Applies to:
SAP GRC NFE 1.0, SAP Process Integration 7.0 / 7.1
For more information, visit the Governance, Risk, and Compliance homepage.

Summary
By default, the outbound B2B interfaces of the SAP GRC NFE solution require that, for each Business Partner that receives the NF-e XML document, a corresponding Party/Service and Communication Channel must be created in the Integration Directory of SAP PI. This article explains how to make it possible to create just one single Party/Service and Communication Channel, by using SAP PI advanced features (Lookup API & Dynamic Configuration) to dynamically define the communication channel details, specifically, the e-mail address of a Receiver Mail Communication Channel.

Author: Henrique Pinto
Company: SAP Brazil
Created on: 22 January 2009

Author Bio
Henrique Pinto works at SAP Brazil, focusing on integration related solutions and development link for Latin-American specific products, including product management for the SAP GRC NFE solution. He is at SAP since Feb/07.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Preparation</td>
<td>3</td>
</tr>
<tr>
<td>RFC Definition</td>
<td>3</td>
</tr>
<tr>
<td>SLD Configuration</td>
<td>4</td>
</tr>
<tr>
<td>Implementation</td>
<td>5</td>
</tr>
<tr>
<td>Integration Repository</td>
<td>5</td>
</tr>
<tr>
<td>Message Mapping</td>
<td>5</td>
</tr>
<tr>
<td>Interface Mapping</td>
<td>8</td>
</tr>
<tr>
<td>Integration Directory</td>
<td>9</td>
</tr>
<tr>
<td>Receiver RFC Communication Channel</td>
<td>9</td>
</tr>
<tr>
<td>Receiver Mail Communication Channel</td>
<td>9</td>
</tr>
<tr>
<td>Interface Determination</td>
<td>11</td>
</tr>
<tr>
<td>Testing</td>
<td>12</td>
</tr>
<tr>
<td>Further Considerations</td>
<td>13</td>
</tr>
<tr>
<td>Additional Dynamic Mail fields</td>
<td>13</td>
</tr>
<tr>
<td>Dynamic Attachment Name</td>
<td>14</td>
</tr>
<tr>
<td>Mail Package</td>
<td>14</td>
</tr>
<tr>
<td>Alternative Adapters</td>
<td>14</td>
</tr>
<tr>
<td>Related Content</td>
<td>15</td>
</tr>
<tr>
<td>Copyright</td>
<td>16</td>
</tr>
</tbody>
</table>
Introduction

Besides the government communication interfaces, the SAP GRC NFE solution includes several interfaces for the exchange of XML documents between the company and its Business Partners (suppliers and customers).

During the configuration of the Outbound B2B interfaces in SAP PI, by default, it is required that, for each Business Partner that is going to receive the NF-e XML documents, it is necessary to create one Party/Service and a specific communication channel, in the Integration Directory.

However, most of the times, it is impractical to create one Party/Service and communication channel for each customer that is going to receive the NF-e XML documents. Even if some tool like Integration Directory API is used to create all ID objects in bulk, it is also not practical to maintain a business field (e.g. the e-mail address of a customer, usually maintained in the Customer Master Data) in a technical tool like SAP PI.

By using a couple of advanced features of SAP PI, in particular, the Mapping Lookup API and Dynamic Configuration of Adapter Specific Message Attributes (ASMA), it is possible to create just one dummy Business Service and communication channel, dynamically retrieving in runtime the e-mail address of each customer that is going to receive the NF-e XML document message. The following steps explain how to achieve this in a simple way.

Also, notice that in our example, we'll use the Receiver Mail adapter, but that in a real implementation, any receiver adapter in XI could be used for this communication; the communication protocol just needs to be agreed between the business partners participating in the integration (XML sender and receiver).

Preparation

RFC Definition

In order to decide for which Business Partner to send the XML message to, the NFE Outbound B2B interface contains a field with the Brazilian government unique identifier of the receiver company (CNPJ code). The standard recommendation is to use this field to select between the several communication parties / business services (conditional routing).

