Idempotency in Services

Applies to:
SAP Business Suite 7.0 and NetWeaver CE 7.1 EhP1

Summary
Idempotency guarantees delivery of synchronous messages exactly once. This document details the why and how of this feature, as implemented in SAP Business Suite and CE 7.1 EhP1

Author: Suresh Ranganathan
Company: SAP America
Created on: 1 December 2009

Author Bio
Suresh Ranganathan has 12 years of application software development and consulting experience and over 10 years of ABAP experience in SAP application software development. Suresh worked for over three years as Development Manager for SAP standard product development. Currently, he assists customers with implementing business scenarios using SOA principles and tools.
# Table of Contents

Prerequisites .......................................................................................................................... 3
Why should Services be Idempotent? ..................................................................................... 3
How to check if SAP delivered services are idempotent? ....................................................... 4
Can I indicate Idempotency for a Service at Design time? .................................................... 4
If I am building custom services (in ABAP), how to do I make them Idempotent? ................. 5
How does this Framework work? ......................................................................................... 6
What configuration is needed? ............................................................................................... 6
How do I test if the feature works for my service implementation? .................................... 7
What should I do when I consume the service from my composite application?
Should I pass the identifier every time? ................................................................................ 8
Some FAQs on Idempotency................................................................................................. 9

Appendix ................................................................................................................................. 11
1. Create the proxy for your Service Interface (using transaction SPROXY) ...................... 11
2. Create the following attributes in your proxy class ......................................................... 11
3. Create the method ONLY_ONE_PREPARE in your proxy class ...................................... 11
4. Create a second method ONLY_ONE_FINALIZE in your proxy class ............................ 16
5. Make the changes to your ‘EXECUTE_SYNCHRONOUS’ of your proxy class to invoke
   the two new methods created above .............................................................................. 19
6. Syntax Check and Activate your proxy class! ................................................................. 19

Related Content ..................................................................................................................... 20

Copyright ................................................................................................................................ 21
Prerequisites

It is assumed that the readers are familiar with the basics of SOA and the tools used in a SOA based development.

Why should Services be Idempotent?

In business scenarios, reliable message transfer is required. This means that it must be guaranteed that a message sent to the receiver to modify an object state is processed exactly once (EO). For example, if a new purchase order is to be created via a service invocation, it is neither acceptable to have no purchase order created nor to have two or more duplicates created. In case of asynchronous services, the XI 3.0 protocol guarantees reliability. However, in synchronous communication, there is no intrinsic mechanism guaranteeing that a message sent arrives at the recipient, nor that a request is processed exactly once by the recipient. A message may be lost or arrive several times in case of network problems. Even if a request reaches the provider, the response may be lost during network transport, in which case the consumer might assume that its request did not arrive and resend it.

To avoid inconsistencies in the provider or consumer system, the relevant SAP Enterprise Services are implemented as idempotent. An idempotent service receiving the exact same request message multiple times within a limited time frame will still process it once only, and return the original response. The restriction to a limited time frame is made to avoid overloading the database, as the system must store the original response message during the time frame.

More details at http://esoadocu.sap.com/socoview(bD1lbiZjPTAwMSZkPW1pbg==)/render.asp?packageid=484F2D49F106E577E10000000A4218AA&edid=ED8CCB8D3C62430681A66DBC6511A15A

Incidentally, Idempotency is not part of the Web Services Standards
How to check if SAP delivered services are idempotent?

The ES Workplace documents if a service is idempotent or not. See example below.

Most SAP standard services (based on Business Suite), that change state (Create, Update and Delete) are designed to be Idempotent.

Can I indicate Idempotency for a Service at Design time?

Yes. Going forward, it should be possible for services modeled in Enterprise Service Repository (ESR) based on CE7.1.1 (EhP1 of CE 7.1). There is a new option to indicate whether a service is Idempotent or not.

Refer details at
If I am building custom services (in ABAP), how to do I make them Idempotent?

If you are developing your custom services in ABAP, you have to do a few things to make them idempotent. Incidentally you do not need to make Query or Read services (those that don’t change the state of the back-end data) idempotent. You would need only to make state-changing services Idempotent.

