



Guía para la migración



Guía para la migración

Nota

Antes de utilizar esta información y el producto que soporta, lea la información en el apartado Avisos al final de esta publicación.

Contenido

Migrar aplicaciones utilizando

WebSphere Integration Developer . . . 1

Versión PDF 1

Guía de aprendizaje: migrar a WebSphere Integration Developer 1

Migrar a WebSphere Integration Developer 1

 Migrar a WebSphere Process Server desde

 WebSphere InterChange Server 1

Migrar a WebSphere Integration Developer desde

WebSphere MQ Workflow 30

Migrar artefactos origen a WebSphere Integration Developer desde WebSphere Studio Application

Developer Integration Edition 36

Avisos 101

Migrar aplicaciones utilizando WebSphere Integration Developer

WebSphere Integration Developer Versión 6.0 proporciona las herramientas necesarias para migrar el entorno actual.

En los temas siguientes se ofrece una descripción del concepto, la referencia y el procedimiento de migración de WebSphere Integration Developer:

Versión PDF

Esta información de migración también está disponible en PDF.

Este documento también está disponible como Archivo PDF

Necesitará Adobe Acrobat para verlo. Puede obtener una versión gratuita de este software en www.adobe.com.

Guía de aprendizaje: migrar a WebSphere Integration Developer

Esta guía de aprendizaje facilita información sobre cómo migrar el proyecto de servicio existente y migrarlo a un proyecto de módulo mediante WebSphere Integration Developer.

La guía de aprendizaje contiene una lección, cuyo formato es el de una película:

- Lección 1: Migrar un proyecto de servicio muestra cómo migrar un proyecto de servicio existente a un proyecto de módulo.

Pulse el enlace siguiente para lanzar la guía de aprendizaje:

Nota: El siguiente enlace no funciona en este sitio Web. Debe instalarse WebSphere Integration Developer para que el enlace funcione correctamente.

Guía de aprendizaje: Migrar a WebSphere Integration Developer

Migrar a WebSphere Integration Developer

WebSphere Integration Developer Versión 6.0 proporciona las herramientas necesarias para migrar el entorno actual.

En los temas siguientes se ofrece una descripción del concepto, la referencia y el procedimiento de migración de WebSphere Integration Developer:

Migrar a WebSphere Process Server desde WebSphere InterChange Server

La migración desde WebSphere InterChange Server a WebSphere Process Server está soportada mediante las siguientes funciones:

Nota: Consulte las Notas del release para obtener información acerca de las limitaciones relacionadas con la migración en este release de WebSphere Process Server.

- Migración automática de artefactos origen mediante herramientas de migración que pueden invocarse desde los siguientes elementos:

- Menú **Archivo** → **Importar** de WebSphere Integration Developer
- Pantalla de bienvenida de WebSphere Integration Developer
- Asistente de migración de primeros pasos de WebSphere Process Server
- El programa de utilidad de línea de mandatos **reposMigrate**
- Soporte nativo en el entorno de ejecución de muchas API de WebSphere InterChange Server
- Soporte para la tecnología actual de WebSphere Business Integration Adapter, mediante el cual los adaptadores existentes serán compatibles con WebSphere Process Server

Aunque la migración de artefactos origen está soportada, es aconsejable realizar análisis y pruebas exhaustivas para determinar si las aplicaciones resultantes funcionarán según lo esperado en WebSphere Process Server, o si será necesario un rediseño de las mismas posterior a la migración. Esta recomendación se basa en las siguientes limitaciones de la paridad funcional entre WebSphere InterChange Server y esta versión de WebSphere Process Server. En esta versión de WebSphere Process Server no hay soporte equivalente a estas funciones de WebSphere InterChange Server:

- Interfaz de acceso
- Respuesta de petición síncrona iniciada por adaptador
- Llamada síncrona de envío de servicio con tiempo de espera
- Llamada de servicio Async_in
- Aislamiento
- Secuenciación de eventos
- Despliegue en caliente/actualización dinámica
- Seguridad - Auditoría
- Seguridad - RBAC de grano fino
- Soporte de grupos
- Planificador - Operación en pausa
- Los descriptores de seguridad no se migran
- Las transformaciones de fragmento de código XML personalizado no están soportadas durante la migración de plantillas de Correlación y colaboración

Vías de migración soportadas para WebSphere InterChange Server

Las herramientas de migración de WebSphere Process Server permiten la migración desde las versiones 4.2.2, 4.2.3 y 4.3 de WebSphere InterChange Server.

Las versiones de WebSphere InterChange Server anteriores a la 4.2.2 deberán migrarse a la versión 4.2.2, 4.2.3 o 4.3 antes de efectuar la migración a WebSphere Process Server.

Tareas de preparación para la migración de WebSphere InterChange Server

Antes de migrar a WebSphere Process Server desde WebSphere InterChange Server, primero debe asegurarse de que ha preparado correctamente el entorno. WebSphere Process Server ofrece las herramientas necesarias para migrar desde WebSphere InterChange Server.

Estas herramientas de migración se pueden invocar desde los elementos siguientes:

- Menú **Archivo** → **Importar** de WebSphere Integration Developer
- Pantalla de bienvenida de WebSphere Integration Developer
- Asistente de migración de primeros pasos de WebSphere Process Server

La entrada de las herramientas de migración es un archivo jar de depósito exportado desde WebSphere

InterChange Server. Por tanto, antes de acceder a las herramientas de migración a través de cualquiera de estas opciones, primero debe realizar las tareas siguientes:

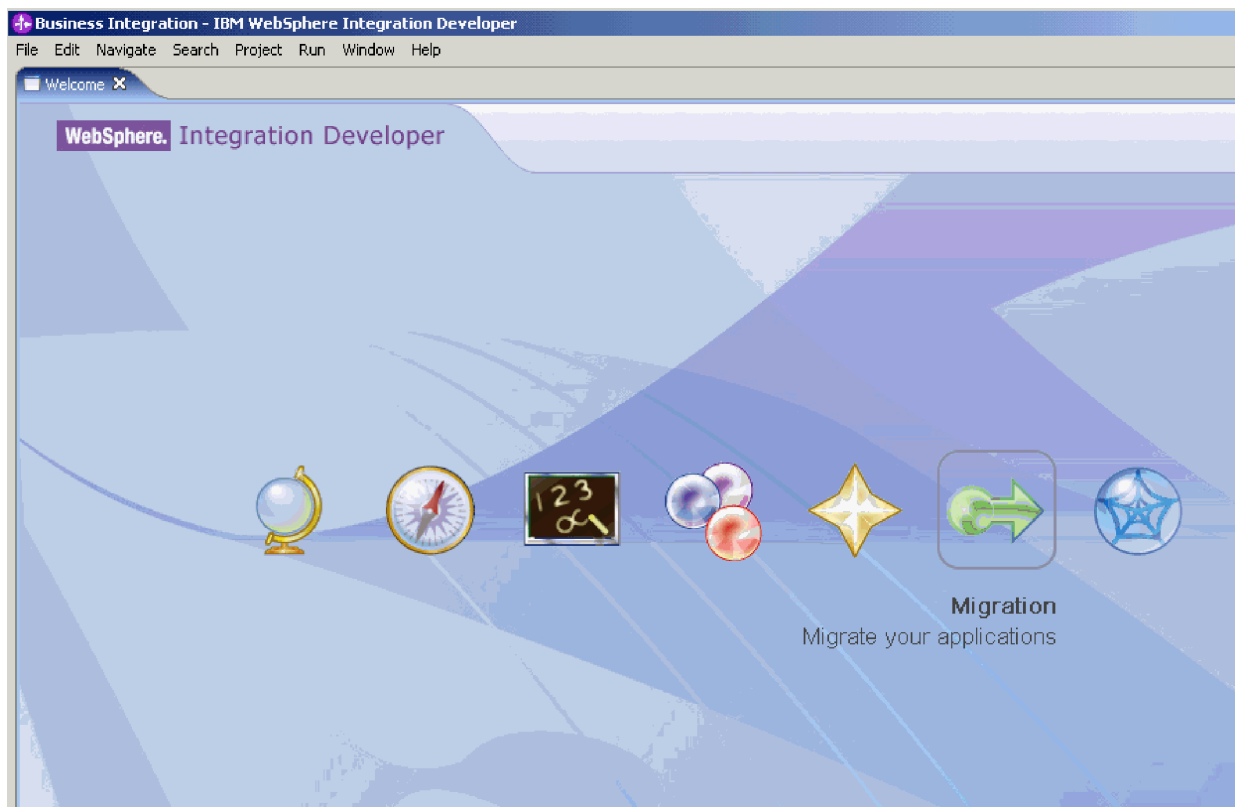
1. Cerciorarse de que ejecuta una versión de WebSphere InterChange Server que pueda migrarse a WebSphere Process Server. Consulte el tema "Vías de migración soportadas para WebSphere InterChange Server".
2. Exportar los artefactos origen de WebSphere InterChange Server a un archivo jar de repositorio mediante el mandato de WebSphere InterChange Server **repos_copy** tal como se describe en la documentación de WebSphere InterChange Server. Este archivo jar se especificará como entrada de las herramientas de migración.

Migrar WebSphere InterChange Server utilizando el asistente de migración

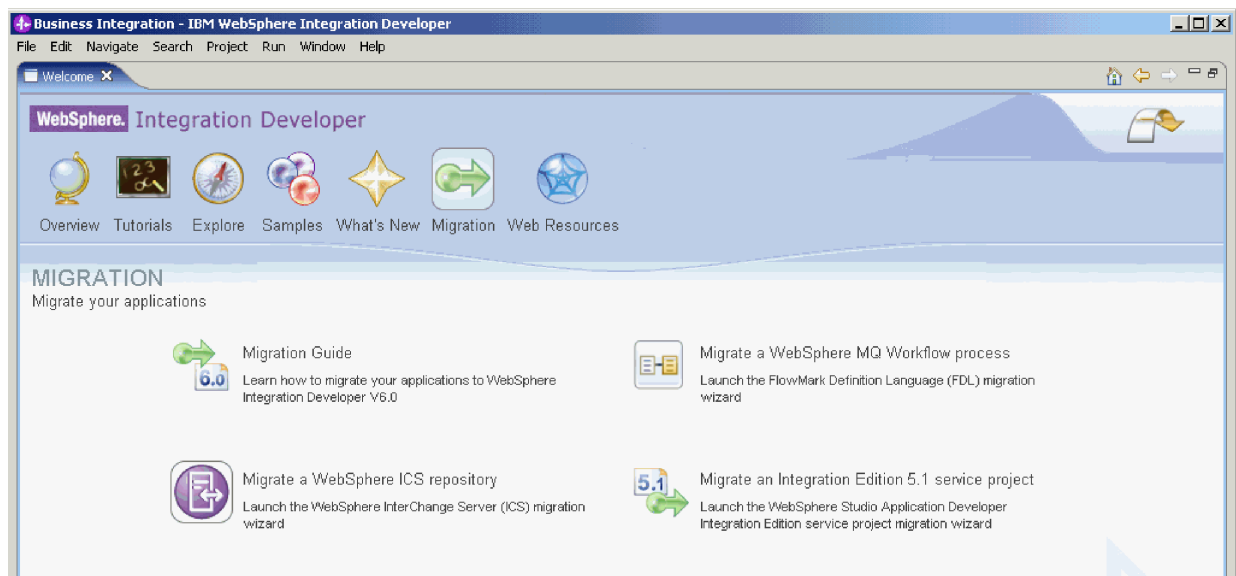
Puede utilizar el asistente de migración de WebSphere Integration Developer para migrar los artefactos actuales de WebSphere InterChange Server.

Siga estos pasos para utilizar el asistente de migración a fin de migrar los artefactos de WebSphere InterChange Server:

1. En la página de bienvenida, pulse  para abrir la página Migración:

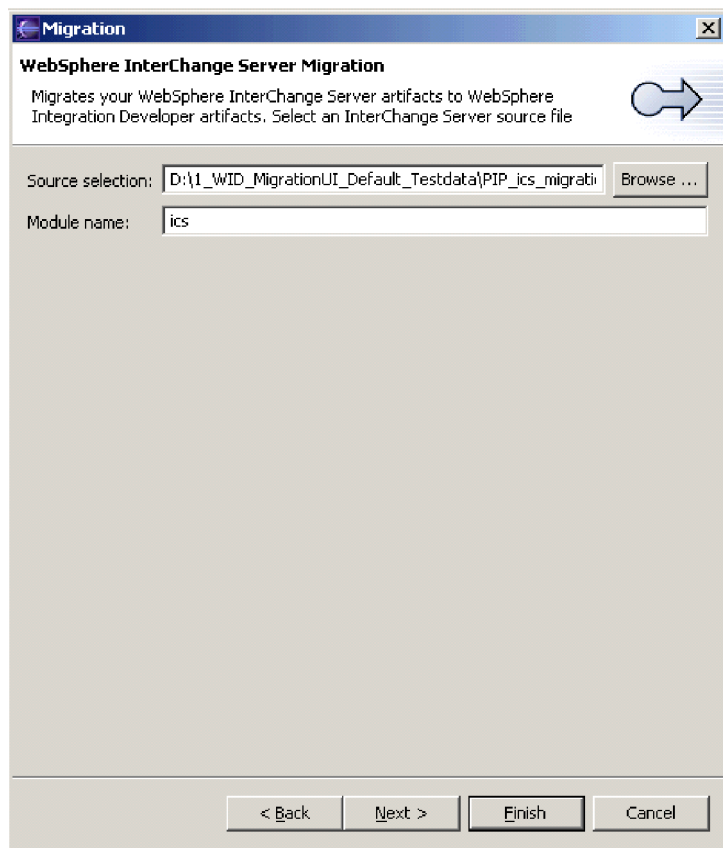


2. En la página Migración, seleccione la opción Migrar un repositorio ICS de WebSphere:

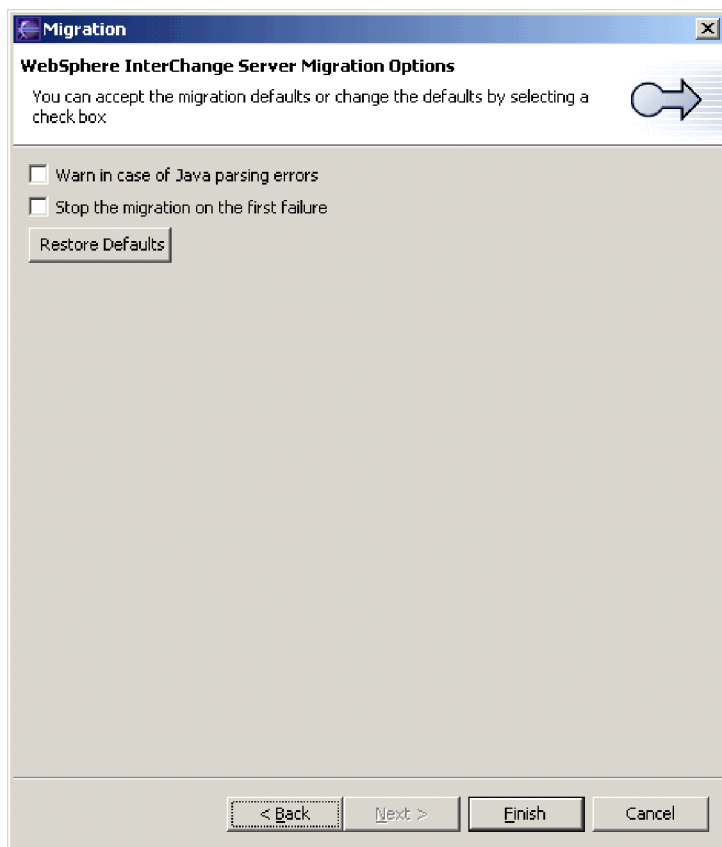


También puede lanzar el asistente de migración desde la opción de menú **Archivo** → **Importar**, seleccionando **Archivo JAR de WebSphere InterChange Server** y pulsando **Siguiente**.

3. Se abre el asistente de migración. Especifique el nombre del archivo origen en el campo **Selección de origen** pulsando el botón **Examinar** y desplazándose hasta el archivo. Especifique el nombre del módulo en el campo correspondiente. Pulse **Siguiente**.



4. Se abre la ventana Opciones de migración. Puede aceptar los valores predeterminados de la migración o marcar un recuadro de selección para cambiar la opción.



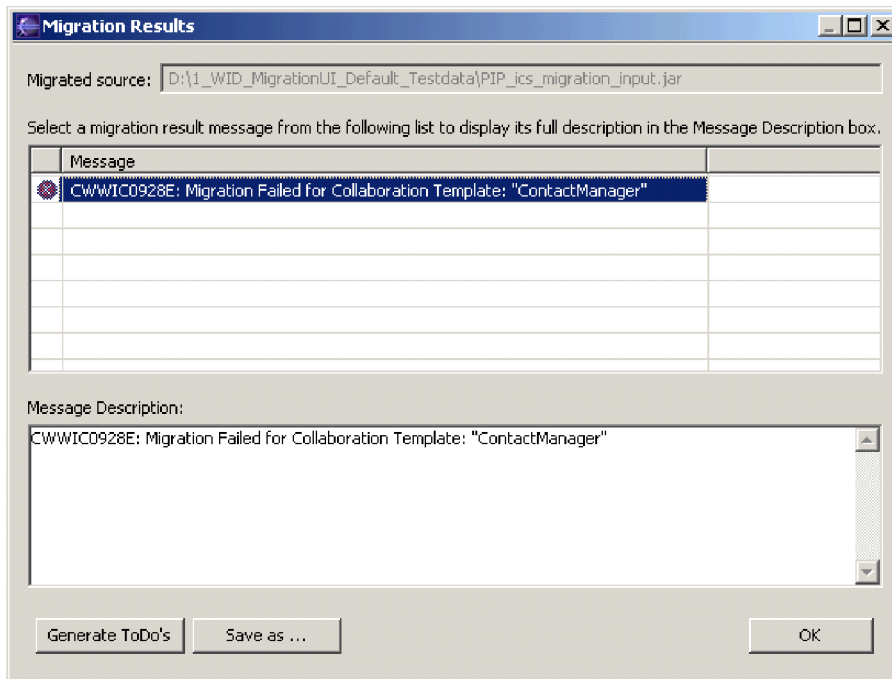
Pulse **Finalizar**.

Verificar la migración de WebSphere InterChange Server

Si no se ha notificado ningún error durante la migración del archivo jar de WebSphere InterChange Server, la migración de los artefactos es correcta. Si no ha finalizado correctamente el asistente de migración, se visualiza una lista de mensajes de error, aviso e informativos. Puede emplear estos mensajes para verificar la migración de WebSphere InterChange Server.

Nota: Debido a la complejidad que supone la migración desde WebSphere InterChange Server a WebSphere Process Server, es muy aconsejable probar por extenso las aplicaciones finales en ejecución en WebSphere Process Server con objeto de cerciorarse de que funcionan según las previsiones antes de ejecutarlas en el entorno de producción.

Al finalizar el asistente de migración se muestra la página siguiente:



En la ventana Resultados de la migración, puede ver una lista de mensajes relacionados con la migración. Pulse el mensaje para ver una descripción completa de sus detalles. Si es necesario, a partir de esta lista de mensajes, puede crear una lista con los aspectos que deberá corregir pulsando el botón **Generar ToDo**.

Trabajar con los errores de migración de WebSphere InterChange Server

Si se producen anomalías en la migración de WebSphere InterChange Server, existen dos métodos para manejar los errores.

Nota: Puede que prefiera la primera opción ya que inicialmente estará más familiarizado con WebSphere InterChange Server. Sin embargo, a medida que adquiera más experiencia con WebSphere Process Server y sus nuevos artefactos, puede optar por reparar los artefactos migrados en WebSphere Integration Developer.

1. Si el tipo de error lo permite, puede ajustar los artefactos de WebSphere InterChange Server mediante el juego de herramientas de WebSphere InterChange Server, exportar el archivo jar nuevamente y volver a intentar la migración.
2. Puede corregir los errores de los artefactos de WebSphere Process Server obtenidos editando los artefactos en WebSphere Integration Developer.

Artefactos de WebSphere InterChange Server manejados por las herramientas de migración

Las herramientas de migración pueden migrar automáticamente algunos de los artefactos de WebSphere InterChange Server.

Pueden migrarse los siguientes artefactos:

- Los **objetos comerciales** pasan a ser objetos comerciales de WebSphere Process Server
- Las **correlaciones** pasan a ser correlaciones de WebSphere Process Server
- Las **relaciones** pasan a ser relaciones y cometidos de WebSphere Process Server
- Las **plantillas de colaboración** pasan a ser BPEL y WSDL de WebSphere Process Server
- Los **objetos de colaboración** pasan a ser artefactos SCA de WebSphere Process Server
- Las **definiciones de conector** pasan a ser artefactos SCA de WebSphere Process Server

Las herramientas de migración pueden configurar automáticamente recursos en WebSphere Process Server para los siguientes artefactos/recursos de WebSphere InterChange Server:

- Agrupaciones de conexiones de BD
- Bases de datos de relaciones
- Entradas del planificador

Las herramientas de migración *no* manejan los siguientes artefactos de WebSphere InterChange Server:

- Artefactos Benchmark

API de WebSphere InterChange Server soportadas

Además de las herramientas de migración de artefactos origen de WebSphere InterChange Server suministradas en WebSphere Process Server y WebSphere Integration Developer, también existe soporte para muchas de las API suministradas en WebSphere InterChange Server. Las herramientas de migración funcionan en combinación con estas API de WebSphere InterChange Server conservando todo lo posible el código personalizado al realizar la migración.

