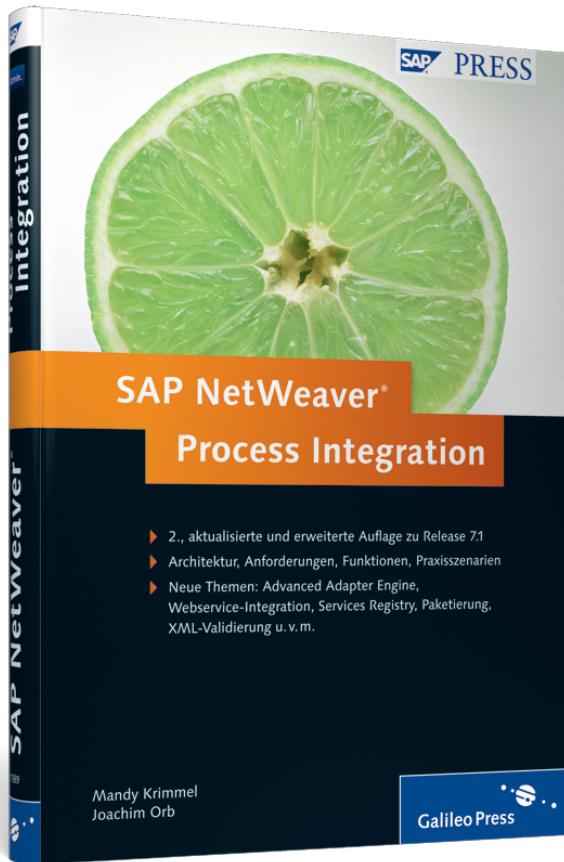


Mandy Krimmel, Joachim Orb

## SAP NetWeaver® Process Integration



•  •  
Galileo Press

Bonn • Boston

# Auf einen Blick

<b>1</b>	<b>Überblick</b>	17
<b>2</b>	<b>Erste Schritte</b>	41
<b>3</b>	<b>Design von kollaborativen Prozessen</b>	61
<b>4</b>	<b>Service-Interfaces, Messages und Proxy-Generierung</b>	93
<b>5</b>	<b>Mappings</b>	137
<b>6</b>	<b>Konfiguration</b>	169
<b>7</b>	<b>Laufzeit</b>	237
<b>8</b>	<b>Integrationsprozesse</b>	275
<b>9</b>	<b>Systemübergreifendes Business Process Management bei der Linde Group</b>	305
<b>10</b>	<b>Unternehmensübergreifende Kommunikation über SAP NetWeaver PI</b>	341
<b>11</b>	<b>Implementierung eines Webservice-Szenarios bei Boehringer Ingelheim</b>	365
<b>A</b>	<b>Glossar</b>	389
<b>B</b>	<b>Die Autoren</b>	397

# Inhalt

Einleitung .....	13
<b>1 Überblick .....</b>	<b>17</b>
1.1 SAP NetWeaver .....	17
1.2 Ebenen der Prozessmodellierung .....	22
1.3 Prozessintegration mit SAP NetWeaver PI .....	24
1.3.1 Kommunikation über den Integration Server ...	25
1.3.2 Design und Konfiguration .....	34
<b>2 Erste Schritte .....</b>	<b>41</b>
2.1 Einführung in die PI-Design- und -Konfigurations- werkzeuge .....	45
2.1.1 Enterprise Services Builder und Integration Builder .....	45
2.1.2 Services Registry .....	52
2.1.3 Webservice Navigator .....	56
2.2 Simple Use Cases und Demo-Beispiele .....	57
<b>3 Design von kollaborativen Prozessen .....</b>	<b>61</b>
3.1 Entwicklungsorganisation .....	61
3.1.1 Beschreibung von Produkten im Software- Katalog .....	62
3.1.2 Organisation der Designobjekte im Enterprise Services Repository .....	65
3.1.3 Objektversionierung und Transport .....	68
3.2 Modellierung des kollaborativen Prozesses .....	71
3.3 Modellierung mithilfe von Integrationsszenarien .....	72
3.3.1 Abbildung von Anwendungskomponenten auf Systeme .....	72
3.3.2 Modellierung des Message-Austauschs .....	78
3.4 Modellierung mithilfe des Prozesskomponenten- Architekturmodells .....	87
3.4.1 Prozesskomponenten-Modelle .....	87
3.4.2 Integrationsszenario-Modelle .....	89
3.4.3 Prozesskomponenten-Interaktionsmodelle .....	91

<b>4 Service-Interfaces, Messages und Proxy-Generierung .....</b>	<b>93</b>
4.1 Entwicklung nach dem Proxy-Modell .....	94
4.1.1 Service-Interface-Entwicklung im Enterprise Services Builder .....	95
4.1.2 Proxy-Generierung .....	108
4.2 Unterstützung für adapterbasierte Kommunikation ....	119
4.2.1 Import von Interfaces und Message-Schemata	121
4.2.2 Entwicklung mit importierten Interface-Objekten .....	126
4.3 Erweiterte Konzepte .....	128
4.3.1 Komponentenübergreifende Verwendung von Message-Typen .....	128
4.3.2 Erweiterung von Datentypen bei Partnern und Kunden .....	130
4.3.3 Zugriff auf Message-Felder über Kontextobjekte .....	135
<b>5 Mappings .....</b>	<b>137</b>
5.1 Mapping-Programme in SAP NetWeaver PI .....	138
5.1.1 Werte-Mappings .....	140
5.1.2 Mappings in Integrationsprozessen .....	141
5.2 Vorkonfiguration und Test von Mapping-Programmen	143
5.3 Java- und XSLT-Mappings .....	147
5.3.1 Java-Mappings .....	148
5.3.2 XSLT-Mappings .....	152
5.4 Entwickeln von Mappings im Enterprise Services Builder .....	153
5.4.1 Einführung in den Mapping-Editor .....	153
5.4.2 Abbildungsfunktionen in Message-Mappings	159
5.4.3 Fortgeschrittene Message-Mapping-Techniken .....	161
5.4.4 Datentyp-Mappings im Enterprise Services Builder entwickeln .....	167
<b>6 Konfiguration .....</b>	<b>169</b>
6.1 Beschreibung von Systemen und ihren Kommunikationskomponenten .....	172
6.1.1 Einstellungen im System Landscape Directory	172

6.1.2	Erste Schritte im Integration Directory .....	176
6.2	Konfiguration unternehmensinterner Prozesse .....	181
6.2.1	Konfiguration über Integrationsszenarien .....	182
6.2.2	Übersicht über Konfigurationsobjekttypen .....	186
6.2.3	Integrierte Konfiguration .....	197
6.2.4	Werte-Mapping .....	200
6.2.5	Direkte Kommunikation .....	203
6.3	Konfiguration unternehmensübergreifender Prozesse .....	205
6.3.1	Von interner zu unternehmensübergreifender Kommunikation .....	207
6.3.2	Partner Connectivity Kit .....	212
6.4	Adapterkonfiguration .....	215
6.4.1	Übersicht .....	215
6.4.2	Besonderheiten des RFC- und des IDoc-Adapters .....	220
6.5	Adapter für Industriestandards .....	226
6.5.1	RosettaNet-Standards .....	226
6.5.2	RosettaNet-Unterstützung mit SAP NetWeaver PI .....	228
6.5.3	Chem eStandards .....	231
6.6	Transporte zwischen Test- und Produktivlandschaft ....	233
<b>7</b>	<b>Laufzeit .....</b>	<b>237</b>
7.1	Integration Server und Integration Engine .....	237
7.1.1	Grundlagen .....	238
7.1.2	Verarbeitungsschritte einer Message .....	241
7.2	Advanced Adapter Engine .....	247
7.2.1	Grundlagen .....	248
7.2.2	Adapter Framework .....	249
7.3	Proxy-Laufzeit .....	254
7.3.1	Besonderheiten bei der Kommunikation über Java-Proxys .....	259
7.3.2	ABAP-Proxys und Webservices .....	262
7.4	Monitoring .....	265
<b>8</b>	<b>Integrationsprozesse .....</b>	<b>275</b>
8.1	Was ist ein Integrationsprozess? .....	276
8.2	Integrationsprozesse und andere Prozesse .....	277

8.3	Design des Integrationsprozesses .....	279
8.3.1	Daten eines Integrationsprozesses .....	280
8.3.2	Verarbeitung von Messages .....	282
8.3.3	Steuerung des Prozessablaufs .....	286
8.3.4	Zeitsteuerung und Ausnahmebehandlung .....	289
8.3.5	Voreinstellung des Laufzeitverhaltens .....	292
8.3.6	Integrationsprozesse importieren oder exportieren .....	293
8.4	Weiterführende Designkonzepte .....	294
8.4.1	Monitoring-Prozess .....	294
8.4.2	Schrittgruppe .....	294
8.4.3	Alert-Kategorie .....	295
8.5	Integrationsprozesse konfigurieren .....	295
8.5.1	Übersicht .....	295
8.5.2	Konfiguration über Integrationsszenarien .....	298
8.5.3	Konfiguration der Eingangsverarbeitung .....	300
8.6	Ausführung eines Integrationsprozesses überwachen .....	302
8.6.1	Laufzeit-Cache analysieren .....	302
8.6.2	Prozess-Monitoring .....	303
8.6.3	Message-Monitoring .....	303

## **9 Systemübergreifendes Business Process Management bei der Linde Group .....** 305

9.1	Betriebswirtschaftlicher Hintergrund des Szenarios .....	305
9.2	Technische Beschreibung .....	307
9.2.1	Versenden der Rückmeldung der Garantieanträge .....	307
9.2.2	Eingang der Nachrichten im Integration Server .....	308
9.2.3	Systemübergreifendes Business Process Management .....	308
9.2.4	Nachrichtenausgang .....	312
9.3	Implementierung des Szenarios bei der Linde Group ...	312
9.3.1	Systemlandschaft und Software-Katalog .....	313
9.3.2	Design im Enterprise Services Repository .....	314
9.3.3	Konfiguration im Integration Directory .....	332
9.4	Fazit .....	339

**10 Unternehmensübergreifende Kommunikation über SAP NetWeaver PI ..... 341**

10.1	Betriebswirtschaftlicher Hintergrund des Szenarios .....	341
10.2	Technische Beschreibung .....	342
10.3	Implementierung des Szenarios .....	344
10.3.1	Komponenten des UCCnet-Szenarios .....	344
10.3.2	Entwicklungs- und Konfigurationsobjekte .....	345
10.3.3	Top-down-Ansatz beim Anlegen der Designobjekte .....	346
10.3.4	Automatisches Generieren der Konfigurationsobjekte .....	353
10.4	Fazit .....	364

**11 Implementierung eines Webservice-Szenarios bei Boehringer Ingelheim ..... 365**

11.1	Betriebswirtschaftlicher Hintergrund des Szenarios .....	365
11.2	Technische Beschreibung .....	366
11.3	Implementierung des Webservices .....	367
11.3.1	Modellierung im Enterprise Services Builder ...	367
11.3.2	Schnittstellendesign im Enterprise Services Builder .....	375
11.3.3	Implementierung der Proxys .....	380
11.3.4	Konfiguration des Webservices .....	381
11.3.5	Publizieren in die Services Registry .....	383
11.3.6	Testen des Webservices im WS Navigator .....	385
11.3.7	Konfiguration im Integration Directory .....	386
11.4	Fazit .....	388

**Anhang**

A	Glossar .....	389
B	Die Autoren .....	397
	Index .....	389

*Wir untersuchen in diesem Kapitel Laufzeitkomponenten von SAP NetWeaver PI und erläutern Ihnen, wie Sie sie mithilfe der Monitoring-Werkzeuge überwachen. Im Fokus stehen die Integration Engine und die Advanced Adapter Engine als wichtigste Komponenten von SAP NetWeaver PI und die Proxy-Laufzeit als Laufzeitumgebung für die Programmierung mit ABAP- und Java-Proxys.*

## 7 Laufzeit

Wir haben im letzten Kapitel gesehen, dass Sie über Adapter eine Vielzahl von unterschiedlichen Anwendungssystemen an den Integration Server anschließen können. Natürlich steht hinter jedem Adapter ein eigenes Protokoll und ein eigenes Programmiermodell im Anwendungssystem, die wir im Rahmen dieses Buches allerdings nicht alle behandeln können. Wir beschränken uns daher in Abschnitt 7.3, »Proxy-Laufzeit«, auf das Programmiermodell für Service-Interfaces aus dem Enterprise Services Repository. Da es sich hierbei um WSDL-basierte Interfaces handelt, erläutern wir in Abschnitt 7.3.2, »ABAP-Proxys und Webservices«, welche Rolle Proxys für Webservices spielen und umgekehrt. Um die Verarbeitung einer Message besser verstehen zu können, konzentrieren wir uns aber vorher in den Abschnitten 7.1 und 7.2 auf technische Aspekte der Integration Engine und der Advanced Adapter Engine. Abschließend geben wir in Abschnitt 7.4 eine Übersicht über das Monitoring.

### 7.1 Integration Server und Integration Engine

Alle Messages, die über SAP NetWeaver PI verarbeitet werden, durchlaufen den Integration Server oder die Advanced Adapter Engine. Der Integration Server ist auf dem AS ABAP implementiert und setzt auf dessen Middleware-Technologie auf, beispielsweise auf das Internet Connection Framework (ICF). Im Unterschied dazu ist die Advanced Adapter Engine auf dem AS Java implementiert und nutzt

dessen Middleware-Technologie. Wir wollen uns in diesem Abschnitt einen Überblick über die Konfiguration der Integration Engine (und des Integration Servers) verschaffen und uns die Message-Verarbeitung ansehen. Auf die Advanced Adapter Engine und deren Konfigurationsmöglichkeiten werden wir in Abschnitt 7.2 noch näher eingehen. Im Gegensatz zur Konfiguration im Integration Directory ist die in den folgenden Abschnitten beschriebene technische Konfiguration hauptsächlich Aufgabe eines Systemadministrators.

### 7.1.1 Grundlagen

**Mandanten** Nach der Installation des SAP NetWeaver Application Servers richten Sie für die ABAP-Seite Mandanten ein. Jeder Mandant kann die Rolle eines Senders oder Empfängers übernehmen. Es gibt zwei Fälle zu unterscheiden:

- ▶ Es sollen Messages über den RFC- oder den IDoc-Adapter mit dem Integration Server ausgetauscht werden. In diesem Fall ist der IDoc-Adapter beziehungsweise die Advanced Adapter Engine für das Messaging mit dem Integration Server verantwortlich.
- ▶ Es sollen Messages über Proxys mit dem Integration Server ausgetauscht werden. In diesem Fall kümmert sich eine lokale Integration Engine auf dem SAP NetWeaver AS ABAP um das Messaging. Wir konzentrieren uns in Abschnitt 7.1 bezüglich des Senders und Empfängers von Messages auf die Kommunikation über ABAP-Proxys.

**Konfiguration als Integration Server** Die für das Business-System lokale Integration Engine übernimmt also die Messaging-Aufgaben bei der Kommunikation mit dem Integration Server. Um einen SAP NetWeaver AS ABAP als Integration Server zu konfigurieren, müssen Sie einen Mandanten bestimmen, in dem die Integration Engine als zentraler Integration Server konfiguriert ist. Der Mandant für den Integration Server nutzt also die gleiche Laufzeitkomponente wie Mandanten, für die Sie die Rolle eines Anwendungssystems festlegen: die *Integration Engine*. Mit dem Unterschied, dass Sie über die als Integration Server konfigurierte Integration Engine neben der Messaging-Logik zum Empfangen und Senden von Messages zusätzliche Dienste aufrufen können (beispielsweise Routing und Mapping). Pro SAP NetWeaver AS ABAP kann es nur einen Mandanten geben, in dem Sie eine Integration Engine als Integration Server konfigurieren. Abbildung 7.1 gibt ein Beispiel. Die

lokalen Integration Engines können nur über die als Integration Server konfigurierte Integration Engine Messages austauschen. Ist ein Mandant auf dem SAP NetWeaver AS ABAP als Integration Server konfiguriert, dürfen Sie in produktiven Szenarien keine weiteren Integration Engines des gleichen AS ABAP nutzen.