Make sure though that your service has the ‘MessageHeader’ element in your service definition. Refer diagram below for how it is done for a SAP standard service.

If you are extending or wrapping your custom service over a SAP Standard Service that is idempotent, you can rest assured that your services are already idempotent.

If on the other hand, you are custom coding the service yourself (like calling a BAPI), you will have to implement the feature in your Proxy Class. Thankfully, you have an underlying framework to ease the implementation. The details of how to use it are provided in the Appendix.
How does this Framework work?

The framework is used to store a message created in response to an incoming service call before the response is actually sent to the consumer. The response is identified by the request message’s ID. For each incoming request, the provider can thus determine if the exact same request has already been processed by checking if a response for the request ID has been stored, and if yes directly return the earlier created response without really reprocessing the request.

What configuration is needed?

The BASIS Administrator will need to set up the transaction WSIDPADMIN to turn on underlying Idempotency framework for services. It is needed to be set up once in each ECC/CRM/ SRM, etc environments (Dev, Quality and Production). It cannot be transported across the landscape.

WSIDPADMIN should be run to configure the frequency of cleaning the ‘temporary’ cluster tables that store message requests and responses. The defaults are 6 and 12 hours. If a very high volume of synchronous request-confirmation service calls is expected in a given system landscape, the values should be reduced to 2hrs / 4hrs.
How do I test if the feature works for my service implementation?

When you test a service, make sure you pass a Universal Identifier value in the UUID sub-element in the MessageHeader element of the service. An example of how to test it using WSNAVIGATOR is shown below.

If you pass the same identifier in two consecutive calls, the second call would not process the request. The response from the first call will persist and the second one will be neglected.
What should I do when I consume the service from my composite application?
Should I pass the identifier every time?

It depends on your tool for composition. Starting CE 7.1.1, you get an excellent feature that allows for Idempotency to be automatically available for service consumers. You will not have to code for it in your individual applications. You can configure the activation of Idempotency, the number or retries and the retry interval. More details at http://help.sap.com/saphelp_nwce711/helpdata/en/47/f8af96f84aa7e1000000000a421937/frameset.htm

Here’s an example of the configuration of an idempotent service consumer created in CE 7.1.1.

First in Design Time configuration,
Now in Runtime configuration,

Remember that these configurations are not visible for non-idempotent services.

If instead of CE, you use .NET or some other environment; you will have to code and pass the identifier in your application.

Some FAQs on Idempotency

1. Is this SAP best practice limited to “duplicate prevention” or does it also help in the area of “guaranteed delivery” or help with other areas?

Idempotency is intended for avoiding duplicate entries. Guaranteed delivery is generally assured for Asynch messages but not for Synch ones. A synch message may be lost or arrive several times in case of network problems. Even if a request reaches the provider, the response may be lost during network transport, in which case the consumer might assume that its request did not arrive and resend it. Idempotency avoids it.

2. Do SAP delivered services automatically come equipped with Idempotency feature and it's only "custom delivered" services that require it?

Most state-changing SAP services are delivered idempotent. Custom Services will have to be made idempotent, by using the SAP delivered framework.

3. Why does this feature not get turned on up front if standard services use it out of the box?

The feature itself is ‘dormant’ in the services. There is an administrative setup that needs to done for the framework to be activated and then you can use the feature.
4. What authorization does SAP recommend to set up batch jobs?

   This is normally a role of a BASIS Administrator who schedules other batch jobs. He should have authority for ‘S_SRT_ADM’ (Administration/Configuration of SOAP Runtime), Activity ‘70’ and WS_NAME ‘*’

5. Is there much of any performance overhead by scheduling and checking GUID's

   Yes. But it should be marginal. There is also a small price for updating the GUID when a successful update happens.

6. Are tables already created or does the transaction create them?

   There are 'auto-generated' tables that get created in each system for storing the messages. These tables (an example is /1SAP1/IDPBD1003) are generated on the fly by the scheduling program and are stored in $TMP and hence do not to be transported across the landscape

7. Is this feature client dependent or independent?

   The tables are client dependent
Appendix

Here are some details on how make your ABAP custom services Idempotent. You can follow using the class L_SLS_SALESORDERCRTRC1 in an ECC EhP2 or higher (with NW 7.0 SP09 or higher) system as an example.