Nota: Estas API sólo se suministran como soporte de las aplicaciones de WebSphere InterChange Server migradas hasta que puedan modificarse para utilizar nuevas API de servidor de procesos. Todas estas API de WebSphere InterChange Server han quedado obsoletas y se eliminarán en un release futuro.

Las API de WebSphere InterChange Server soportadas en el servidor de procesos se listan más abajo. Estas API suministran funciones del servidor de procesos similares a las funciones que suministran en WebSphere InterChange Server. Consulte la documentación de WebSphere InterChange Server para obtener una descripción funcional de estas API.

BusObj

Collaboration/

- BusObj(DataObject)
- BusObj(String)
- BusObj(String, Locale)
- copy(BusObj)
- duplicate():BusObj
- equalKeys(BusObj):boolean
- equals(Object):boolean
- equalsShallow(BusObj):boolean
- exists(String):boolean
- get(int):Object
- get(String):Object
- getBoolean(String):boolean
- getBusObj(String):BusObj
- getBusObjArray(String):BusObjArray
- getCount(String):int
- getDouble(String):double
- getFloat(String):float
- getInt(String):int
- getKeys():String
- getLocale():java.util.Locale
- getLong(String):long
- getLongText(String):String

- getString(String):String
- getType():String
- getValues():String
- getVerb():String
- isBlank(String):boolean
- isKey(String):boolean
- isNull(String):boolean
- isRequired(String):boolean
- keysToString():String
- set(BusObj)
- set(int, Object)
- set(String, boolean)
- set(String, double)
- set(String, float)
- set(String, int)
- set(String, long)
- set(String, Object)
- set(String, String)
- setContent(BusObj)
- setDefaultAttrValues()
- setKeys(BusObj)
- setLocale(java.util.Locale)
- setVerb(String)
- setWithCreate(String, BusObj)
- setWithCreate(String, BusObjArray)
- setWithCreate(String, Object)
- toString():String
- validData(String, boolean):boolean
- validData(String, BusObj):boolean
- validData(String, BusObjArray):boolean
- validData(String, double):boolean
- validData(String, float):boolean
- validData(String, int):boolean
- validData(String, long):boolean
- validData(String, Object):boolean
- validData(String, String):boolean

BusObjArray

Collaboration/

- addElement(BusObj)
- duplicate():BusObjArray
- elementAt(int):BusObj
- equals(BusObjArray):boolean
- getElements():BusObj[]
- getLastIndex():int

- max(String):String
- maxBusObjArray(String):BusObjArray
- maxBusObjs(String):BusObj[]
- min(String):String
- minBusObjArray(String):BusObjArray
- minBusObjs(String):BusObj[]
- removeAllElements()
- removeElement(BusObj)
- removeElementAt(int)
- setElementAt(int, BusObj)
- size():int
- sum(String):double
- swap(int, int)
- toString():String

BaseDLM

DLM/

- BaseDLM(BaseMap)
- getDBConnection(String):CwDBConnection
- getDBConnection(String, boolean):CwDBConnection
- getName():String
- getRelConnection(String):DtpConnection
- implicitDBTransactionBracketing():boolean
- isTraceEnabled(int):boolean
- logError(int)
- logError(int, Object[])
- logError(int, String)
- logError(int, String, String)
- logError(int, String, String, String)
- logError(int, String, String, String, String)
- logError(int, String, String, String, String, String)
- logError(String)
- logInfo(int)
- logInfo(int, Object[])
- logInfo(int, String)
- logInfo(int, String, String)
- logInfo(int, String, String, String)
- logInfo(int, String, String, String, String)
- logInfo(int, String, String, String, String, String)
- logInfo(String)
- logWarning(int)
- logWarning(int, Object[])
- logWarning(int, String)
- logWarning(int, String, String)
- logWarning(int, String, String, String)

- logWarning(int, String, String, String, String)
- logWarning(int, String, String, String, String, String)
- logWarning(String)
- raiseException(RuntimeEntityException)
- raiseException(String, int)
- raiseException(String, int, Object[])
- raiseException(String, int, String)
- raiseException(String, int, String, String)
- raiseException(String, int, String, String, String)
- raiseException(String, int, String, String, String, String)
- raiseException(String, int, String, String, String, String, String)
- raiseException(String, String)
- releaseRelConnection(boolean)
- trace(int, int)
- trace(int, int, Object[])
- trace(int, int, String)
- trace(int, int, String, String)
- trace(int, int, String, String, String)
- trace(int, int, String, String, String, String)
- trace(int, int, String, String, String, String, String)
- trace(int, String)
- trace(String)

CwDBConnection

CwDBConnection/

CxCommon/

- beginTransaction()
- commit()
- executePreparedSQL(String)
- executePreparedSQL(String, Vector)
- executeSQL(String)
- executeSQL(String, Vector)
- executeStoredProcedure(String, Vector)
- getUpdateCount():int
- hasMoreRows():boolean
- inTransaction():boolean
- isActive():boolean
- nextRow():Vector
- release()
- rollback()

CwDBConstants

CwDBConnection/

CxCommon/

- PARAM_IN - 0
- PARAM_INOUT - 1

- PARAM_OUT - 2

CwDBStoredProcedureParam

CwDBConnection/

CxCommon/

- CwDBStoredProcedureParam(int, Array)
- CwDBStoredProcedureParam(int, BigDecimal)
- CwDBStoredProcedureParam(int, boolean)
- CwDBStoredProcedureParam(int, Boolean)
- CwDBStoredProcedureParam(int, byte[])
- CwDBStoredProcedureParam(int, double)
- CwDBStoredProcedureParam(int, Double)
- CwDBStoredProcedureParam(int, float)
- CwDBStoredProcedureParam(int, Float)
- CwDBStoredProcedureParam(int, int)
- CwDBStoredProcedureParam(int, Integer)
- CwDBStoredProcedureParam(int, java.sql.Blob)
- CwDBStoredProcedureParam(int, java.sql.Clob)
- CwDBStoredProcedureParam(int, java.sql.Date)
- CwDBStoredProcedureParam(int, java.sql.Struct)
- CwDBStoredProcedureParam(int, java.sql.Time)
- CwDBStoredProcedureParam(int, java.sql.Timestamp)
- CwDBStoredProcedureParam(int, Long)
- CwDBStoredProcedureParam(int, String)
- CwDBStoredProcedureParam(int, String, Object)
- getParamType():int getValue():Object

DtpConnection

Dtp/

CxCommon/

- beginTran()
- commit()
- executeSQL(String)
- executeSQL(String, Vector)
- executeStoredProcedure(String, Vector)
- getUpdateCount():int
- hasMoreRows():boolean
- inTransaction():boolean
- isActive():boolean
- nextRow():Vector
- rollback()

DtpDataConversion

Dtp/

CxCommon/

- BOOL_TYPE - 4
- CANNOTCONVERT - 2

- DATE_TYPE - 5
- DOUBLE_TYPE - 3
- FLOAT_TYPE - 2
- INTEGER_TYPE - 0
- LONGTEXT_TYPE - 6
- OKTOCONVERT - 0
- POTENTIALDATALOSS - 1
- STRING_TYPE - 1
- UNKNOWN_TYPE - 999
- getType(double):int
- getType(float):int
- getType(int):int
- getType(Object):int
- isOKToConvert(int, int):int
- isOKToConvert(String, String):int
- toBoolean(boolean):Boolean
- toBoolean(Object):Boolean
- toDouble(double):Double
- toDouble(float):Double
- toDouble(int):Double
- toDouble(Object):Double
- toFloat(double):Float
- toFloat(float):Float
- toFloat(int):Float
- toFloat(Object):Float
- toInteger(double):Integer
- toInteger(float):Integer
- toInteger(int):Integer
- toInteger(Object):Integer
- toPrimitiveBoolean(Object):boolean
- toPrimitiveDouble(float):double
- toPrimitiveDouble(int):double
- toPrimitiveDouble(Object):double
- toPrimitiveFloat(double):float
- toPrimitiveFloat(int):float
- toPrimitiveFloat(Object):float
- toPrimitiveInt(double):int
- toPrimitiveInt(float):int
- toPrimitiveInt(Object):int
- toString(double):String
- toString(float):String
- toString(int):String
- toString(Object):String

DtpDate

Dtp/

CxCommon/

- DtpDate()
- DtpDate(long, boolean)
- DtpDate(String, String)
- DtpDate(String, String, String[], String[])
- addDays(int):DtpDate
- addMonths(int):DtpDate
- addWeekdays(int):DtpDate
- addYears(int):DtpDate
- after(DtpDate):boolean
- before(DtpDate):boolean
- calcDays(DtpDate):int
- calcWeekdays(DtpDate):int
- get12MonthNames():String[]
- get12ShortMonthNames():String[]
- get7DayNames():String[]
- getCWDate():String
- getDayOfMonth():String
- getDayOfWeek():String
- getHours():String
- getIntDay():int
- getIntDayOfWeek():int
- getIntHours():int
- getIntMilliseconds():int
- getIntMinutes():int
- getIntMonth():int
- getIntSeconds():int
- getIntYear():int
- getMaxDate(BusObjArray, String, String):DtpDate
- getMaxDateBO(BusObj[], String, String):BusObj[]
- getMaxDateBO(BusObjArray, String, String):BusObj[]
- getMinDate(BusObjArray, String, String):DtpDate
- getMinDateBO(BusObj[], String, String):BusObj[]
- getMinDateBO(BusObjArray, String, String):BusObj[]
- getMinutes():String
- getMonth():String
- getMSSince1970():long
- getNumericMonth():String
- getSeconds():String
- getShortMonth():String
- getYear():String
- set12MonthNames(String[], boolean)
- set12MonthNamesToDefault()

- set12ShortMonthNames(String[])
- set12ShortMonthNamesToDefault()
- set7DayNames(String[])
- set7DayNamesToDefault()
- toString():String
- toString(String):String
- toString(String, boolean):String

DtpMapService

Dtp/

CxCommon/

- runMap(String, String, BusObj[], CxExecutionContext):BusObj[]

DtpSplitString

Dtp/

CxCommon/

- DtpSplitString(String, String)
- elementAt(int):String
- firstElement():String
- getElementCount():int
- getEnumeration():Enumeration
- lastElement():String
- nextElement():String
- prevElement():String
- reset()

DtpUtils

Dtp/

CxCommon/

- padLeft(String, char, int):String
- padRight(String, char, int):String
- stringReplace(String, String, String):String
- truncate(double):int
- truncate(double, int):double
- truncate(float):int
- truncate(float, int):double
- truncate(Object):int
- truncate(Object, int):double

IdentityRelationship

relationship/

utilities/

crossworlds/

com/

- addMyChildren(String, String, BusObj, String, Object, CxExecutionContext)
- deleteMyChildren(String, String, BusObj, String, CxExecutionContext)
- deleteMyChildren(String, String, BusObj, String, Object, CxExecutionContext)
- foreignKeyLookup(String, String, BusObj, String, BusObj, String, CxExecutionContext)

- foreignKeyXref(String, String, String, BusObj, String, BusObj, String, CxExecutionContext)
- maintainChildVerb(String, String, String, BusObj, String, BusObj, String, CxExecutionContext, boolean, boolean)
- maintainCompositeRelationship(String, String, BusObj, Object, CxExecutionContext)
- maintainSimpleIdentityRelationship(String, String, BusObj, BusObj, CxExecutionContext)
- updateMyChildren(String, String, BusObj, String, String, String, String, CxExecutionContext)

MapExeContext

Dtp/

CxCommon/

- ACCESS_REQUEST - "SUBSCRIPTION_DELIVERY"
- ACCESS_RESPONSE - "ACCESS_RETURN_REQUEST"
- EVENT_DELIVERY - "SUBSCRIPTION_DELIVERY"
- getConnName():String
- getGenericBO():BusObj
- getInitiator():String
- getLocale():java.util.Locale
- getOriginalRequestBO():BusObj
- SERVICE_CALL_FAILURE - "CONSUME_FAILED"
- SERVICE_CALL_REQUEST - "CONSUME"
- SERVICE_CALL_RESPONSE - "DELIVERBUSOBJ"
- setConnName(String)
- setInitiator(String)
- setLocale(java.util.Locale)

Participant

RelationshipServices/

Server/

- Participant(String, String, int, BusObj)
- Participant(String, String, int, String)
- Participant(String, String, int, long)
- Participant(String, String, int, int)
- Participant(String, String, int, double)
- Participant(String, String, int, float)
- Participant(String, String, int, boolean)
- Participant(String, String, BusObj)
- Participant(String, String, String)
- Participant(String, String, long)
- Participant(String, String, int)
- Participant(String, String, double)
- Participant(String, String, float)
- Participant(String, String, boolean)
- getBoolean():boolean
- getBusObj():BusObj
- getDouble():double
- getFloat():float
- getInstanceId():int

- getInt():int
- getLong():long
- getParticipantDefinition():String
- getRelationshipDefinition():String
- getString():String INVALID_INSTANCE_ID
- set(boolean)
- set(BusObj)
- set(double)
- set(float)
- set(int)
- set(long)
- set(String)
- setInstanceId(int)
- setParticipantDefinition(String)
- setRelationshipDefinition(String)
- setParticipantDefinition(String)
- setRelationshipDefinition(String)

Relationship

RelationshipServices/

Server/

- addMyChildren(String, String, BusObj, String, Object, CxExecutionContext)
- addParticipant(Participant):int
- addParticipant(String, String, boolean):int
- addParticipant(String, String, BusObj):int
- addParticipant(String, String, double):int
- addParticipant(String, String, float):int
- addParticipant(String, String, int):int
- addParticipant(String, String, int, boolean):int
- addParticipant(String, String, int, BusObj):int
- addParticipant(String, String, int, double):int
- addParticipant(String, String, int, float):int
- addParticipant(String, String, int, int):int
- addParticipant(String, String, int, long):int
- addParticipant(String, String, int, String):int
- addParticipant(String, String, long):int
- addParticipant(String, String, String):int
- create(Participant):int
- create(String, String, boolean):int
- create(String, String, BusObj):int
- create(String, String, double):int
- create(String, String, float):int
- create(String, String, int):int
- create(String, String, long):int
- create(String, String, String):int
- deactivateParticipant(Participant)

- deactivateParticipant(String, String, boolean)
- deactivateParticipant(String, String, BusObj)
- deactivateParticipant(String, String, double)
- deactivateParticipant(String, String, float)
- deactivateParticipant(String, String, int)
- deactivateParticipant(String, String, long)
- deactivateParticipant(String, String, String)
- deactivateParticipantByInstance(String, String, int)
- deactivateParticipantByInstance(String, String, int, boolean)
- deactivateParticipantByInstance(String, String, int, BusObj)
- deactivateParticipantByInstance(String, String, int, double)
- deactivateParticipantByInstance(String, String, int, float)
- deactivateParticipantByInstance(String, String, int, int)
- deactivateParticipantByInstance(String, String, int, long)
- deactivateParticipantByInstance(String, String, int, String)
- deleteMyChildren(String, String, BusObj, String, CxExecutionContext)
- deleteMyChildren(String, String, BusObj, String, Object, CxExecutionContext)
- deleteParticipant(Participant)
- deleteParticipant(String, String, boolean)
- deleteParticipant(String, String, BusObj)
- deleteParticipant(String, String, double)
- deleteParticipant(String, String, float)
- deleteParticipant(String, String, int)
- deleteParticipant(String, String, long)
- deleteParticipant(String, String, String)
- deleteParticipantByInstance(String, String, int)
- deleteParticipantByInstance(String, String, int, boolean)
- deleteParticipantByInstance(String, String, int, BusObj)
- deleteParticipantByInstance(String, String, int, double)
- deleteParticipantByInstance(String, String, int, float)
- deleteParticipantByInstance(String, String, int, int)
- deleteParticipantByInstance(String, String, int, long)
- deleteParticipantByInstance(String, String, int, String)
- getNewID(String):int
- maintainCompositeRelationship(String, String, BusObj, Object, CxExecutionContext)
- maintainSimpleIdentityRelationship(String, String, BusObj, BusObj, CxExecutionContext)
- retrieveInstances(String, boolean):int[]
- retrieveInstances(String, BusObj):int[]
- retrieveInstances(String, double):int[]
- retrieveInstances(String, float):int[]
- retrieveInstances(String, int):int[]
- retrieveInstances(String, long):int[]
- retrieveInstances(String, String):int[]
- retrieveInstances(String, String, boolean):int[]
- retrieveInstances(String, String, BusObj):int[]

- retrieveInstances(String, String, double):int[]
- retrieveInstances(String, String, float):int[]
- retrieveInstances(String, String, int):int[]
- retrieveInstances(String, String, long):int[]
- retrieveInstances(String, String, String):int[]
- retrieveInstances(String, String[], boolean):int[]
- retrieveInstances(String, String[], BusObj):int[]
- retrieveInstances(String, String[], double):int[]
- retrieveInstances(String, String[], float):int[]
- retrieveInstances(String, String[], int):int[]
- retrieveInstances(String, String[], long):int[]
- retrieveInstances(String, String[], String):int[]
- retrieveParticipants(String, int):Participant[]
- retrieveParticipants(String, String, int):Participant[]
- retrieveParticipants(String, String[], int):Participant[]
- updateMyChildren(String, String, BusObj, String, String, String, String, CxExecutionContext)
- updateParticipant(String, String, BusObj)
- updateParticipantByInstance(Participant)
- updateParticipantByInstance(String, String, int)
- updateParticipantByInstance(String, String, int, BusObj)

UserStoredProcedureParam

Dtp/

CxCommon/

- UserStoredProcedureParam(int, String, Object, String, String)
- getParamDataTypeJavaObj():String
- getParamDataTypeJDBC():int
- getParamIndex():int
- getParamIOType():String
- getParamName():String
- getParamValue():Object
- setParamDataTypeJavaObj(String)
- setParamDataTypeJDBC(int)
- setParamIndex(int)
- setParamIOType(String)
- setParamName(String)
- setParamValue(Object)

In StoredProcedureParam

- PARAM_TYPE_IN - "IN"
- PARAM_TYPE_OUT - "OUT"
- PARAM_TYPE_INOUT - "INOUT"
- DATA_TYPE_STRING - "String"
- DATA_TYPE_INTEGER - "Integer"
- DATA_TYPE_DOUBLE - "Double"
- DATA_TYPE_FLOAT - "Float"

- DATA_TYPE_BOOLEAN - "Boolean"
- DATA_TYPE_TIME - "java.sql.Time"
- DATA_TYPE_DATE - "java.sql.Date"
- DATA_TYPE_TIMESTAMP - "java.sql.Timestamp"
- DATA_TYPE_BIG_DECIMAL - "java.math.BigDecimal"
- DATA_TYPE_LONG_INTEGER - "Long"
- DATA_TYPE_BINARY - "byte[]"
- DATA_TYPE_CLOB - "Clob"
- DATA_TYPE_BLOB - "Blob"
- DATA_TYPE_ARRAY - "Array"
- DATA_TYPE_STRUCT - "Struct"
- DATA_TYPE_REF - "Ref"

BaseCollaboration Collaboration/

- BaseCollaboration(com.ibm.bpe.api.ProcessInstanceData)
- AnyException - "AnyException"
- AppBusObjDoesNotExist - "BusObjDoesNotExist"
- AppLogOnFailure - "AppLogOnFailure"
- AppMultipleHits - "AppMultipleHits"
- AppRequestNotYetSent - "AppRequestNotYetSent"
- AppRetrieveByContentFailed - "AppRetrieveByContent"
- AppTimeOut - "AppTimeOut"
- AppUnknown - "AppUnknown"
- AttributeException - "AttributeException"
- existsConfigProperty(String):boolean
- getConfigProperty(String):String
- getConfigPropertyArray(String):String[]
- getCurrentLoopIndex():int
- getDBConnection(String):CwDBConnection
- getDBConnection(String, boolean):CwDBConnection getLocale():java.util.Locale
- getMessage(int):String
- getMessage(int, Object[]):String
- getName():String
- implicitDBTransactionBracketing():boolean
- isCallerInRole(String):boolean
- isTraceEnabled(int):boolean
- JavaException - "JavaException"
- logError(int)
- logError(int, Object[])
- logError(int, String)
- logError(int, String, String)
- logError(int, String, String, String)
- logError(int, String, String, String, String)
- logError(int, String, String, String, String, String)

- `logError(String)`
- `logInfo(int)`
- `logInfo(int, Object[])`
- `logInfo(int, String)`
- `logInfo(int, String, String)`
- `logInfo(int, String, String, String)`
- `logInfo(int, String, String, String, String)`
- `logInfo(int, String, String, String, String, String)`
- `logInfo(String)`
- `logWarning(int)`
- `logWarning(int, Object[])`
- `logWarning(int, String)`
- `logWarning(int, String, String)`
- `logWarning(int, String, String, String)`
- `logWarning(int, String, String, String, String)`
- `logWarning(int, String, String, String, String, String)`
- `logWarning(String)`
- `not(boolean):boolean` `ObjectException` - "`ObjectException`"
- `OperationException` - "`OperationException`"
- `raiseException(CollaborationException)`
- `raiseException(String, int)`
- `raiseException(String, int, Object[])`
- `raiseException(String, int, String)`
- `raiseException(String, int, String, String)`
- `raiseException(String, int, String, String, String)`
- `raiseException(String, int, String, String, String, String)`
- `raiseException(String, int, String, String, String, String, String)`
- `raiseException(String, String)`
- `ServiceCallException` - "`ConsumerException`"
- `ServiceCallTransportException` - "`ServiceCallTransportException`"
- `SystemException` - "`SystemException`"
- `trace(int, int)`
- `trace(int, int, Object[])`
- `trace(int, int, String)`
- `trace(int, int, String, String)`
- `trace(int, int, String, String, String)`
- `trace(int, int, String, String, String, String)`
- `trace(int, int, String, String, String, String, String)`
- `trace(int, String)`
- `trace(String)`
- `TransactionException` - "`TransactionException`"