For our case, though, we'll use this field in order to retrieve the receiver e-mail address. In order to retrieve this data, we'll use a Z RFC that receives the CNPJ code as input parameter and returns the e-mail address based on this code. To store the e-mail address, it could be used a Z-table with just 2 fields (CNPJ & e-mail address) or even the standard customer master data table (the CNPJ code is part of the Brazilian-localization standard customer master data). We'll not focus on the RFC code here.

On the other hand, the RFC XML message structure is relevant for our development. In order to make it easier to visualize its structure, you could import the RFC definition into PI. It should be something like:

<table>
<thead>
<tr>
<th>Display Message Type</th>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Category</td>
</tr>
<tr>
<td>ZNFE_READ_EMAIL</td>
<td>Element</td>
</tr>
<tr>
<td>CNPJ</td>
<td>Element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display Message Type</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Category</td>
</tr>
<tr>
<td>ZNFE_READ_EMAIL_Reso</td>
<td>Element</td>
</tr>
<tr>
<td>EMAIL_ADDRS</td>
<td>Element</td>
</tr>
</tbody>
</table>
SLD Configuration

In the SLD, it is necessary to create a Software Component Version (SWCV) that will include our custom developments. However, in order to make our job a lot easier and avoid redefining standard objects which we'll not change, it is possible to add the standard SLL-NFE SWCV as a dependency of our customer SWCV. This way, we'll be able to refer to all objects in the standard SWCV in our custom objects.

In order to add a dependency, go to the SLD home page and click on Software Components (under the Software Catalog section). Browse your custom SWCV and, on the Dependencies tab, search and include the standard SLL-NFE SWCV, as pictured below. The Dependency Context is not really important for us, so you can leave it in the default option (Installation Time dependency).

Note: Make sure that your SLD CR Content Version is at least 3.10, or else you won't be able to see the SAP NFE Product and Software Components in the SLD Software Catalog. In order to check that, go to the SLD home page -> Administration -> Details -> Data tab and check the SAP CR Content Version value.
Finally, it is also necessary to create the Business Systems in SLD that will be part of the integration process. At least, you need the Business System of the GRC NFE System (as described in the NFE configuration guide) and the Business System of the system where the RFC will be executed. In our example, the RFC system is the GRC NFE system itself, so there is no need to create any extra Business Systems.

**Implementation**

In order to implement the dynamic e-mail determination in SAP PI, it is necessary to develop/enhance some objects in the Integration Repository and then perform some configuration steps in the Integration Directory, as outlined below.

**Integration Repository**

In our example, we’ll use the RFC Lookup functionality from the SAP PI Mapping Lookup API in order to execute the RFC and retrieve the e-mail address. In order to pass this value to the Receiver Mail Communication Channel, we’ll use Dynamic Configuration to fill the “to” field in the Adapter Specific Message Attributes of the Mail Adapter.

Both the RFC Lookup and the Dynamic Configuration can be done within a dummy message mapping (“dummy” in the sense that it will not change the XML message anyhow). Finally, this message mapping needs to be included in the interface mapping of the NFE B2B Outbound interface.

**Message Mapping**

In your custom SWCV, create a new namespace to hold your message mapping. Create it and give a meaningful name.

This message mapping will not change anything in the XML message structure, it will only execute the RFC and fill the Mail Adapter “to” field dynamic configuration. So, both the source and target message definitions are the same; it is the procNFe External Message, included in the NTB2B_NFeToB2BReceiver External Definition of the http://sap.com/xi/NFE/common namespace (in the standard SLL-NFE SWCV). By following the steps described in the SLD Configuration section above, it should be available to be included in your custom message mapping.