1. Create the proxy for your Service Interface (using transaction SPROXY)

2. Create the following attributes in your proxy class
   a. Name: MV_IS_EO_REQUESTED
      Level: Instance Attribute
      Visibility: Private
      Associate Type: ABAP_BOOL
   b. Name: MV_MSG_ID
      Level: Instance Attribute
      Visibility: Private
      Associate Type: STRING
   c. Name: MV_MSG_UUID
      Level: Instance Attribute
      Visibility: Private
      Associate Type: GUID_32
   d. Name: MR_IDP_HELPER
      Level: Instance Attribute
      Visibility: Private
      Associate Type: IF_WS_IDP_HELPER

3. Create the method ONLY_ONE_PREPARE in your proxy class.
   Name: ONLY_ONE_PREPARE
   Level: Instance
   Visibility: Private
   Description: Only-One: Preparatory steps for request processing
   Parameters:
      a. IS_INPUT, Type: Importing
         Associated Type: <Use the same type as the input structure of your proxy class>
      b. ES_OUTPUT, Type: Exporting
         Associated Type: <Use the same type as the output structure of your proxy class>
      c. EV_IS_PROCESSED, Type: Exporting
         Associated Type: ABAP_BOOL
   Exceptions:
      a. Exception of type CX_SAPPLCO_STANDARD_MSG_FAULT
Code: Introduce the following code. Replace <InputStructure> with appropriate structure of your input

```
method ONLY_ONE_PREPARE.

DATA: lcx_idp_enqueue_failure TYPE REF TO cx_soap_idp_enqueue_failure,
lr_response_data TYPE REF TO data,
lv_msgtext TYPE symsgv, "#EC NEEDED
ls_exc_message TYPE sapplico_exchange_fault_data,
ls_exc_message_line TYPE sapplico_exchange_log_data.

* 1 - get message ID and UUID from message header
me->mv_msg_id = is_input-><InputStructure>-message_header-ID-CONTENT.

TRY.

TRY.

* 2 - check if idempotency requested ...
  IF is_input-><InputStructure>-message_header-uuid IS NOT INITIAL.
    CALL METHOD cl_gdt_conversion=>guid_inbound
      EXPORTING
        im_value = is_input-><InputStructure>-message_header-uuid
      IMPORTING
        ex_guid_c = me->mv_msg_uuid.
  ENDIF.

* 3- use NW helper class to implement idempotency
  me->mv_is_eo_requested = abap_true.
  me->mr_idp_helper = cl_ws_idp_factory=>create_idp_helper( me-
  >mv_msg_uuid ).
  * First attempt to check whether request was processed already
    ev_is_processed = me->mr_idp_helper->is_message_processed( ).

ENDIF.

CATCH cx_soap_idp_enqueue_failure INTO lcx_idp_enqueue_failure.

  request message ID can’t be locked ? can have two causes:
  CASE lcx_idp_enqueue_failure->exception_nr.
    WHEN cx_soap_idp_enqueue_failure=>co_exception_nr_foreign_lock.
      * foreign lock on the document -
        > wait a few moments and try again
        WAIT UP TO '2' SECONDS.
      * Second attempt to check whether request was processed already
        ev_is_processed = me->mr_idp_helper->is_message_processed( ).
      WHEN OTHERS.
        * system error during lock attempt -
          > throw exception, but allow retry
          ls_exc_message-fault_text = lcx_idp_enqueue_failure->get_text( ).
          MESSAGE e024(appl_common) WITH ls_exc_message-fault_text
            lcx_idp_enqueue_failure->exception_nr
            INTO lv_msgtext. "Exactly-
        once request cannot be locked: $1 (error code $2)
          CALL METHOD cl_proxy_fault=>get_fault_detail
```

*
EXPORTING
  msgty = sy-msgty
  msgid = sy-msgid
  msgno = sy-msgno
  msgv1 = sy-msgv1
  msgv2 = sy-msgv2
  msgv3 = sy-msgv3
  msgv4 = sy-msgv4
IMPORTING
  severity = ls_exc_message_line-severity
  text = ls_exc_message_line-text
  url = ls_exc_message_line-url
  id = ls_exc_message_line-id.
APPEND ls_exc_message_line TO ls_exc_message-fault_detail.

RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
EXPORTING
  automatic_retry = abap_false
  no_retry = abap_false
  standard = ls_exc_message.
ENDCASE.
ENDTRY.

IF ev_is_processed = abap_true.
* 4 - request was already processed -> return stored response
  GET REFERENCE OF es_output INTO lr_response_data.
  CALL METHOD me->mr_idp_helper->retrieve( CHANGING cr_msg_data = lr_response_data ).
ENDIF.

CATCH cx_soap_idp_enqueue_failure INTO lcx_idp_enqueue_failure.
*  error locking request after second attempt -
  > throw exception, but allow retry
  ls_exc_message-fault_text = lcx_idp_enqueue_failure->get_text( ).
  MESSAGE e024(appl_common) WITH ls_exc_message-fault_text
    lcx_idp_enqueue_failure->exception_nr
    INTO lv_msgtext. "Exactly-once request cannot be locked: &1 (error code &2)
  CALL METHOD cl_proxy_fault=>get_fault_detail
  EXPORTING
    msgty = sy-msgty
    msgid = sy-msgid
    msgno = sy-msgno
    msgv1 = sy-msgv1
    msgv2 = sy-msgv2
    msgv3 = sy-msgv3
    msgv4 = sy-msgv4
IMPORTING
  severity = ls_exc_message_line-severity
  text = ls_exc_message_line-text
  url = ls_exc_message_line-url
  id = ls_exc_message_line-id.
APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
EXPORTING
  automatic_retry = abap_false
  no_retry = abap_false
  standard = ls_exc_message.

CATCH cx_soap_idp_time_out.
*  Request was processed before, but stored response already cleaned up -
>  unrecoverable exception
MESSAGE e025(appl_common) WITH me->mv_msg_uuid INTO lv_msgtext. "Exactly-
once request timed out (message UUID &1)"
CALL METHOD cl_proxy_fault=>get_fault_detail
EXPORTING
  msgty = sy-msgty
  msgid = sy-msgid
  msgno = sy-msgno
  msgv1 = sy-msgv1
  msgv2 = sy-msgv2
  msgv3 = sy-msgv3
  msgv4 = sy-msgv4
IMPORTING
  severity = ls_exc_message_line-severity
  text = ls_exc_message_line-text
  url = ls_exc_message_line-url
  id = ls_exc_message_line-id.
APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
ls_exc_message-fault_text = ls_exc_message_line-text.

RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
EXPORTING
  automatic_retry = abap_false
  no_retry = abap_true
  standard = ls_exc_message.

CATCH cx_soap_idp_not_found.
MESSAGE e026(appl_common) WITH me->mv_msg_uuid INTO lv_msgtext. "Internal error in exactly-
once processing (message UUID &1)"
CALL METHOD cl_proxy_fault=>get_fault_detail
EXPORTING
  msgty = sy-msgty
  msgid = sy-msgid
  msgno = sy-msgno
  msgv1 = sy-msgv1
  msgv2 = sy-msgv2
  msgv3 = sy-msgv3
  msgv4 = sy-msgv4
IMPORTING
  severity = ls_exc_message_line-severity
  text = ls_exc_message_line-text
  url = ls_exc_message_line-url
  id = ls_exc_message_line-id.
APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
ls_exc_message-fault_text = ls_exc_message_line-text.
RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
   EXPORTING
   automatic_retry = abap_false
   no_retry = abap_false
   standard = ls_exc_message.