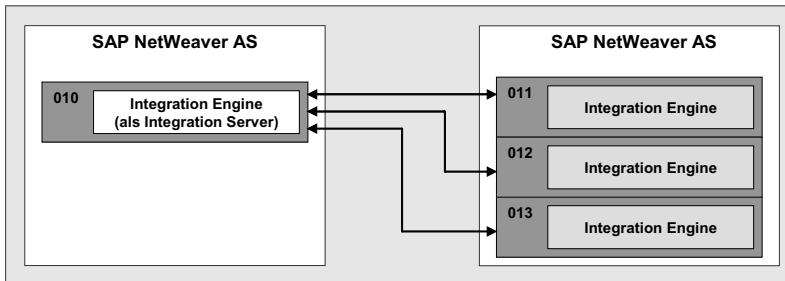


Abbildung 7.1 Integration Engines auf dem SAP NetWeaver AS ABAP

Im Urzustand eines Mandanten ist die Integration Engine weder als lokale Engine noch als Integration Server konfiguriert. Um die Rolle eines Business-Systems festzulegen, müssen in dem jeweiligen Mandanten die *globalen* Konfigurationsdaten konfiguriert werden. Diese Konfiguration wird für den zentralen Integration Server direkt im Anschluss an die Installation von SAP NetWeaver PI im Rahmen der Post-Installation-Aufgaben durch Ausführen eines Configuration Wizards automatisch durchgeführt. Möchten Sie einen Mandanten mit lokaler Integration Engine konfigurieren, führen Sie die Schritte manuell aus. Dazu rufen Sie in der Transaktion SXMB\_ADMIN den Eintrag INTEGRATION ENGINE KONFIGURIEREN auf. Die globalen Konfigurationsdaten sind zwingend notwendig für den Betrieb der Integration Engine. Über die gleiche Transaktion können Sie optional *spezifische* Konfigurationsdaten konfigurieren, um den Message-Austausch zu optimieren beziehungsweise Ihren Anforderungen anzupassen. Die globalen und spezifischen Konfigurationsdaten sind mandantenabhängig und werden von der Transaktion in einer Customizing-Tabelle gespeichert.

Globale und spezifische Konfigurationsdaten

Pro Integration Engine sind die Verarbeitungsschritte der Message in einer *Pipeline* zusammengefasst. Ein Verarbeitungsschritt wird über ein Pipeline-Element abgehandelt, das einen Pipeline-Service aufruft. Auf diese Weise können Pipeline-Services von verschiedenen Pipeline-Elementen aufgerufen werden. Die Zusammenstellung der Pipeline-Elemente ist dabei von SAP fest vorgegeben. Die PI-Laufzeit

Aufbau der Integration Engine

erzeugt für jede zu verarbeitende Message eine eigene Instanz der Pipeline, die exklusiv für die Verarbeitung der Message genutzt wird, bis alle Pipeline-Elemente abgearbeitet worden sind. Eine Pipeline im Sinne des Wortes entsteht deswegen, weil die Laufzeit die Sender-Pipeline, die zentrale Pipeline (also die als Integration Server konfigurierte Integration Engine) und die Empfänger-Pipeline hintereinander abarbeitet.

#### Aufbau einer Message

Abschließend wollen wir uns in diesem Abschnitt ansehen, wie eine Message aufgebaut ist, die von der Integration Engine verarbeitet wird. Das XI-Message-Protokoll der Exchange Infrastructure basiert auf der W3C-Note *SOAP Messages with Attachments*<sup>1</sup>. Der Integration Server erwartet eine Message, die den in Abbildung 7.2 dargestellten Aufbau hat.

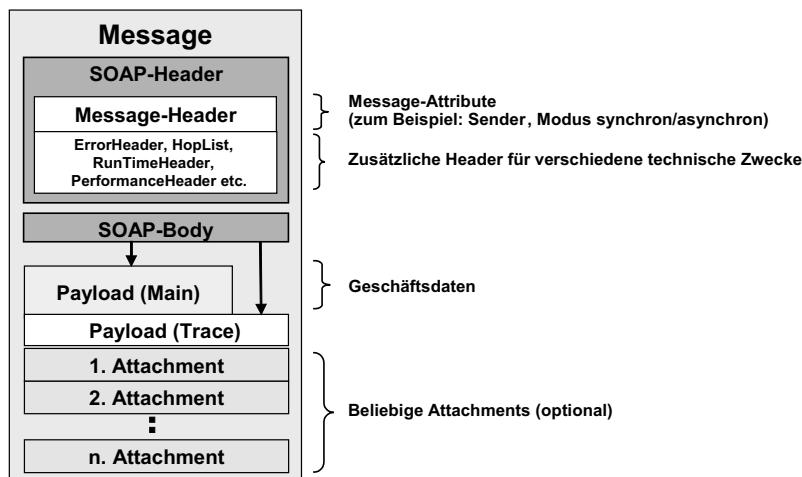


Abbildung 7.2 Message-Format des XI-Message-Protokolls

Alle Sender-Adapter wandeln daher einen Aufruf oder eine Message eines Senders in dieses Format um; die Proxy-Laufzeit erzeugt das Format unmittelbar. Der SOAP-Header einer Message enthält dabei alle für den Integration Server wichtigen Informationen, um eine Message weiterzuleiten, während mit der *Payload* die eigentlichen Geschäftsdaten übertragen werden. Zusätzlich können Sie bei der Proxy-Kommunikation beliebig viele Attachments an die Message anhängen, bevor sie versendet wird. Bei Attachments handelt es sich

<sup>1</sup> Siehe im Internet unter [www.w3.org/TR/SOAP-attachments](http://www.w3.org/TR/SOAP-attachments).

typischerweise um Nicht-XML-Daten, wie Bilder, Textdokumente und Binärdateien. Entscheidend für die Verarbeitung auf dem Integration Server ist, dass die Informationen des Message-Headers das korrekte Format haben. Die Payload bleibt unangetastet, außer es wird ein Mapping darauf ausgeführt. Wenn Sie sich im Message-Monitoring Messages ansehen, wird Ihnen der Aufbau der Message aus Abbildung 7.2 wieder begegnen.

Nach diesen Grundlagen wollen wir uns im nächsten Abschnitt ansehen, wie die Integration Engine Messages verarbeitet.

### 7.1.2 Verarbeitungsschritte einer Message

Wie die Integration Engine Messages verarbeitet, hängt von der *Quality of Service* (QoS) ab. Proxy-Laufzeit (ABAP und Java), lokale Integration Engine, Integration Server und die Advanced Adapter Engine zusammengekommen unterstützen folgende Zustellungsarten:

Quality of Service

- ▶ **Best Effort (BE)**

Synchrone Verarbeitung der Message; der Sender wartet auf eine Antwort, bevor er seine Verarbeitung fortsetzt.

- ▶ **Exactly Once (EO)**

Asynchrone Verarbeitung der Message; der Sender wartet also nicht auf eine Antwort. Die Integration Engine beziehungsweise die Advanced Adapter Engine garantiert, dass die Message genau einmal zugestellt und verarbeitet wird.

- ▶ **Exactly Once In Order (EOIO)**

Wie Zustellungsart EO, nur dass die Anwendung hier Messages über einen Queue-Namen serialisieren kann. Die Integration Engine oder die Advanced Adapter Engine stellen die Messages dieser Queue in der Reihenfolge zu, in der sie vom Sendersystem geschickt werden.

Da bei der Kommunikation mit Proxys die lokale Integration Engine Messages mit dem Integration Server austauscht, werden bei der Proxy-Kommunikation alle hier aufgeführten Qualitys of Service unterstützt. Welche Quality of Service bei anderen Adaptertypen unterstützt wird, ist vom Adapter abhängig. In diesem Fall garantiert das Messaging der Advanced Adapter Engine die jeweilige Quality of Service.

## Asynchrone Verarbeitung

Zur Vereinfachung konzentrieren wir uns im Folgenden auf die Message-Verarbeitung über ABAP-Proxys. Hierzu wollen wir uns zunächst ansehen, wie die Integration Engine asynchrone Messages verarbeitet. Sowohl für die QoS *Exactly Once* als auch für die QoS *Exactly Once In Order* greift die Integration Engine auf den *qRFC-Inbound-Scheduler* des SAP NetWeaver AS ABAP zurück. Es werden folgende qRFC-Queues verwendet:

### ► **Queues zum Senden und Empfangen von Messages**

Für die QoS *Exactly Once* verteilt die Integration Engine die Messages auf verschiedene Queues. Die Namen der Queues werden dabei über festgelegte Präfixe auseinandergehalten. Die Verteilung der Messages auf verschiedene Queues wird über das Suffix dieses Namens gesteuert.

Für die QoS *Exactly Once In Order* teilen sich alle Messages eine Queue, deren Suffix im Anwendungsprogramm explizit vor dem Aufruf des Client-Proxys durch einen *Serialisierungskontext* gesetzt werden muss (siehe auch Abschnitt 7.3, »Proxy-Laufzeit«). Die EOIO-Queue ist also blockierend. Bei EO verteilt die Integration Engine die Messages nach dem Zufallsprinzip. Ist in einer EO-Queue eine Message fehlerhaft, hängt es vom Fehlertyp ab, ob diese Message aus der Queue genommen wird oder nicht. Da es je nach Fehlertyp mehr oder weniger wahrscheinlich ist, ob nachfolgende Messages mit dem gleichen Fehler abbrechen, wird so die Queue entweder gestoppt (zum Beispiel bei Verbindungsproblemen zum Empfänger), oder die fehlerhafte Message wird mit einem Fehler abgebrochen und aus der Queue genommen, sodass die folgenden Messages weiter verarbeitet werden können.

### ► **Queues, um Acknowledgments für asynchrone Messages an den Sender zurückzuschicken**

Um sicherzustellen, dass Acknowledgments auf dem gleichen Weg zurückgeschickt werden, über den vorher die zugehörige Request-Message in Gegenrichtung transportiert wurde, verwendet die Integration Engine die *Hoplist* der Message und sogenannte *Backward-Pipelines*.

### ► **Queues, die speziell für extra große EO-Messages reserviert sind**

Sie können über spezifische Konfigurationsparameter festlegen, ab welcher Größe Messages über diese getrennte Queue verarbeitet werden sollen.

Bevor Sie asynchrone Messages beim Sender, Empfänger und auf dem Integration Server austauschen können, müssen Sie die qRFC-Queues über die Transaktion SXMB\_ADMIN registrieren. Beim Integration Server werden die Queues beim Ausführen des Configuration Wizards automatisch registriert.

Wir verfolgen nun die Verarbeitung einer asynchronen Message anhand von Abbildung 7.3 und gehen zunächst auf die Verarbeitung bei der lokalen Integration Engine beim Sender ein.

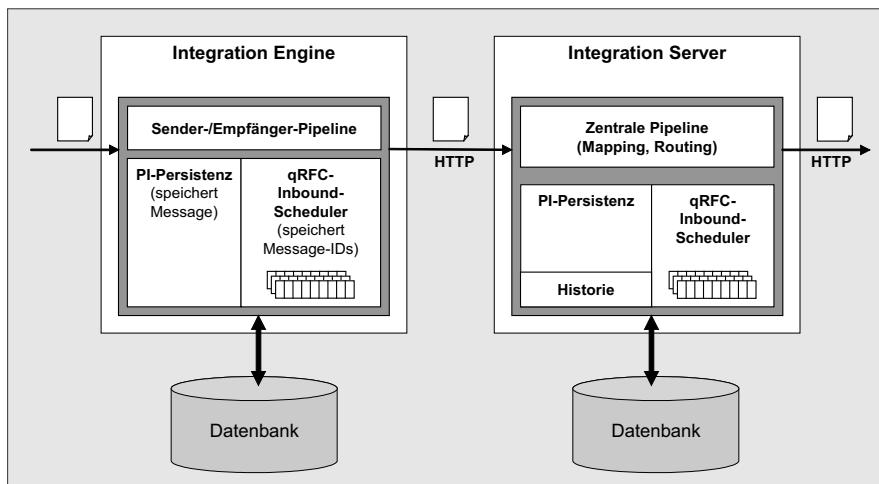


Abbildung 7.3 Asynchrone Message-Verarbeitung

1. Ausgehend von einem Proxy-Aufruf im Anwendungsprogramm (in der Abbildung nicht dargestellt) erhält die lokale Integration Engine die Payload, Attachments und weitere Angaben für den Message-Header, über die die Integration Engine die Message wie in Abbildung 7.2 dargestellt aufbauen kann. Dabei wird eine Message-ID für die Message erzeugt.
2. Die Integration Engine persistiert die gesamte Message über eine Persistenzschicht und plant die Message-Verarbeitung über einen Funktionsbaustein beim qRFC-Inbound-Scheduler ein. Der Funktionsbaustein verweist lediglich auf die Message-ID.
3. Nach der nächsten COMMIT WORK-Anweisung wird die Abarbeitung des Anwendungsprogramms fortgesetzt.
4. Der qRFC-Inbound-Scheduler arbeitet die eingeplanten Funktionsbaustinaufrufe nach dem Round-Robin-Algorithmus ab: Die

Priorisierte  
Message-  
Verarbeitung

Queues haben in der Voreinstellung die gleiche Zeitscheibe; für die Abarbeitung nimmt sich der Scheduler also für jede Queue die gleiche Zeit. Zur priorisierten Message-Verarbeitung können Sie die Zeitscheibe für bestimmte Queues erhöhen und Messages über einen Filter auf diese Queues verteilen.

5. Sobald der qRFC-Inbound-Scheduler den eingeplanten Funktionsbaustein aufruft, liest dieser die Message aus der Persistenzschicht. Die Sender-Pipeline verschickt die Message daraufhin über HTTP.

Am Eingang des Integration Servers laufen diese Schritte ähnlich ab, allerdings mit folgenden Unterschieden:

- ▶ Beim Integration Server ist der Aufrufer, über den die Messages an die Integration Engine übergeben werden, nicht der Proxy aus dem Anwendungsprogramm, sondern die lokale Integration Engine des Senders.
- ▶ Der qRFC-Inbound-Scheduler des Integration Servers plant die Messages für die Verarbeitung in der zentralen Pipeline ein. Die *Historie* ist eine Tabelle, in der die Integration Engine die Message-ID und den Status der Message am Ende eines Bearbeitungsschritts sichert. Auf diese Weise garantiert die Integration Engine die Zustellungsart *Exactly Once* auch dann noch, wenn die Message aus der Persistenzschicht bereits gelöscht oder archiviert wurde.
- ▶ Neue Message-IDs entstehen nur, wenn es beim logischen Routing mehrere Empfänger für die Message gibt. Dann erzeugt die Integration Engine pro Empfänger eine neue Message mit neuer Message-ID und persistiert sie ebenfalls.