CxExecutionContext

CxCommon/

- `CxExecutionContext()`

- getContext(String):Object
- MAPCONTEXT - "MAPCONTEXT"
- setContext(String, Object)

CollaborationException

Collaboration/

- getMessage():String
- getMsgNumber():int
- getSubType():String
- getText():String
- getType():String
- toString():String

Filter

crossworlds/

com/

- Filter(BaseCollaboration)
- filterExcludes(String, String):boolean
- filterIncludes(String, String):boolean
- recurseFilter(BusObj, String, boolean, String, String):boolean
- recursePreReqs(String, Vector):int

Globals

crossworlds/

com/

- Globals(BaseCollaboration)
- callMap(String, BusObj):BusObj

SmartCollabService

crossworlds/

com/

- SmartCollabService()
- SmartCollabService(BaseCollaboration)
- doAgg(BusObj, String, String, String):BusObj
- doMergeHash(Vector, String, String):Vector
- doRecursiveAgg(BusObj, String, String, String):BusObj
- doRecursiveSplit(BusObj, String):Vector
- doRecursiveSplit(BusObj, String, boolean):Vector
- getKeyValues(BusObj, String):String
- merge(Vector, String):BusObj
- merge(Vector, String, BusObj):BusObj
- split(BusObj, String):Vector

StateManagement

crossworlds/

com/

- StateManagement()
- beginTransaction()
- commit()

- deleteBO(String, String, String)
- deleteState(String, String, String, int)
- persistBO(String, String, String, String, BusObj)
- recoverBO(String, String, String):BusObj
- releaseDBConnection()
- resetData()
- retrieveState(String, String, String, int):int
- saveState(String, String, String, String, int, int, double)
- setDBConnection(CwDBConnection)
- updateBO(String, String, String, String, BusObj)
- updateState(String, String, String, String, int, int)

EventKeyAttrDef
EventManagement/
CxCommon/

- EventKeyAttrDef()
- EventKeyAttrDef(String, String)
- public String keyName
- public String keyValue

EventQueryDef
EventManagement/
CxCommon/

- EventQueryDef()
- EventQueryDef(String, String, String, String, int)
- public String nameConnector
- public String nameCollaboration
- public String nameBusObj
- public String verb
- public int ownerType

FailedEventInfo
EventManagement/
CxCommon/

- FailedEventInfo()
- FailedEventInfo(String x6, int, EventKeyAttrDef[], int, int, String, String, int)
- public String nameOwner
- public String nameConnector
- public String nameBusObj
- public String nameVerb
- public String strTime
- public String strMessage
- public int wipIndex
- public EventKeyAttrDef[] strbusObjKeys
- public int nKeys
- public int eventStatus
- public String expirationTime

- public String scenarioName
- public int scenarioState

CwBiDiEngine

- BiDiBOTransformation(BusinessObject, String, String, boolean):BusinessObj
- BiDiBusObjTransformation(BusObj, String, String, boolean):BusObj
- BiDiStringTransformation(String, String, String):String

Correlacionar el DataObject de WebSphere Process Server desde el XML de WebSphere InterChange Server

Si utiliza los Adaptadores de legado para conectarse a WebSphere Process Server, el algoritmo siguiente le permitirá una comprensión más profunda de cómo se ha creado el DataObject de WebSphere Process Server a partir del XML de WebSphere InterChange Server. Esta información muestra dónde se han colocado los valores de datos y también qué valores de datos se han elegido para sustituir a los utilizados en WebSphere InterChange Server.

General

- Para establecer el verbo de ChangeSummary, todos los valores se establecerán con las API **markCreate/Update/Delete**.
- Para establecer el verbo de ChangeSummary/EventSummary, los verbos **Create**, **Update** y **Delete** se establecerán en ChangeSummary, mientras que todos los demás verbos se establecerán en EventSummary.
- Para obtener el verbo de ChangeSummary:
 - Si isDelete es true, el verbo será **Delete**.

Nota: El valor de isCreate se pasará por alto si isDelete es true. Esto puede ocurrir si la creación se ha marcado antes de que el DataObject entrara en el concentrador en el punto en que se ha suprimido, o si la creación y la supresión se han producido ambas en el concentrador.

- Si isDelete es false e isCreate es true, el verbo será **Create**.
- Si isDelete es false e isCreate es false, el verbo será **Update**.
- Para evitar que un DataObject se identifique como **Create** en lugar de un **Update** esperado, si la anotación está habilitada, debe:
 - Suspender la anotación durante la creación del DataObject.
 - Reanudar la anotación para la actualización (update) del DataObject (o utilizar la API **markUpdated**).

Carga

La carga cargará un XML de tiempo de ejecución de WebSphere

InterChange Server en una instancia de WebSphere

Business Integration BusinessGraph AfterImage.

- Se creará una instancia del BusinessGraph adecuado.
- Se activará la anotación de ChangeSummary, a fin de que su activación posterior no borre las entradas.
- La anotación de ChangeSummary quedará en pausa para evitar que entre información no deseada en ChangeSummary.
- Los atributos del BusinessObject de nivel superior se crearán en el DataObject (consulte la sección "Proceso de atributos" que figura más adelante).
- Si el BusinessObject de nivel superior tiene BusinessObjects hijos, estos se procesarán recursivamente.

- Los atributos de estos BusinessObjects hijos se crearán en el DataObject (consulte la sección "Proceso de atributos" que figura más adelante).
- El verbo del BusinessObject de nivel superior se establecerá en el verbo de nivel superior del BusinessGraph y se establecerá en los resúmenes.
- El verbo de los BusinessObjects hijos se establecerá en los resúmenes.

Guardado

El guardado guardará una instancia de WebSphere

Business Integration BusinessGraph AfterImage en un XML de tiempo de ejecución de WebSphere

InterChange Server. Se lanzará una excepción si el BusinessGraph de entrada no es AfterImage.

- Se creará una instancia del documento XML de WebSphere InterChange Server.
- Todos los DataObjects suprimidos del ChangeSummary se tratarán como si existieran en sus posiciones originales.

Nota: Este proceso de "desupresión" no conservará el orden de las listas.

- El verbo del BusinessObject de nivel superior se establecerá en el verbo de nivel superior del BusinessGraph.
- Los atributos del BusinessObject de nivel superior se establecerán a partir del DataObject (consulte la sección "Proceso de atributos" que figura más adelante).
- Si el BusinessObject tiene DataObjects hijos, estos se procesarán recursivamente.
- El verbo de los DataObjects hijos se manejará del siguiente modo:
 - Si no hay ningún verbo para el DataObject hijo en EventSummary o ChangeSummary, el verbo se establecerá en "" (serie vacía).
 - Si hay un verbo en EventSummary pero no en ChangeSummary, se utilizará este verbo.
 - Si hay un verbo en ChangeSummary pero no en EventSummary, se utilizará este verbo.
 - Si hay un verbo tanto en ChangeSummary como en EventSummary, el valor que figura en ChangeSummary tendrá preferencia sobre el de EventSummary.
- Los atributos de los DataObjects hijos se establecerán en el BusinessObject (consulte la sección "Proceso de atributos" que figura más adelante).

Proceso de atributos

- Todos los valores no indicados a continuación serán ASIS cargados/guardados.
- ObjectEventId se cargará en/guardará desde EventSummary.
- Para **CxBlank** y **CxIgnore**:
 - En el lado del BusinessObject de WebSphere Business Integration de la conversión, **CxBlank** y **CxIgnore** se establecerán/identificarán del siguiente modo:
 - **CxIgnore** - desestablecer o establecer con el valor Java null
 - **CxBlank** - valor dependiente del tipo, como se muestra en la tabla que figura más adelante
 - En el lado del XML de WebSphere InterChange Server de la conversión, **CxBlank** y **CxIgnore** se establecerán/identificarán del siguiente modo:

Tabla 1. Establecer CxBlank y CxIgnore

Tipo	CxIgnore	CxBlank
Int	Integer.MIN_VALUE	Integer.MAX_VALUE
Float	Float.MIN_VALUE	Float.MAX_VALUE
Double	Double.MIN_VALUE	Double.MAX_VALUE

Tabla 1. Establecer CxBlank y CxIgnore (continuación)

Tipo	CxIgnore	CxBlank
String/date/longtext	"CxIgnore"	""
BusinessObjects hijos	(elemento vacío)	N/A

Procedimientos recomendados para el proceso de migración de WebSphere InterChange Server

Las siguientes directrices están destinadas al desarrollo de artefactos de desarrollo para WebSphere InterChange Server. Al ajustarse a estas directrices, facilitará la migración de artefactos de WebSphere InterChange Server a WebSphere Process Server.

Estas recomendaciones sólo deben utilizarse como guía para el desarrollo de soluciones de integración nuevas. El contenido existente puede no ajustarse a estas directrices. También puede haber casos en los que sea necesario desviarse de estas directrices. En estos casos, debe tener cuidado de limitar el ámbito de la desviación para minimizar la cantidad de trabajo que debe realizarse de nuevo para migrar los artefactos. Tenga en cuenta que las directrices indicadas aquí no especifican las mejores prácticas para el desarrollo de artefactos de WebSphere

InterChange Server en general. Su ámbito está limitado a aquellas consideraciones que pueden afectar a la facilidad de los artefactos que pueden migrarse en el futuro.

Desarrollo general

Existen varias consideraciones que se aplican ampliamente a la mayoría de los artefactos de integración. En general, los artefactos que aprovechan los recursos suministrados por las herramientas y que se ajustan a los modelos de metadatos obligatorios en ellas se migrarán con mayor fluidez. También es probable que los artefactos con extensiones y dependencias externas significativas requieran más intervención manual durante la migración.

La lista siguiente resume las mejores prácticas para el desarrollo general de soluciones basadas en WebSphere InterChange Server para facilitar la migración futura:

- Utilizar WebSphere InterChange Server para soluciones de integración de procesos automatizados en tiempo real
- Documentar el diseño del sistema y los componentes
- Utilizar las herramientas de desarrollo para editar los artefactos de integración
- Aprovechar las mejores prácticas de definición de normas con las herramientas y fragmentos de código Java

Aunque pueda parecer obvio, es importante que las soluciones de integración se ajusten al modelo de programación y a la arquitectura suministrados por WebSphere InterChange Server. Ésta es más adecuada a las soluciones de integración de procesos automatizados en tiempo real. Asimismo, cada uno de los componentes de integración de WebSphere InterChange Server juega un papel bien definido dentro de la arquitectura. Desviaciones significativas con respecto a este modelo dificultarán la migración de contenido a los artefactos adecuados de WebSphere Process Server.

Otra práctica genérica recomendada que mejorará en gran medida la migración de proyectos futuros consiste en documentar el diseño del sistema. Asegúrese de capturar la arquitectura y el diseño de integración, incluyendo el diseño funcional y los requisitos de calidad del servicio, las interdependencias de los artefactos compartidos en los proyectos y también las decisiones de diseño tomadas durante el despliegue. Esto le ayudará en el análisis del sistema durante la migración y minimizará la necesidad de volver a realizar trabajos.

Para crear, configurar y modificar definiciones de artefactos, es esencial utilizar sólo las herramientas de desarrollo suministradas. Evite la manipulación manual de metadatos de artefactos (por ejemplo, editar archivos XML directamente, que pueden dañar los artefactos para la migración).

Al desarrollar código Java dentro de plantillas de colaboración, correlaciones, programas de utilidad de código habituales y otros componentes, debe tener en cuenta varias consideraciones:

- Utilice sólo las API publicadas
- Utilice el editor de actividades
- Utilice adaptadores para acceder a EIS
- Evite dependencias externas en el fragmento de código Java
- Respete los procedimientos de desarrollo de J2EE con respecto a la portabilidad
- No genere hebras ni utilice primitivos de sincronización de hebras. Si necesita hacerlo, deberá reconvertirlas para utilizar beans asíncronos al realizar la migración.
- No realice operaciones de E/S de disco mediante `java.io.*`. Utilice JDBC para almacenar los datos.
- No ejecute funciones que puedan estar reservadas para un contenedor EJB, como por ejemplo E/S de socket, carga de clases, carga de bibliotecas nativas, etc. Si debe hacerlo, estos fragmentos de código necesitarán conversión manual para utilizar funciones de contenedor EJB al realizar la migración.

Utilice sólo las API publicadas en la documentación del producto para los artefactos. Estas API se indican con detalle en las guías de desarrollo de WebSphere InterChange Server. Aunque en muchos casos se suministrarán API de compatibilidad en WebSphere Process Server, sólo se incluirán las API publicadas. Aunque WebSphere InterChange Server tiene muchas interfaces internas que los desarrolladores pueden utilizar, no puede garantizarse que estén soportadas en el futuro.

Al diseñar la lógica comercial y las normas de transformación en correlaciones y plantillas de colaboración, asegúrese de utilizar la herramienta Editor de actividades siempre que sea posible. Con ello se asegurará de que la lógica se describa mediante metadatos que puedan convertirse más fácilmente a los artefactos nuevos. Para las operaciones que desee reutilizar en las herramientas, utilice la característica “Mis colecciones” del Editor de actividades siempre que sea posible. Intente evitar bibliotecas de programas de utilidad de código común desarrolladas por campo, incluidas como archivadores Java (archivos *.jar) en la vía de acceso de clases de WebSphere InterChange Server, ya que deberá migrarlas manualmente.

En los fragmentos de código Java que deban desarrollarse, es aconsejable que el código sea lo más simple y atomizado posible. El nivel de sofisticación del código Java debe ir por orden de scripts, implicación de evaluaciones básicas, operaciones y cálculos, formateo de datos, conversiones de tipos, etc. Si es necesaria una lógica de aplicación más extensa o sofisticada, considere la posibilidad de utilizar EJB ejecutados en WebSphere Application Server para encapsular la lógica y utilizar llamadas de servicio Web para invocarla desde WebSphere InterChange Server. Utilice bibliotecas JDK estándar en lugar de bibliotecas externas o de terceros, que deberían migrarse por separado. Reúna también toda la lógica relacionada dentro de un solo fragmento de código, y evite utilizar lógica en la que los contextos de transacción y conexión estén divididos en varios fragmentos de código. En operaciones de base de datos, por ejemplo, el código relacionado con la obtención de una conexión, el inicio y finalización de una transacción y la liberación de la conexión debe estar en un solo fragmento de código.

En general, asegúrese de que el código diseñado para interactuar con un Enterprise Information System (EIS) esté colocado dentro de adaptadores, y no dentro de correlaciones o plantillas de colaboración. Este es generalmente un procedimiento recomendado para el diseño de la arquitectura. Esto también le ayudará a evitar los prerrequisitos de bibliotecas de terceros y las consideraciones relacionadas dentro del código, como por ejemplo la gestión de conexiones y las posibles implementaciones de JNDI (Java Native Interface).

Haga que el código sea lo más seguro posible utilizando el manejo de excepciones adecuado. Haga también que el código sea compatible para la ejecución dentro de un entorno de servidor de aplicaciones

J2EE, aunque se esté ejecutando actualmente en un entorno J2SE. Respete los procedimientos de desarrollo de J2EE, como por ejemplo evitar variables estáticas, generación de hebras y E/S de disco. Estos son procedimientos recomendados excelentes que respetar en general, pero serán pertinentes en el caso de la portabilidad.

Programas de utilidad de código comunes

Como se ha indicado anteriormente, es aconsejable evitar el desarrollo de bibliotecas de programas de utilidad de código comunes para utilizarlas en los artefactos de integración dentro del entorno de WebSphere

InterChange Server. Cuando sea necesario reutilizar código en los artefactos de integración, es aconsejable utilizar la característica “Mis colecciones” de la herramienta Editor de actividades. Considere también la posibilidad de utilizar EJB ejecutados en WebSphere

Application Server para encapsular la lógica y utilizar llamadas de servicio de web para invocarlos desde WebSphere

InterChange Server. Aunque es posible que algunas bibliotecas de programas de utilidad de código comunes puedan funcionar correctamente en WebSphere

Process Server, el usuario será responsable de la migración de los programas de utilidad personalizados.

Agrupaciones de conexiones de base de datos

Las agrupaciones de conexiones de base de datos definidas por usuario son muy útiles en las correlaciones y plantillas de colaboración para búsquedas de datos simples y para la gestión de estados más sofisticados en las instancias de proceso. Una agrupación de conexiones de base de datos de WebSphere

InterChange Server se representará como un recurso JDBC estándar en WebSphere

Process Server y la función básica será la misma. Sin embargo, la forma de gestionar las conexiones y transacciones puede variar.

Para maximizar la portabilidad futura, evite mantener activas las transacciones de base de datos en los nodos de fragmento de código Java dentro de una plantilla de colaboración o una correlación. Por ejemplo, el código relacionado con la obtención de una conexión, el inicio y finalización de una transacción y la liberación de la conexión debe estar en un solo fragmento de código.

Objetos comerciales

Las recomendaciones principales para el desarrollo de objetos comerciales son utilizar sólo las herramientas suministradas para configurar artefactos, utilizar tipos y longitudes de datos explícitos para los atributos de datos y utilizar sólo las API documentadas.

Los objetos comerciales de WebSphere Process Server se basan en SDO (Service Data Objects), que utilizan atributos de datos fuertemente tipificados. En los objetos comerciales de WebSphere InterChange Server y adaptadores, los atributos de datos no están fuertemente tipificados y los usuarios especifican a veces tipos de datos serie (string) para atributos de datos no serie. Para evitar problemas en WebSphere Process Server, sea explícito en la especificación de tipos de datos.

Dado que los objetos comerciales de WebSphere Process Server pueden serializarse en tiempo de ejecución a medida que se pasan entre componentes, es importante ser explícito con las longitudes necesarias para los atributos de datos para minimizar la utilización de recursos del sistema. Por esta razón, no utilice la longitud máxima de 255 caracteres para un atributo de tipo serie, por ejemplo,

Tampoco especifique atributos de longitud cero que actualmente tengan el valor por omisión de 255 caracteres. En lugar de ello, especifique la longitud exacta necesaria para los atributos.

Los nombres de atributos de objeto comercial de WebSphere Process Server están sujetos a las normas de XSD NCName, por lo que no debe utilizar espacios ni signos ":" en los nombres de atributos de objeto comercial. Los nombres de atributo de objeto comercial con espacios o signos ":" no son válidos en WebSphere Process Server. Redenomine los atributos de objetos comerciales antes de la migración.

Si utiliza una matriz en un objeto comercial, no puede basarse en el orden de la matriz al indexar en la matriz en correlaciones y/o relaciones. La construcción que se migra a WebSphere Process Server no garantiza el orden del índice, particularmente cuando se suprimen entradas.

De nuevo es importante, como se ha indicado anteriormente, utilizar sólo la herramienta Diseñador de objetos comerciales para editar definiciones de objeto comercial, y utilizar sólo las API publicadas para los objetos comerciales dentro de los artefactos de integración.

Plantillas de colaboración

Muchas de las directrices descritas anteriormente se aplican al desarrollo de plantillas de colaboración.

Para asegurar que los procesos se describan adecuadamente con metadatos, utilice siempre la herramienta Diseñador de procesos para la creación y modificación de plantillas de colaboración, y evite editar directamente archivos de metadatos. Utilice la herramienta Editor de actividades siempre que sea posible para maximizar el uso de metadatos para describir la lógica necesaria.

Para minimizar la cantidad de trabajo manual que puede requerirse en la migración, utilice sólo las API documentadas en las plantillas de colaboración. Evite el uso de variables estáticas. En su lugar, utilice variables no estáticas y propiedades de colaboración para satisfacer los requisitos de la lógica comercial. Evite el uso de calificadores Java finales, transitorios y nativos en fragmentos de código Java. Estos no pueden aplicarse en los fragmentos de código Java de BPEL resultantes de la migración de plantillas de colaboración.

Para maximizar la portabilidad futura, evite utilizar llamadas de liberación de conexión explícitas y corchetes de transacción explícitos (es decir, compromisos explícitos & retrotracciones explícitas) para agrupaciones de conexiones de base de datos definidas por usuario. En su lugar, utilice la limpieza de conexiones implícita gestionada por contenedor y los corchetes de transacción implícitos. Evite también mantener activas las conexiones de sistema y transacciones en los nodos de fragmento de código Java dentro de una plantilla de colaboración. Esto se aplica a cualquier conexión a un sistema externo, así como a agrupaciones de conexiones de base de datos definidas por usuario. Como se ha indicado anteriormente, las operaciones con un EIS externo deben gestionarse dentro de un adaptador, y el código relacionado con la operación de base de datos debe encontrarse dentro de un fragmento de código. Esto puede ser necesario dentro de una colaboración que, cuando se representa como componente de proceso comercial BPEL, puede desplegarse selectivamente en forma de flujo interrumpible. En este caso, el proceso puede componerse de varias transacciones independientes, en la que se pasa sólo información de estado y de variable global entre las actividades. El contexto de cualquier conexión de sistema o transacción relacionada que abarque estas transacciones de proceso se perderá.