The mapping layout should be something like:

<table>
<thead>
<tr>
<th>Design</th>
<th>Test</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="External Message: procNFe" /></td>
<td><img src="image2" alt="External Message: procNFe" /></td>
<td><img src="image3" alt="External Message: procNFe" /></td>
</tr>
</tbody>
</table>

Both the RFC Lookup and the Dynamic Configuration can be done within a UDF (User-Defined Function), which enables us to insert custom code in the standard interface processing. This UDF needs to receive the CNPJ code value in order to trigger the RFC; hence it must be executed for the CNPJRec field.

Additionally, the RFC Lookup needs the name of a Business System / Communication Channel that it will use to call the RFC (the Receiver RFC communication channel configuration is discussed in the Integration Directory section). So, when creating your UDF, make sure to add three input arguments:
Using SAP PI Lookup API and Dynamic Configuration in SAP GRC NFE Outbound B2B Interface for Dynamic E-mail Determination

- CNPJ code (from the \textit{CNPJRec} field);
- Business System (previously created in the SLD) where the RFC will be executed (e.g. your ERP system or your GRC NFE system);
- Receiver RFC communication channel name.

In our example, the input arguments were created as shown below:

![New Function](image)

Regarding the \textit{CNPJRec} field mapping layout, it goes like this:

- \textit{cnpj}: map the content of the \textit{CNPJRec} field of the source message;
- \textit{busSystem}: since the Business System name will vary through the landscape (DEV/QAS/PRD), we need to evaluate the proper BS during runtime. In our example, since the RFC is being executed in the GRC NFE system, which also happens to be the sender system, we've used the \textit{Sender} standard mapping field to retrieve its name; in the case a different system is used, e.g. ERP, just add a FixValues field to map from the GRC NFE Business System name to the respective ERP Business System name;
- \textit{commChannel}: Constant standard field with the name of the Receiver RFC communication channel, which can be the same throughout the whole landscape (DEV/QAS/PRD);
- \textit{output}: goes into the \textit{CNPJRec} field of the target message.

In the end, it should look like this:

![Diagram](image)

For the \textit{procNFeStr} field, just map it directly from the source to the target message.
As for the UDF, here follows a sample code:

```java
Channel channel = null;
RfcAccessor accessor = null;
//send the NFe to an admin in case the retrieve fails
String email = "nfe_admin@company.com";
AbstractTrace trace = container.getTrace();
try {
    //Retrive the communication channel
    channel = LookupService.getChannel(busSystem, commChannel);
    //Get a RFC accessor for the channel
    accessor = LookupService.getRfcAccessor(channel);
    //Define the RFC Request Message
    String req = "<ns0:ZNFE_READ_EMAIL xmlns:ns0='urn:sap-com:document:sap:rfc:functions'><CNPJ>" + cnpj + "</CNPJ>"/ns0:ZNFE_READ_EMAIL>";
    //Create the xml inputstream
    InputStream inputStream = new ByteArrayInputStream(req.getBytes("UTF-8");
    //Create xml payload
    XmlPayload payload = LookupService.getXmlPayload(inputStream);
    //Execute lookup
    XmlPayload result = accessor.call(payload);
    //Get the response
    InputStream resp = result.getContent();
    //Parse the response
    DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document doc = builder.parse(resp);
    Node node = (Node) doc.getElementsByTagName("EMAIL_ADDRS").item(0);
    if (node.hasChildNodes() && !node.getFirstChild().getNodeValue().equals("") {  
        email = node.getFirstChild().getNodeValue();
    }  
} catch (Exception e) {  
    trace.addWarning("RFC Lookup Failed: " + e);
} finally {  
    //Close the accessor in order to free resources.
    if (accessor!=null) {
        try {  
            accessor.close();
        } catch (Exception e) {  
            trace.addWarning("Error while closing accessor: " + e);
        }
    }
}
```
Using SAP PI Lookup API and Dynamic Configuration in SAP GRC NFE Outbound B2B Interface for Dynamic E-mail Determination

```java
DynamicConfiguration conf = (DynamicConfiguration) container
    .getTransformationParameters()
    .get(StreamTransformationConstants.DYNAMIC_CONFIGURATION);
DynamicConfigurationKey key = DynamicConfigurationKey.create(
    "http://sap.com/xi/XI/System/Mail", "THeaderTO");
conf.put(key, email);
```

Make sure to add the following entries in the `imports` section:

```java
com.sap.aii.mapping.api.*;com.sap.aii.mapping.lookup.*;javax.xml.parsers.*;org.w3c.dom.*;
```

Save your message mapping.