**Commit-Handling** Beim Empfänger beginnt das gleiche Spiel mit der lokalen Integration Engine wieder von neuem. Insgesamt verhält sich die Integration Engine bei der asynchronen Message-Verarbeitung *weitestgehend* wie bei einem tRFC- (für EO) beziehungsweise einem qRFC-Aufruf (für EOIO): Temporäre Anwendungsdaten und Aufrufe der Integration Engine werden mit einem expliziten `COMMIT WORK` gemeinsam in der Datenbank festgeschrieben. Im Gegensatz zum tRFC beziehungsweise qRFC werden jedoch verschiedene Aufrufe der Integration Engine innerhalb einer Transaktion auch in verschiedenen Messages verschickt. Jeder Aufruf der Integration Engine erzeugt also eine eigene unabhängige Message. Die transaktionale Klammer über die einzelnen Aufrufe wird dabei nicht zum Ziel des Aufrufs transportiert.

Im Gegensatz zur asynchronen Verarbeitung werden synchrone Aufrufe nicht in eine Queue des qRFC-Inbound-Schedulers gestellt, sondern wirken blockierend für den jeweiligen Aufrufer (Anwendungsprogramm und Integration Engine). Nach einer erfolgreichen Verarbeitung der Request-Message beim Empfänger wird eine neue Message-ID für die Response-Message erzeugt; der Bezug zur ursprünglichen Request-Message bleibt dadurch erhalten, dass die Laufzeit im Message-Header der Response-Message auf die Message-ID der Request-Message verweist (über das Header-Feld `RefTo-MessageId`). Die Integration Engine verhält sich bezüglich des Datenbank-Commits (`DB_COMMIT`) genauso wie bei einem synchronen RFC-Aufruf.

## Synchrone Verarbeitung

### Weitere Konfigurationsmöglichkeiten

Abschließend wollen wir eine Übersicht geben, wie Sie die Message-Verarbeitung zusätzlich beeinflussen können:

#### ► Zeitgesteuerte Message-Verarbeitung

Sie können bei den Zustellungsarten EO und EOIO die Verarbeitung von Messages auf einen späteren Zeitpunkt verschieben. Dazu müssen Sie für die gewünschten Messages einen Filter definieren und einen Job, der die Verarbeitung der über den Filter bestimmten Messages einplant.

#### ► Priorisierte Message-Verarbeitung

Wie schon kurz erwähnt, können Sie für bestimmte Anwendungsfälle die Verarbeitung von Messages mit den Zustellungsarten EO oder EOIO nach Anwendungsfällen unterscheiden und priorisieren. Dazu konfigurieren Sie Filter auf der Basis von Sender- und Empfänger-IDs und weisen ein Queue-Präfix zu. Sie können Messages sowohl höher als auch niedriger priorisieren.

#### ► Message-Paketierung

Zur Verbesserung der Performance der Message-Verarbeitung können asynchrone Messages zu Paketen zusammengefasst und gemeinsam verarbeitet werden. Die Paketerstellung können Sie im Sendersystem und in der zentralen Integration Engine konfigurieren. Im Empfängersystem können Pakete nur empfangen und gespeichert werden, sie werden dann aber als Einzel-Messages verarbeitet. Beachten Sie, dass der Plain-HTTP-Adapter keine Pakete verarbeiten kann und die Advanced Adapter Engine zwar welche

empfangen, jedoch keine erstellen kann. Die Message-Paketierung aktivieren Sie über den Parameter `RUNTIME/PACKAGING` in der Transaktion `SXMB_ADM`. Um die Message-Paketierung Ihren Bedürfnissen anzupassen, haben Sie die Möglichkeit, vorhandene Konfigurationstypen zu ändern oder neue zu erstellen und diese dann bestimmten Sendern und Empfängern zuzuweisen. Die Paketierung kann auch für bestimmte Empfänger deaktiviert werden.

#### ► **XML-Validierung**

Mit der XML-Validierung können Sie die Struktur einer empfangenen oder einer zu sendenden PI-Message-Payload überprüfen. Um die XML-Validierung nutzen zu können, stellen Sie die XML-Schemata aus dem Enterprise Services Builder im Filesystem zur Verfügung und aktivieren die Validierung entweder in der Sendervereinbarung oder der Empfängervereinbarung, je nachdem, ob Sie eine empfangene Message oder eine zu sendende Message überprüfen möchten. Die XML-Validierung kann sowohl in der Advanced Adapter Engine als auch in der Integration Engine durchgeführt werden.

#### ► **Logging**

Das Logging protokolliert den Zustand der zu verarbeitenden Message vor dem ersten Verarbeitungsschritt (Eingangs-Message) sowie nach jedem Aufruf eines Pipeline-Services. Die Integration Engine persistiert hierbei die gesamte Message sowie Informationen zum Status der Verarbeitung, sodass Sie die Message-Verarbeitung im Message-Monitoring überwachen können. In der Voreinstellung ist das Logging für asynchrone und synchrone Messages deaktiviert. Für synchrone Messages bedeutet das, dass sie im Message-Monitoring überhaupt nicht zu finden sind, wenn die Verarbeitung fehlerfrei verlief.

Sie können das Logging in verschiedenen Stufen aktivieren: für eine Integration Engine (beim Sender, Integration Server oder Empfänger), für alle Integration Engines, gezielt für einzelne Pipeline-Services oder über ein Feld im Message-Header. Im letzten Fall werden die Logging-Informationen auch dann im Message-Header gespeichert, wenn das Logging per Konfiguration explizit ausgeschaltet ist.

#### ► **Tracing**

Zur Laufzeit schreiben verschiedene PI-Laufzeitkomponenten Informationen in den Trace, um den Ablauf von Verarbeitungsschritten zu verfolgen.

ten zu dokumentieren. Wie in den Abschnitten 5.2, »Vorkonfiguration und Test von Mapping-Programmen«, 5.3.1, »Java-Mappings«, und 5.4.3, »Fortgeschrittene Message-Mapping-Techniken«, beschrieben, können Sie auch während eines Mapping-Programms Informationen in den Trace schreiben. Je nach konfiguriertem Trace-Level (0: kein Trace; 3: Trace von allen Verarbeitungsschritten) finden Sie mehr oder weniger detaillierte Informationen im Trace. In der Voreinstellung wird der Trace mit Trace-Level 1 geschrieben. Ähnlich wie beim Logging gibt es verschiedene Stufen, um den Trace zu aktivieren: für eine Integration Engine, für alle an der Message-Verarbeitung beteiligten Integration Engines und gezielt für eine Message im Message-Header.

#### ► **Verweilzeiten, Archivieren und Löschen von Messages**

Korrekt verarbeitete Messages werden in der Voreinstellung gelöscht. Sie können allerdings in der Transaktion SXMB\_ADMIN konfigurieren, wie lange die Integration Engine Messages sowie Einträge in der Historie in der Datenbank halten soll, bevor sie gelöscht werden.

Alle Messages, die nicht gelöscht werden sollen, müssen Sie archivieren. Manuell modifizierte oder beendete Messages werden automatisch archiviert. Zum Archivieren legen Sie die Interfaces fest, deren Messages Sie archivieren möchten, und planen einen Job zum Schreiben der Archive sowie einen Job zum Löschen der archivierten Messages ein. (Sollen die Messages lediglich periodisch gelöscht werden, ist nur ein Job erforderlich.) Über diese Mechanismen verhindern Sie ein Überlaufen der Datenbanktabellen.

Nachdem wir uns nun detailliert mit der Konfiguration der Integration Engine beschäftigt haben, sehen wir uns jetzt die Message-Verarbeitung in der Advanced Adapter Engine an.

## 7.2 Advanced Adapter Engine

Mit der Advanced Adapter Engine können Sie einerseits die Integration Engine über Adapter an SAP- (RFC-Adapter) oder an Fremdsysteme anbinden; dazu stehen Ihnen verschiedene Adapter zur Verfügung, mit deren Hilfe XML- und HTTP-basierte Messages in die spezifischen Protokolle und Formate dieser Systeme konvertiert werden und umgekehrt. Andererseits können Sie die Advanced Adapter

Engine nutzen, um lokal Messages zu verarbeiten: Unter den schon in Abschnitt 6.2.3, »Integrierte Konfiguration«, beschriebenen Voraussetzungen können Sie die Performance des Message-Austauschs verbessern, indem Sie das Szenario so konfigurieren, dass die Message nur auf der Advanced Adapter Engine, ohne Beteiligung der Integration Engine, verarbeitet wird. Dazu stellt die Advanced Adapter Engine Mapping und Routing lokal zur Verfügung. Auf die Konfiguration eines solchen Szenarios über das Konfigurationsobjekt *Integrierte Konfiguration* sind wir in Abschnitt 6.2.3 schon detailliert zu sprechen gekommen. In den folgenden Abschnitten möchten wir uns die Laufzeitaspekte der Advanced Adapter Engine etwas genauer anschauen.

### 7.2.1 Grundlagen

#### Zentrale und dezentrale Advanced Adapter Engine

Die Advanced Adapter Engine ist eine eigene Software-Komponente, die bei der Installation von SAP NetWeaver Process Integration automatisch auf dem Integration Server installiert wird. In diesem Fall handelt es sich um die *zentrale* Advanced Adapter Engine. Sie können die Advanced Adapter Engine aber auch separat auf einem anderen Host installieren. Dann handelt es sich um eine *dezentrale* Advanced Adapter Engine.

#### Vorteile von dezentralen Advanced Adapter Engines

Sie können mehrere dezentrale Advanced Adapter Engines an einem Integration Server anbinden. Dadurch bieten sich Ihnen einige Vorteile, die Sie schon bei der Planung der zu konfigurierenden Integrationsszenarien berücksichtigen sollten:

- ▶ **Szenarien isolieren**

Sie haben die Möglichkeit, einzelne Szenarien voneinander zu trennen, um zum Beispiel zeitkritische Szenarien auf eigene Advanced Adapter Engines auszulagern.

- ▶ **Szenarien priorisieren**

Sie können einzelne Szenarien höher priorisieren, indem Sie der entsprechenden Advanced Adapter Engine mehr Hardware-Ressourcen zuweisen.

- ▶ **Szenarien verteilen aufgrund von Netzwerk- oder Sicherheitsaspekten**

Befinden sich anzubindende Fremdsysteme in anderen geografischen Regionen beziehungsweise in anderen Netzwerkzonen,

kann es sinnvoll sein, dezentrale Advanced Adapter Engines zu verwenden.

#### ► **Adaptertypen isolieren**

Zur besseren Ressourcenverteilung kann es unter Umständen vorteilhaft sein, bestimmte Adaptertypen auf separate Advanced Adapter Engines auszulagern.

Mit diesen Konfigurationsoptionen haben Sie die Möglichkeit, störende Einflüsse parallel laufender Szenarien zu minimieren und die Ausfallsicherheit kritischer Szenarien zu verbessern.

Vom funktionellen Standpunkt aus können beide Arten von Advanced Adapter Engines, ob zentral oder dezentral, gleichermaßen für den Message-Austausch verwendet werden. Zur Verarbeitung von Messages in der Advanced Adapter Engine sind im Gegensatz zum Integration Server keine weiteren technischen Einstellungen nötig. Nach der Installation melden sich sowohl die zentrale als auch die dezentralen Advanced Adapter Engines automatisch am angebundenen SLD an und hinterlegen dort ihre Daten, die später in der Laufzeit dafür verwendet werden, die Messages an die richtige Adresse zu senden. Die angemeldeten Advanced Adapter Engines können dann beim Anlegen eines Kommunikationskanals im Integration Directory ausgewählt werden; darüber legen Sie fest, welche Advanced Adapter Engine in dem Szenario zur Laufzeit angesprochen wird. Im Folgenden möchten wir uns nun die Verarbeitung von Messages in der Advanced Adapter Engine etwas genauer anschauen.

Kommunikations-  
daten im SLD

### **7.2.2 Adapter Framework**

Die zentrale Komponente der Advanced Adapter Engine ist das Adapter Framework, dessen Grundlagen wiederum der AS Java und die Connector Architecture (JCA) des AS Java sind. Das Adapter Framework stellt Interfaces zur Konfiguration, Verwaltung und Überwachung von Adaptersn zur Verfügung. Die Konfigurations-Interfaces werden sowohl vom SLD als auch vom Integration Directory verwendet, um Daten der Advanced Adapter Engine zu verwalten beziehungsweise der Advanced Adapter Engine Konfigurationsdaten zur Verfügung zu stellen. Die Interfaces zur Verwaltung und Überwachung werden von der Runtime Workbench im Rahmen des Monitorings und der Administration der Advanced Adapter Engine verwendet.

det. In den folgenden Abschnitten werden wir darauf noch genauer eingehen.

#### Verarbeitung einer Message

Das Adapter Framework hat eigene Queueing- und Protokolldienste, die der Adapter Engine einen Betrieb ohne direkte Verbindung zum Integration Server ermöglichen. Diese Dienste werden zur Verarbeitung einer Message verwendet. Anhand von Abbildung 7.4 verfolgen wir jetzt die Verarbeitung einer Message in der Advanced Adapter Engine:

1. In Senderrichtung ruft der Adapter den Modul-Prozessor und übergibt das Message-Objekt entweder als XI-Message oder in einem eigenen Format. Im letzten Fall muss anschließend die Konvertierung in eine XI-Message in einem adapterspezifischen Modul erfolgen.
2. Aufgrund der Senderinformationen wird die entsprechende Modulkette im Modul-Prozessor zur weiteren Verarbeitung ausgewählt. Das Adapter Framework enthält zwei Standardmodulketten, eine für die Senderrichtung, eine für die Empfängerrichtung. Die Standardmodulketten können durch kundenspezifische Module ergänzt werden. Die Abarbeitung der Message wird durch den Modul-Prozessor kontrolliert.
3. Das letzte Modul in der Modulkette leitet die Message an den Messaging Service weiter, der dann wiederum die Message über HTTP an den Integration Server sendet.
4. Eine Message, die vom Integration Server kommt, wird im Adapter Framework vom Messaging Service entgegengenommen. Aufgrund der Empfängerinformationen wird die entsprechende Modulkette im Modul-Prozessor zur weiteren Verarbeitung ausgewählt. Auch in diese Richtung können kundenspezifische Module aufgerufen werden.
5. Der Modul-Prozessor kontrolliert die Schritte in der Modulkette und arbeitet die Module entsprechend der Konfiguration im Kommunikationskanal ab. Das letzte Modul in der Modulkette leitet die Message an den Adapter weiter.
6. Der Adapter übergibt die Message in Empfängerrichtung an das angeschlossene System.

Diese Verarbeitung einer Message, die entweder vom Integration Server kommt oder an den Integration Server gesendet wird, ent-

spricht der schon seit XI 3.0 implementierten Technologie. Seit SAP NetWeaver PI 7.1 gibt es zusätzlich die Möglichkeit, Messages lokal in der Advanced Adapter Engine zu prozessieren. Die entsprechende Konfiguration über das Konfigurationsobjekt *Integrierte Konfiguration* haben wir in Abschnitt 6.2.3, »Integrierte Konfiguration«, schon besprochen.