No utilice caracteres especiales en nombres de propiedad de plantillas de colaboración. Estos caracteres especiales no son válidos en los nombres de propiedad BPEL a los que se migrarán. Redenomine las propiedades para eliminar estos caracteres especiales antes de la migración para evitar errores sintácticos en el BPEL generado por la migración.

No haga referencia a variables utilizando "this." Por ejemplo, en lugar de "this.inputBusObj" utilice "inputBusObj"

Utilice ámbitos a nivel de clase en variables en lugar de variables a nivel de ámbito. El ámbito a nivel de escenario no se conserva durante la migración.

Inicialice todas las variables declaradas en fragmentos de código Java con un valor por omisión. Por ejemplo, "Object myObject = null;" Asegúrese de que todas las variables se inicialicen durante la declaración antes de la migración.

Asegúrese de que no haya sentencias de importación Java en las secciones modificables por el usuario de las plantillas de colaboración. En la definición de la plantilla de colaboración, utilice los campos de importación para especificar los paquetes Java que deben importarse.

Correlaciones

Muchas de las directrices descritas anteriormente para las plantillas de colaboración se aplican también a las correlaciones.

Para asegurarse de que las correlaciones se describan adecuadamente con metadatos, utilice siempre la herramienta Diseñador de correlaciones para la creación y modificación de correlaciones, y evite editar directamente los archivos de metadatos. Utilice la herramienta Editor de actividades siempre que sea posible para maximizar el uso de metadatos para describir la lógica necesaria.

Al hacer referencia a un objeto comercial hijo en una correlación, utilice una subcorrelación para los objetos comerciales hijos.

Evite utilizar código Java como "valor" en SET ya que no es válido en WebSphere Process Server. En lugar de ello, utilice constantes. Por ejemplo, si el valor de establecimiento es "xml version=" + "1.0" + " encoding=" + "UTF-8", no se validará en WebSphere Process Server. En lugar de ello, cámbielo por "xml version=1.0 encoding=UTF-8" antes de la migración.

Para minimizar la cantidad de trabajo manual que puede requerirse en la migración, utilice sólo las API documentadas en las plantillas de colaboración. Evite el uso de variables estáticas. En su lugar, utilice variables no estáticas y propiedades de colaboración para satisfacer los requisitos de la lógica comercial. Evite el uso de calificadores Java finales, transitorios y nativos en fragmentos de código Java.

Si utiliza una matriz en un objeto comercial, no puede basarse en el orden de la matriz al indexar en la matriz en correlaciones. La construcción que se migra a WebSphere Process Server no garantiza el orden del índice, particularmente cuando se suprimen entradas.

Para maximizar la portabilidad futura, evite utilizar llamadas de liberación de conexión explícitas y corchetes de transacción explícitos (es decir, compromisos explícitos & retrotracciones explícitas) para agrupaciones de conexiones de base de datos definidas por usuario. En su lugar, utilice la limpieza de conexiones implícita gestionada por contenedor y los corchetes de transacción implícitos. Evite también mantener activas las conexiones de sistema y transacciones en los nodos de fragmento de código Java en los límites de nodo de la transformación. Esto se aplica a cualquier conexión a un sistema externo, así como a agrupaciones de conexiones de base de datos definidas por usuario. Como se ha indicado anteriormente, las operaciones con un EIS externo deben gestionarse dentro de un adaptador, y el código relacionado con la operación de base de datos debe encontrarse dentro de un fragmento de código.

Relaciones

En las relaciones, recuerde que, aunque las definiciones de relación podrán migrarse para utilizarlas en WebSphere Process Server, el esquema de tabla y los datos de instancia de relación pueden reutilizarse en WebSphere Process Server, y también se compartirán de forma concurrente entre WebSphere InterChange Server y WebSphere Process Server.

Las recomendaciones clave para las relaciones son utilizar sólo las herramientas suministradas para configurar los componentes relacionados y utilizar sólo las API publicadas para las relaciones dentro de los artefactos de integración.

Es importante utilizar sólo la herramienta Diseñador de relaciones para editar definiciones de relación. Además, permita sólo a WebSphere InterChange Server configurar el esquema de relaciones, que se genera automáticamente al desplegar las definiciones de relación. No modifique el esquema de tabla de relaciones directamente con herramientas de base de datos o scripts SQL.

Asimismo, si debe modificar manualmente datos de instancia de relación dentro del esquema de tabla de relaciones, asegúrese de utilizar los recursos suministrados por el Diseñador de relaciones.

Como se ha indicado anteriormente, es importante utilizar sólo las API publicadas para las relaciones dentro de los artefactos de integración.

Clientes de infraestructura de acceso

No desarrolle clientes nuevos que adopten las API de interfaz IDL CORBA. Esto no estará soportado en WebSphere

Process Server.

Migrar a WebSphere Integration Developer desde WebSphere MQ Workflow

WebSphere Integration Developer ofrece las herramientas necesarias para migrar desde WebSphere MQ Workflow.

El asistente de migración permite convertir las definiciones FDL de los procesos de negocio que ha exportado del componente de tiempo de construcción (buildtime) de WebSphere MQ Workflow en los artefactos correspondientes del coreógrafo de procesos de negocio (BPC). Los artefactos de Coreógrafo de procesos de negocio generados comprenden las definiciones de esquemas XML de los objetos de negocio, las definiciones WSDL, BPEL y las definiciones TEL.

La herramienta de conversión requiere una definición FDL semánticamente completa de un modelo de proceso exportado desde WebSphere MQ Workflow Buildtime con la opción **export deep**. Esta opción garantiza la inclusión de todas las especificaciones de datos, programas y subprocesos necesarias. Asimismo, asegúrese de que también se seleccionan todas las definiciones de servidores de ejecución de procesos definidos por el usuario (UPES) con referencias en el modelo de proceso de WebSphere MQ Workflow al exportar FDL desde WebSphere MQ Workflow Buildtime.

Nota: El asistente de migración no cubre la migración de los elementos siguientes:

- Instancias de tiempo de ejecución de WebSphere MQ Workflow
- Aplicaciones de programa invocadas por un agente de ejecución de programas (PEA) de WebSphere MQ Workflow o un servidor de ejecución de programas (PES para z/OS) de WebSphere MQ Workflow

Tareas de preparación para la migración de WebSphere MQ Workflow

Antes de migrar a WebSphere Integration Developer desde WebSphere MQ Workflow, primero debe asegurarse de que ha preparado correctamente el entorno.

El ámbito y la integridad de la correlación dependerán del seguimiento de las siguientes directrices sobre migración:

- Compruebe que las actividades de programa FDL están asociadas a un servidor de ejecución de programas definido por el usuario (UPES) si no son actividades de **personal** puras.

- Asegúrese de que las asignaciones de personal de las actividades de programa de WebSphere MQ Workflow se ajusten a los **verbos de personal** predeterminados de TEL.
- Utilice nombres cortos y sencillos para facilitar la lectura de los modelos de procesos migrados. Tenga presente que los nombres FDL pueden ser nombres BPEL no permitidos. El asistente de migración le ayudará a convertir automáticamente los nombres FDL en nombres BPEL válidos.

El asistente de migración generará construcciones del editor de procesos de negocio sintácticamente correctas incluso en el caso de las construcciones de WebSphere MQ Workflow que no se pueden migrar (actividades de programa PEA o PES, algunas asignaciones de personal dinámicas, etc.), que deberán adaptarse manualmente a los artefactos del editor de procesos de negocio ejecutables.

En la tabla siguiente figuran las reglas de correlación aplicadas:

Tabla 2. Reglas de correlación

WebSphere MQ Workflow	Coreógrafo de procesos de negocio
Proceso	<i>Proceso con modalidad de ejecución: longRunning; enlaces de socio para interfaces de proceso entrantes y salientes</i>
Origen y llegada	<i>Variables para la entrada y la salida de procesos; actividad de recepción y actividad de respuesta</i>
Actividad de programa	Actividad de <i>invocación</i>
Actividad de proceso	Actividad de <i>invocación</i>
Actividad vacía	Actividad <i>vacía</i>
Bloque	<i>Ámbito con actividades BPEL incorporadas</i>
Condición de salida de actividad	Actividad <i>while</i> (delimitando la actividad)
Condición de inicio de actividad	<i>Condición de unión</i> de actividad
Asignación de personal de actividad	Actividad de <i>tarea manual</i>
Contenedor de entrada y contenedor de salida de actividad	<i>Variables</i> empleadas para especificar la entrada/salida de la actividad de <i>invocación</i>
Conector de control; condición de transición	<i>Enlace; condición de transición</i>
Conector de datos	Actividad de <i>asignación</i>
Contenedor de datos global	<i>Variable</i>

Inicialmente se recomienda probar el proceso de migración con proyectos pequeños, si es posible. El asistente de migración simplificará la conversión de los modelos de proceso de WebSphere MQ Workflow en los modelos de proceso del editor de procesos de negocio, pero tenga presente que los procesos no pueden correlacionarse de forma unívoca ya que se crea un nuevo modelo de programación. Los ámbitos semánticos de los lenguajes de especificación de procesos subyacentes (FDL y BPEL) comparten una área de intersección, pero no se solapan por completo. De lo contrario, no podría esperar ninguna ventaja adicional del editor de procesos de negocio. Los servicios web constituyen una nueva y prometedora tecnología que pretende sustituir las soluciones obsoletas por otras nuevas.

Como norma general, siempre deberá revisar y posiblemente modificar los artefactos generados. Puede que tenga que llevar a cabo algunas tareas adicionales para que la migración sea satisfactoria o logre completarse.

Migrar WebSphere MQ Workflow utilizando el asistente de migración

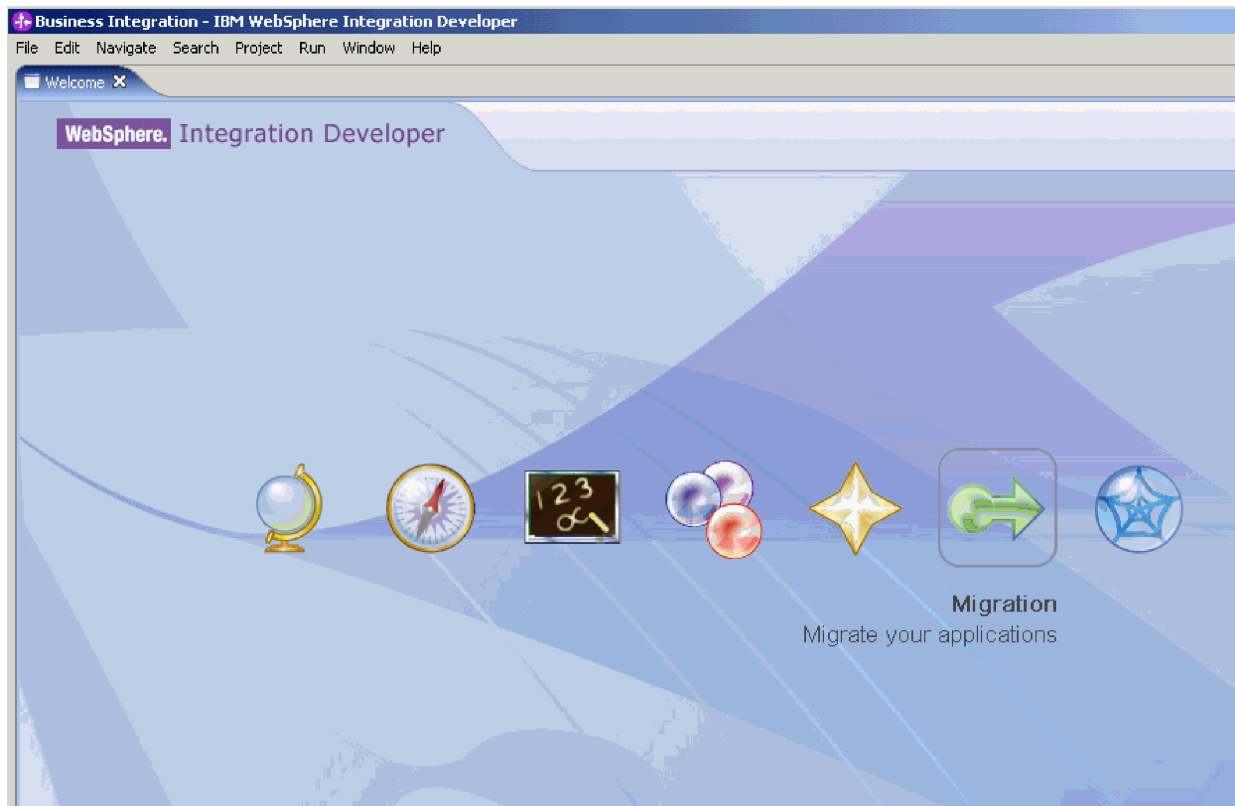
El asistente de migración permite convertir las definiciones FDL de los procesos de negocio que ha exportado del componente de tiempo de construcción (buildtime) de WebSphere MQ Workflow en los artefactos correspondientes del coreógrafo de procesos de negocio (BPC). Los artefactos de Coreógrafo de procesos de negocio generados comprenden las definiciones de esquemas XML de los objetos de negocio, las definiciones WSDL, BPEL y las definiciones TEL.

Nota: El asistente de migración no cubre la migración de los elementos siguientes:

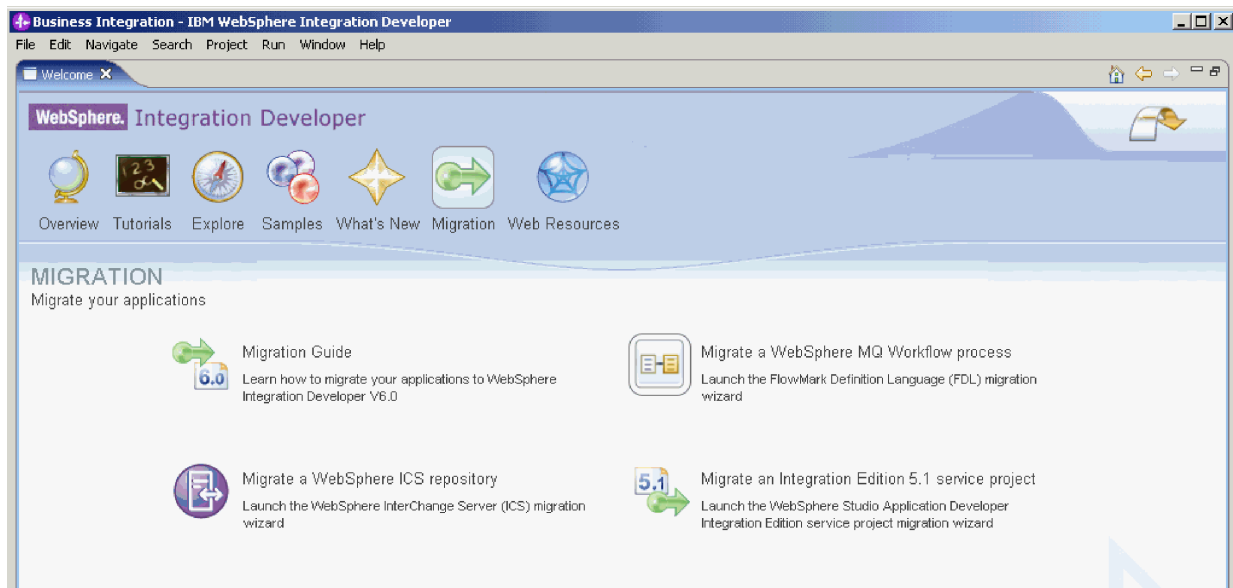
- Instancias de tiempo de ejecución de WebSphere MQ Workflow
- Aplicaciones de programa invocadas por un agente de ejecución de programas (PEA) de WebSphere MQ Workflow o un servidor de ejecución de programas (PES para z/OS) de WebSphere MQ Workflow

Siga estos pasos para utilizar el asistente de migración a fin de migrar los artefactos de WebSphere MQ Workflow:

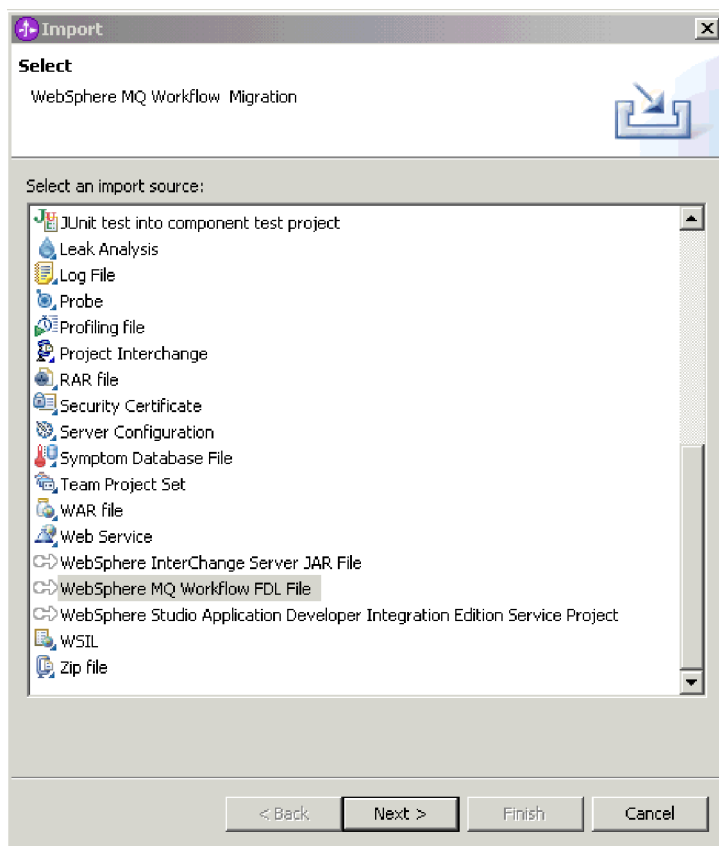
1. En la página de bienvenida, pulse  para abrir la página Migración.



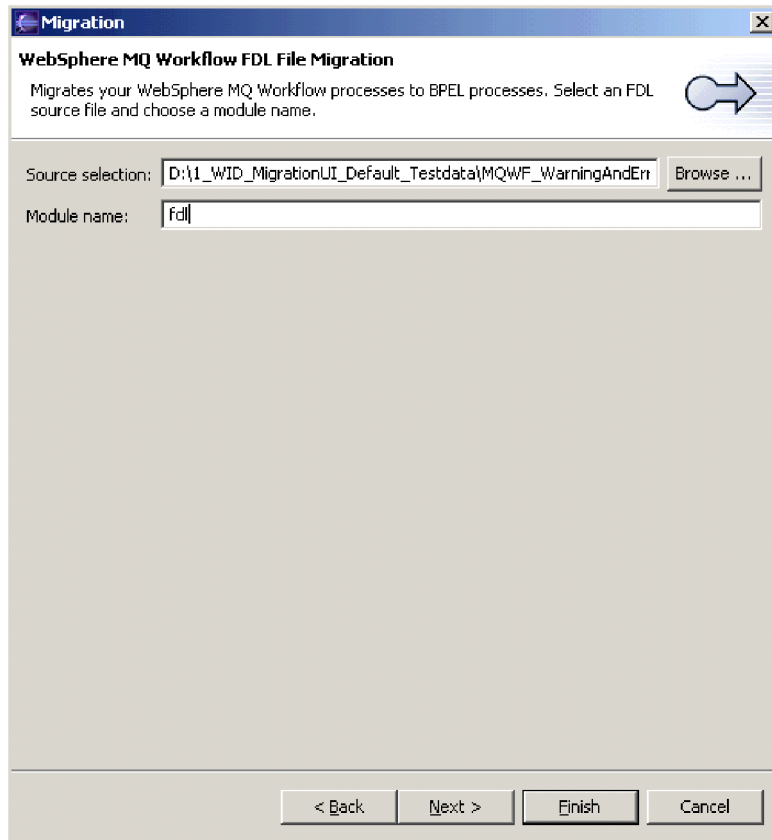
2. En la página Migración, seleccione la opción para migrar un proceso de WebSphere MQ Workflow:



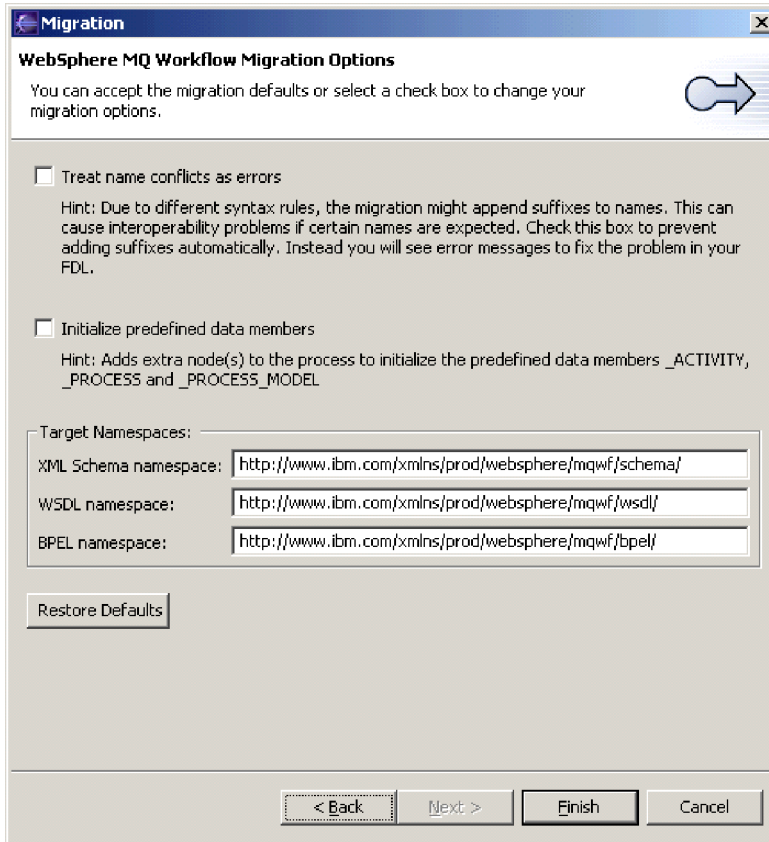
(Nota: también puede abrir el asistente de migración pulsando **Archivo** → **Importar** → **Archivo FDL de WebSphere MQ Workflow**:



3. Se abre el asistente de migración. Especifique el nombre del archivo .fdl en el campo **Selección de origen** pulsando el botón **Examinar** y desplazándose hasta el archivo. Especifique el nombre del módulo en el campo correspondiente. Pulse **Siguiente**.



