführen. Die Nachricht wird anschließend der Reihe nach durch die angegebenen Transformer geschleust, wobei jeder Transformer die Nachricht lesen, verändern oder auch komplett austauschen kann. Nach dem letzten Transformer wird die Nachricht an den End-Konnektor übergeben, der für die weitere Verarbeitung im Zielsystem sorgt. Bei eingehenden Prozessen wird der Start-Konnektor von einer externen Quelle beschickt und der End-Konnektor verbucht Daten in i.s.h.med; bei einem ausgehenden Prozess ist es umgekehrt.

MCI-Prozesse sind immer linear aufgebaut. Es gibt keine Verzweigungen, Schleifen oder andere Kontrollfluss-Konstrukte. Die einzige Möglichkeit der Steuerung besteht darin, dass Komponenten die Verarbeitung von Nachrichten durch die Meldung von temporären oder permanenten Fehlersituationen zeitweise oder final verhindern können.

Die Konfiguration des Prozesses erfolgt in der Transaktion `N1MCI_CONFIG`. Bei der Anlage des Prozesses geben Sie bereits den Start- und End-Konnektor an. Anschließend können Sie aus der Palette der verfügbaren Transformer die gewünschten Komponenten in den Prozess ziehen. Abbildung 6.7 zeigt den in diesem Beispiel verwendeten Prozess.

In den ausgelieferten Beispielen finden Sie diesen Prozess im BC-Set `/PMDBUCH/MCI_ORU_IMPORT_CONFIG`. Dieses BC-Set ist **nicht** im hierarchischen BC-Set `/PMDBUCH/BAND2` enthalten, da es nicht über die PMD-Transportworkbench installiert wird. Verwenden Sie in der Transaktion `N1MCI_CONFIG` den Menüpunkt Konfiguration » Importieren » Aus BC-Set... zum Import in Ihr System. Beachten Sie, dass Sie die Einstellungen in der Regel noch nachbearbeiten müssen.

7.2 Linkbausteine

7.2.1 Zielsetzung

Linkbaustein / Linkelement (Konzept)
1 / 2.3.3.1 / S. 72

Im ersten Band haben Sie das Konzept der Linkelemente und Linkbausteine kennengelernt; dort finden Sie auch eine Liste der im Standardumfang von i.s.h.med ausgelieferten Linkbausteine. In Kapitel 1 auf Seite 13 haben Sie einige dieser Bausteine näher kennengelernt. Es ist prinzipiell möglich, eigene Linkbausteine zu erstellen; in diesem Kapitel erhalten Sie die notwendigen Hintergrundinformationen dazu.

Die Schnittstelle zur Bereitstellung von Linkbausteinen ist nicht offiziell für Kundenentwicklungen freigegeben und nur schwach dokumentiert. Es handelt sich allerdings um eine aktiv gewartete und sowohl im Produkt als auch in Hersteller- und Partner-Erweiterungen vielfach genutzte Schnittstelle, die eine hinreichende Stabilität und Zukunftssicherheit bietet, um auch in Kundenentwicklungen zum Einsatz zu kommen.

7.2.2 Voraussetzungen

Um Linkbausteine zu erstellen, müssen Sie sicher in der prozeduralen und objektorientierten ABAP-Entwicklung sein. Da Sie mit Komponenten des Control Frameworks arbeiten müssen, benötigen Sie gute Kenntnisse im Umgang mit diesen Komponenten; diese werden etwa im Kurs `BC412` (ABAP Dialogprogrammierung mit EnjoySAP-Controls) vermittelt. Zusätzlich müssen Sie sehr oft auf die generische Programmierung zurückgreifen, wenn Sie den Linkbaustein wiederverwendbar gestalten möchten, sodass Sie sich auch mit dieser Technologie vertraut machen sollten.

Generische Programmierung
2.3 / S. 46

Planen Sie in jedem Fall hinreichend Zeit für die Erstellung eines eigenen Linkbausteins ein – je nach Komplexität der Aufgabenstellung kann es selbst erfahrene Entwickler mehrere Tage oder Wochen kosten, einen Linkbaustein zu erstellen.