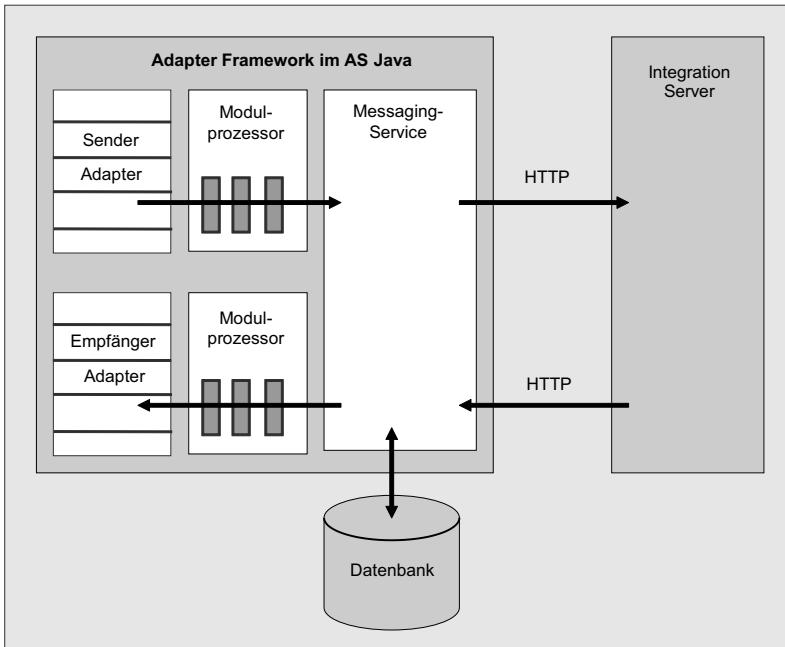


Abbildung 7.4 Message-Verarbeitung in der Advanced Adapter Engine

Wie sich die Verarbeitung in der Advanced Adapter Engine im Detail verhält, werden wir uns jetzt etwas genauer anschauen. Anhand von Abbildung 7.5 verfolgen wir die lokale Verarbeitung einer Message in der Advanced Adapter Engine:

Lokale  
Verarbeitung einer  
Message

1. Der Sender-Adapter ruft den Modul-Prozessor und übergibt das Message-Objekt entweder als XI-Message oder in einem eigenen Format. Im letzten Fall muss anschließend die Konvertierung in eine XI-Message in einem adapterspezifischen Modul erfolgen.
2. Aufgrund der Senderinformationen wird die entsprechende Modulkette im Modul-Prozessor zur weiteren Verarbeitung ausgewählt. Auch die lokale Message-Verarbeitung kann durch kunden-

spezifische Module ergänzt werden. Die Abarbeitung der Message wird durch den Modul-Prozessor kontrolliert.

3. Das letzte Modul in der Modulkette leitet die Message an den Messaging Service weiter. Der Messaging Service ruft dann die nur für die lokale Message-Verarbeitung benötigten Services Receiver- und Interface-Determination auf. Ist ein Mapping in der *Integrierten Konfiguration* hinterlegt, wird die Message zur Ausführung des Mappings an die Mapping-Laufzeit übergeben.
4. Für die Ausgangsverarbeitung übergibt der Messaging Service die Message wieder an den Modul-Prozessor zur weiteren Verarbeitung, der die Module entsprechend der Konfiguration im Kommunikationskanal abarbeitet. Das letzte Modul in der Modulkette leitet die Message an den Adapter weiter.
5. Der Adapter übergibt die Message in Empfängerrichtung an das angeschlossene System.

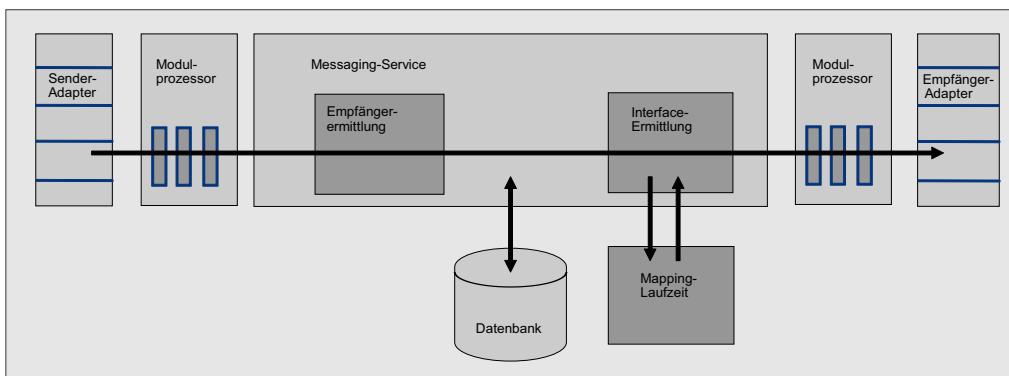


Abbildung 7.5 Lokale Message-Verarbeitung in der Advanced Adapter Engine

Nachdem wir uns eingehend mit der Verarbeitung einer Message in der Advanced Adapter Engine beschäftigt haben, möchten wir noch auf weitergehende Konfigurationsmöglichkeiten eingehen und einige Worte zum Löschen und Archivieren von Messages in der Advanced Adapter Engine hinzufügen.

### Weitere Konfigurationsmöglichkeiten

Um die Message-Verarbeitung zu beeinflussen, stehen Ihnen folgende Möglichkeiten zur Verfügung:

## ► **Message-Priorisierung**

Für die Message-Verarbeitung auf der Advanced Adapter Engine können Sie Regeln definieren, nach denen bestimmte Messages mit unterschiedlicher Priorität (niedrig, normal oder hoch) verarbeitet werden sollen. Zur Definition der Regeln im *Komponenten-Monitoring* in der Runtime Workbench stehen Ihnen die Header-Daten einer Message, also Sender-Partner und -Komponente, Empfänger-Partner und -Komponente und das Interface zur Verfügung. Wird für eine eingehende Message zur Laufzeit keine Regel gefunden, wird sie mit normaler Priorität bearbeitet.

## ► **Bereitschaftszeiten**

Möchten Sie bestimmte Kommunikationskanäle nur während bestimmter oder immer wiederkehrender Zeiten aktivieren, planen Sie für diese Kommunikationskanäle Bereitschaftszeiten. Damit haben Sie die Möglichkeit, Kommunikationskanäle automatisch und unabhängig voneinander zu steuern. Die Bereitschaftszeitenplanung konfigurieren Sie im *Kommunikationskanal-Monitoring* in der Runtime Workbench.

## ► **XML-Validierung**

Wie schon für die Integration Engine vorgestellt, haben Sie auch in der Advanced Adapter Engine die Möglichkeit, mithilfe der XML-Validierung die Struktur einer empfangenen PI-Message-Payload zu überprüfen. Der Sender-Adapter erzeugt dazu erst die PI-MESSAGE und führt anschließend die Validierung der PI-Payload durch. In der Ausgangsverarbeitung wird die XML-Validierung nur in der Integration Engine durchgeführt, die Empfänger-Adapter bieten diese Möglichkeit nicht. Um die XML-Validierung in der Advanced Adapter Engine nutzen zu können, stellen Sie die XML-Schemata aus dem Enterprise Services Builder im Filesystem der Advanced Adapter Engine zur Verfügung und aktivieren die Validierung entweder in der Sendervereinbarung oder der integrierten Konfiguration.

## ► **Archivieren und Löschen von Messages**

Korrekt verarbeitete Messages werden in der Voreinstellung gelöscht. Dazu wird automatisch ein Lösch-Job im AS Java erzeugt, der einmal am Tag ausgeführt wird und alle korrekt verarbeiteten Messages, die älter als 30 Tage sind, löscht. Sie können allerdings im SAP NetWeaver Administrator den Java-EE-Service SAP XI Adapter dahingehend konfigurieren, wie lange die Advanced

Adapter Engine Messages in der Datenbank halten soll, bevor sie gelöscht werden. Alle Messages, die nicht gelöscht werden sollen, müssen Sie archivieren. Manuell modifizierte oder beendete Messages werden automatisch archiviert; dazu wird automatisch ein Archivierungs-Job erzeugt, der täglich einmal ausgeführt wird. Beachten Sie, dass dieser Job in der Voreinstellung inaktiv ist und erst von Ihnen aktiviert werden muss. Zur Archivierung der korrekt verarbeiteten Messages erstellen Sie im *Komponenten-Monitoring* in der Runtime Workbench Archivierungs-Jobs, für die Sie Regeln mit Bedingungen definieren, die eine Message erfüllen muss, um von dem Job archiviert zu werden.

Nach dieser Übersicht über die Integration und die Advanced Adapter Engine wollen wir uns das Programmiermodell für die Proxy-Kommunikation genauer ansehen.

### 7.3 Proxy-Laufzeit

Die Motivation für den Outside-In-Ansatz haben wir bereits in Abschnitt 4.1, »Entwicklung nach dem Proxy-Modell«, erläutert: Ausgehend von einem Service-Interface im Enterprise Services Repository generieren Sie einen Proxy in einem Anwendungssystem, um einen Message-Austausch mit dem Integration Server zu implementieren.

#### Komponenten zur Proxy-Kommunikation

Abbildung 7.6 zeigt, welche Laufzeitkomponenten die Proxy-Kommunikation mit dem Integration Server ermöglichen:

##### ► Proxys

Proxys sind softwarelogistisch Teil der Anwendung. Sie müssen die Proxy-Objekte daher zusammen mit dem Anwendungsprogramm kompilieren und transportieren. Technisch gesehen ist ein Proxy eine Klasse (Outbound) beziehungsweise ein zu implementierendes Interface (Inbound).

##### ► Proxy-Laufzeit

Die Proxy-Laufzeit ist Teil von SAP NetWeaver PI, wobei die ABAP-Proxy-Laufzeit eine Komponente des AS ABAP ist (ab Release 6.40) und die Java-Proxy-Laufzeit zusammen mit SAP NetWeaver PI installiert werden muss. Die Proxy-Laufzeit erzeugt aus

den an einen Consumer-Proxy übergebenen Daten die zu versendende Message beziehungsweise liest empfangene Messages ein, um einen zugehörigen Provider-Proxy aufzurufen.

#### ► Lokale Integration Engine

Die lokale Integration Engine haben wir im letzten Abschnitt kennengelernt, sie ist ebenfalls Teil von SAP NetWeaver PI. Sie stellt in Kooperation mit dem Integration Server das Messaging sicher, also den Empfang und das Versenden von Messages inklusive der gewählten Quality of Service und die Zustandsverwaltung für Messages. Für die Kommunikation über ABAP-Proxys ist sie Teil des SAP NetWeaver AS ABAP. Für die Kommunikation über Java-Proxys für XI-3.0-kompatible Interfaces auf dem Standard EJB 2.0 liefert SAP zusammen mit SAP NetWeaver PI ein *Messaging-System* aus, das die Aufgaben der lokalen Integration Engine auf dem AS Java-Server übernimmt.

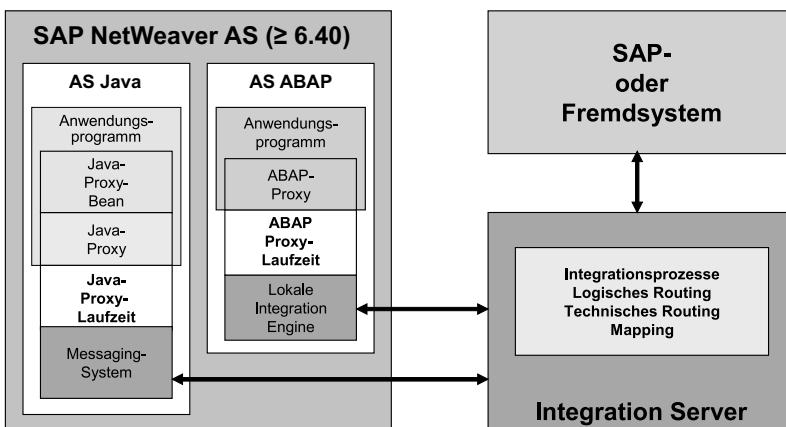


Abbildung 7.6 Kommunikation über Java- oder ABAP-Proxy-Laufzeit

Darüber hinaus wird die Laufzeitkomponente *Web Services Framework* zusammen mit der SAP-Java-EE-Engine ausgeliefert. Sie bildet die Laufzeitumgebung für Java-Proxys, die basierend auf dem Standard EJB 3.0 implementiert wurden.

Im Folgenden konzentrieren wir uns darauf, wie Sie im Anwendungsprogramm mit Proxys programmieren. Dazu gehen wir zunächst auf Konzepte ein, die sowohl für ABAP als auch für die Java-Proxy-Laufzeit gelten.

## Senden und Empfangen von Messages

Um eine Message an den Integration Server zu senden, müssen Sie zunächst ihren Inhalt festlegen und sie dann über einen Proxy-Aufruf versenden. Den Inhalt legen Sie im Anwendungsprogramm über die generierten Proxy-Objekte fest: in Java über die Zugriffsmethoden der generierten Objekte und in ABAP, indem Sie generierten Strukturen und Feldern die gewünschten Werte zuweisen. Schließlich übergeben Sie die Daten über einen Methodenaufruf der generierten Klasse an die Proxy-Laufzeit, die daraus die Message erzeugt (Proxy-Aufruf). Die Proxy-Laufzeit schreibt automatisch den Sender in den Message-Header, die Empfängerinformationen ergeben sich aus der Konfiguration im Integration Directory.

### Synchrone und asynchrone Kommunikation

Ein Proxy-Aufruf entspricht also dem Versenden einer Message. Wenn Sie beim Design ein synchrones Service-Interface angelegt haben, ist die Ausführung des Programms so lange unterbrochen, bis eine Response-Message empfangen wird und die Proxy-Laufzeit dem Anwendungsprogramm die Werte dieser Message über die Rückgabeparameter übergibt. Entsprechend gibt es keine Rückgabeparameter bei asynchroner Kommunikation. In der Voreinstellung stellt die Proxy-Laufzeit solche Messages *Exactly Once* zu. Bei der Übermittlung ist dann unerheblich, in welchen Eingangs- und Ausgangs-Queues der Integration Engine die Messages verarbeitet werden: Die Integration Engine verteilt die Messages dann nach Performance-Gesichtspunkten optimiert auf die Queues. Wenn die Zustellungsart *Exactly Once In Order* garantiert werden soll, dürfen hingegen alle Messages jeweils nur in einer Queue verarbeitet werden, damit die Reihenfolge erhalten bleibt. Den Queue-Namen müssen Sie im Anwendungsprogramm der Proxy-Laufzeit vor dem Proxy-Aufruf als *Serialisierungskontext* mitgeben. Die Java-Proxy-Laufzeit übergibt dabei asynchrone Messages direkt an das Messaging-System, während Sie im ABAP-Anwendungsprogramm asynchrone Messages über mehrere Proxy-Aufrufe bündeln und über eine abschließende `COMMIT WORK`-Anweisung absenden.

Sowohl die ABAP-Proxy-Laufzeit als auch die Java-Proxy-Laufzeit bieten Möglichkeiten, um neben der Payload weitere Informationen an die Proxy-Laufzeit zu übergeben beziehungsweise beim Empfänger abzufragen. Wir wollen kurz die wichtigsten nennen:

## ► Exactly Once In Order und Acknowledgments

Zusätzlich zum bereits erwähnten Serialisierungskontext für *Exactly Once In Order* kann die Proxy-Laufzeit für asynchrone Messages *Acknowledgments* verarbeiten, mit denen Sie sich den Empfang (System-Acknowledgment) und die erfolgreiche Verarbeitung beim Empfänger (Anwendungs-Acknowledgment) bestätigen lassen können. Alle in Abschnitt 6.4.1, »Übersicht«, vorgestellten Empfänger-Adapter unterstützen System-Acknowledgments. Integrationsprozesse und Proxy-Laufzeit unterstützen auch Anwendungs-Acknowledgments. Als einzige Adapter der Advanced Adapter Engine unterstützen die RNIF- und der CIDX-Adapter (*Chemical Industry Data Exchange*) szenarioabhängig auch Anwendungsfehler-Acknowledgments.