4. Se abre la página Opciones de migración. Puede aceptar los valores predeterminados de la migración o marcar un recuadro de selección para cambiar la opción. Si marca el recuadro de selección **Tratar conflictos de nombres como errores**, puede evitar la adición automática de sufijos que puede provocar errores de interoperatividad. El recuadro de selección **Inicializar miembros de datos predefinidos** añade nodos adicionales al proceso para inicializar los miembros de datos predefinidos.



The image shows a Windows-style dialog box titled "Migration" with a close button (X) in the top right corner. The main title is "WebSphere MQ Workflow Migration Options". Below the title, there is a text box that says "You can accept the migration defaults or select a check box to change your migration options." To the right of this text is a blue arrow icon pointing to the right. There are two checkboxes with associated hints. The first checkbox is labeled "Treat name conflicts as errors" and its hint says: "Hint: Due to different syntax rules, the migration might append suffixes to names. This can cause interoperability problems if certain names are expected. Check this box to prevent adding suffixes automatically. Instead you will see error messages to fix the problem in your FDL." The second checkbox is labeled "Initialize predefined data members" and its hint says: "Hint: Adds extra node(s) to the process to initialize the predefined data members _ACTIVITY, _PROCESS and _PROCESS_MODEL". Below these checkboxes is a section titled "Target Namespaces:" which contains three text input fields. The first field is labeled "XML Schema namespace:" and contains the value "http://www.ibm.com/xmlns/prod/websphere/mqwf/schema/". The second field is labeled "WSDL namespace:" and contains the value "http://www.ibm.com/xmlns/prod/websphere/mqwf/wsdl/". The third field is labeled "BPEL namespace:" and contains the value "http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/". Below these fields is a button labeled "Restore Defaults". At the bottom of the dialog box are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Migration

WebSphere MQ Workflow Migration Options

You can accept the migration defaults or select a check box to change your migration options.

☐ Treat name conflicts as errors

Hint: Due to different syntax rules, the migration might append suffixes to names. This can cause interoperability problems if certain names are expected. Check this box to prevent adding suffixes automatically. Instead you will see error messages to fix the problem in your FDL.

☐ Initialize predefined data members

Hint: Adds extra node(s) to the process to initialize the predefined data members _ACTIVITY, _PROCESS and _PROCESS_MODEL

Target Namespaces:

XML Schema namespace:

WSDL namespace:

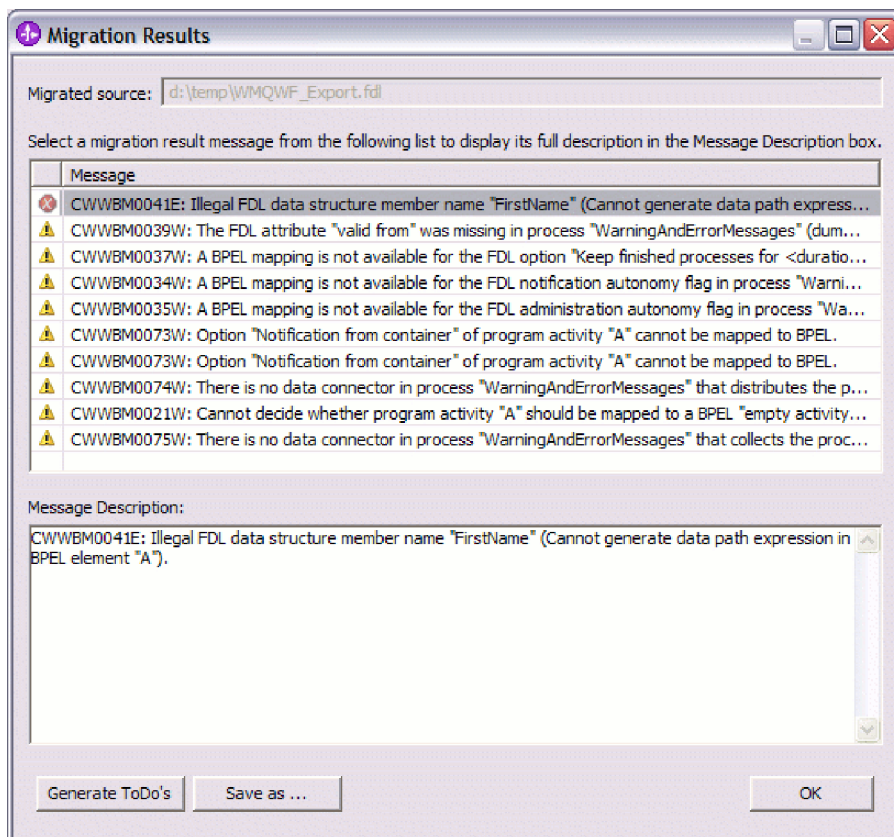
BPEL namespace:

Pulse **Finalizar**.

Verificar la migración de WebSphere MQ Workflow

Una vez finalizado correctamente el asistente de migración, se visualiza una lista de mensajes de error, aviso e informativos. Puede emplear estos mensajes para verificar la migración de WebSphere MQ Workflow.

Al finalizar el asistente de migración se muestra la página siguiente:



En la ventana Resultados de la migración, puede ver una lista de mensajes relacionados con la migración. Pulse el mensaje para ver una descripción completa de sus detalles. Si es necesario, a partir de esta lista de mensajes, puede crear una lista con los aspectos que deberá corregir pulsando el botón **Generar ToDo**.

Limitaciones del proceso de migración (desde WebSphere MQ Workflow)

Existe una serie de limitaciones en el proceso de migración de WebSphere MQ Workflow.

La migración de FDL generará actividades de invocación para actividades de UPES y los WSDL correspondientes. Sin embargo, el entorno de tiempo de ejecución difiere significativamente entre IBM WebSphere MQ Workflow y IBM WebSphere Process Server en términos de las técnicas utilizadas para correlacionar los mensajes de invocación y sus respuestas. Por tanto, el código BPEL generado no puede ejecutarse satisfactoriamente si no hay una capa de compatibilidad de UPES disponible.

Migrar artefactos origen a WebSphere Integration Developer desde WebSphere Studio Application Developer Integration Edition

Es posible migrar artefactos origen de WebSphere Studio Application Developer Integration Edition a WebSphere Integration Developer. Al migrar los artefactos origen de una aplicación, estos se migran al nuevo modelo de programación de WebSphere Integration Developer para que se puedan utilizar las nuevas funciones. A continuación, la aplicación puede desplegarse de nuevo e instalarse en el servidor WebSphere Integration Developer Versión 6.0.

Muchas características disponibles en WebSphere Business Integration Server Foundation 5.1 han pasado a la base WebSphere Application Server 6.0. Consulte este sitio para obtener consejos sobre la migración de estas características: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/rins_migratepme.html

Para migrar por completo un proyecto de servicio de WebSphere Studio Application Developer Integration Edition, deben realizarse dos tareas fundamentales:

1. Utilice el asistente Migración para migrar automáticamente los artefactos al proyecto Módulo de Integración empresarial.
2. Utilice WebSphere Integration Developer para completar manualmente la migración. Esto implica el arreglo del código Java que no se haya podido migrar automáticamente y volver a conectar los artefactos migrados.

Nota: La migración en tiempo de ejecución (vía de acceso de actualización) no se proporciona con WebSphere Process Server 6.0.0, por lo tanto, esta vía de acceso de migración de artefactos fuente será la única opción para migrar proyectos de servicio de WebSphere Studio Integration Edition en 6.0.0.

Vías de migración soportadas para migrar artefactos origen

Antes de empezar a migrar artefactos origen de WebSphere Studio Application Developer Integration Edition, examine las vías de migración que admite WebSphere Integration Developer.

El asistente de migración permite migrar un proyecto de servicio de WebSphere Studio Application Developer Integration Edition Versión 5.1 (o posterior) cada vez. *No* migrará toda una área de trabajo.

El asistente de migración no migra binarios de aplicaciones; solamente migra los artefactos origen que se encuentran en un proyecto de servicio de WebSphere Studio Application Developer Integration Edition.

Preparar los artefactos origen para la migración

Antes de migrar los artefactos origen a WebSphere Integration Developer desde WebSphere Studio Application Developer Integration Edition, primero debe asegurarse de que ha preparado correctamente el entorno.

El procedimiento siguiente describe cómo preparar el entorno antes de migrar artefactos origen a WebSphere

Integration Developer desde WebSphere

Studio Application Developer Integration Edition:

1. Asegúrese de tener una copia de seguridad de todo el área de trabajo de 5.1 antes de empezar la migración.
2. Revise la sección de migración del Centro de información de Rational Application Developer para determinar la mejor forma de migrar los proyectos no específicos de WBI que haya en el área de trabajo: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.rad.migration.doc/topics/tmigratefrom51x.html>
3. Revise la sección Servicio Web del Centro de información de Rational Application Developer para obtener más información acerca de la función de servicio Web proporcionada por Rational Application Developer: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.webservices.rad.nav.doc/developingweb.html>
4. Asegúrese de tener todas las características adecuadas de WebSphere Integration Developer habilitadas. Si no tiene estas características habilitadas, no verá las opciones de menú que se tratarán a continuación. Para habilitar las características importantes:
 - En WebSphere Integration Developer, vaya al elemento de menú **Ventana** y seleccione **Preferencias**.
 - Vaya a **Entorno de trabajo** y seleccione la categoría **Posibilidades**.
 - Seleccione todas las características situadas bajo las categorías siguientes:
 - J2EE avanzado
 - Java de empresa
 - Desarrollador de integración
 - Desarrollador Java
 - Desarrollador Web (típico)

- Desarrollador de servicios Web
 - Desarrollador XML
 - Pulse **Aceptar**.
5. Utilice un nuevo directorio de área de trabajo para WebSphere Integration Developer. No se recomienda abrir WebSphere Integration Developer en una antigua área de trabajo de WebSphere Studio Application Developer Integration Edition que contenga proyectos de servicio, ya que esos proyectos primero deben migrarse a un formato legible por WebSphere Integration Developer. A continuación se indica el procedimiento recomendado para ello:
- a. Copie todos los proyectos que no son de servicio de la antigua área de trabajo en la nueva. No copie los proyectos EJB, Web y EAR 5.1 creados al generarse el código de despliegue para un proyecto de servicio 5.1. El nuevo código de despliegue de 6.0 se volverá a generar automáticamente al construir el módulo de integración empresarial.
 - b. Abra WebSphere Integration Developer en el área de trabajo en blanco e importe todos los proyectos que no son de servicio pulsando en **Archivo** → **Importar** → **Proyecto existente en área de trabajo** y seleccione los proyectos que ha copiado en la nueva área de trabajo.
 - Si el proyecto es un proyecto J2EE, debe migrarlo al nivel 1.4 utilizando el asistente Migración de Rational Application Developer:
 - 1) Pulse el proyecto con el botón derecho y seleccione **Migración** → **Asistente Migración de J2EE...**
 - 2) Revise las sentencias de aviso en la primera página y pulse **Siguiente**.
 - 3) Asegúrese de que el **Proyecto J2EE** está seleccionado en la lista Proyectos. Deje seleccionados **Migrar estructura de proyecto** y **Migrar nivel de especificación J2EE**. Elija **J2EE Versión 1.4** y **Servidor destino WebSphere Process Server v6.0**.
 - 4) Seleccione cualesquiera otras opciones que puedan ser adecuadas para el proyecto J2EE y pulse **Finalizar**. Si este paso se lleva a cabo adecuadamente, verá un mensaje indicando que **La migración ha finalizado satisfactoriamente**.
 - 5) Si hay errores en el proyecto J2EE después de la migración, debe eliminar todas las entradas de vía de acceso de clases que hagan referencia a los archivos v5 .jar o a las bibliotecas y añadir las bibliotecas **Biblioteca del sistema JRE** y **Destino del servidor WPS** a la vía de acceso de clases en su lugar (se trata a continuación.) Esto debe resolver la mayoría, si no todos los errores.
 - Para los proyectos EJB de WebSphere Business Integration con Mensajería ampliada (CMM) o Persistencia contenida por contenedor sobre cualquier cosa (CMP/A), los archivos de descriptor de Extensión Jar de EJB de IBM deben migrarse después de importar el proyecto 5.1 en el área de trabajo 6.0. Consulte "Migrar proyectos EJB de WebSphere Business Integration" para obtener más información.
 - Corrija la vía de acceso de clases para cada uno de los proyectos que no son de servicio importados en el área de trabajo. Para añadir las bibliotecas JRE y WebSphere Process Server a la vía de acceso de clases, pulse con el botón derecho en el proyecto importado y seleccione **Propiedades**. Vaya a la entrada **Vía de construcción Java** y seleccione la pestaña **Bibliotecas**. Después, haga lo siguiente:
 - 1) Seleccione **Añadir biblioteca** → **Biblioteca del sistema JRE** → **JRE alternativo - JRE de Servidor WPS v6.0** → **Finalizar**.
 - 2) A continuación, seleccione **Añadir biblioteca** → **Destino del servidor WPS** → **Configurar vía de acceso de clases del servidor wps** → **Finalizar**.
6. Para obtener un resultado óptimo, antes de ejecutar el asistente de migración diríjase al elemento de menú **Proyecto** y compruebe que el valor **Construcción automática** no está seleccionado. WebSphere Integration Developer difiere de WebSphere Studio Application Developer Integration Edition en que las opciones de despliegue de servicio se especifican en tiempo de diseño. Cuando se construye el proyecto, el código de despliegue se actualiza automáticamente en el EJB generado y los proyectos Web de modo que ya no hay ninguna opción para ejecutar manualmente **Generar código de despliegue**.

7. Para migrar por completo los archivos .bpel dentro de un proyecto de servicio, debe comprobar que todos los archivos .wsdl y .xsd a los que hacen referencia los archivos .bpel se pueden resolver en un proyecto de integración empresarial de la nueva área de trabajo:
 - Si los archivos .wsdl o .xsd están en el mismo proyecto de servicio que el archivo .bpel, no es precisa ninguna acción adicional.
 - Si los archivos .wsdl y/o .xsd están en un proyecto de servicio distinto del que está migrando, los artefactos 5.1 deben reorganizarse utilizando WebSphere Studio Application Developer Integration Edition antes de la migración. La razón de esto es que los proyectos de módulo de integración empresarial no pueden compartir artefactos. Estas son las dos opciones para reorganizar los artefactos 5.1:
 - En WebSphere Studio Application Developer Integration Edition, cree un proyecto Java nuevo que albergará todos los artefactos comunes. Ponga todos los archivos .wsdl y .xsd compartidos por más de un proyecto de servicio en este proyecto Java nuevo. Añada una dependencia de este proyecto Java nuevo para todos los proyectos de servicio que utilicen estos artefactos comunes. En WebSphere Integration Developer, cree un proyecto de biblioteca de integración empresarial con el mismo nombre que el proyecto Java compartido 5.1 antes de migrar cualquiera de los proyectos de servicio. Copie manualmente los archivos .wsdl y .xsd antiguos del proyecto Java compartido de 5.1 en esta carpeta de proyecto de biblioteca BI. Esto debe hacerse antes de migrar los proyectos de servicio de BPEL.
 - Otra opción consiste en guardar una copia local de estos artefactos .wsdl y .xsd compartidos en cada proyecto de servicio de modo que no haya dependencias entre proyectos de servicio.
 - Si los archivos .wsdl o .xsd están en cualquier otro tipo de proyecto (normalmente otros proyectos Java), debe crear un proyecto de biblioteca de integración empresarial con el mismo nombre que el del proyecto 5.1. También debe configurar la vía de acceso de clases del proyecto de biblioteca nuevo añadiendo las entradas del proyecto Java 5.1 si existen. Este tipo de proyecto resulta de utilidad para almacenar artefactos compartidos.

Ahora ya está preparado para iniciar el proceso de migración.

Migrar proyectos de servicio utilizando el asistente de migración de WebSphere Integration Developer

El asistente de migración de WebSphere Integration Developer permite migrar proyectos de servicio.

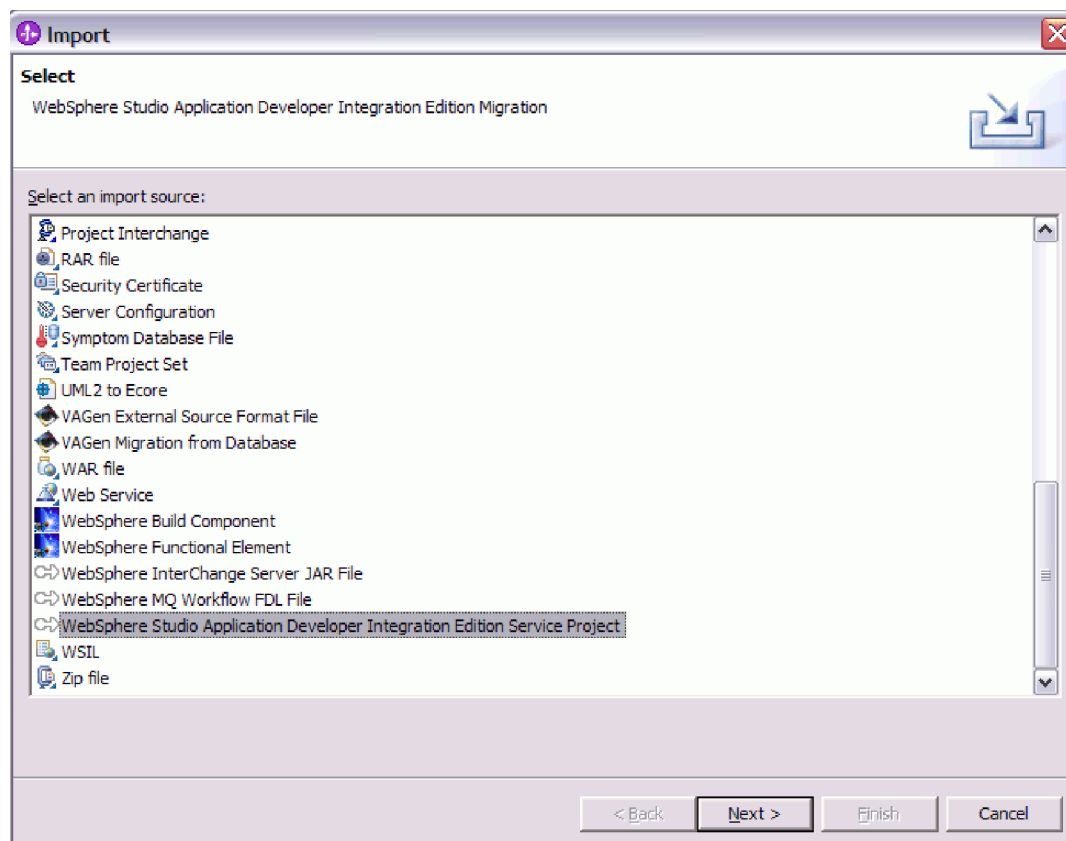
El asistente de migración lleva a cabo las acciones siguientes:

1. Crea un módulo de integración de negocio nuevo (el nombre del módulo lo define usted)
2. Migra las entradas de vía de acceso de clases del proyecto de servicio al nuevo módulo.
3. Copia todos los artefactos origen de WebSphere Business Integration Server Foundation del proyecto origen seleccionado a este módulo.
4. Migra las extensiones BPEL en archivos WSDL.
5. Migra los procesos de negocio (archivos .bpel) de BPEL4WS, versión 1.1 al nivel nuevo soportado por el servidor de procesos de WebSphere que está incorporado en BPEL4WS, versión 1.1 con las prestaciones principales de la próxima especificación WS-BPEL, versión 2.0
6. Crea un componente SCA para cada proceso .bpel.
7. Genera un archivo .mon de supervisión para cada proceso BPEL a fin de conservar el procedimiento de supervisión predeterminado de WebSphere Studio Application Developer Integration Edition (si es necesario).