CATCH cx_soap_idp_failure.
   * unqualified error when retrieving stored response (e.g. authorization issue)
   MESSAGE e026(appl_common) WITH me->mv_msg_uuid INTO lv_msgtext.
   "Internal error in exactly-once processing (message UUID &1)
   CALL METHOD cl_proxy_fault=>get_fault_detail
   EXPORTING
   msgty = sy-msgty
   msgid = sy-msgid
   msgno = sy-msgno
   msgv1 = sy-msgv1
   msgv2 = sy-msgv2
   msgv3 = sy-msgv3
   msgv4 = sy-msgv4
   IMPORTING
   severity = ls_exc_message_line-severity
   text = ls_exc_message_line-text
   url = ls_exc_message_line-url
   id = ls_exc_message_line-id.
   APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
   EXPORTING
   automatic_retry = abap_false
   no_retry = abap_true
   standard = ls_exc_message.

CATCH cx_gdt_conversion.
   * message UUID passed, but it's not really a UUID -> raise exception
   MESSAGE e021(appl_common) WITH is_input-<InputStructure>-message_header-uuid
      'MessageHeader/UUID'
   "#EC NOTEXT
   INTO lv_msgtext.
   "UUID &1 for element &2 does not comply to required pattern
   CALL METHOD cl_proxy_fault=>get_fault_detail
   EXPORTING
   msgty = sy-msgty
   msgid = sy-msgid
   msgno = sy-msgno
   msgv1 = sy-msgv1
   msgv2 = sy-msgv2
   msgv3 = sy-msgv3
   msgv4 = sy-msgv4
   IMPORTING
   severity = ls_exc_message_line-severity
   text = ls_exc_message_line-text
   url = ls_exc_message_line-url
   id = ls_exc_message_line-id.
   APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
   ls_exc_message-fault_text = ls_exc_message_line-text.
RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
EXPORTING
  automatic_retry = abap_false
  no_retry = abap_true
  standard = ls_exc_message.

ENDTRY.
endmethod.

4. Create a second method ONLY_ONE_FINALIZE in your proxy class.
Name: ONLY_ONE_FINALIZE
Level: Instance
Visibility: Private
Description: Only-One : Finalizing steps after request processing
Parameters:
  a. IV_FLAG_ERROR, Type: Importing
     Associated Type: ABAP_BOOL
  b. CS_OUTPUT, Type: Changing
     Associated Type: <Use the same type as the output structure of your proxy class>
Exceptions:
  b. Exception of type CX_SAPPLCO_STANDARD_MSG_FAULT

Code: Introduce the following code. Replace <OutputStructure> with appropriate structure of your output structure

method ONLY_ONE_FINALIZE.

* exception CX_SAPPLCO_STANDARD_MSG_FAULT is not caught
* if the exception cx_gdt_conversion is thrown the
* exception CX_SAPPLCO_STANDARD_MSG_FAULT will be
* propagated

data: lv_error_uuid type guid_32,
  lv_error_field type SYMSGV.

DATA: lr_response_data TYPE REF TO data,
  lv_mgtext TYPE string,
  ls_exc_message TYPE sapplco_exchange_fault_data,
  ls_exc_message_line TYPE sapplco_exchange_log_data,
  lv_uuid type guid_32.

TRY.
  * prepare to fill identifiers for output message itself in output message header
  * the elements will only be filled if corresponding message identifiers were passed in input message
    CALL FUNCTION 'GUID_CREATE'
    IMPORTING
    ev_guid_32 = lv_uuid.
fill reference to input message in output message's header.

IF me->mv_msg_id IS NOT INITIAL.
cs_output-><OutputStructure>->message_header-reference_id-content = me->mv_msg_id.
ENDIF.

IF me->mv_msg_uuid IS NOT INITIAL.

move : lv_uuid to lv_error_uuid, "MessageHeaderUUID" to lv_error_field.

CALL METHOD c1_gdt_conversion=>guid_outbound
EXPORTING
  im_guid_c = lv_uuid
IMPORTING
  ex_value = cs_output-><OutputStructure>->message_header-uuid.

move : me->mv_msg_uuid to lv_error_uuid, "MessageHeaderReferenceUUID" to lv_error_field.

CALL METHOD c1_gdt_conversion=>guid_outbound
EXPORTING
  im_guid_c = me->mv_msg_uuid
IMPORTING
ENDIF.

store output message if idempotency is requested

IF iv_flag_error = abap_false AND me->mv_is_eo_requested = abap_true.

store response in case request is received a second time
GET REFERENCE OF cs_output INTO lr_response_data.
CALL METHOD me->mr_idp_helper->save( ir_msg_data = lr_response_data ).
ENDIF.