## ► Setzen des Empfängers

Sie können den Empfänger einer Message bereits im Anwendungsprogramm setzen. Dies überschreibt aber nicht das logische Routing im Integration Directory, sondern erweitert es: Die Routing-Laufzeit übernimmt den gesetzten Empfänger nur, wenn es dort eine gültige Routing-Regel in der Empfängerermittlung gibt, bei der angegeben ist, dass der Empfänger aus dem Message-Header übernommen werden soll.

## ► Message-Attachments

Die Proxy-Laufzeit erlaubt es, beliebige Text- oder Binärdateien an die Message zu hängen beziehungsweise beim Empfänger abzufragen.

## ► Abfragen der Payload

Sie können die Payload der Message abfragen, beispielsweise um sie zu archivieren.

## ► Abfragen der Message-ID

Nach dem Versenden der Message können Sie die Message-ID abfragen, beispielsweise um sie in ein Anwendungslog zu schreiben.

Beim Empfänger implementiert die Anwendungsentwicklung das von der Proxy-Generierung erzeugte ABAP-Objekt beziehungsweise Java-Interface. Wenn die lokale Integration Engine eine Message an die Proxy-Laufzeit weiterleitet, ist im Message-Header allerdings nicht dieses Interface, sondern das im Design angelegte Service-Interface enthalten. Die Proxy-Laufzeit muss zu diesem Service-Interface die implementierende Klasse und die darin aufzurufende Methode ermitteln. Daher müssen Sie empfangende Service-Interfaces bei der

Implementierung  
des Provider-  
Proxys

Proxy-Laufzeit registrieren. (Wir haben uns dies schon ansatzweise in Abschnitt 4.1.2, »Proxy-Generierung«, angesehen.) Nach einer erfolgreichen Verarbeitung der Message beim Empfänger löst die ABAP-Proxy-Laufzeit ein COMMIT WORK aus. Bevor wir auf weitere Besonderheiten in den folgenden Abschnitten eingehen, wollen wir uns noch der Fehlerbehandlung bei der Kommunikation mit Proxys zuwenden.

## **Fehlerbehandlung**

Mit der Proxy-Laufzeit können Sie auf zwei Fehlertypen reagieren:

### ► **Systemfehler**

Diese Fehler treten beim Transport der Message auf und werden von einer PI-Laufzeitkomponente ausgelöst, beispielsweise wenn kein Empfänger ermittelt werden konnte. Der Sender sollte diesen Fehler unbedingt über die Ausnahmeklasse CX\_AI\_SYSTEM\_FAULT (ABAP) beziehungsweise SystemFaultException (Java) abfangen. Beim Empfänger persistiert die Proxy-Laufzeit Systemfehler für das Monitoring (asynchrone Kommunikation) beziehungsweise schickt den Fehler zurück zum Sender (synchrone Kommunikation).

### ► **Anwendungsfehler**

Diese Fehler treten beim Empfänger auf und sind anwendungsspezifisch, beispielsweise wenn eine Anfrage beim Empfänger nicht beantwortet werden kann, weil die übermittelte Kundennummer im Zielsystem nicht bekannt ist. Dieser Fehlertyp ist primär für synchrone Kommunikation bedeutend, bei asynchroner Kommunikation persistiert die Laufzeit Fault-Messages für das Monitoring.

Fault-Message-Typ	Die Struktur von Fault-Messages legen Sie im Enterprise Services Builder fest: Sie besteht immer aus einem Standardteil und optional aus einem anwendungsspezifischen Teil. Im Standardteil legen Sie essenzielle Informationen über den Fehler ab (Fehlercode, Art des Fehlers und möglicherweise eine URL mit weiterführenden Hinweisen). Die Struktur des anwendungsspezifischen Teils legen Sie über einen beliebigen Datentyp fest, den Sie dem Fault-Message-Typ zuweisen. Die Proxy-Generierung erzeugt für jeden Fault-Message-Typ eine Ausnahmeklasse, über die der Sender den Fehler im Anwendungsprogramm über einen Try-Catch-Block abfängt. Abbildung 7.7 gibt ein Beispiel mit einer Java-Anwendung als Sender. Alle Ausnah-
-------------------	--

meklassen haben die gleiche Superklasse: CX\_AI\_APPLICATION\_FAULT (ABAP) beziehungsweise ApplicationFaultException (Java).

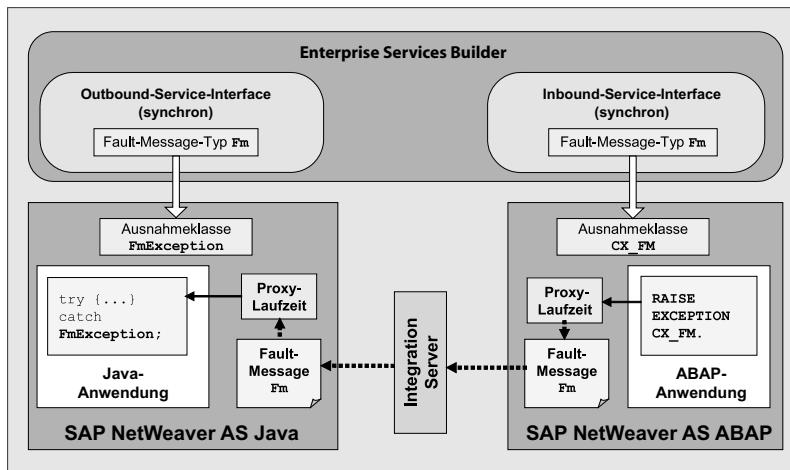


Abbildung 7.7 Fehlerbehandlung mit Fault-Messages

### 7.3.1 Besonderheiten bei der Kommunikation über Java-Proxys

In Abschnitt 4.1, »Entwicklung nach dem Proxy-Modell«, haben wir gesehen, dass bereits bestehende Java-Proxys für XI-3.0-kompatible Interfaces auf dem Standard EJB 2.0 nachgeneriert werden können. Die Neuentwicklung von Java-Proxys erfolgt im Rahmen von EJB 3.0 und unterstützt die Webservice-Standards. Die Unterschiede im Programmiermodell schlagen sich natürlich auch in unterschiedlichem Laufzeitverhalten nieder.

Für bestehende XI-3.0-kompatible Java-Proxy-Implementierungen unterstützt die Java-Proxy-Laufzeit J2EE-Anwendungen auf der SAP-J2EE-Engine unter Verwendung von Enterprise JavaBeans 2.0. Die Java-Proxy-Generierung erzeugt hierzu folgende Klassen:

Java-Proxys mit EJB 2.0

- ▶ Proxy-Klassen, die über die Java-Proxy-Laufzeit Messages versenden beziehungsweise empfangen, und Java-Klassen für die verwendeten Datentypen.
- ▶ Bean-Klassen als äußere Hülle, die den J2EE-Standard erfüllt. Die Bean-Klassen rufen die Proxy-Klassen für die Kommunikation auf. Sie sind die in der Bean-Programmierung üblichen home-, remote-, local home- und local-Interfaces.

In der J2EE-Anwendung programmieren Sie den Message-Austausch über die generierten Bean-Klassen, die Sie wie die Java-Klassen zusammen mit der Anwendung deployen.

Abbildung 7.8 zeigt die Verarbeitung einer eingehenden Message. Wie wir bereits festgestellt haben, muss die Proxy-Laufzeit ausgehend von dem Service-Interface im Message-Header den letztlich aufzurufenden Service ermitteln können. Im Fall von J2EE-Anwendungen müssen Sie daher für jedes Service-Interface eine Server-Bean und den Namen der zugehörigen Bean-Methode in der *JPR-Registry* registrieren. Für den synchronisierten Zugriff in der J2EE-Cluster-Umgebung verwenden Sie ein Servlet des sogenannten *Proxy-Servers*. Sie registrieren die Interfaces in der Regel einmal als initiale Konfiguration der Laufzeit, können aber auch zur Laufzeit weitere Interfaces registrieren beziehungsweise bestehende Interfaces deregistrieren. Der Proxy-Server liest bei jeder Änderung der Registry diese neu aus, ohne dass der Proxy-Server neu gestartet werden muss. Alle Kommandos, die Sie dem Proxy-Server-Servlet schicken, haben den folgenden Aufbau:

`http://<host>:<port>/ProxyServer/<Kommando>`

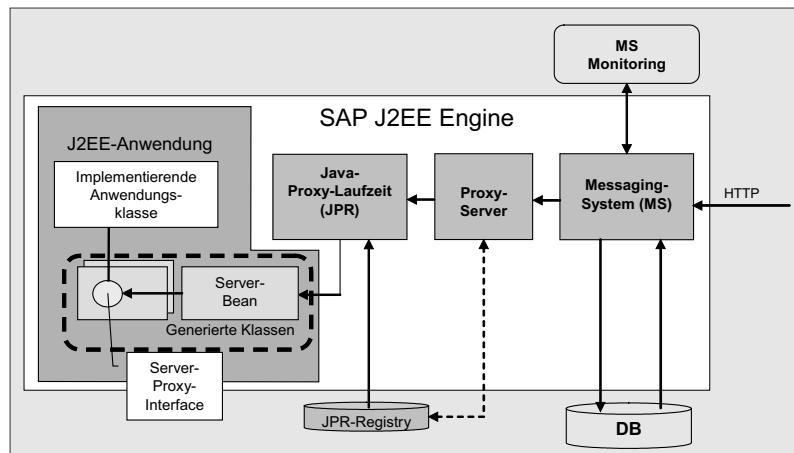


Abbildung 7.8 Java-Proxy-Laufzeit als Empfänger

Beispielsweise gibt das Kommando `listAll` alle registrierten Service-Interfaces in alphabetischer Reihenfolge aus. Mit dem Kommando `jprtransportservlet` führt das Proxy-Server-Servlet einen Selbsttest durch.

Abschließend wollen wir noch ein paar allgemeine Besonderheiten der Java-Proxy-Laufzeit erwähnen:

► **Co-located Beans**

Da die Java-Proxy-Laufzeit EJB 2.0 unterstützt, können Sie `co-located Beans` einsetzen. Diese Beans werden in dem gleichen EJB-Containersystem deployt und auf derselben Java VM (*Java Virtual Machine*) ausgeführt und versprechen daher eine bessere Performance. Um `co-located Beans` von `remote Beans` zu unterscheiden, verwenden Sie das Präfix `localejbs/`.

► **Indirekte Outbound-Kommunikation**

Die Client-Proxy-Bean kann nicht nur von einer J2EE-Anwendung, sondern auch von einer J2SE-Anwendung gerufen werden. Dazu müssen Sie die Bean beim J2EE-Server über die Datei `jndi.properties` registrieren.

Vielleicht haben Sie sich zudem schon gefragt, inwiefern Proxys beziehungsweise SAP NetWeaver PI die Verwendung von Webservices unterstützen, schließlich basieren Service-Interfaces auf der Web Service Description Language (WSDL). Wir gehen auf diesen Punkt in den nächsten Abschnitten ein. Die dort beschriebene Harmonisierung bezüglich der Programmierung beziehungsweise Kommunikation über Webservices oder über PI-Proxys ist sowohl für Java- als auch für ABAP-Proxys realisiert.

Falls Java-Proxys basierend auf dem Standard EJB 3.0 implementiert werden, werden diese im Outbound-Fall als *Webservice-Clients* und im Inbound-Fall als *Webservices* bezeichnet. Die Änderungen beschränken sich jedoch nicht auf eine reine Umbenennung der Objekte: Mit dem neuen Programmiermodell ergeben sich auch neue Möglichkeiten der Kommunikation. Nun ist es neben dem Versenden von Messages über SAP NetWeaver PI auch möglich, direkt Messages über Webservice-Protokolle mit den Webservices auszutauschen.

Java-Webservices mit EJB 3.0

Das konkrete Vorgehen bei der Service-Implementierung sowie die Details zu den generierten Klassen und Methoden wurden bereits in Abschnitt 4.1, »Entwicklung nach dem Proxy-Modell«, vorgestellt. Wir wollen daher an dieser Stelle nur auf die Konfiguration von Java-Webservices und Java-Webservice-Clients eingehen. Sobald der Webservice beziehungsweise der Webservice-Client auf dem Java-Applikationssystem *deployt* ist und sofern Sie Administrator-Berech-

tigung haben, können Sie mit der Konfiguration mithilfe des SAP NetWeaver Administrators (NWA) beginnen. Die konkrete Vorgehensweise hängt von folgenden Faktoren ab:

- ▶ Man unterscheidet zwischen der Konfiguration von Provider- (Webservices) und Consumer-Proxys (Webservice-Clients).
- ▶ Die Konfiguration kann zentral auf dem Integration Server oder lokal in den Java-Anwendungssystemen durchgeführt werden. Öffnen Sie hierfür den NWA auf dem Integration Server im zentralen Modus (alias /nwapi) beziehungsweise auf dem Applikationssystem im lokalen Modus (alias /nwa).
- ▶ Java-Webservices können einzeln oder als Webservice-Gruppen konfiguriert werden. Im zweiten Fall werden Webservices zu Business-Szenarien zusammengefasst und können dann gemeinsam konfiguriert werden.

Nachdem wir uns eingehend mit den ABAP- und Java-Proxys auseinandergesetzt haben, schauen wir uns nun die Verbindung zwischen ABAP-Proxys und Webservices an.

### 7.3.2 ABAP-Proxys und Webservices

Bisher haben wir uns bei der Proxy-Kommunikation auf den Austausch von Messages mit dem Integration Server konzentriert. Die ABAP-Proxy-Laufzeit unterstützt aber eigentlich zwei verschiedene Szenarien:

#### ▶ PI-Laufzeit

Über die PI-Laufzeit tauschen Sie Messages über den Integration Server aus. Sie konfigurieren den oder die Empfänger der Message in diesem Fall zentral über das Integration Directory und können auf das Routing, Mapping und auf Integrationsprozesse zurückgreifen.

#### ▶ Webservice-Laufzeit

Die Webservice-Laufzeit ist Teil des SAP NetWeaver AS ABAP. Sie können unabhängig von SAP NetWeaver PI die Webservice-Laufzeit nutzen, um Point-to-Point-Services aufzurufen. Die wesentliche Idee dahinter ist, dass Geschäftspartner eine sprachunabhängige Beschreibung dieses Services als WSDL-Dokument veröffentlichen. Um so einen Service aufzurufen, generieren Sie einen ABAP-Consumer-Proxy.

Wir beschreiben im Folgenden die ABAP-Proxy-Lauffzeit aus PI-Sicht, das heißt: Wir gehen von einer Proxy-Kommunikation über die PI-Lauffzeit aus und untersuchen, welche Erweiterungen notwendig sind, um die Webservice-Lauffzeit zu nutzen. Ansonsten ist das Programmiermodell identisch. Abbildung 7.9 stellt diese beiden Szenarien dar. Prinzipiell können Sie zwischen beiden Lauffzeitumgebungen umschalten, wenn die Integrationslogik des Integration Servers nicht benötigt wird. In diesen Fällen darf die Anwendung aber nur diejenigen Funktionen der PI-Lauffzeit nutzen, die auch durch die Webservice-Lauffzeit unterstützt werden, und umgekehrt.

