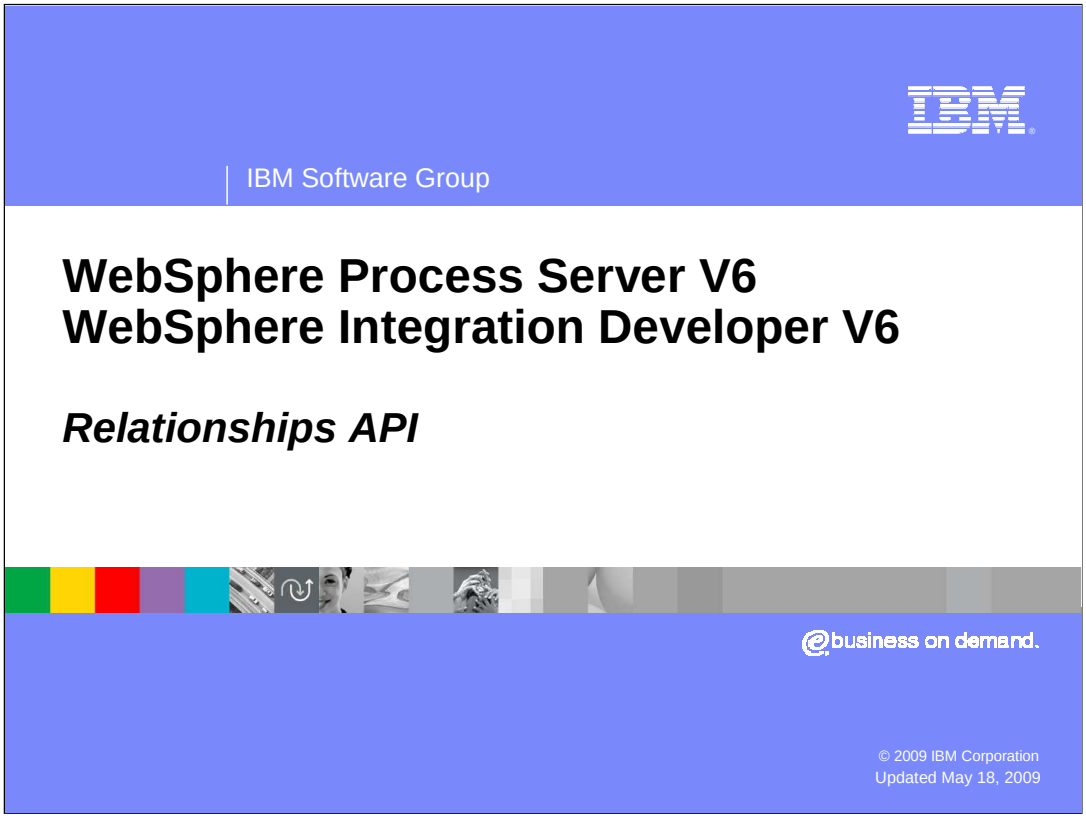


This presentation will provide an overview of relationships and the relationship service of WebSphere® Process Server and WebSphere Integration Developer version 6.

Agenda

- Relationships API overview
- Summary

This section will provide an overview of the APIs for relationships.

Correlate() API replaces maintainIdentityRelationship() API

- The maintainIdentity() method is deprecated in V6.1
- correlate() APIs replace maintainIdentityRelationship() API
  - correlate()
  - Correlates an inputBO of the given inputRoleName with the outputBO of the given outputRoleName and maintains the relationship between them according to the meta data contained in the inputBO (currently either verb or change summary information or both).
  - correlateToList()
    Used to correlate the children of a parent/child input role with a list of business objects of a simple output role.
  - correlateFromList()
    Used to correlate a list of business objects of a simple input role with the children of parent/child output role.

In version 6.1, the maintainIdentityRelationship method is deprecated and replaced with the more robust correlate() methods. The relationship service uses these correlate methods to manage and maintain the relationships. The calling context and verb scenarios you reviewed in the overview are all handled by the relationship service through the use of the application programming interface.