**Interface Mapping**

Since we’ve added the standard SLL-NFE SWCV as a dependency of our customer SWCV, we can create modified versions of the standard objects without actually modifying the standard SWCV. For more details, check the **Modifying Objects** reference below.

Once you’ve added the dependency, a new **Basis Objects** node should appear in your custom SWCV, right below your namespaces & Imported Objects. Within this new node, you can find all the standard objects in the SLL-NFE SWCV, however here they’re included in your custom SWCV. It’s more or less like if the system had automatically created Z versions of all standard objects within your custom SWCV.

Go to the NTB2B_procNFe_TO_procNFe interface mapping in the **Basis Objects** node. Enter into **Edit** mode and click on **Modify**. In the **Mapping Program** list, include a new Message Mapping and search for the message mapping you’ve create in the previous step. Notice that it must be the first of the list; click on the arrows if necessary to move the mappings across the list.

Save your interface mapping, click on **Modify** again and activate it. It should look something like this:

```
SAP COMMUNITY NETWORK
© 2009 SAP AG  8
```
Also, notice the little "m" right next to your modified object. It means that this object was modified from the standard version. If you delete your modification, it will just retrieve the original version of the object.

**Integration Directory**

In the SAP PI Integration Directory, we need to maintain the necessary communication channels (one for the RFC communication and one for the e-mail B2B interface). Additionally, we need to change the interface mapping added to the Interface Determination of the NFE B2B Outbound scenario the interface mapping we’ve modified in our custom SWCV.

**Receiver RFC Communication Channel**

Under the Business System of the system where the RFC for e-mail retrieving will be executed, create a Receiver RFC communication channel. Make sure it has the same name that was maintained in the Constant field of the message mapping described above.

![PSUCLNT300 Communication Channel](WAS_RFC_RCV)

Maintain all the logon details as for a regular Receiver RFC scenario.

**Receiver Mail Communication Channel**

Under a generic Party / Business Service (e.g. "NFE_B2B_PARTNER"), create the Receiver Mail communication channel.

Maintain the parameters in the General tab as for a regular Receiver Mail scenario (with the XIPAYLOAD Message protocol). You can leave the "To" field empty, since it will be retrieved from the Dynamic Configuration. Alternatively, you can also fill it with a default e-mail address (e.g. an administrator’s e-mail) that will receive the XML message in case the dynamic configuration is empty (which may happen for example, if the RFC execution fails). In our case, however, this “default” e-mail address was already considered in the UDF code.

In order to receive the NF-e as an attachment, set the Keep Attachments check box.
Finally, in the Advanced tab, make sure to select the Use Adapter-Specific Message Attributes and Variable Transport Binding check boxes, in order for the dynamic configuration to be used by the adapter.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Identifiers</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter Type *</td>
<td>Mail</td>
<td><a href="http://sap.com/xi/XJ/System">http://sap.com/xi/XJ/System</a></td>
</tr>
<tr>
<td>Sender</td>
<td>Receiver</td>
<td></td>
</tr>
<tr>
<td>Transport Protocol *</td>
<td>SMTP</td>
<td></td>
</tr>
<tr>
<td>Message Protocol *</td>
<td>XIPAYLOAD</td>
<td></td>
</tr>
<tr>
<td>Adapter Engine *</td>
<td>Integration Server</td>
<td></td>
</tr>
</tbody>
</table>

### Connection Parameters for Mail Server

- **URL**: smtp.mail.sap-ag.de
- **Configure User Authentication**: 
- **Send Delivery Status Notification**: 