ABAP-Proxy-Lauffzeit aus PI-Sicht

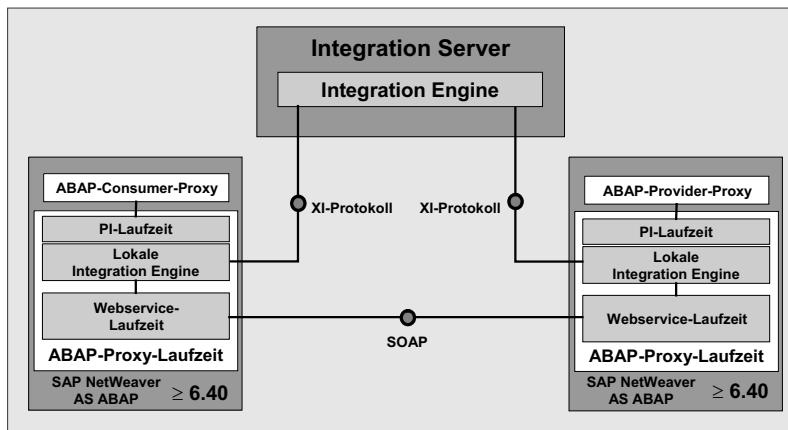


Abbildung 7.9 PI- und Webservice-Lauffzeit

Bei einem Proxy-Aufruf über die PI-Lauffzeit ist keine Angabe eines Empfängers notwendig, denn der wird über die Konfiguration des Routings im Integration Directory bestimmt. Für die direkte Kommunikation über die Webservice-Lauffzeit konfigurieren Sie diesen Empfänger im Sendersystem über einen logischen Port mithilfe der Transaktion SOAMANAGER oder über das Konfigurationsobjekt *Direkte Verbindung*, das wir uns in Abschnitt 6.2.5, »Direkte Kommunikation«, schon angeschaut haben. Die Einstellungen eines Ports enthalten neben der Adressierung des Empfängers beispielsweise noch Optionen für das Logging und Tracing sowie Sicherheitseinstellungen. Im Anwendungsprogramm geben Sie dann diesen Port bei der Instantiierung des Proxys an, beispielsweise:

```
CREATE OBJECT
    lo_clientProxy('LOGICAL_PORT_NAME').
```

Logischer Port

Die Angabe des Ports ist für die PI-Laufzeit optional. Wird der Port angegeben, bestimmen Sie auf der Registerkarte XI INTEGRATION in der Definition des logischen Ports in der Transaktion SOAManager, ob Sie die PI- oder die Webservice-Laufzeit nutzen wollen. Wird der Port bei der Instantiierung nicht angegeben, ist die PI-Laufzeit aktiv.

Point-to-Point-Verbindungen	Wenn bei einer Kommunikation zwischen zwei ABAP-Proxys keine Dienste des Integration Servers benötigt werden, können Sie über den logischen Port auf die Webservice-Laufzeit umschalten und dadurch den Message-Austausch beschleunigen. Zusätzlich zum logischen Port benötigen Sie dann einen über die Transaktion SOAManager freigegebenen Webservice zum generierten Provider-Proxy. Um den Empfänger beim logischen Port zu adressieren, geben Sie die WSDL-URL des freigegebenen Webservices beim logischen Port ein.
Protokolle	Ansonsten unterscheidet sich das Programmiermodell nur durch die zur Verfügung stehenden Protokolle. Da die PI-Laufzeit Routing unterstützt, gibt es beispielsweise ein Routing-Protokoll, über das Sie den Empfänger setzen können. Der Zugriff auf das Protokoll ist in beiden Fällen gleich: Sie holen sich über die Methode <code>GET_PROTOCOL</code> des Proxys eine Protokollinstanz, die die erforderlichen Attribute und Methoden für das Protokoll anbietet. Unterstützt die aktive Laufzeit das Protokoll nicht, löst sie eine Ausnahme aus.
WSDL und Services-Interfaces	Sie können wie eben dargestellt von einer PI-Kommunikation über ABAP-Proxys auf eine Kommunikation über Webservices umschalten. Umgekehrt ist das nicht möglich, denn Service-Interfaces unterstützen nur eine Teilmenge des WSDL-Sprachumfangs, wie wir bereits in Abschnitt 4.1.1, »Service-Interface-Entwicklung im Enterprise Services Builder«, gesehen haben. Das XI-Protokoll erwartet beispielsweise eine vorgegebene Struktur von Fault-Messages, in WSDL ist diese Struktur beliebig.
Erweiterter Webservice	Zusätzlich zu der Point-to-Point-Verbindung über die Webservice-Laufzeit und der Verbindung über die PI-Laufzeit gibt es noch eine dritte Variante: Da der Integration Server SOAP-Messages verarbeiten kann, können Sie einen Webservice auch über den Integration Server aufrufen. Der Integration Server empfängt die SOAP-Message des Aufrufers und leitet die Message gemäß der Konfiguration im Integration Directory an einen Empfänger weiter. Der Empfänger muss daher auch kein Proxy sein, sondern ist beliebig. Da Sie bei die-

sem Webservice-Aufruf die Dienste des Integration Servers nutzen können, spricht man von einem *erweiterten* Webservice. Folgende Schritte sind notwendig, um einen solchen Webservice zu definieren:

1. Sie benötigen ein Service-Interface, um eine WSDL-Beschreibung für den Aufrufer generieren zu können. Dies kann sowohl ein Outbound- als auch ein Inbound-Service-Interface sein. In der weiteren Konfiguration wird das Service-Interface nicht unbedingt benötigt; es beschreibt lediglich die Signatur für den Aufrufer.
2. Rufen Sie im Integration Builder für die Konfiguration das Menü WERKZEUGE • WSDL ANZEIGEN auf. Mit diesem Assistenten erzeugen Sie eine WSDL-Beschreibung über die folgenden Angaben:
  - ▶ Die Adresse des Integration Servers oder eines anderen Webservers, der den Webservice-Aufruf entgegennehmen soll.
  - ▶ Das Service-Interface aus dem ersten Schritt, um die Aufruf-Signatur über das WSDL-Dokument zu publizieren.
  - ▶ Angaben über den Sender der SOAP-Message (Partner, Service, Outbound-Interface). Wenn der Integration Server die SOAP-Message empfängt, benötigt er diese Angaben für die Auswertung der Konfigurationsdaten. Sie geben die Informationen über den Sender beim logischen Routing, gegebenenfalls beim Operation-Mapping und in den Kommunikationsvereinbarungen an.
3. Der Aufrufer kann mithilfe der generierten WSDL-Beschreibung einen Consumer-Proxy generieren und über seine Webservice-Laufzeit die SOAP-Message an den Integration Server senden. Dort konfigurieren Sie für die SOAP-Message den Empfänger.

Damit wollen wir die Übersicht über die Programmierung mit Proxys abschließen und uns im nächsten Abschnitt die Möglichkeiten zur Message-Überwachung ansehen.

## 7.4 Monitoring

Wie wir bisher in diesem Kapitel gesehen haben, sind an einem systemübergreifenden Message-Austausch neben den beteiligten Sender- und Empfängersystemen eine ganze Reihe von Laufzeitkomponenten beteiligt. Dieses Kapitel gibt eine Übersicht über die Möglichkeiten, diese Laufzeitkomponenten zu überwachen.

**Runtime Workbench** In Abbildung 7.10 sind im oberen Bereich die Komponenten des Monitorings abgebildet. Die *Runtime Workbench* ist ein Java-basiertes Werkzeug, das über eine Weboberfläche einen zentralen Zugriff auf alle Monitoring-Bereiche bietet und das Sie wie die PI-Konfigurationswerkzeuge und das SLD über die PI-Startseite aufrufen. Die Runtime Workbench nutzt bereits vorhandene Monitoring-Komponenten des SAP NetWeaver AS: das *Computer Center Management System* (CCMS), die *Process Monitoring Infrastructure* (PMI) und das Alert Framework. Das Prozess-Monitoring des PMI ermöglicht unter anderem die Überwachung eines durchgängigen Prozesses, der mehrere Komponenten umfasst, und hat keine Bedeutung für das Monitoring eines Integrationsprozesses. Wir gehen auf Letzteres in Abschnitt 8.6, »Ausführung eines Integrationsprozesses überwachen«, ein.

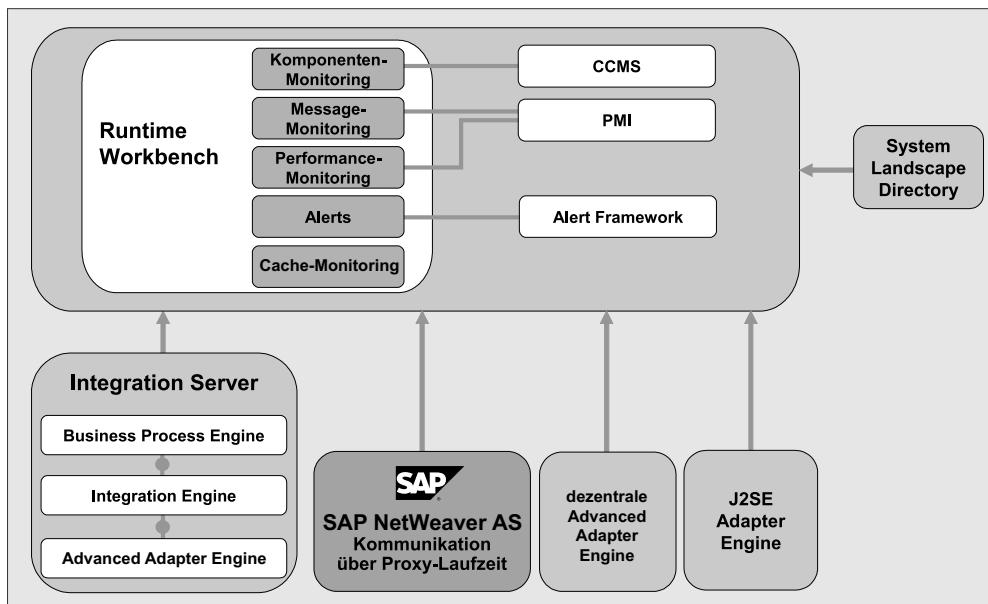


Abbildung 7.10 Bereiche des Monitorings

Mit dem Monitoring über die Runtime Workbench überwachen Sie alle PI-Laufzeitkomponenten (*Komponenten-Monitoring*), die Message-Verarbeitung auf dem Integration Server und übergreifend (End-to-End) über alle PI-Laufzeitkomponenten (*Message-Monitoring*) sowie die Geschwindigkeit und den Durchsatz der Message-Verarbeitung (*Performance-Monitoring*). Zusätzlich können sich Administratoren über Alerts über auftretende Fehler benachrichtigen lassen (*Alert Framework*).

Für das End-to-End-Monitoring und das Performance-Monitoring wertet die Runtime Workbench Daten aus dem PMI (*Process Monitoring Infrastructure*) aus. Dazu müssen Sie in der Runtime Workbench im Bereich KONFIGURATION festlegen, welche Komponenten mit welchem Monitoring-Level überwacht werden sollen. Alle anderen Laufzeitkomponenten (Integration Server, Advanced Adapter Engines und Proxy-Laufzeit) übermitteln ihre Daten, ohne dass eine Konfiguration in der Runtime Workbench nötig ist. Über das CACHE-MONITORING können Sie sich auch den Cache der Integration Engine beziehungsweise der Advanced Adapter Engine anzeigen lassen. Auf diese Weise lassen Sie sich Werte-Mapping-Gruppen anzeigen und erfahren, welche Mapping-Programme beziehungsweise Software-Komponentenversionen im Cache vorhanden sind.

Daten für das Monitoring

Die verschiedenen Monitoring- und Konfigurationsoberflächen sind neben der Runtime Workbench auch im SAP Solution Manager aufrufbar. Der *SAP Solution Manager* wird als eigenständiges, zentrales System installiert und kann Sie während des gesamten Lebenszyklus Ihrer Lösungen – vom Business Blueprint über die Konfiguration bis hin zum Produktivbetrieb – unterstützen. Der SAP Solution Manager bietet einen zentralen Zugriff auf Werkzeuge, Methoden und vorkonfigurierte Inhalte, die Sie während der Evaluierung und Implementierung sowie beim operativen Betrieb Ihrer Systeme nutzen können. Im Work Center *Systemadministration* können Sie auf die verschiedenen PI-Werkzeuge und Monitore verzweigen, die dann auf dem angeschlossenen PI-System aufgerufen werden. Wir werden im Folgenden etwas genauer auf die wichtigsten Monitoring-Bereiche eingehen, die sowohl in der Runtime Workbench als auch im SAP Solution Manager verfügbar sind.

Weitere Monitoring-Werkzeuge

## Komponenten-Monitoring

Mit dem Komponenten-Monitoring überwachen Sie alle PI-Laufzeitkomponenten. In Abbildung 7.11 ist die Statusübersicht für den Integration Server dargestellt. Der Status der Advanced Adapter Engine enthält Warnungen (gelb), der Status der Integration Engine ist fehlerhaft (rot), die Status der Business Process Engine und der Mapping Runtime sind fehlerfrei (grün), und der Status der dezentralen Advanced Adapter Engine ist unbekannt (grau). Ein grauer Status bedeutet, dass keine Informationen über die jeweilige Komponente vorliegen, etwa weil das Monitoring für diese Komponente nicht aktiviert

Statusübersicht

wurde. Um sich weitere Details anzeigen zu lassen, klicken Sie auf den Status. Im unteren Bereich der Runtime Workbench können Sie dann je nach Laufzeitkomponente einen Selbsttest durchführen, Konfigurationsdaten zur Komponente abrufen, in das Kommunikationskanal-Monitoring verzweigen oder an die Komponente eine Test-Message senden.

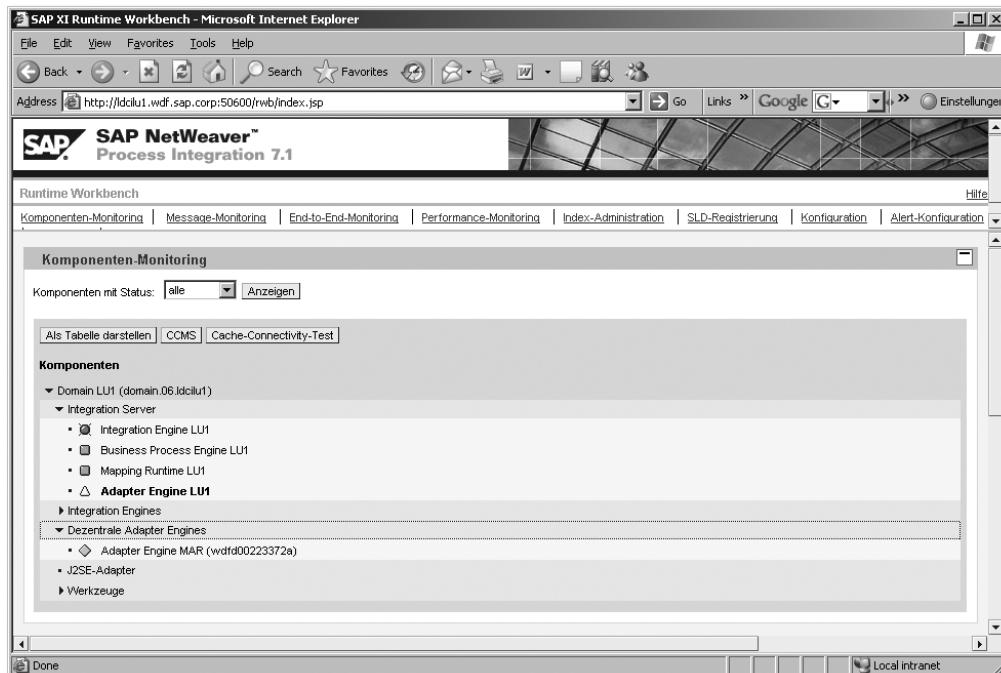


Abbildung 7.11 Übersicht über den Status der PI-Laufzeitkomponenten

**CCMS** Um Probleme einer Komponente zu beheben, starten Sie das CCMS, das über das SAP GUI for HTML in die Runtime Workbench integriert ist. Mit dem CCMS überwachen Sie alle Systeme Ihrer Systemlandschaft, rufen statistische Informationen Ihrer Systeme ab, starten diese Systeme beziehungsweise fahren sie herunter und vieles mehr. Abbildung 7.12 zeigt den Systemstatus für das System an, auf dem der Integration Server installiert ist. Die Queues mit dem Präfix XBTS\* im Mandanten 105 des qRFC-Inbound-Schedulers, den wir in Abschnitt 7.1.2, »Verarbeitungsschritte einer Message«, kennengelernt haben, sind blockiert. Um sich genauere Informationen anzeigen zu lassen, können Sie von hier direkt in den qRFC-Monitor verzweigen.