Exceptions thrown by correlate() method

- RelationshipServiceException – Indicates an exception internal to the Relationship Service, such as a missing database connection or other runtime failures.
- RelationshipUserException – Indicates an error in the usage of this API, such as non-valid relationship / role names, business objects that are null or invalid type.
- DataNotFoundException – Indicates that a participant was not found when attempting to retrieve it
- DuplicateDataException – Indicates a duplicate

The correlate() method can create these exceptions. RelationshipServiceException indicates a problem with the relationship service itself. RelationshipUserException indicates a user error in the use of the API. DataNotFoundException indicates a participant was not found when trying to retrieve it, and DuplicateDataException, indicating duplicated data.

Correlate method

correlate

```
void correlate(java.lang.String relationshipName,
               java.lang.String inputRoleName,
               java.lang.String outputRoleName,
               commonj.sdo.DataObject inputBO,
               commonj.sdo.DataObject outputBO,
               commonj.sdo.DataObject originalInputBO,
               commonj.sdo.DataObject originalOutputBO,
               java.lang.String callingContext)
               throws RelationshipServiceException,
                      RelationshipUserException,
                      DataNotFoundException,
                      DuplicateDataException
```

- Parameters
  - relationshipName - The fully qualified name of the relationship
  - inputRoleName - The fully qualified name of the input role
  - outputRoleName - The fully qualified name of the output role
  - inputBO - The input business object of the input role type
  - outputBO - The output business object of the output role type
  - originalInputBO - The original input business object. For calling context SERVICE_CALL_RESPONSE or SERVICE_CALL_FAILURE this BO can not be null and must be of the output role type.
  - originalOutputBO - The original output business object. For calling context SERVICE_CALL_RESPONSE or SERVICE_CALL_FAILURE this BO can not be null and must be of the input role type.
  - callingContext - The calling context for the invocation. One of SERVICE_CALL_REQUEST, EVENT_DELIVERY, SERVICE_CALL_RESPONSE, and SERVICE_CALL_FAILURE

Here you see the signature of the correlate method.

CorrelateToList method

correlateToList

```
void correlateToList(java.lang.String relationshipName,
                     java.lang.String inputRoleName,
                     java.lang.String outputRoleName,
                     commonj.sdo.DataObject inputBO,
                     java.util.List outputBOs,
                     java.util.List originalInputBOs,
                     commonj.sdo.DataObject originalOutputBO,
                     java.lang.String callingContext)
                     throws RelationshipServiceException,
                            RelationshipUserException,
                            DataNotFoundException,
                            DuplicateDataException
```

- Parameters
  - relationshipName - The fully qualified name of the relationship
  - inputRoleName - The fully qualified name of the input role
  - outputRoleName - The fully qualified name of the output role
  - inputBO - The input business object of the input role type
  - outputBOs – A list of output business objects of the output role type
  - originalInputBOs – A list of original input business objects. For calling context SERVICE_CALL_RESPONSE or SERVICE_CALL_FAILURE this list can not be null and must be of the output role type.
  - originalOutputBO - The original output business object. For calling context SERVICE_CALL_RESPONSE or SERVICE_CALL_FAILURE this BO can not be null and must be of the input role type.
  - callingContext - The calling context for the invocation. One of SERVICE_CALL_REQUEST, EVENT_DELIVERY, SERVICE_CALL_RESPONSE, and SERVICE_CALL_FAILURE

Here you see the signature of the correlateToList method. This method is used to correlate the children of a parent/child input role with a list of business objects of a simple output role.