CATCH cx_soap_idp_failure.

unqualified error when storing response
MESSAGE e027(apl_common) WITH me->mv_msg_uuid INTO lv_msgtext. "Internal error storing response for request message UUID &1"
CALL METHOD c1_proxy_fault=>get_fault_detail
EXPORTING
  msgty = sy-msgty
  msgid = sy-msgid
  msgno = sy-msgno
  msgv1 = sy-msgv1
  msgv2 = sy-msgv2
  msgv3 = sy-msgv3
  msgv4 = sy-msgv4
IMPORTING
  severity = ls_exc_message_line-severity
text = ls_exc_message_line-text
url = ls_exc_message_line-url
id = ls_exc_message_line-id.

APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
  EXPORTING
    automatic_retry = abap_false
    no_retry = abap_true
    standard = ls_exc_message.

CATCH cx_gdt_conversion.
  MESSAGE e021(appl_common)
    WITH lv_error_uuid
    lv_error_field
    INTO lv_msgtext. "UUID &1 for element &2 does not comply to required pattern

CALL METHOD cl_proxy_fault=>get_fault_detail
  EXPORTING
    msgty = sy-msgty
    msgid = sy-msgid
    msgno = sy-msgno
    msgv1 = sy-msgv1
    msgv2 = sy-msgv2
    msgv3 = sy-msgv3
    msgv4 = sy-msgv4
  IMPORTING
    severity = ls_exc_message_line-severity
    text = ls_exc_message_line-text
    url = ls_exc_message_line-url
    id = ls_exc_message_line-id.

APPEND ls_exc_message_line TO ls_exc_message-fault_detail.
ls_exc_message-fault_text = ls_exc_message_line-text.

RAISE EXCEPTION TYPE cx_sapplco_standard_msg_fault
  EXPORTING
    automatic_retry = abap_false
    no_retry = abap_true
    standard = ls_exc_message.

ENDTRY.

endmethod.
5. Make the changes to your 'EXECUTE_SYNCHRONOUS' of your proxy class to invoke the two new methods created above.

   Make the first executable line in this method to invoke the method only_one_prepare as below:

   ```abap
   DATA : lv_flag_is_processed TYPE abap_BOOL.
   "definition of local working area
   * Check whether this particular input-message has already been
   * processed using the "only once"-processing
   * if the exception CX_SAPPLCO_STANDARD_MSG_FAULT is thrown
   * in this method the processing of this proxy will be
   * terminated immediately
   CALL METHOD me->only_one_prepare
      EXPORTING
      is_input = input
      IMPORTING
      es_output = output
      ev_is_processed = lv_flag_is_processed.
   * If a recent messages with the identifier has been processed already,
   * no further processing will be done -> leave this method
   if lv_flag_is_processed = abap_TRUE.
   return.  " leave this method immediately
   endif.
   Make the last executable line in this method to invoke the method only_one_finalize
   as below:
   * finalize "only-once"-processing (checking handle)
   * if the exception CX_SAPPLCO_STANDARD_MSG_FAULT is thrown
   * in this method the processing of this proxy will be
   * terminated immediately
   CALL METHOD me->only_one_finalize
      EXPORTING
      iv_flag_error = ''
      CHANGING
      cs_output = output.
   ```

6. Syntax Check and Activate your proxy class!
Related Content

http://esoadocu.sap.com/socoview/bD1ibZjPTAwMSZhPW1pbg==/render.asp?packageid=484F2D49F106E577E10000000A4218AA&id=ED8CCB8D3C62430681A66DBC6511A15A
http://help.sap.com/saphelp_nwce711/helpdata/en/47/f8af96f8b84aa7e10000000a421937/frameset.htm

For more information, visit the ABAP homepage.
Copyright

© 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, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

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.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, 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 other countries.

BusinessObjects and the BusinessObjects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other BusinessObjects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of BusinessObjects S.A. in the United States and in other countries. BusinessObjects is an SAP company.

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.