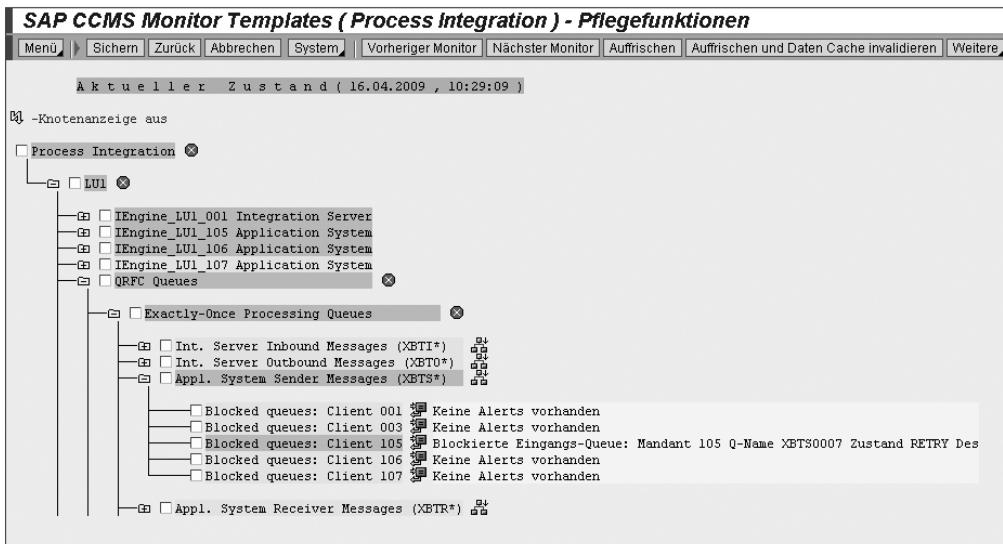


Abbildung 7.12 Arbeiten mit dem CCMS über die Runtime Workbench

Im CCMS können Sie eine automatische Alert-Benachrichtigung definieren, über die Administratoren über E-Mail, Pager oder Fax informiert werden, dass es Probleme mit einer Komponente gibt. Wir sehen uns im Folgenden an, welche Möglichkeiten die Runtime Workbench für die Überwachung der Message-Verarbeitung bietet, und im Anschluss, wie Sie speziell hierfür Alerts in der Runtime Workbench konfigurieren.

Alerts

### Message-Monitoring

Im Message-Monitoring überwachen Sie die Verarbeitung von Messages. Wie umfangreich die Informationen sind, die im Message-Monitoring angezeigt werden, hängt von den Einstellungen beim Logging und Tracing der jeweiligen Laufzeitkomponente ab. Beispielsweise sind in der Voreinstellung keine fehlerfrei verarbeiteten synchronen Messages im Monitoring sichtbar.

Nachdem Sie den Link MESSAGE-MONITORING gewählt haben, müssen Sie die Anzahl der verarbeiteten Messages einschränken. Dazu wählen Sie zunächst eine Laufzeitkomponente aus und selektieren, von wo Sie Messages anzeigen möchten. Sie haben hier die Möglichkeit, Messages direkt aus der Datenbank zu selektieren, eine Übersicht über einen bestimmten Zeitraum anzeigen zu lassen, archivierte

Messages selektieren

Messages im Archiv zu suchen und indizierte Messages zu selektieren. Bei der Suche über einen Index benötigen Sie die Search and Classification Engine (TREX) und müssen die Indizierung zuvor über die Index-Administration einrichten.

Grenzen Sie nun die zu selektierenden Messages über den Verarbeitungszeitpunkt und weitere Felder im Message-Header ein. Mit dem Feld STATUS können Sie sich beispielsweise alle fehlerhaften Messages eines bestimmten Zeitraumes anzeigen lassen. Über die daraufhin angezeigte Liste können Sie folgende Aktionen für einzelne Messages ausführen:

- ▶ Sie können fehlerhafte Messages über WIEDERHOLEN manuell erneut an den Empfänger senden, beispielsweise wenn vorübergehend ein Empfänger nicht erreichbar war.
- ▶ Fehlerhafte Messages können über BEENDEN aus der Verarbeitung genommen werden, um sie danach archivieren zu können.
- ▶ Über DETAILS lassen Sie sich weitere Informationen zu einer ausgewählten Message in der Liste anzeigen. Dies beinhaltet die Anzeige der Message in den verschiedenen Verarbeitungsschritten der Pipeline durch die Integration Engine.<sup>2</sup>
- ▶ Über den MESSAGE-EDITOR können Sie die Payload einer fehlerhaften asynchronen Message editieren. Das gibt Ihnen die Möglichkeit, Messages, die wegen eines Fehlers in der Payload zu einem Fehler in der Laufzeit geführt haben, zum Beispiel beim Mapping, anzupassen und nochmals zu verarbeiten.
- ▶ Da die Message auf dem Transportweg vom Sender zum Empfänger über mehrere Laufzeitkomponenten durchgereicht wird, gibt es von der Liste im Message-Monitoring einen direkten Absprung zum End-to-End-Monitoring. Voraussetzung hierfür ist, dass Sie das PMI für alle beteiligten Laufzeitkomponenten im Bereich KONFIGURATION der Runtime Workbench aktiviert haben.

#### End-to-End-Monitoring

Das End-to-End-Monitoring können Sie auch direkt in der Runtime Workbench aufrufen. Sie bekommen dann für einen optional festzulegenden Sender und Empfänger folgende Informationen angezeigt:

---

<sup>2</sup> Die Monitoring-Transaktionen der Integration Engine sind alternativ im System über die Transaktion SXMB\_MONI erreichbar.

## ► Prozessübersicht

Die Gesamtzahl der (fehlerhaft) verarbeiteten Messages aller Komponenten. Sobald es bei der Verarbeitung durch eine Komponente eine oder mehrere fehlerhafte Messages gibt, wechselt der Status dieser Komponente von Grün auf Rot.

## ► Instanzsicht

Die Instanzsicht zeigt den Weg einer bestimmten Message durch die beteiligten Komponenten (es geht also um die Message-Instanz). Abbildung 7.13 zeigt ein Beispiel, in dem die Message-Verarbeitung auf dem Weg zum Empfänger abgebrochen wurde. Die zugehörige Komponente ist in Rot dargestellt.

Der Rahmen STATISTIKDATEN zeigt neben Performancedaten die Anzahl der fehlerhaften, offenen und erfolgreich beendeten Messages an. Um sich diese Messages auflisten zu lassen, klicken Sie auf die jeweilige Statuskategorie. Für eine so ermittelte Message in der Liste können Sie die grafische Übersicht aus Abbildung 7.13 aktualisieren.

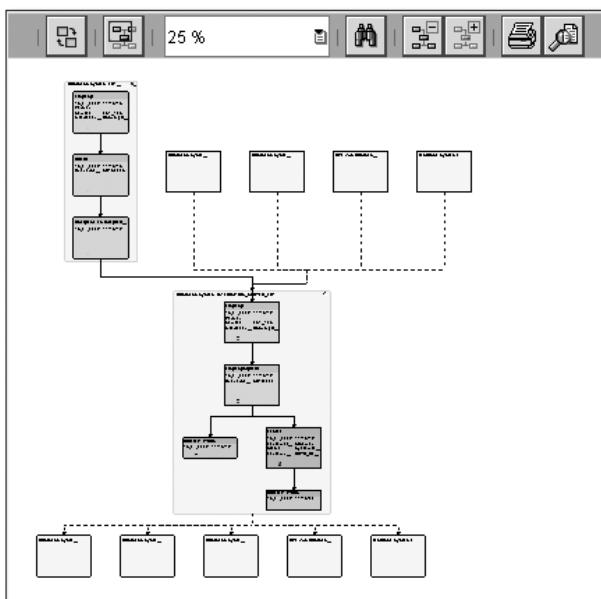


Abbildung 7.13 Übersicht im End-to-End-Monitoring

Schließlich wollen wir uns am Ende dieses Abschnitts noch ansehen, welche Möglichkeiten Administratoren haben, um sich im Fehlerfall benachrichtigen zu lassen.

## Alerts

**Konfiguration** Das Alerting von SAP NetWeaver PI stützt sich auf die Alert-Infrastruktur (CCMS) des SAP NetWeaver AS, wobei nun zusätzlich message-orientierte Alerts (Benachrichtigungen) konfiguriert werden können. Um über Fehler während der Message-Verarbeitung per E-Mail, Fax oder SMS informiert zu werden, verwenden Sie die **ALERT-KONFIGURATION** der Runtime Workbench. Die Konfiguration setzt vereinfacht ausgedrückt einen ausgewählten Fehlercode in eine Benachrichtigung um. Alerts werden dabei über *Alert-Kategorien* klassifiziert.

Abbildung 7.14 gibt eine Übersicht über die Konfiguration von Alerts. Sie ermöglicht die Verteilung von Alerts auf verschiedene Benutzer(gruppen), die sich für eine Kategorie von Alerts abonnieren können. Die Zuordnung zu einem Benutzer erfolgt dabei in mehreren Schritten:

- Benutzerzuordnung**
1. Zunächst muss der Alert selbst definiert werden. Dazu legen Sie eine Alert-Kategorie mit folgenden Informationen an:
    - Die Alert-Definition enthält allgemeine Attribute zum Alert, beispielsweise wie oft der Alert maximal zugestellt werden soll.
    - Im Benachrichtigungstext können Sie über *Containerelemente* Informationen aus dem Message-Header in den Alert mit aufnehmen.
    - Schließlich vergeben Sie eine *Subskriptionsberechtigung* (eine Benutzerrolle). Diese Berechtigung benötigen alle Benutzer, die sich später in der *Alert-Inbox* der Runtime Workbench für die Alert-Kategorie abonnieren wollen. Tut dies ein Benutzer, trägt das System ihn automatisch in die *feste Empfängerliste* ein. Alternativ können Sie die Alert-Kategorie allen Benutzern einer Rolle zuordnen. Den Benutzern muss unabhängig davon auch die Rolle der Subskriptionsberechtigung zugewiesen sein.
  2. Um Fehlercodes in Alerts umzusetzen, müssen Sie eine *Alert-Regel* definieren. Sie beschreiben die Bedingungen, unter denen ein Alert ausgelöst werden soll, und ordnen die Regel einer Alert-Kategorie zu.
  3. Ist eine Alert-Regel einer Alert-Kategorie gültig, werden alle Benutzer benachrichtigt, die sich für die Kategorie in ihrer Alert-Inbox abonniert haben: Die Alerts erscheinen in jedem Fall in der

Alert-Inbox und können zusätzlich per E-Mail, Fax oder SMS an den Empfänger weitergeleitet werden (*Personalisierung*); bei der zeitabhängigen Alert-Zustellung zu einem bestimmten Zeitpunkt, ansonsten direkt. Die Weiterleitung an einen Stellvertreter ist ebenfalls möglich.

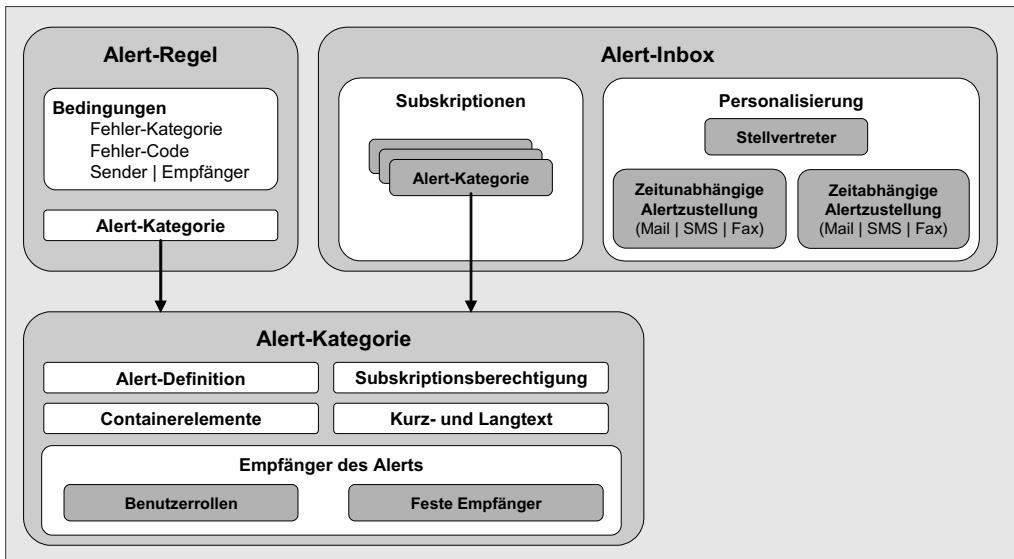


Abbildung 7.14 Konfiguration von Alerts

Mit der Alert-Konfiguration wollen wir dieses Kapitel abschließen. Wir haben damit alle Aspekte der zustandslosen Message-Verarbeitung in den Kapiteln 1 bis 7 behandelt. Zustandsbehaftete Message-Verarbeitung bedeutet, dass auf dem Integration Server Informationen über einen systemübergreifenden Prozess gehalten werden. Wir sehen uns diese Integrationsprozesse im nächsten Kapitel an.