CorrelateFromList method

correlateFromList

```
void correlateFromList(java.lang.String relationshipName,
                       java.lang.String inputRoleName,
                       java.lang.String outputRoleName,
                       java.util.List inputBOs,
                       commonj.sdo.DataObject outputBO,
                       commonj.sdo.DataObject originalInputBO,
                       java.util.List originalOutputBOs,
                       java.lang.String callingContext)
                       throws RelationshipServiceException,
                              RelationshipUserException,
                              DataNotFoundException,
                              DuplicateDataException
```

- Parameters
  - relationshipName - The fully qualified name of the relationship
  - inputRoleName - The fully qualified name of the input role
  - outputRoleName - The fully qualified name of the output role
  - inputBO - The input business object of the input role type
  - outputBO - The output business object of the output role type
  - originalInputBO - The original input business object. For calling context SERVICE_CALL_RESPONSE or SERVICE_CALL_FAILURE this BO can not be null and must be of the output role type.
  - originalOutputBO - The original output business object. For calling context SERVICE_CALL_RESPONSE or SERVICE_CALL_FAILURE this BO can not be null and must be of the input role type.
  - callingContext - The calling context for the invocation. One of SERVICE_CALL_REQUEST, EVENT_DELIVERY, SERVICE_CALL_RESPONSE, and SERVICE_CALL_FAILURE.

Here you see the signature for the correlateFromList method. This method is used to correlate a list of business objects of a simple input role with the children of parent/child output role.

Relationship service enhancements

	New script-based access to the Relationship Service repository tables
New 6.2	▪ New APIs for the pre-population of instance data in the repository tables ▪ Sample SQL scripts to help users write their own SQL scripts to pre-populate the data using views ▪ Sample Java™ code to help users write their own Java code to pre-populate the data using APIs
Benefits	The administrator can pre-populate the relationship instance data in the database using these new APIs.

Relationship database tables contain relationship runtime data that is created during relationship installation. In previous releases, the only way to enter new data into the Relationship tables was using the Relationship Manager in the administrative console, or through the database GUI, one entry at a time. New in V6.2 are APIs to allow the relationship instance data to be pre-populated in the database multiple entries at a time. Also some sample scripts and sample Java code are provided to help users use these new views and APIs.

New relationship service views

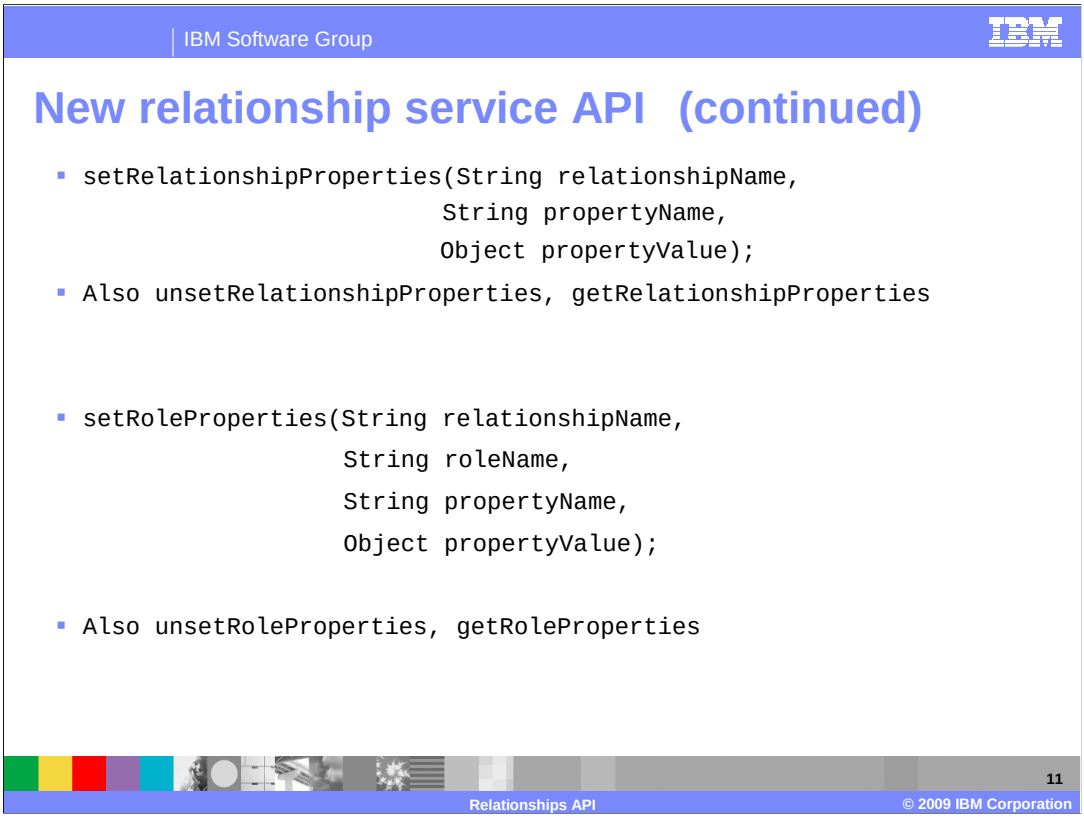
- View is a virtual table
  - Shows all or part of the database table columns
- Allows users to manipulate table data without needing to know the physical table schema
- One view is generated for each relationship instance table
- Derby does not support the use of views for table manipulation, so views can only be used for lookup in Derby
  - Users can still use existing Relationship Service APIs to manipulate data outside of the view