### Mail Attributes

- **Use Mail Package**: 
- **From**: henrique.pinto@sap.com
- **To**: you_may_leave_this_empty
- **Cc**: 
- **Bcc**: 
- **Subject**: NFE XML Message
- **Content Encoding**: None
- **Keep Attachments**: 

### Adapter-Specific Message Attributes

- **Use Adapter-Specific Message Attributes**: 
- **Variable Transport Binding**: 
- **Variable Header (XHeaderName1)**: 
- **Variable Header (XHeaderName2)**: 
- **Variable Header (XHeaderName3)**: 
Using SAP PI Lookup API and Dynamic Configuration in SAP GRC NFE Outbound B2B Interface for Dynamic E-mail Determination

Interface Determination

You may run the Configuration Scenario wizard from the standard NTB2B_WebAS_Outbound_B2B_NFe Integration Scenario. Once the wizard is completed, open the Interface Determination which was just created and change the NTB2B_procNFe_TO_procNFe interface mapping in the standard SLL-NFE SWCV for the one you modified in your custom SWCV.

<table>
<thead>
<tr>
<th>Quality of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑ Maintain Order At Runtime</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>05aроботка NTB2B_procNFe_TO_procNFe</td>
</tr>
</tbody>
</table>

**Note:** In the match code, it may appear two interface mappings with the same name within your custom SWCV, in addition to the one in the standard SWCV. That's because, for each SWCV, the system stores two versions of modified objects: the original one and the modified one. But regardless of which one you choose, it will execute the modified one (unless you choose the one in the standard SWCV). If you remove your modification, the system restores the original version and you'll see just one entry for your custom SWCV in the match code.
Testing

In order to test the application, simply go to the Integration Engine Test tab, in the Component Monitoring of the Runtime Workbench. Enter the Sender and Interface details, and insert a valid payload for that interface.

### Runtime Workbench

<table>
<thead>
<tr>
<th>Component Monitoring</th>
<th>Message Monitoring</th>
<th>End-to-End Monitoring</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Message</td>
<td>Open Message</td>
<td>Delete Message</td>
<td>Send Message</td>
</tr>
</tbody>
</table>

Message loaded successfully

Send Message To: * http://powdf2709:50009/sap/xi/engine?type=entry

**Header Information**

<table>
<thead>
<tr>
<th>Sender Party</th>
<th>Interface</th>
<th>Interface Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NTB2B_procNFe_OB</td>
<td></td>
</tr>
</tbody>
</table>

User: 1613314
Password: ********

Quality of Service: Exactly Once

**Payload**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ns0:procNFe xmlns:ns0="http://sap.com/xi/NFe/common">
  <ns0:CNPJRec>12345</ns0:CNPJRec>
  <ns0:procNFeStr>&lt;TEST/&gt;</ns0:procNFeStr>
</ns0:procNFe>
```

Once the message is sent, if the RFC returns the correct e-mail address, you should see the message in the account Inbox.
Further Considerations

Below, there are some alternative implementation strategies and some additional features that may enhance your dynamic scenario.

Additional Dynamic Mail fields

Other than the “To” field, you could evaluate other Mail fields dynamically, e.g. “From”, “Cc”, “Bcc”, “Reply-to” etc. Just change your RFC to return these extra fields and add more dynamic configuration entries in the UDF code.

You could also set the “Subject” field dynamically. For example, you could retrieve some data from the NF-e XML in the procNFest field, e.g. the NF-e Id (Access Key), the customer name, date of invoice emission etc., and include those information in the e-mail subject. For that, you’ll need to parse the NF-e XML and browse for these fields (similar to what was done with the RFC Lookup response message).

This is a sample code for dynamic subject (UDF to be added between the source & target procNFest fields). It retrieves the NF-e Access Key from the Id attribute of the infNFe tag and places it in the subject:

```java
AbstractTrace trace = container.getTrace();
try {
    InputStream in = new ByteArrayInputStream(xml.getBytes("utf-8"));
    // Parse the xml
    DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document doc = builder.parse(in);
    Node node = (Node) doc.getElementsByTagName("infNFe").item(0);
    if (node.hasAttributes() && !node.getAttributes().getNamedItem("Id").equals("")) {
        String subject = node.getAttributes().getNamedItem("Id").getNodeValue();
        DynamicConfiguration conf = (DynamicConfiguration) container
            .getTransformationParameters()
            .get(StreamTransformationConstants.DYNAMIC_CONFIGURATION);
        DynamicConfigurationKey key = DynamicConfigurationKey.create(
            "http://sap.com/xi/XI/System/Mail", "THeaderSUBJECT");
        conf.put(key, subject);
    } else {
        trace.addWarning("Dynamic Subject failed: empty or missing NF-e Id!");
    }
} catch (Exception e) {
    trace.addWarning("Dynamic Subject failed: " + e);
}
return xml;
```

This UDF requires just one input parameter, xml.

Also, make sure to import the following packages:

```java
com.sap.aii.mapping.api.*;javax.xml.parsers.*;org.w3c.dom.*;
```
Dynamic Attachment Name

In the testing section above, in the screenshot of the received e-mail, it is possible to notice that the XML attachment’s name is “Untitled.xml”. By default, it is always like that.

However, through a set of simple developments in PI, it is also possible to set the attachment name dynamically. For that, you’ll need to develop a custom adapter module that reads some parameter (e.g., some previously filled dynamic configuration field) or parses the attachment XML message to get the dynamic information and then sets the attachment name.

For more references on Module Development, check the Adapter and Module Development reference in the Related Content section; for a sample code for this custom module, check the Adapter Module PI 7.0 Set Attachment Name reference.

Mail Package

Instead of using dynamic configuration, it is also possible to set various e-mail parameters in the mapping by using the Mail Package message (including dynamic attachment name, by using the Content-type field.

However, SAP recommends using Adapter Specific Message Attributes (ASMA) instead of the Mail Package, which is the older format. Check the "Should I use MailPackage or ASMA?" question in the SAP Note 856599 (link in the Related Content section).

Alternative Adapters

Dynamic Configuration of Adapter Specific Message Attributes works not just for the Mail Adapter, but for several fields of all standard adapters. Even customer-specific adapters and modules can read/write their own specific attribute (just define a customer Technical Name for the field and namespace for the adapter). Check the Adapter-Specific Message Attributes in the Message Header reference in the Related Content section.

If you need to use different communication protocols to send the NF-e XML message to your customers (e.g. Mail for some customers, SOAP (web service) for others, FTP for others etc.), you can enhance the RFC which we described above in order to include the fields for these other parameters of other adapters.

For example, you could add the SOAP URL, SOAP Action and SOAP Authorization Key fields to the RFC and create a specific mapping to fill the dynamic configuration of these fields. In the Integration Directory, based on the CNPJ (content-based routing in the Interface Determination), it’d be possible to select which mapping & which interface (SOAP or Mail) to trigger on the receiver side. It’d also be possible to define two separate generic Receiver systems (e.g. one generic party for SOAP receivers and another one for Mail receivers) and use content-based routing in the Receiver Determination.
Related Content

SAP NFE 1.0 Weblog
Configuring the Receiver Mail Adapter
Modifying an Object
Mapping Lookup API Javadoc
Adapter-Specific Message Attributes in the Message Header
Accessing Adapter-Specific Attributes
Adapter and Module Development
Adapter Module PI 7.0 Set Attachment Name
Note 856599 - FAQ: XI 3.0 / PI 7.0 / PI 7.1 Mail Adapter
For more information, visit the Governance, Risk, and Compliance homepage.
Using SAP PI Lookup API and Dynamic Configuration in SAP GRC NFE Outbound B2B Interface for Dynamic E-mail Determination

Copyright

© 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors. Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.