Der Dokumenttyp muss zur Aufnahme des Linkbausteins lediglich um ein Linkelement erweitert werden. Um dieses Element unterzubringen, sollte im Dialog an geeigneter Stelle Platz geschaffen werden. Weitere Anforderungen an den Dokumenttyp ergeben sich unter Umständen aus der fachlichen Anforderung.

Die hier vorgestellten Linkelemente sind so nur mit Dokumenttypen kompatibel, die mit Generatorversion 2 entwickelt werden. Bei Verwendung der Generatorversion 1 gibt es subtile Unterschiede in der Ansteuerung, auf die hier nicht eingegangen werden kann.

Da die Neuentwicklung von Dokumenttypen mit Generatorversion 1 nicht mehr unterstützt wird, stellt das allerdings in der Regel kein Problem dar.

7.2.3 Beispiel-Szenario

Die bisher verwendeten Beispiel-Dokumente eignen sich nicht sonderlich gut zur Demonstration eines selbstentwickelten Linkbausteins. Deshalb kommt hier ein neuer Dokumenttyp zum Einsatz, in dem patientenbezogen das Gewicht zu unterschiedlichen Zeitpunkten erfasst wird. Aus diesen Daten soll eine grafische Darstellung der Gewichtsentwicklung erstellt werden. Die tabellarische Erfassung von Datum und Gewicht ist mit herkömmlichen Dokelementen leicht zu realisieren, nicht aber die grafische Darstellung. Dazu ist die Einbindung einer Dialogkomponente erforderlich, die normalerweise nicht im PMD-Standardumfang zur Verfügung steht. Abbildung 7.1 zeigt den fertigen Linkbaustein innerhalb eines Dokuments. In den nächsten Abschnitten erfahren Sie schrittweise, wie dieser Linkbaustein realisiert werden kann.

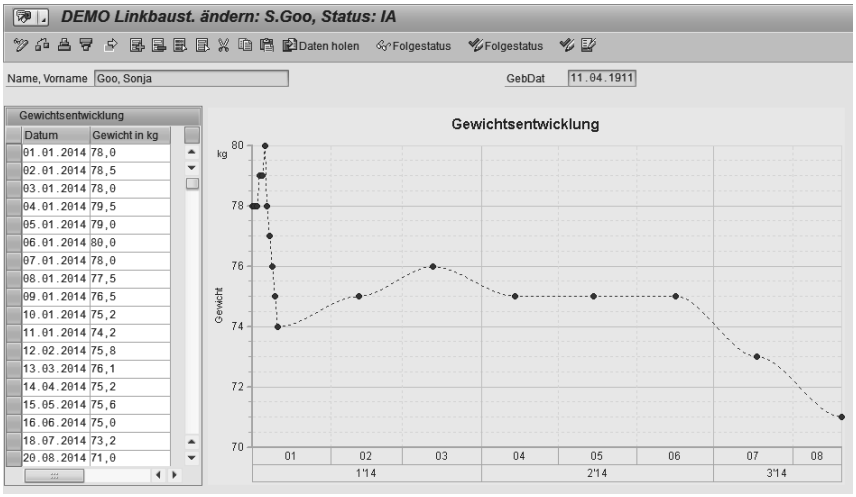


Abbildung 7.1: Beispieldokument mit eigenem Linkbaustein

Die Entwicklungsumgebungsobjekte zu diesem Kapitel finden Sie im Paket `/PMDBUCH/LNK`. Der Beispiel-Dokumenttyp `DEMOLNK` kann mit dem Installationpaket, das sich im BC-Set `/PMDBUCH/LNK_DEMOLNK_V01` befindet, installiert werden. Der Druck dieses Dokumenttyps ist mit SIFbA realisiert und kann deshalb nur getestet werden, wenn in der Systemlandschaft ein ADS verfügbar ist.