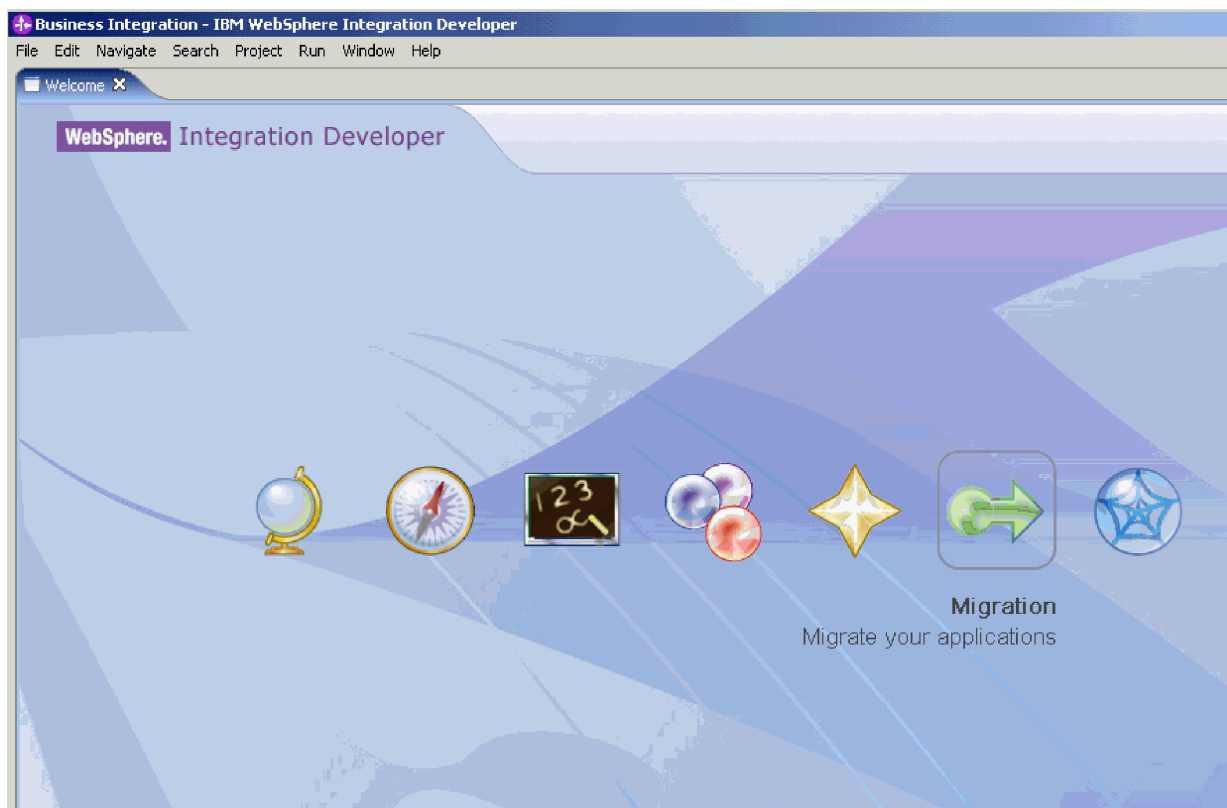
Siga estos pasos para migrar proyectos de servicio utilizando el asistente de migración de WebSphere

Integration Developer:

1. Invoque el asistente seleccionando **Archivo → Importar → Proyecto de servicio de WebSphere Studio Application Developer Integration Edition**.



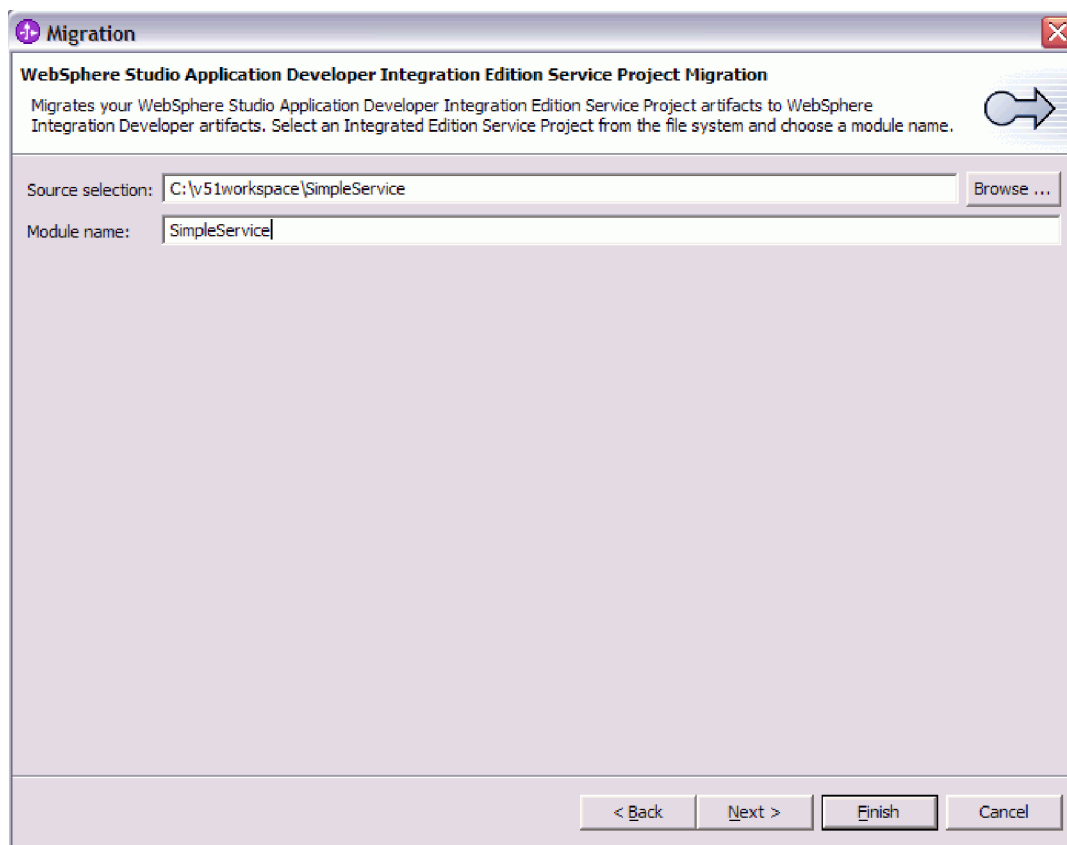
También puede abrir el asistente Migración desde la página Bienvenida pulsando  :



En la página Migración, seleccione la opción Migrar un proyecto de servicio de Integration Edition 5.1:

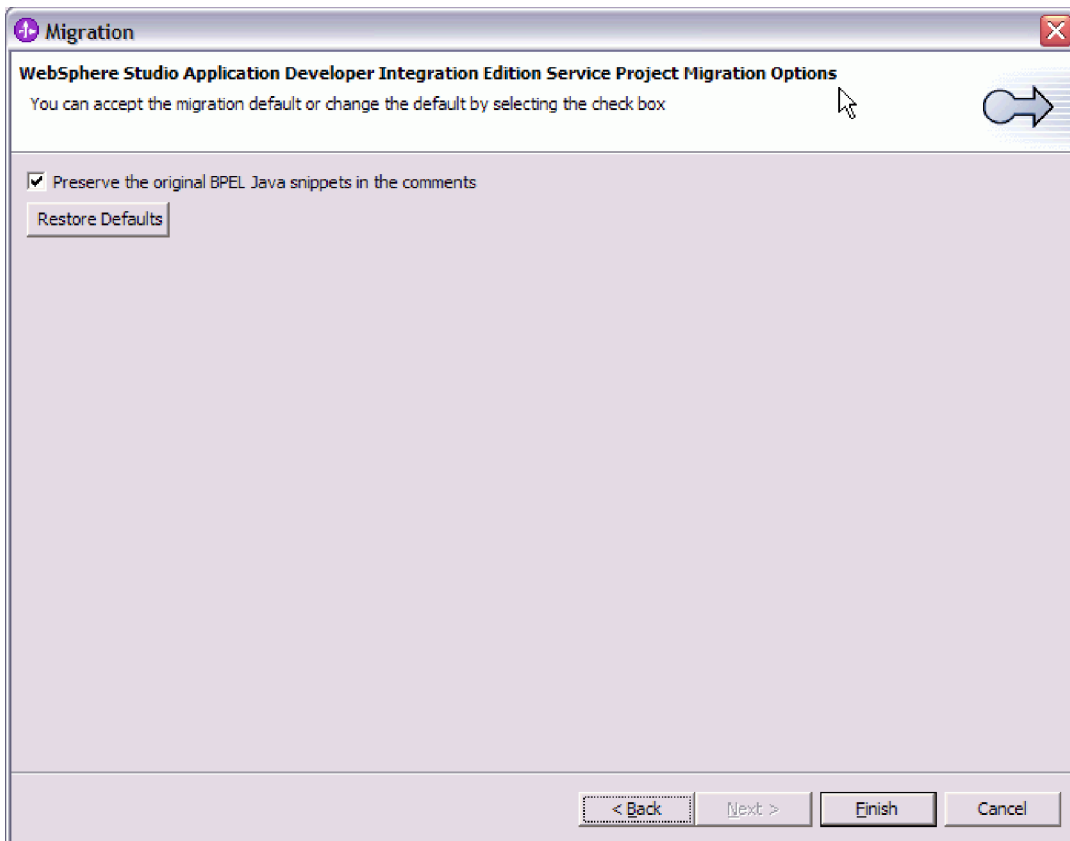


2. Se abre el asistente de migración. Especifique la vía de acceso para la Selección origen o pulse el botón **Examinar** para buscarla. Especifique también el nombre de Módulo de la ubicación del Proyecto de servicio de WebSphere Studio Application Developer Integration Edition a migrar:



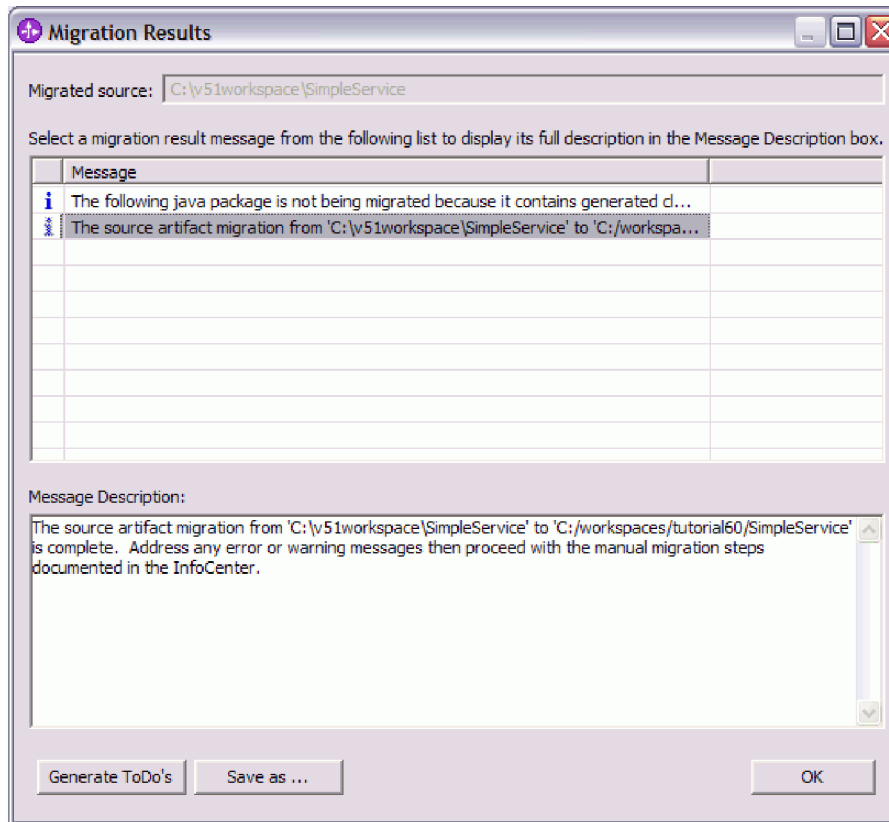
Note: se recomienda seleccionar el nombre del proyecto de servicio como nombre de módulo ya que si hay otros proyectos en el espacio de trabajo de WebSphere Studio Application Developer Integration Edition que dependen de este proyecto, no será necesario actualizar las vías de acceso de clases de los proyectos dependientes tras importarlos en WebSphere Integration Developer.

3. En las opciones de migración, marque el recuadro de selección Preservar fragmentos de código Java:



Pulse **Finalizar**.

4. Una vez finalizado el proceso de migración, se abre la ventana Resultados de la migración:



Se generará automáticamente un archivo de anotaciones que contenga estos mensajes de migración en la carpeta .metadata del área de trabajo de 6.0. El archivo de anotaciones se llamará ".log".

Una vez finalizado el asistente de migración, construya el módulo de integración empresarial que se ha creado e intente resolver los errores de construcción que se produzcan. Examine todos los archivos .bpel migrados: compruebe que se han migrado por completo y que se pueden abrir en el editor BPEL de WebSphere

Integration Developer. Hay algunos fragmentos de código Java

BPEL que no se pueden migrar automáticamente. Si aprecia errores en los fragmentos de código Java

BPEL, consulte la sección "Migrar al modelo de programación de SCA" para conocer los pasos necesarios para corregir los errores. Asimismo, si ha utilizado el asistente de migración para migrar un proyecto de servicio a un módulo de integración empresarial, abra el editor de dependencias de módulos para asegurarse de que las dependencias se han establecido correctamente. Para ello, vaya a la perspectiva Integración empresarial y pulse dos veces en el proyecto de módulo de integración empresarial. En esta ubicación puede añadir dependencias a los proyectos de biblioteca de integración empresarial, proyectos Java

y proyectos J2EE.

Migrar manualmente la aplicación

Una vez que el asistente de migración ha migrado correctamente los artefactos al nuevo módulo de integración empresarial, los artefactos deben conectarse entre sí para crear una aplicación conforme al modelo de SCA.

1. Abra WebSphere Integration Developer y vaya a la perspectiva Integración empresarial. Verá los módulos que ha creado el asistente de migración (un módulo por proyecto de servicio migrado). El primer artefacto que aparece bajo el proyecto de módulo es el archivo de ensamblaje del módulo (denominado igual que el módulo).
2. Pulse dos veces en el archivo de ensamblaje para abrirlo en el editor de ensamblajes donde los componentes SCA se pueden crear y conectar entre sí para obtener funciones similares a las de la aplicación de la versión 5.1. Si había procesos BPEL en el proyecto de servicio de WebSphere Studio Application Developer Integration Edition, el asistente de migración habrá creado componentes SCA predeterminados para cada uno de esos procesos y estarán en el editor de ensamblajes.
3. Seleccione un componente y vaya a la vista Propiedades donde aparecen y pueden editarse las propiedades de descripción, detalles e implementación.

La información siguiente describe con mayor detalle cómo volver a conectar manualmente la aplicación mediante las herramientas disponibles en WebSphere Integration Developer:

Crear componentes SCA e Importaciones SCA para los servicios en la aplicación para volver a conectar:

Será necesario volver a conectar algo para todos los proyectos después de la migración para volver a conectar los servicios tal y como estaban en 5.1. Por ejemplo, es necesario volver a conectar todos los procesos de negocio a los socios de negocio. Es necesario crear un componente o una importación SCA para todos los demás tipos de servicio. Para los proyectos de servicio de WebSphere Studio Application Developer Integration Edition que interactúan con sistemas o entidades externas al proyecto, puede crearse una Importación SCA para que el proyecto migrado acceda a esas entidades como servicios de acuerdo con el modelo SCA.

Para los proyectos de servicio de WebSphere Studio Application Developer Integration Edition que interactúan con entidades en el proyecto (por ejemplo, un proceso de negocio, un servicio transformador o una clase Java), puede crearse una Importación SCA para que el proyecto migrado acceda a esas entidades como servicios de acuerdo con el modelo SCA.

Las secciones siguientes proporcionan detalles acerca de la Importación SCA o los Componentes SCA que hay que crear basándose en el tipo de servicio que debe migrarse:

Migrar un servicio Java:

Puede migrar un servicio Java a un componente Java SCA.

Si el proyecto de servicio WebSphere

Studio Application Developer Integration Edition era dependiente de otros proyectos Java

, copie los proyectos existentes en el directorio nuevo de área de trabajo e impórtelos en WebSphere

Integration Developer utilizando el asistente **Archivo** → **Importar** → **Proyecto existente en el espacio de trabajo**.

En WebSphere Studio Application Developer Integration Edition al generar un nuevo servicio Java a partir de una clase Java existente, se proporcionaban las opciones siguientes:

- Crear esquemas XSD para tipos de datos complejos:
 - Dentro del archivo WSDL de interfaz
 - Como un nuevo archivo para cada tipos de datos
- Dar soporte a la función de manejo de errores:
 - Generar falta

- No generar falta
- Otros detalles sobre el servicio por generar como los nombres de enlace y servicio

Hay muchos componentes nuevos en 6.0 que ofrecen nuevas funciones como por ejemplo correlación de datos, mediación de interfaces, máquinas de estado de negocio, selectores, reglas de negocio y más. Primero debe determinar si uno de estos tipos de componente nuevos pueden sustituir el componente Java personalizado. Si eso no es posible, siga la vía de acceso de migración que se describe a continuación.

Importe el proyecto de servicio utilizando el asistente Migración. Esto resultará en la creación de un módulo de integración de negocio con los mensajes, tipos de puerto, enlaces y servicios de WSDL generados en WebSphere Studio Application Developer Integration Edition.

En la perspectiva Integración empresarial, expanda el módulo para ver el contenido correspondiente. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)

Tiene las opciones siguientes:

Pros y contras de las opciones de reconexión de servicios Java:

Hay pros y contras para las opciones de reconexión de servicios Java.

La lista siguiente describe las opciones y los pros y contras de cada una:

- La primera opción proporciona un mejor rendimiento en tiempo de ejecución porque invocar un servicio Web es más lento que invocar un componente Java.
- La primera opción puede propagar el contexto mientras que una invocación de servicio Web no propaga el contexto de la misma forma.
- La segunda opción no implica la creación de código personalizado.
- La segunda opción no es aplicable para algunas definiciones de interfaz Java ya que la generación de un servicio Java tiene limitaciones. Consulte la documentación de Rational Application Developer aquí: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.doc/ref/rlimit.html>
- La segunda opción puede implicar un cambio de interfaz y por lo tanto un cambio para el consumidor de SCA.
- La segunda opción requiere la instalación de un servidor WebSphere Process Server 6.0 y se ha configurado para trabajar con WebSphere Integration Developer. Para ver los tiempos de ejecución instalados que están configurados para trabajar con WebSphere Integration Developer, vaya a **Ventana** → **Preferencias** → **Server** → **Tiempos de ejecución instalados**, seleccione la entrada **WebSphere Process Server v6.0** si es que existe y asegúrese de que señala la ubicación en la que está instalado el producto. Asegúrese de que esta entrada está marcada si el servidor existe y de que no lo está si el servidor no está instalado. También puede pulsar el botón **Añadir...** si desea añadir otro servidor.
- Si el componente Java se construyó en WebSphere Studio Application Developer Integration Edition utilizando el procedimiento de arriba abajo en el que se genera el esqueleto Java a partir de un WSDL, los parámetros in y out de esta clase Java serán probablemente una subclase de WSIFFormatPartImpl. En este caso, elija la opción 1 para generar un esqueleto Java de estilo SCA nuevo del WSDL/XSD original o la opción 2 para generar un esqueleto Java genérico nuevo (no dependiente de las API WSIF o DataObject) de la interfaz WSDL original.

Crear el componente Java personalizado: opción 1:

La técnica de migración recomendada consiste en utilizar el tipo de componente Java WebSphere Integration Developer que permite representar el servicio Java como un componente SCA. Durante la migración, debe escribir código Java personalizado para convertir entre el estilo de interfaz Java SCA y el estilo de interfaz del componente Java existente.

Para crear el componente Java

personalizado:

1. En el proyecto del módulo, expanda **Interfaces** y seleccione la interfaz WSDL generada para esta clase Java en WebSphere Studio Application Developer Integration.
2. Arrastre y suelte esta interfaz en el Editor de ensamblaje. Aparecerá un diálogo en el que se le pedirá que seleccione el tipo de componente a crear. Seleccione **Componente sin tipo de implementación** y pulse **Aceptar**.
3. Aparecerá un componente en el diagrama de ensamblaje. Selecciónelo y vaya a la vista **Propiedades**.
4. En la pestaña **Descripción** puede cambiar el nombre y el nombre de visualización del componente por algo más descriptivo.
5. En la pestaña **Detalles** verá que este componente tiene una interfaz, la interfaz que arrastró y soltó en el Editor de ensamblaje.
6. Asegúrese de que la clase Java a la que está intentando acceder está en la vía de acceso de clases del proyecto de servicio si no está contenida en el mismo proyecto de servicio.
7. Pulse con el botón derecho sobre el proyecto del módulo y seleccione **Abrir editor de dependencias...** Bajo la sección **Java**, asegúrese de que aparece el proyecto que contiene la clase Java antigua. Si no es así, añádala pulsando el botón **Añadir...**
8. En el editor de ensamblaje, pulse con el botón derecho el componente que acaba de crear y seleccione **Generar implementación...** → **Java** A continuación seleccione el paquete en el que se generará la implementación Java. Esto crea un servicio Java de esqueleto que se adhiere a la interfaz WSDL de acuerdo con el modelo de programación SCA, en el que los tipos complejos están representados por un objeto que es un `commonj.sdo.DataObject` y los tipos simples están representados por los equivalentes de objeto Java.