# Index

## A

---

ABAP-Proxy-Generierung 111, 114  
ABAP-Proxy-Objekt 114  
Abschnitt, paralleler 309  
Acknowledgment 242, 257  
Acknowledgment-Message 98, 389  
Action 78, 80, 81, 347, 389  
Adapter 119  
Adapter Engine 389  
Adapter Framework 28, 249  
Adapter-Metadaten 215, 233  
Advanced Adapter Engine 26, 28, 58, 197, 247, 389  
  *dezentrale* 248, 390  
  *zentrale* 248  
Agentur 201  
  *vergebende* 207, 395  
Alert 272, 273  
ALE-Szenario Linde Group 307  
Änderungsliste 48, 68, 389  
Anwendungsfehler 258  
Anwendungskomponente 72, 347, 389  
Archiv, importiertes 352  
Ausgangsverarbeitung 390  
Ausnahme 291  
Authentifizierung 218

## B

---

B2B-Kommunikation 341  
Bean-Klasse 259  
Bearbeitungsprotokoll 49  
Benutzerentscheidung 287  
Bereitschaftszeit 253  
Best Effort 241  
Black-Box-Ansatz 90  
Bottom-up-Entwicklung 85  
BPM 277, 278, 305, 308  
Business Process Engine 26, 276, 295, 390  
Business Process Management → BPM  
Business-Komponente 209

Business-Objekt 88, 390  
Business-Prozess → Integrationsprozess  
Business-System 75, 174, 313, 355, 390  
Business-System-Gruppe 235  
Business-Szenario → Integrations-szenario

## C

---

Cache 52  
ccBPM 278, 305, 390  
CCMS 266  
CCTS 103  
Change and Transport System 70, 236  
Change Management Service → CMS  
Chem eStandards 226, 231, 390  
CIDX-Adapter 226  
CMS 70, 235, 390  
Collaboration Knowledge 390  
Collect Pattern 309  
Commit-Handling 244  
Component View 77, 360  
Computer Center Management System  
  → CCMS  
Consumer-Proxy 380, 390  
Containerelement  
  *für Alerts* 272  
  *im Integrationsprozess* 280, 331  
Containerinhalt sortieren 325  
Containeroperation 281, 311  
Content-Manager 65  
Conversation-ID 283  
Core-Datentyp 103, 104  
Cross-component Business Process  
  Management → ccBPM  
CTS 70, 236

## D

---

Darstellung 201  
Data Warehousing 20  
Datenfluss-Editor 154, 159, 160

- Datentyp 98, 103  
*aggregierter* 105  
*einfacher Typ* 107  
*eingebauter* 108  
*Erweiterung* 98, 130  
*globaler* 376  
*komplexer Typ* 107  
*Mapping* 103, 139, 167  
*Namensraum* 106  
*XSD importieren* 106
- Definition, externe 124
- Demo-Beispiel 58  
*Übungsaufgaben* 60
- Denormalisierung 208, 211
- Deployment Unit 90, 372
- Designobjekt 34, 390
- Designzeit 34, 390
- Direktverbindung 29, 203, 390
- Document Object Model (DOM) 148
- Document Type Definition (DTD) 227
- Doppelaktion 228
- 
- E**
- Eingangsverarbeitung 390
- Einzelaktion 228
- EJB 259
- Empfänger-Adapter 391
- Empfängerermittlung 177, 195, 334, 391  
*dynamische* 191  
*für Integrationsprozesse* 295  
*generische* 184  
*operationsspezifische* 192  
*spezifischere* 184  
*Standard* 191
- Empfängerliste 272
- Empfängerregel 191
- Empfängervereinbarung 177, 184, 190, 391
- Endlosschleife 323
- Endpunkt 382
- End-to-End-Monitoring 391
- Enterprise JavaBean 259
- Enterprise Service 391
- Enterprise Services Browser 380
- Enterprise Services Builder 34, 45, 391
- Enterprise Services Repository 32, 34, 314, 346
- Entwicklungsobjekt 345
- ES Workplace 57
- ESR-Namensraum 65
- EU-Verordnung 365
- Exportfunktion  
*in Operation-Mappings* 147  
*von BPEL* 86  
*von WSDL* 96
- Exportparameter 165
- Extensible Markup Language → XML
- Extensible Stylesheet Language Transformations → XSLT
- 
- F**
- Fault-Message 98, 258
- Feld, generisches 188
- Fremdprodukt 64
- Fremdsystem 127, 173
- FTP-Senderadapter 306
- Funktion, benutzerdefinierte 162
- Funktionsbibliothek, lokale 163
- 
- G**
- Garantieantragsszenario 305
- Generierungsprotokoll 185
- Geschäftsszenario 71
- Geschäftsvorgang 227
- 
- H**
- Hauptinstanz 74
- Historie 48
- Hoplist 242
- HTTPS 218
- 
- I**
- IBM WebSphere Message Queue 357
- Idempotenz 102
- Identifikationsschema 201, 207, 391
- Identifikator 207, 391  
*adapterspezifischer* 221  
*alternativer* 208
- IDoc  
*Adapter* 220

IDoc (Forts.)  
*empfangen* 324  
*Import* 316  
*Inhalte sammeln* 325  
*Partner* 222  
*XML* 220

IDoc-IDoc-Kommunikation 194, 306

Importparameter 165

Inbound-Interface 30, 99, 391

Inbound-Service-Interface 378

Informationsmanagement 20

Inside-Out-Entwicklung 32, 120, 391

Integration Builder 45

Integration Directory 34, 391  
*Linde Group* 332

Integration Engine 26, 238, 295  
*lokale* 26, 255  
*Pipeline* 239

Integration Server 238, 391

Integration-Directory-Objekt 353

Integrationsobjekt 314

Integrationsprozess 22, 76, 141, 170, 276, 391  
*Block* 279, 325  
*Containerelement* 280  
*dynamische Verarbeitung* 288  
*Empfangsschritt* 282  
*Export* 293  
*Fristüberwachung* 289  
*Konfiguration* 295  
*Korrelation* 283, 289, 319, 323  
*Laufzeit-Cache* 302  
*Linde Group* 319, 321  
*Mehrfachbedingung* 287  
*paralleler Abschnitt* 286, 328  
*Schleife* 287  
*Schrittyp* 280  
*Sendeschritt* 282  
*Signatur* 319  
*Transformationsschritt* 286, 329  
*vordefinierter* 276

Integrationsszenario 71, 78, 170, 262, 347, 391  
*Konfigurator* 181, 343, 359  
*Modell* 366, 372, 392  
*Verweis* 81

Integrierte Konfiguration 38, 197, 389, 392

Interaktion 90

Interface  
*Beschreibung* 93, 121  
*Ermittlung* 143, 177, 191, 193, 195, 334, 392

iWay 343

**J**


---

Java Proxy Runtime 260

Java Web Start 45

Java-Mapping 150

Java-Proxy-Generierung 115, 118

JCA 343

JMS-Adapter 344, 357

JPR 260

**K**


---

Klasse AbstractTransformation 150

Klassifizierung 54

Kommunikation  
*asynchrone* 83  
*direkte* 203  
*Komponente* 178, 332, 354, 392  
*Partner* 79, 208, 353, 392  
*Profil* 172, 177, 186, 392  
*synchrone* 82  
*Vereinbarung* 177, 184, 189, 392

Kommunikationskanal 184, 189, 334, 356, 392  
*Monitoring* 253  
*Vorlage* 181

Konfiguration  
*Daten* 239  
*kollaborativer Prozess* 42, 169  
*Objekt* 34, 186, 345, 392  
*Profil* 55  
*Szenario* 171, 332, 392  
*technische* 41  
*Test* 196  
*Übersicht* 195  
*Zeit* 34, 393

Kontext 160  
*Objekt* 135, 297

Kopierfunktion 130

Kopplung  
*enge* 32  
*lose* 31

Korrelation 285, 309  
*aktivieren* 322  
*Linde Group* 319

## L

---

Laufzeitkonstante der Mapping-  
 Laufzeit 152  
 Linde Group 305  
 Logging 246  
 Lookup 166

## M

---

Mandant 221  
 Mapping 137, 393  
*Importiertes Archiv* 148, 352  
*Linde Group* 329  
*Testumgebung* 146  
*Trace-Level* 146  
*Vorlage* 166, 167  
 Mapping-Editor 142, 153  
*benutzerdefinierte Funktion* 162  
*Datenfluss-Editor* 154  
*Kontext* 159  
*Queue* 158  
*Standardfunktion* 160  
*Strukturübersicht* 154  
*Testwerkzeug* 196  
*Übersichtsfunktion* 155  
*Zieldfeld-Mapping* 154  
*Zielstruktur* 157  
 Mapping-Programm 138, 213, 393  
*in Java* 151  
*in Operation-Mappings* 144  
*in XSLT* 152, 352  
 Mapping-Trace in Java-Programmen  
 151  
 Massenkonfiguration 55  
 Mehrfachbedingung 310, 326  
 Message 393  
*Definition* 316  
*Editor* 270  
*Header* 135  
*ID* 243, 244, 245, 257  
*Instanz* 128, 134  
*Interface* 260  
*Monitoring* 196  
*Paketierung* 245

Message (Forts.)  
*Priorisierung* 253  
*sammeln* 308  
*Schema* 124, 127  
*sortieren* 310  
*Typ* 97, 98, 393  
*verschmelzen* 311  
*versenden* 311  
 Message Mapping 153, 393  
*Kontext* 160  
*Queue* 158  
*Standardfunktion* 160  
*vollständiges* 158  
 Message-Split 158  
*mapping-basierter* 193  
 Message-Verarbeitung  
*asynchrone* 242  
*priorisierte* 243, 245  
*synchrone* 245  
*zeitgesteuerte* 245  
 Messaging-System 255  
 Modellierung, konzeptionelle 22, 393  
 Monitoring  
*Alert* 272  
*der Performance* 266, 271  
*End-to-End* 270  
*von Caches* 267  
*von Integrationsprozessen* 303  
*von Komponenten* 267  
*von Messages* 269, 303  
 Multi-Mapping 141, 142, 148, 158,  
 193, 329, 393

## N

---

Nachrichtenchoreografie 278  
 Namensraum 314, 393  
*Repository* 65, 128  
*XML* 128, 134  
 Normalisierung 208

## O

---

Objekteigenschaft 315  
 Objektprüfung 49  
 Operation 88, 393  
*Mapping* 142, 144, 352, 393  
*Pattern* 101  
 Outbound-Interface 99, 393

Outbound-Service-Interface 380  
 Outside-In-Entwicklung 33, 94, 393

---

**P**

Paket 64  
 Parameter 165  
 Partner Connectivity Kit (PCK) 206, 212  
 Partner Interface Process (PIP) 227  
 Payload 107, 240, 393  
 PI-Benutzerrolle 43  
 PI-Content 42, 62, 68, 394  
 PI-Laufzeit 262  
 Pipeline 239, 394  
 Principal Propagation 218, 220, 394  
 Process Integration Content → PI-Content  
 Process Monitoring Infrastructure (PMI) 266  
 Produkt 63  
 Provider-Proxy 380, 394  
 Proxy 93, 108, 133, 254, 312, 394  
     *ABAP-Consumer-Proxy* 111  
     *ABAP-Provider-Proxy* 112  
     *Aufruf* 256  
     *Consumer-Proxy* 109  
     *Generierung* 129, 380, 394  
     *Laufzeit* 26, 180, 254, 258  
     *Nachgenerierung* 110, 118  
     *Objekt* 394  
     *Provider-Proxy* 109  
     *Server* 260  
 Prozess, kollaborativer 24, 61, 170, 392  
 Prozesskomponente 87, 89, 366, 368, 373, 394  
 Publizieren 383

---

**Q**

qRFC 242, 244  
 Quality of Service 241, 255, 394

---

**R**

Release-Übernahme 67, 394  
 Remote Function Call 30, 220  
 Repository-Namensraum 128

Representation-Term 103  
 Request 143  
 Request-Message 97, 394  
 Response 143  
 Response-Message 97, 394  
 RFC 30, 220  
 RFC-RFC-Kommunikation 194  
 RNIF-Adapter 219  
 Rolle 73, 76  
 RosettaNet 226, 394  
 RosettaNet Implementation Framework (RNIF) 227  
 Round-Robin-Algorithmus 243  
 Routing  
     *empfängerabhängiges* 211  
     *logisches* 194, 334  
     *payload-basiertes* 211  
 Runtime Workbench 266, 394

---

**S**

Sanktionsliste 365  
 SAP  
     *Business Package* 229, 232  
     *Interface* 121, 122  
     *Knowledge Management* 20  
     *SAP Alert Management* 290  
     *SAP Business Suite* 22  
     *SAP Business Workflow* 24, 278  
     *SAP Business Workplace* 287  
     *SAP Integration Scenario Model* 372  
     *SAP ProComp Interaction Model* 373  
     *SAP ProComp Model* 368, 370  
     *SAP Solution Manager* 23, 267, 395  
     *SAP System* 173, 221  
 SAP NetWeaver 17  
     *SAP NetWeaver Administrator (NWA)* 262, 395  
     *SAP NetWeaver Application Server* 18  
     *SAP NetWeaver Business Process Management* 19  
     *SAP NetWeaver Business Rules Management* 19  
     *SAP NetWeaver Collaboration* 21  
     *SAP NetWeaver Developer Studio* 116  
     *SAP NetWeaver Master Data Management* 20

- SAP NetWeaver Mobile* 21  
*SAP NetWeaver Portal* 21  
*SAP NetWeaver Process Integration* 19  
 Secure Sockets Layer 218  
 SEI 117  
 Sendekontext 282, 296  
 Sender-Adapter 395  
 Sendervereinbarung 177, 395  
 Sendeschritt 296  
 Sequenz 83  
*dynamische* 310  
 Serialisierungskontext 242, 256, 395  
 Service  
*Definition* 112  
*Endpunkt* 55  
*Gruppe* 55  
*Orchestrierung* 278  
 Service Endpoint Interface (SEI) 117  
 Service-Interface 33, 88, 94, 95, 99, 100, 144, 316, 349, 377, 395  
*abstraktes* 99, 232, 280, 282, 389  
*Fault-Message-Typ* 258  
*Freigabezustand* 102  
*importieren* 123  
*Interface-Pattern* 100  
*Kategorie* 99  
*Modus* 101  
*Operation* 97  
*Sicherheitsprofil* 100  
 Services Registry 42, 45, 52, 383, 395  
 Shared Collaboration Knowledge 25  
 Simple Use Case 57  
 SLD 62, 172, 313  
 SOAP-Message 264  
 Software-Katalog 172, 173, 313  
 Software-Komponente 63, 346  
 Software-Komponentenversion 69, 313, 314, 395  
*ABAP-Proxy-Generierung* 111  
*für Mappings* 144  
*importieren* 65  
*kundenspezifische* 132  
 SSL 218  
 Subskriptionsberechtigung 272  
 Suchfunktion im Navigationsbaum 50  
 Suchhilfe 50  
 SXI\_CACHE 52  
 System Landscape Directory → SLD
- System, logisches 221  
 Systemfehler 258  
 Systemlandschaft 313, 333  
 Szenario → Konfigurationsszenario
- 
- T**
- Tabellenstruktur importieren 125  
 Tentative Update & Confirm/  
*Compensate* 100  
 Top-down-Entwicklung 85, 345  
 Trace 246  
 Transaktion SOAManager 381, 383  
 Transformation 312  
 Transport  
*filesystem-basiert* 235  
*mithilfe des CMS* 70, 235  
*über CTS* 70, 236  
*Ziel* 235  
 tRFC 244  
 TU&C/C 100
- 
- U**
- UCCnet Data Pool Service 342, 343, 358  
 UCCnet-Adapter 343, 357  
 UN/CEFACT Core Components Technical Specification 103
- 
- V**
- Verarbeitung, zustandsbehaftete 275  
 Verbindung 82, 395  
 Verdichtung 329  
 Verwendungsnachweis 50
- 
- W**
- Web Services Framework 255  
 Webservice 53, 96, 261, 395  
*aktivierter/deployter* 53  
*Client* 261  
*erweiterter* 264  
*konfigurierter* 53  
*konsumieren* 56  
*Laufzeit* 112, 262  
*logischer Port* 263  
*modellierter* 53

- Wesbservice (Forts.)  
  *Navigator* 56, 385, 395  
  *Publikation* 53  
  *Webservice-Wizard* 116  
WebSphere Message Queue 344  
Werte-Mapping 200  
  *Gruppe* 202  
  *Kontext* 141  
  *Massenbefüllung* 202  
  *Tabelle* 140, 202  
While-Schleife 287  
Workflow 278  
WSDL 55, 95, 395  
  *Export* 96  
  *externe Definition* 124
- XI-Message-Protokoll 26, 240  
XML 25, 395  
  *Validierung* 190, 246, 253, 396  
XML Schema 103, 107  
XML-Namensraum 128  
  *Datentyp-Erweiterung* 134  
XPath 135, 152, 363  
XSD-Editor 103  
  *Export* 97  
  *Häufigkeit* 108  
  *Spalte* 107  
XSLT 25, 152  
  *Java-Erweiterung* 153  
  *Mapping* 350

## Z

---

- Zielfeld-Mapping 154

## X

---

- XI-3.0-Protokoll 206