7.2.4 Exkurs: IGS und Geschäftsgrafiken

Chart Engine

⇒ [HP-CEN]

Zur Erstellung und Anzeige der Grafik wird eine bereits existierende Komponente verwendet, da ansonsten der Aufwand untragbare Größenordungen erreichen würde. Es gibt im System mehrere Bibliotheken, die sich zur Erstellung von sogenannten Geschäftsgrafiken eignen. In diesem Fall wird eine Grafikbibliothek verwendet, die in den Internet Gateway Service (IGS) integriert ist. In aktuellen Releases ist diese Komponente praktisch immer verfügbar, so dass auf die Konfiguration hier nicht weiter eingegangen wird.

Die Chart Engine kann in mehreren Modi betrieben werden, die in der Dokumentation beschrieben sind. In diesem Beispiel wird der ActiveX-Modus verwendet, in dem eine Anzeigekomponente direkt in die SAP GUI eingebettet wird. Dafür existiert bereits eine entsprechende Hüllklasse in der ABAP-Entwicklungsumgebung. Sie können die Funktion mit dem Programm `GRAPHICS_GUI_CE_DEMO` testen.

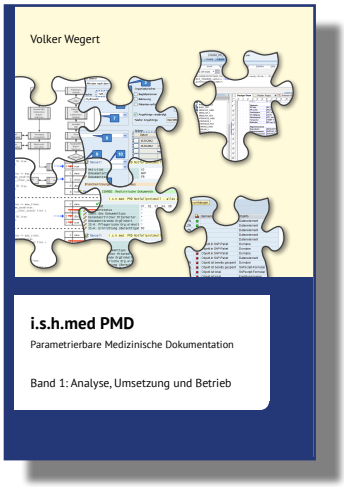
Chart Designer

⇒ [HP-CDE]

Die Chart Engine verwendet zwei Eingabedatensätze: die anzuzeigenden Daten und einen Satz Einstellungen, die die Aufbereitung steuern. Die Eingabeformate sind nur spärlich dokumentiert, können aber mit dem extern herunterzuladenden Chart Designer relativ bequem erstellt werden. In die ActiveX-Komponente ist eine einfache Version dieses Chart Designers integriert, mit der für das hier vorgestellte Beispiel eine hinreichend flexible Aufbereitung der Grafik möglich ist. Damit ist es nicht weiter erforderlich, sich mit dem Format der Einstellungen auseinanderzusetzen. Die anzuzeigenden Daten werden in Form eines XML-Dokuments bereitgestellt, das folgenden Aufbau hat:

```
<?xml version="1.0" encoding="UTF-16"?>
<ChartData>
  <Categories>
    <Category>20140101</Category>
    <Category>20140102</Category>
    <!-- ... -->
  </Categories>
  <Series>
    <Point>
      <Value type="time">20140101;120000</Value>
      <Value type="y">78,0</Value>
    </Point>
    <Point>
      <Value type="time">20140102;120000</Value>
      <Value type="y">78,5</Value>
    </Point>
    <!-- ... -->
  </Series>
</ChartData>
```

Die genauen Eingabedatenformate sind je nach Grafiktyp unterschiedlich; für das vorliegende Beispiel sind aber die hier gezeigten Daten ausreichend.



Aus dem Inhalt:

Analyse und Entwurf
- Einsatzbereiche und Prozessübersicht
- Funktionsumfang
- Ablauf der Analyse