Los ejemplos de código siguientes muestran:

1. Definiciones relevantes de la interfaz WSDL 5.1
2. Los métodos Java de WebSphere Studio Application Developer Integration Edition 5.1 correspondientes al WSDL
3. Los métodos Java de WebSphere Integration Developer 6.0 para el mismo WSDL

El código siguiente muestra las definiciones relevantes de la interfaz WSDL de 5.1:

```
<types>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="qualified"
elementFormDefault="unqualified"
targetNamespace="http://migr.practice.ibm.com/"
xmlns:xsd="http://migr.practice.ibm.com/">
<complexType name="StockInfo">
<all>
<element name="index" type="int"/>
<element name="price" type="double"/>
<element name="symbol" nillable="true"
type="string"/>
</all>
</complexType>
</schema>
</types>
<message name="getStockInfoRequest">
<part name="symbol" type="xsd:string"/>
</message>
```

```

<message name="getStockInfoResponse">
<part name="result" type="xsd:StockInfo"/>
</message>
<operation name="getStockInfo" parameterOrder="symbol">
<input message="tns:getStockInfoRequest"
name="getStockInfoRequest"/>
<output message="tns:getStockInfoResponse"
name="getStockInfoResponse"/>
</operation>

```

El código siguiente muestra los métodos Java

de WebSphere

Studio Application Developer Integration Edition 5.1 correspondientes al WSDL:

```

public StockInfo getStockInfo(String symbol)
{
return new StockInfo();
}
public void setStockPrice(String symbol, float newPrice)
{
// establecer cosas
}

```

El código siguiente muestra los métodos Java

de WebSphere

Integration Developer 6.0 para el mismo WSDL:

```

public DataObject getStockInfo(String aString) {
//TODO Hay que implementar.
return null;
}
public void setStockPrice(String symbol, Float newPrice) {
//TODO Hay que implementar.
}

```

Ahora necesitará cumplimentar el código en el que verá los códigos “//TODO” en la clase de implementación Java

generada. Hay dos opciones:

1. Mueva la lógica de la clase Java original a esta clase, adaptándola para utilizar DataObjects
 - Esta es la opción recomendada si ha elegido el procedimiento de arriba abajo en WebSphere Studio Application Developer Integration Edition y desea que el componente Java trabaje con parámetros de DataObject. Esto es necesario porque las clases Java generadas a partir de definiciones WSDL en WebSphere Studio Application Developer Integration Edition tienen dependencias WSIF que deben eliminarse.
2. Cree una instancia privada de la antigua clase Java dentro de esta clase Java generada y escriba código para:
 - a. Convertir todos los parámetros de la clase de implementación Java generada en parámetros esperados por la clase Java antigua
 - b. Invocar la instancia privada de la clase Java antigua con los parámetros convertidos
 - c. Convertir el valor de retorno de la clase Java antigua en el tipo de valor de retorno declarado por el método de implementación Java generado
 - d. Esta opción se recomienda para los casos de consumo en los que los componentes Java de estilo 6.0 nuevos deben consumir proxys de servicio WSIF.

Una vez completada una de las opciones anteriores, debe volver a conectar el servicio Java. No debería haber referencias, por lo tanto, necesita volver a conectar la interfaz del componente Java:

- Si un proceso de negocio invoca este servicio en el mismo módulo, se crea una conexión desde la referencia de proceso de negocio adecuada a esta interfaz del componente Java.
- Si un proceso de negocio invoca este servicio en otro módulo, cree una **Exportación con enlace SCA** y desde el otro módulo, arrastre y suelte esta exportación en el Editor de ensamblaje de ese módulo para crear la **Importación con enlace SCA** correspondiente. Conecte la referencia de proceso de negocio adecuada a esa Importación.
- Si este servicio se publicó en WebSphere Studio Application Developer Integration Edition para exponerlo externamente, consulte la sección "Crear exportaciones SCA para acceder al servicio migrado" para obtener instrucciones acerca de cómo volver a publicar el servicio.

Crear un servicio Web Java: opción 2:

Una opción alternativa a tener en cuenta son las herramientas de servicios Web de Rational Application Developer que permiten crear un servicio Web alrededor de una clase Java.

Nota: Consulte la información del sitio siguiente antes de intentar migrar utilizando este método:
<http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.was.creation.ui.doc/tasks/twsbeanw.html>

Nota: Esta opción requiere configurar un tiempo de ejecución de servicio Web a través de WebSphere Integration Developer antes de invocar el asistente de servicio Web.

Si siguió un procedimiento de abajo arriba en WebSphere

Studio Application Developer Integration Edition para generar WSDL alrededor de la clase Java

, siga estos pasos:

1. Cree un proyecto Web nuevo y copie la clase Java que desea construir como un servicio alrededor de la carpeta fuente Java de este proyecto Web.
2. Pulse con el botón derecho del ratón sobre el proyecto de aplicación de empresa que es el contenedor de la clase Java alrededor de la cuál está creando un servicio.
3. Seleccione **Propiedades**, vaya a las propiedades del **Servidor** y asegúrese de que **Tiempo de ejecución destino** esté establecido en **WebSphere Process Server v6.0** y que **Servidor por omisión** esté establecido en el **WebSphere Process Server v6.0** instalado.
4. Inicie el servidor de prueba, despliegue esta aplicación en el servidor y asegúrese de que se inicia satisfactoriamente.
5. A continuación, pulse con el botón derecho sobre la clase Java alrededor de la cuál desea crear un servicio y seleccione **Servicios Web** → **Crear servicio Web**.
6. Para **Tipo de servicio Web** seleccione **Servicio Web de bean Java** y quite la marca de la opción **Iniciar servicio Web en el proyecto Web** a menos que desee desplegar inmediatamente el servicio Web. También puede optar por generar un proxy de cliente. Pulse **Siguiente**.
7. Se mostrará la clase Java que ha pulsado con el botón derecho, pulse **Siguiente**.
8. Ahora debe configurar las opciones de despliegue de servicio. Pulse el botón **Editar...** Para el tipo de servidor, elija **Servidor WPS v6.0** y para el tiempo de ejecución de servicio Web, elija **IBM WebSphere** y J2EE versión **1.4**. Si no es capaz de seleccionar una combinación válida haciendo esto, consulte la sección "Prepararse para la migración" para obtener información acerca de cómo migrar los proyectos J2EE al nivel de v1.4. Pulse **Aceptar**.
9. Para el proyecto de servicio, especifique el nombre del proyecto Web. Además, seleccione el proyecto EAR adecuado. Pulse **Siguiente**. Tenga en cuenta que posiblemente deberá esperar varios minutos.
10. En el panel Identidad del bean Java del servicio Web, seleccione el archivo WSDL que contendrá las definiciones WSDL. Elija los métodos que desea exponer en el servicio Web y elija el estilo/la

codificación adecuados (Documento/Literal, RPC/Literal o RPC/Codificado.) Seleccione la opción **Definir correlación personalizada para paquete a espacio de nombres** y seleccione un espacio de nombres que sea exclusivo de la clase Java migrada para todos los paquetes Java utilizados por esta interfaz de la clase Java (el espacio de nombres por omisión será exclusivo del nombre de paquete, lo que puede provocar conflictos si crea otro servicio Web que utilice las mismas clases Java.) Cumplimente los demás parámetros si procede.

11. Pulse **Siguiente** y, en el panel **Correlación de paquete de servicio Web con espacio de nombres**, pulse el botón **Añadir** y, en la fila que se crea, especifique el nombre del bean Java y luego el espacio de nombres personalizado que identifica de forma exclusiva a la clase Java. Continúe añadiendo correlaciones para todos los paquetes Java utilizados por la interfaz de bean Java.
12. Pulse **Siguiente**. Tenga en cuenta que posiblemente deberá esperar varios minutos.
13. Pulse **Finalizar**. Después de completar el asistente, debe copiar el archivo WSDL generado que describe el servicio Java para el proyecto de módulo de integración de negocio si el proyecto de servicio era un consumidor del servicio Java. Se encuentra en el proyecto Web de direccionador generado bajo la carpeta WebContent/WEB-INF/wsdl. Renovar/reconstruir el proyecto de módulo de integración de negocio.
14. Pase a la perspectiva Integración de negocio, expanda el módulo y después la categoría lógica **Puertos de servicio Web**.
15. Seleccione el puerto creado en los pasos anteriores, arrástrelo y suéltelo en el Editor de ensamblaje y seleccione crear una **Importación con enlace de servicio Web**. Seleccione la interfaz WSDL de la clase Java si se le solicita. Ahora, el componente SCA que consumía el componente Java en 5.1 puede conectarse a esta Importación para completar los pasos de migración de reconexión manual.

Tenga en cuenta que la interfaz puede ser ligeramente distinta de la interfaz de 5.1 y que puede ser necesario insertar un componente de mediación de interfaz entre el consumidor de 5.1 y la nueva Importación. Para hacerlo, pulse la herramienta de **conexión** del Editor de ensamblaje y conecte el componente origen de SCA con esta **Importación con enlace de servicio Web**. Como las interfaces son diferentes, se le indicará lo siguiente: **Los nodos origen y destino no tienen interfaces coincidentes. Elija crear una correlación de interfaces entre el nodo destino y el origen**. Efectúe una doble pulsación sobre el componente de correlación creado en el Editor de ensamblaje. Esto abrirá el editor de correlaciones. Consulte el Centro de información para obtener instrucciones acerca de cómo crear una correlación de interfaces.

Si siguió un procedimiento de arriba abajo en WebSphere Studio Application Developer Integration Edition, la generación de clases Java a partir de una definición WSDL, siga estos pasos:

1. Cree un proyecto Web nuevo y copie el archivo WSDL a partir del cuál desea generar el esqueleto Java en esta carpeta fuente del proyecto Web.
2. Pulse el archivo WSDL que contiene el PortType a partir del cuál desea generar un esqueleto Java y seleccione **Servicios Web** → **Generar esqueleto de bean Java**.
3. Elija el tipo de servicio Web **Servicio Web de bean Java de esqueleto** y complimente el asistente.

Después de completar el asistente, debe tener clases Java que implementen la interfaz de servicio y no sean dependientes de las API de WSIF.

Migrar un servicio EJB:

Puede migrar un servicio EJB a una Importación SCA con enlace de bean de sesión sin estado.

Si el proyecto de servicio de WebSphere

Studio Application Developer Integration Edition dependía de otro EJB, cliente EJB o proyecto Java

, importe esos proyectos utilizando el asistente **Archivo** → **Importar** → **Proyecto existente en el espacio de trabajo**. Este caso se daba habitualmente cuando se hacía referencia a un EJB desde un proyecto de servicio. Si los archivos WSDL o XSD a los que se hace referencia desde el proyecto de servicio se

encuentran en otro tipo de proyecto, cree una nueva biblioteca de integración empresarial con el mismo nombre que el antiguo proyecto que no era de servicio y copie todos estos artefactos en la biblioteca.

Importe el proyecto de servicio utilizando el asistente Migración. Esto resultará en la creación de un módulo de integración de negocio con los mensajes, tipos de puerto, enlaces y servicios de WSDL generados en WebSphere Studio Application Developer Integration Edition.

En la perspectiva Integración empresarial, expanda el módulo para ver el contenido correspondiente. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)

Tiene las opciones siguientes:

Pros y contras de las opciones de reconexión de servicios EJB:

Hay pros y contras para las opciones de reconexión de servicios EJB.

La lista siguiente describe las opciones y los pros y contras de cada una:

- La primera opción proporciona un mejor rendimiento en tiempo de ejecución porque invocar un servicio Web es más lento que invocar un EJB.
- La primera opción puede propagar el contexto mientras que una invocación de servicio Web no propaga el contexto de la misma forma.
- La segunda opción no implica la creación de código personalizado.
- La segunda opción no es aplicable para algunas definiciones de interfaz EJB ya que la generación de un servicio EJB tiene limitaciones. Consulte la documentación de Rational Application Developer aquí: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.doc/ref/rlimit.html>
- La segunda opción puede implicar un cambio de interfaz y por lo tanto un cambio para el consumidor de SCA.
- La segunda opción requiere la instalación de un servidor WebSphere Process Server 6.0 y se ha configurado para trabajar con WebSphere Integration Developer. Para ver los tiempos de ejecución instalados que están configurados para trabajar con WebSphere Integration Developer, vaya a **Ventana** → **Preferencias** → **Server** → **Tiempos de ejecución instalados**, seleccione la entrada **WebSphere Process Server v6.0** si es que existe y asegúrese de que señala la ubicación en la que está instalado el producto. Asegúrese de que esta entrada está marcada si el servidor existe y de que no lo está si el servidor no está instalado. También puede pulsar el botón **Añadir...** si desea añadir otro servidor.
- Si el componente Java se construyó en WebSphere Studio Application Developer Integration Edition utilizando el procedimiento de arriba abajo en el que se genera el esqueleto EJB a partir de un WSDL, los parámetros in y out de esta clase Java serán probablemente una subclase de `WSIFFormatPartImpl`. En este caso, elija la opción 2 para generar un esqueleto EJB genérico (no dependiente del WSIF ni de de las API de DataObject) de la interfaz WSDL original.

Crear el componente EJB personalizado: opción 1:

La técnica de migración recomendada consiste en utilizar el tipo de Importación con enlace de sesión sin estado de WebSphere Integration Developer que permite invocar un EJB de sesión sin estado como un componente SCA. Durante la migración, el código Java personalizado debe escribirse para convertirlo entre el estilo de interfaz Java SCA y el estilo de interfaz EJB existente.

Para crear el componente EJB personalizado:

1. En el proyecto de módulo, expanda **Interfaces** y seleccione la interfaz WSDL generada para este EJB en WebSphere Studio Application Developer Integration.

2. Arrastre y suelte esta interfaz en el Editor de ensamblaje. Aparecerá un diálogo en el que se le pedirá que seleccione el tipo de componente a crear. Seleccione **Componente sin tipo de implementación** y pulse **Aceptar**.
3. Aparecerá un componente en el diagrama de ensamblaje. Selecciónelo y vaya a la vista **Propiedades**.
4. En la pestaña **Descripción** puede cambiar el nombre y el nombre de visualización del componente por algo más descriptivo. Elija un nombre como el del EJB pero añada un sufijo como por ejemplo "JavaMed" ya que esto será un componente Java que medie entre la interfaz WSDL generada para el EJB en WebSphere Studio Application Developer Integration y la interfaz Java del EJB.
5. En la pestaña **Detalles** verá que este componente tiene una interfaz, la interfaz que arrastró y soltó en el Editor de ensamblaje.
6. En el editor de ensamblaje, pulse con el botón derecho el componente que acaba de crear y seleccione **Generar implementación...** → **Java** A continuación seleccione el paquete en el que se generará la implementación Java. Esto crea un servicio Java de esqueleto que se adhiere a la interfaz WSDL de acuerdo con el modelo de programación SCA, en el que los tipos complejos están representados por un objeto que es un `commonj.sdo.DataObject` y los tipos simples están representados por los equivalentes de objeto Java.

Los ejemplos de código siguientes muestran:

1. Definiciones relevantes de la interfaz WSDL 5.1
2. Los métodos Java de WebSphere Studio Application Developer Integration Edition 5.1 correspondientes al WSDL
3. Los métodos Java de WebSphere Integration Developer 6.0 para el mismo WSDL

El código siguiente muestra las definiciones relevantes de la interfaz WSDL de 5.1:

```
<types>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="qualified"
elementFormDefault="unqualified"
targetNamespace="http://migr.practice.ibm.com/"
xmlns:xsd1="http://migr.practice.ibm.com/">
<complexType name="StockInfo">
<all>
<element name="index" type="int"/>
<element name="price" type="double"/>
<element name="symbol" nillable="true"
type="string"/>
</all>
</complexType>
</schema>
</types>
<message name="getStockInfoRequest">
<part name="symbol" type="xsd:string"/>
</message>
<message name="getStockInfoResponse">
<part name="result" type="xsd1:StockInfo"/>
</message>
<operation name="getStockInfo" parameterOrder="symbol">
<input message="tns:getStockInfoRequest"
name="getStockInfoRequest"/>
<output message="tns:getStockInfoResponse"
name="getStockInfoResponse"/>
</operation>
```

El código siguiente muestra los métodos Java

de WebSphere

Studio Application Developer Integration Edition 5.1 correspondientes al WSDL:

```

public StockInfo getStockInfo(String symbol)
{
    return new StockInfo();
}
public void setStockPrice(String symbol, float newPrice)
{
    // establecer cosas
}

```

El código siguiente muestra los métodos Java

de WebSphere

Integration Developer 6.0 para el mismo WSDL:

```

public DataObject getStockInfo(String aString) {
    //TODO Hay que implementar.
    return null;
}
public void setStockPrice(String symbol, Float newPrice) {
    //TODO Hay que implementar.
}

```

También deberá cumplimentar código real cuando vea códigos “//TODO” en la clase de implementación Java

. Primero debe crear una referencia de este componente Java

al EJB real para que pueda acceder al EJB de acuerdo con el modelo de programación SCA:

1. Deje el editor de ensamblaje abierto y pase a la perspectiva J2EE. Busque el proyecto EJB que contiene el EJB para el que está creando un servicio.
2. Expanda el elemento **Descriptor de despliegue: <nombre-proyecto>** y busque el EJB. Arrástrelo y suéltelo en el Editor de ensamblaje. Si recibe un aviso acerca de la necesidad de actualizar las dependencias de proyecto, marque el recuadro de selección **Abrir el editor de dependencias de módulo** y pulse **Aceptar**.
3. Asegúrese de que el proyecto EJB aparece bajo la sección J2EE y si no es así, añádalo pulsando el botón **Añadir...**
4. Guarde las dependencias de módulo y cierre el editor. Verá que se creó una Importación nueva en el Editor de ensamblaje. Puede seleccionarlo e ir a la vista Propiedades en la pestaña Descripción para cambiar el nombre de la importación y el nombre de visualización por algo más significativo. En la pestaña Enlace verá que el tipo de importación se establece automáticamente en **Enlace de bean de sesión sin estado** y que el nombre JNDI del EJB ya se ha establecido adecuadamente.
5. Seleccione la herramienta Conectar en la paleta, en el Editor de ensamblaje.
6. Pulse el componente Java y suelte el ratón.
7. A continuación pulse la Importación EJB y suelte el ratón.
8. Cuando se le pregunte **Se creará una referencia coincidente en el nodo origen. ¿Desea continuar?**, pulse **Aceptar**. Esto crea una conexión entre los dos componentes.
9. Seleccione el componente Java en el Editor de ensamblaje y en la vista Propiedades bajo la pestaña Detalles, expanda Referencias y seleccione la referencia al EJB recién creado. Puede actualizar el nombre de la referencia si el nombre generado no es muy descriptivo o adecuado. Recuerde el nombre de esta referencia para utilizarla en el futuro.
10. Guarde el diagrama de ensamblaje.

Debe utilizar el modelo de programación SCA para invocar el EJB a partir de la clase Java

generada. Abra la clase Java

generada y siga estos pasos para escribir el código que invocará el servicio EJB. Para la clase de implementación Java

generada:

1. Cree una variable privada (cuyo tipo sea el de la interfaz EJB remota):
`private YourEJBInterface ejbService = null;`
2. Si hay tipos complejos en la interfaz EJB, cree también una variable privada para BOFactory:
`private BOFactory boFactory = (BOFactory)
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo
/BOFactory");`
3. En el constructor de la clase de implementación Java, utilice las API SCA para resolver la referencia EJB (recuerde cumplimentar el nombre de la referencia EJB que anotó en un paso anterior) y establecer la variable privada igual a esta referencia:
`// Buscar el servicio EJB
this.ejbService = (YourEJBInterface)
ServiceManager.INSTANCE.locateService("name-of-your-ejb-reference");`

Por cada “//TODO” de la clase de implementación Java

generada:

1. Convierta todos los parámetros en los tipos de parámetro esperados por el EJB.
2. Invoque el método adecuado en la referencia EJB utilizando el modelo de programación SCA, enviando los parámetros convertidos.
3. Convierta el valor de retorno del EJB en el tipo de valor de retorno declarado por el método de implementación Java generado

```
/**  
 * Método generado para soportar el tipo de puerto WSDL de implementación llamado  
 * "interface.MyBean".  
 */  
public BusObjImpl getStockInfo(String aString) {  
    BusObjImpl boImpl = null;  
    try {  
        // invocar el método EJB  
        StockInfo stockInfo = this.ejbService.getStockInfo(aString);  
        // formular el objeto de datos SCA para retorno.  
        boImpl = (BusObjImpl)  
            this.boFactory.createClass(StockInfo.class);  
        // convertir manualmente todos los datos del tipo de retorno EJB en el  
        // objeto de datos SCA a retornar  
        boImpl.setInt("index", stockInfo.getIndex());  
        boImpl.setString("symbol", stockInfo.getSymbol());  
        boImpl.setDouble("price", stockInfo.getPrice());  
    } catch (RemoteException e) {  
        e.printStackTrace();  
    }  
    return boImpl;  
}  
/**  
 * Método generado para soportar el tipo de puerto WSDL de implementación llamado  
 * "interface.MyBean".  
 */  
public void setStockPrice(String symbol, Float newPrice) {  
    try {  
        this.ejbService.setStockPrice(symbol, newPrice.floatValue());  
    } catch (RemoteException e) {  
        e.printStackTrace();  
    }  
}
```

Crear un servicio Web EJB: opción 2:

Una opción alternativa a tener en cuenta son las herramientas de servicios Web de Rational Application Developer que permiten crear un servicio Web alrededor de un EJB.