New in version 6.2 is the implementation of views, which are actually virtual tables. You can treat this view as if it is the underlying table, and therefore can manipulate the table data (that is, add, change, or delete the table data). You can write your own SQL scripts to do the table manipulation, and you do not need to know the physical table schema to do so. A view is generated for each relationship instance table when the table is created. There are not views created for property tables. The one caveat here is that Derby database does not support the use of updateable views. If you are using Derby as your database for relationships, you will only be able to use the views as a data lookup. You are not able to manipulate the data for Derby tables through the SQL scripts. You can still use the Relationship Service APIs to manipulate your data if you use Derby as your database.

New relationship service APIs

- Pre-V6.2 Relationship Service API allowed users to work with relationship and role instance data, but only one instance at a time
- New in V6.2 is an API that allows users to set/unset/retrieve a property for all relationship or role instances that belong to the same relationship or role
- You now only need to make one API call to manipulate a large set of property values!

The views are not created for property tables. Instead, you have the ability to work with the relationship and role properties using APIs. Before V6.2, you can use the APIs to set, unset, and retrieve properties but only one instance at a time. New in V6.2 is the ability to set, unset, and retrieve a property for all relationship or role instances that belong to the same relationship or role. This will allow users to populate a large set of data with one API call.



Here the new APIs are briefly described. More details are available in the information center. These API names will seem familiar if you have worked with them before. Note that in the previous release these were singular names (as in setRelationshipProperty) while the new APIs are plural names (setRelationshipProperties). It is this multiple instance capability that has been added in V6.2.

The setRelationshipProperties API sets a relationship property for all instances of a specified relationship. The unsetRelationshipProperties API unsets a relationship property for all instances, and the getRelationshipProperties will retrieve a relationship propertyvalue of all instances of a specified relationship. The setRoleProperties, unsetRoleProperties, and getRoleProperties work similarly for all instances of a specified relationship and role.

New relationship service API (continued)

- addParticipantsWithID(String relationshipName,
  int instanceid,
  HashMap roleName_Value);
- Also addParticipantsWithoutID, addParticipants
- getNewInstanceID(String relationshipName);

These new methods allow multiple instances to be manipulated and will work for both dynamic and static relationships. You can add participants with ID (when you know the instance ID), or without ID (which will return a new instance ID). The addParticipants method inserts a participant for each specified role into each specified relationship instance that is provided. The getNewInstanceID is used to insert a participant that is the first participant in the specified relationship; it returns a new instanceID.

Section

Summary

This section will provide a summary of this presentation.

Summary

- Relationships API overview

This module briefly covered the APIs that are available for the relationship service. You can refer to the information center for detailed information on the relationship APIs.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WPSWIDv6_RelationshipAPI.ppt

This module is also available in PDF format at: ../WPSWIDv6_RelationshipAPI.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Java, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.