Entwicklungsumgebung
- Konzepte und Objekte
- ABAP für Umsteiger
- ABAP Objects

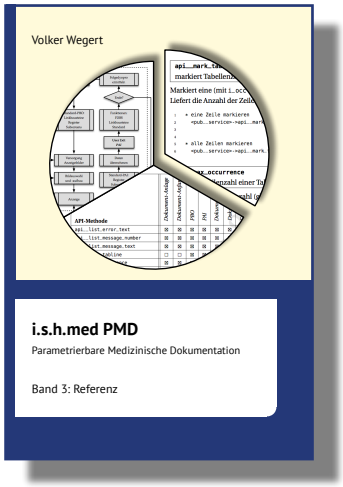
Umsetzung, Betrieb und Weiterentwicklung
- Planung der technischen Umsetzung
- Dokumentarten
- Dokumentationselemente und Dokumenttypen
- Werteversorgung
- Einfache Zusatzprogrammierung
- Gestaltung der Druckausgabe
- Test, Produktivsetzung und Betrieb
- Anpassung und Weiterentwicklung
- Dokumenttypen kopieren und verteilen

Volker Wegert

i.s.h.med PMD
Band 1: Analyse, Umsetzung und Betrieb

Dieses Buch bietet eine umfassende Einführung in den Entwurfs- und Entwicklungsprozess PMD-basierter Anwendungen. Sie erhalten grundlegende Informationen über die Einsatzbereiche, Möglichkeiten und Grenzen der PMD sowie einen Überblick über die Arbeitsschritte, die zur Erstellung eines PMD-Dokumenttyps erforderlich sind. Anhand eines durchgängig verwendeten Beispiels werden die Schritte erläutert, die zur Erarbeitung eines umsetzbaren Entwurfs erforderlich sind. Das Buch enthält eine Einführung in die ABAP-Entwicklungsumgebung für Leser, die noch nie oder nur in geringem Umfang mit den Entwicklungswerkzeugen des SAP NetWeaver-Systems in Berührung gekommen sind. Anschließend wird aus dem Entwurf schrittweise ein einsetzbarer PMD-Dokumenttyp erstellt. Anhand des Beispiels aus dem ersten Teil erlernen Sie die Anlage und Einstellung der verschiedenen Objekte, die Erweiterung der Anwendung durch zusätzliche Programmierung sowie die Erstellung der Druckausgabe. Auf die Schritte, die zur erfolgreichen Inbetriebnahme eines Dokumenttyps erforderlich sind, wird ebenso eingegangen wie auf die Anpassung und Weiterentwicklung bestehender Dokumenttypen. Ein umfangreicher Anhang mit ausführlichem Schlagwortverzeichnis erleichtert die tägliche Arbeit mit der PMD.

621 Seiten, Hardcover, 2014, 69,90 €
ISBN 978-3-8495-9921-8
http://www.pmd-buch.de



Aus dem Inhalt:

- Funktionen der Dokumenttypbearbeitung
- Einstellungen der Dokumentart
- Dokelementtypen mit Kennzeichen und Größenangaben
- Fremddatenbausteine
- Dialogerweiterungen: User Exits, Ablauf und Verarbeitungsmodi
- Druckausgabe: User Exits, Ablauf und Anwendungen
- API-Referenz: Variablen und Methoden
- Benutzereinstellungen und Berechtigungen
- Generische Programmierung
- Technische Objektnamen
- Beispiel-Dokumenttypen

Volker Wegert

i.s.h.med PMD
Band 3: Referenz

Dieses Buch stellt eine Referenz zur täglichen Arbeit mit der PMD dar. Hier finden Sie in komprimierter Form die wichtigsten Informationen, die Sie zur Erstellung und Anpassung von Dokumenttypen benötigen. Auf die Einstellungen der Dokumentart und der Dokelementtypen wird ebenso eingegangen wie auf die Abläufe der Dialog- und Druckausgabe und die verfügbaren User Exits. Zentraler Bestandteil der Referenz ist eine Übersicht über die verfügbaren API-Variablen und -Methoden mit Beschreibung und Anwendungsbeispielen. In diesen Buch sind weiterhin die wichtigsten Berechtigungsobjekte und Benutzerparameter, die Strukturen und Konstanten zur generischen Programmierung sowie häufig benötigte Transaktionen, Tabellen und Programme aufgeführt.

65 Seiten, Paperback, 2014, 9,90 €
ISBN 978-3-7323-0104-1
http://www.pmd-buch.de