Nota: Consulte la información del sitio siguiente antes de intentar migrar utilizando este método:
<http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.was.creation.ejb.ui.doc/tasks/twsejwb.html>

Nota: Esta opción requiere configurar un tiempo de ejecución de servicio Web a través de WebSphere Integration Developer antes de invocar el asistente de servicio Web.

Para crear un servicio Web alrededor de un EJB, siga estos pasos:

1. Pulse con el botón derecho del ratón sobre el proyecto de aplicación de empresa que es el contenedor del EJB alrededor del cuál está creando un servicio.
2. Seleccione **Propiedades**, vaya a las propiedades del **Servidor** y asegúrese de que **Tiempo de ejecución destino** esté establecido en **WebSphere Process Server v6.0** y que **Servidor por omisión** esté establecido en el **WebSphere Process Server v6.0** instalado.
3. Inicie el servidor de prueba, despliegue esta aplicación en el servidor y asegúrese de que se inicia satisfactoriamente.
4. En la perspectiva J2EE, expanda el **Proyecto EJB** en la vista Explorador de proyectos. Expanda el **Descriptor de despliegue** y la categoría **Beans de sesión**. Seleccione el bean alrededor del cuál desea generar el servicio Web.
5. Pulse con el botón derecho del ratón y seleccione **Servicios Web** → **Crear servicio Web**.
6. Para **Tipo de servicio Web** seleccione **Servicio Web EJB** y quite la marca de la opción **Iniciar servicio Web en el proyecto Web** a menos que desee desplegar inmediatamente el servicio Web. Pulse **Siguiente**.
7. Asegúrese de que el EJB que pulsó con el botón derecho está seleccionado aquí y pulse **Siguiente**.
8. Ahora debe configurar las opciones de despliegue del servicio. Pulse el botón **Editar....** Para el tipo de servidor, elija **Servidor WPS v6.0** y para el tiempo de ejecución de servicio Web, elija **IBM WebSphere** y J2EE versión **1.4**. Si no es capaz de seleccionar una combinación válida haciendo esto, consulte la sección "Prepararse para la migración" para obtener información acerca de cómo migrar los proyectos J2EE al nivel de v1.4. Pulse **Aceptar**.
9. Para el proyecto de servicio, especifique el nombre del proyecto EJB que contiene el EJB. Además, seleccione el proyecto EAR adecuado. Pulse **Siguiente**. Tenga en cuenta que posiblemente deberá esperar varios minutos.
10. En el panel Configuración EJB de servicio Web, seleccione el proyecto de direccionador adecuado a utilizar (elija el nombre del proyecto Web de direccionador que desearía crear y este proyecto se añadirá a la misma aplicación de empresa que el EJB original. Seleccione el transporte deseado (**SOAP sobre HTTP** o **SOAP sobre JMS**). Pulse **Siguiente**.
11. Seleccione el archivo WSDL que contendrá las definiciones WSDL. Elija los métodos que desea exponer en el servicio Web y elija el estilo/la codificación adecuados (Documento/Literal, RPC/Literal o RPC/Codificado.) Seleccione la opción **Definir correlación personalizada para paquete a espacio de nombres** y seleccione un espacio de nombres que sea exclusivo del EJB migrado para todos los paquetes Java utilizados por el EJB (el espacio de nombres por omisión será exclusivo del nombre de paquete, lo que puede provocar conflictos si crea otro servicio Web que utilice las mismas clases Java). Complimente los demás parámetros si procede. Hay limitaciones para cada combinación de estilo/codificación. Consulte las limitaciones para obtener más información: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.doc/ref/rlimit.html>
12. Pulse **Siguiente** y, en el panel **Correlación de paquete de servicio Web con espacio de nombres**, pulse el botón **Añadir** y, en la fila que se crea, especifique el nombre del paquete del EJB y luego el espacio de nombres personalizado que identifica de forma exclusiva a este EJB. Continúe añadiendo correlaciones para todos los paquetes Java utilizados por la interfaz EJB.
13. Pulse **Siguiente**. Tenga en cuenta que posiblemente deberá esperar varios minutos.

14. Pulse **Finalizar**. Después de completar el asistente, debe copiar el archivo WSDL generado que describe el servicio EJB para el proyecto de módulo de integración de negocio si el proyecto de servicio era un consumidor del servicio EJB. Se encuentra en el proyecto Web de direccionador generado bajo la carpeta WebContent/WEB-INF/wsdl. Renovar/reconstruir el proyecto de módulo de integración de negocio.
15. Pase a la perspectiva Integración de negocio, expanda el módulo migrado y después la categoría lógica **Puertos de servicio Web**.
16. Seleccione el puerto generado en los pasos anteriores, arrástrelo y suéltelo en el Editor de ensamblaje y seleccione crear una **Importación con enlace de servicio Web**. Seleccione la interfaz WSDL del EJB si se le solicita. Ahora, el componente SCA que consumía el EJB en 5.1 puede conectarse a esta Importación para completar los pasos de migración de reconexión manual.

Si siguió un procedimiento de arriba abajo en WebSphere Studio Application Developer Integration Edition, la generación de un esqueleto EJB a partir de una definición WSDL, siga estos pasos:

1. Cree un proyecto Web y copie el archivo WSDL a partir del cuál desea generar el esqueleto EJB en esta carpeta origen del proyecto Web.
2. Pulse el archivo WSDL que contiene el PortType a partir del cuál desea generar un esqueleto EJB y seleccione **Servicios Web** → **Generar esqueleto de bean Java**.
3. Elija el tipo de servicio Web **Servicio Web EJB esqueleto** y complimente el asistente.

Después de completar el asistente, debe tener un EJB que implemente la interfaz de servicio y que no sea dependiente de las API de WSIF.

Tenga en cuenta que la interfaz puede ser ligeramente distinta de la interfaz de 5.1 y que puede ser necesario insertar un componente de mediación de interfaz entre el consumidor de 5.1 y la nueva Importación. Para hacerlo, pulse la herramienta de **conexión** del Editor de ensamblaje y conecte el componente origen de SCA con esta **Importación con enlace de servicio Web**. Como las interfaces son diferentes, se le indicará lo siguiente: **Los nodos origen y destino no tienen interfaces coincidentes**. Elija **crear una correlación de interfaces entre el nodo destino y el origen**. Efectúe una doble pulsación sobre el componente de correlación creado en el Editor de ensamblaje. Esto abrirá el editor de correlaciones. Consulte el Centro de información para obtener instrucciones acerca de cómo crear una correlación de interfaces.

Una vez finalizado esto, debe volver a conectar el servicio EJB. No debería haber referencias, por lo tanto, necesita volver a conectar la interfaz del componente Java:

- Si este servicio lo invoca un proceso de negocio en el mismo módulo, cree una conexión a partir de la referencia de proceso de negocio adecuada a este componente EJB.
- Si un proceso de negocio invoca este servicio en otro módulo, cree una **Exportación con enlace SCA** y desde el otro módulo, arrastre y suelte esta exportación en el Editor de ensamblaje de ese módulo para crear la **Importación con enlace SCA** correspondiente. Conecte la referencia de proceso de negocio adecuada a esa Importación.
- Si este servicio se publicó en WebSphere Studio Application Developer Integration Edition para exponerlo externamente, consulte la sección "Migración de servicio no BPEL entrante" para obtener instrucciones acerca de cómo volver a publicar el servicio.

Migrar un proceso de negocio a una invocación de servicio de proceso de negocio:

Este escenario se aplica a un proceso de negocio que invoca otro proceso de negocio, donde el segundo proceso de negocio se invoca utilizando un enlace de proceso WSIF. En esta sección se muestra cómo migrar una invocación de BPEL a servicio BPEL utilizando una conexión o una Importación/Exportación con enlace SCA.

Para migrar un proyecto de servicio de enlace de proceso (BPEL) para un servicio saliente, siga estos pasos:

1. En la perspectiva Integración empresarial, expanda el módulo para ver el contenido correspondiente. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)
2. Hay varios casos en los que un proceso BPEL puede invocar otro proceso BPEL. Identifique qué caso de los siguientes es aplicable a su aplicación:
 - Si el BPEL que se invoca está en el mismo módulo, cree una conexión desde la referencia adecuada del primer componente BPEL a la interfaz adecuada del componente BPEL destino.
 - Si el BPEL que se invoca está en otro módulo (siendo el otro módulo un proyecto de servicio migrado):
 - a. Cree una **Exportación con enlace SCA** para el segundo proceso de negocio en el diagrama de ensamblaje de módulo correspondiente.
 - b. Expanda el segundo icono de ensamblaje de módulo en el navegador en la vista Integración empresarial. Debe ver la exportación que acaba de crear.
 - c. Arrastre y suelte la exportación de la vista Integración empresarial bajo el segundo módulo en el editor de ensamblaje abierto del primer módulo. Con esto se creará una Importación con enlace SCA en el primer módulo. Si este servicio se publicó en WebSphere Studio Application Developer Integration Edition para exponerlo externamente, consulte la sección "Crear Exportaciones SCA para acceder al servicio migrado".
 - d. Conecte la referencia adecuada en el primer proceso de negocio con la importación que acaba de crear en ese módulo.
 - e. Guarde el diagrama de ensamblaje.
 - Para obtener un enlace tardío al invocar el segundo proceso comercial:
 - a. Deje la referencia del primer componente de proceso comercial sin conectar. Abra el primer proceso en el editor de BPEL y, bajo la sección **Asociados de referencia**, seleccione el asociado que corresponda al segundo proceso BPEL para invocarlo mediante enlace tardío.
 - b. En la vista Propiedades de la pestaña **Descripción**, especifique el nombre del segundo proceso comercial en el campo **Plantilla de proceso**.
 - c. Guarde el proceso comercial. Ha finalizado la configuración de la invocación de enlace tardío.

Migrar un servicio Web (SOAP/JMS):

Puede migrar un servicio Web (SOAP/JMS) a una Importación SCA con enlace de servicio Web.

Para migrar un proyecto de servicio SOAP/JMS para una migración de servicio saliente, siga estos pasos:

1. En primer lugar, debe importar el proyecto de servicio utilizando el asistente Migración. Con esta operación se creará un módulo de integración empresarial con los mensajes WSDL, los tipos de puerto, los enlaces y los servicios generados en WebSphere Studio Application Developer Integration Edition. Tenga en cuenta que si el servicio Web de IBM (SOAP/JMS) que invocará esta aplicación también es un servicio Web de WebSphere Studio Application Developer Integration Edition que se migrará, puede que haya habido actualizaciones en ese servicio Web durante la migración. En ese caso, utilice aquí los archivos WSDL migrados de ese servicio Web.
2. En la perspectiva Integración empresarial, expanda el módulo para poder ver su contenido. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)
3. A continuación, añada una Importación que permita a la aplicación interactuar con el servicio Web IBM (a través de SOAP/JMS) de acuerdo con el modelo de programación SCA. Asegúrese de que la interfaz, el enlace y las definiciones de servicio WSDL están presentes en el módulo migrado o en una biblioteca de la que depende el módulo migrado.
4. En la perspectiva Integración empresarial, expanda el módulo migrado y abra el Diagrama de ensamblaje en el Editor de ensamblaje.
5. Expanda la categoría lógica de Puertos de servicio Web y arrastre y suelte el puerto correspondiente al servicio que desea invocar en el Editor de ensamblaje.

6. Elija crear una **Importación con enlace de servicio Web**.
7. Después de crear la importación, selecciónela en el Editor de ensamblajes y vaya a la vista Propiedades. En la pestaña Enlace verá el puerto y el servicio a los que está enlazada la importación.
8. Guarde el diagrama de ensamblaje.

Una vez que haya cumplimentado esto, debe volver a conectar el servicio:

- Si un proceso de negocio invoca este servicio en el mismo módulo, cree una conexión de la referencia de proceso de negocio adecuada a esta Importación.
- Si un proceso de negocio invoca este servicio en otro módulo, cree una **Exportación con enlace SCA** y desde el otro módulo, arrastre y suelte esta exportación en el Editor de ensamblaje de ese módulo para crear la **Importación con enlace SCA** correspondiente. Conecte la referencia de proceso de negocio adecuada a esa Importación.
- Guarde el diagrama de ensamblaje.

Migrar un servicio Web (SOAP/HTTP):

Puede migrar un servicio Web (SOAP/HTTP) a una Importación SCA con enlace de servicio Web.

Para migrar un proyecto de servicio SOAP/HTTP para una migración de servicio saliente, siga estos pasos:

1. En primer lugar, debe importar el proyecto de servicio utilizando el asistente Migración. Con esta operación se creará un módulo de integración empresarial con los mensajes WSDL, los tipos de puerto, los enlaces y los servicios generados en WebSphere Studio Application Developer Integration Edition. Tenga en cuenta que si el servicio Web de IBM (SOAP/HTTP) que invocará esta aplicación también es un servicio Web de WebSphere Studio Application Developer Integration Edition que se migrará, puede que haya habido actualizaciones en ese servicio Web durante la migración. En ese caso, utilice aquí los archivos WSDL migrados de ese servicio Web.
2. En la perspectiva Integración empresarial, expanda el módulo para poder ver su contenido. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)
3. A continuación, añada una Importación que permita a la aplicación interactuar con el servicio Web IBM (a través de SOAP/HTTP) de acuerdo con el modelo de programación SCA. Asegúrese de que la interfaz, el enlace y las definiciones de servicio WSDL están presentes en el módulo migrado o en una biblioteca de la que depende el módulo migrado.
4. En la perspectiva Integración empresarial, expanda el módulo migrado y abra el Diagrama de ensamblaje en el Editor de ensamblaje.
5. Expanda la categoría lógica de Puertos de servicio Web y arrastre y suelte el puerto correspondiente al servicio que desea invocar en el Editor de ensamblaje.
6. Elija crear una **Importación con enlace de servicio Web**.
7. Después de crear la importación, selecciónela en el Editor de ensamblajes y vaya a la vista Propiedades. En la pestaña Enlace verá el puerto y el servicio a los que está enlazada la importación.
8. Guarde el diagrama de ensamblaje.

Una vez que haya cumplimentado esto, debe volver a conectar el servicio:

- Si un proceso de negocio invoca este servicio en el mismo módulo, cree una conexión de la referencia de proceso de negocio adecuada a esta Importación.
- Si un proceso de negocio invoca este servicio en otro módulo, cree una **Exportación con enlace SCA** y desde el otro módulo, arrastre y suelte esta exportación en el Editor de ensamblaje de ese módulo para crear la **Importación con enlace SCA** correspondiente. Conecte la referencia de proceso de negocio adecuada a esa Importación.
- Guarde el diagrama de ensamblaje.

Migrar un servicio JMS:

Puede migrar un servicio JMS a una Importación SCA con enlace JMS.

Nota: Si el mensaje JMS se envía a un WebSphere Business Integration Adapter, consulte la sección "Migrar interacciones con WebSphere Business Integration Adapter".

Para migrar un proyecto de servicio JMS para una migración de servicio saliente, siga estos pasos:

1. En primer lugar, debe importar el proyecto de servicio utilizando el asistente Migración. Con esta operación se creará un módulo de integración empresarial con los mensajes WSDL, los tipos de puerto, los enlaces y los servicios generados en WebSphere Studio Application Developer Integration Edition.
2. En la perspectiva Integración empresarial, expanda el módulo para poder ver su contenido. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)
3. A continuación, añada una Importación que permita a la aplicación interactuar con una cola JMS de acuerdo con el modelo de programación SCA.
4. En el Editor de ensamblaje, expanda el proyecto de módulo migrado, expanda la categoría **Interfaces** y busque el PortType WSDL que describe el servicio Web que invocará la aplicación. Arrástrelo y suéltelo en el Editor de ensamblaje.
5. Un diálogo **Creación de componente** permitirá seleccionar el tipo de componente a crear. Elija **Importar sin enlaces**.
6. Verá que se habrá creado una Importación nueva en el Editor de ensamblaje y que si la selecciona y va a la vista Propiedades, en la pestaña Descripción podrá cambiar el nombre y el nombre de visualización de la importación por algo más significativo.
7. Puede consultar los archivos WSDL de enlace y servicio 5.1 para encontrar detalles acerca del servicio JMS que está migrando y utilizarlos para cumplimentar los detalles de la opción 6.0 "Importar con enlace JMS". Localice los archivos WSDL de enlace y servicio JMS 5.1 dentro del proyecto de servicio 5.1 (generalmente se denominan *JMSBinding.wsdl y *JMSService.wsdl). Examine la información de enlace y servicio capturada allí. A partir del enlace, puede determinar si se han utilizado mensajes de texto u objeto y si se han utilizado enlaces de formato de datos de cliente. Si existen, debe considerar la posibilidad de escribir también un enlace de datos personalizado para la opción "Exportar con enlace JMS" de 6.0. Desde el servicio, puede buscar la fábrica de contexto inicial, el nombre de conexión JNDI, el nombre de destino JNDI y el estilo de destino (cola).
8. Pulse la importación con el botón derecho del ratón, seleccione **Generar enlace** y después **Enlace JMS**. Se le solicitará que especifique los parámetros siguientes:

Seleccione el dominio de mensajería JMS:

- Punto a punto
- Publicación-suscripción
- Independiente del dominio

Seleccione cómo se serializan los datos entre Objeto comercial y Mensaje JMS:

- Texto
- Objeto
- Proporcionado por el usuario

Si se selecciona Proporcionado por el usuario:

Especifique el nombre completamente calificado de la clase de implementación `com.ibm.websphere.sca.jms.data.JMSDataBinding`. Deberá especificar un enlace de datos suministrado por usuario si la aplicación necesita establecer propiedades de cabecera JMS que no están generalmente disponibles en el enlace de importación JMS. En este caso, a puede crear una clase de enlace de datos personalizado que amplíe el enlace de datos JMS estándar "`com.ibm.websphere.sca.jms.data.JMSDataBinding`" y añadir código personalizado para

acceder a JMSMessage directamente. Consulte los ejemplos de JMS de la sección "Crear y modificar enlaces para componentes de importación y exportación" cuyo enlace figura más abajo.

La conectividad entrante utiliza la clase de selector de función JMS por omisión

<selected> o <deselected>

9. Seleccione la importación que acaba de crear. En la vista Propiedades, vaya a la pestaña Enlace. Puede complimentar manualmente toda la información de enlace que aparece allí con los mismos valores especificados anteriormente en WebSphere Studio Application Developer Integration Edition. La información de enlace que puede especificar es:
 - Enlace de importación JMS (esto es lo más importante)
 - Conexión
 - Adaptador de recursos
 - Destinos JMS
 - Enlaces de método

Una vez que haya cumplimentado esto, debe volver a conectar el servicio:

- Si un proceso de negocio invoca este servicio en el mismo módulo, cree una conexión de la referencia de proceso de negocio adecuada a esta Importación.
- Si un proceso de negocio invoca este servicio en otro módulo, cree una **Exportación con enlace SCA** y desde el otro módulo, arrastre y suelte esta exportación en el Editor de ensamblaje de ese módulo para crear la **Importación con enlace SCA** correspondiente. Conecte la referencia de proceso de negocio adecuada a esa Importación.
- Guarde el diagrama de ensamblaje.

Migrar un servicio J2C-IMS:

Puede migrar un servicio J2C-IMS a una Importación SCA con enlace EIS o a una Importación SCA con enlace de servicio Web.

No utilice ninguno de los artefactos de WebSphere

Studio Application Developer Integration Edition generados para este servicio IMS

. Necesitará recrear el servicio utilizando los asistentes disponibles en WebSphere

Integration Developer y volver a conectar manualmente la aplicación.

Nota: Active la construcción automática o construya el módulo manualmente.

Tiene las opciones siguientes:

Nota: Para ambas opciones, tenga en cuenta que si un servicio BPEL invoca este servicio IMS, el BPEL deberá cambiar ligeramente ya que la interfaz expuesta por el servicio EIS será ligeramente diferente a la interfaz 5.1 antigua. Para hacerlo, abra el editor BPEL y ajuste el enlace de socio correspondiente al servicio EIS y utilice la interfaz nueva (archivo WSDL) generada al realizar los pasos anteriores. Haga los cambios necesarios en las actividades de BPEL para la interfaz WSDL nueva del servicio EIS.

Pros y contras de las opciones de reconexión de servicios J2C-IMS:

Hay pros y contras para las opciones de reconexión de servicios J2C-IMS.

La lista siguiente describe las opciones y los pros y contras de cada una:

- La primera opción utiliza los componentes SCA estándar para invocar el servicio IMS.
- La primera opción tiene algunas limitaciones:
 - La API de especificación de SDO versión 1 no proporciona acceso a la matriz de bytes COBOL o C, lo que afectará a los clientes que trabajen con segmentos múltiples de IMS.
 - La especificación de SDO versión 1 para la serialización no soporta redefiniciones COBOL ni uniones C.
- La segunda opción utiliza el procedimiento JSR 109 estándar para conectar con el servicio IMS. Esta funcionalidad está disponible como parte de Rational Application Developer.

Crear una Importación SCA para invocar el servicio IMS: opción 1:

Puede crear una Importación SCA con enlace EIS que utilizará DataObjects para almacenar el mensaje/los datos para comunicarse con el sistema IMS.

Para crear una Importación SCA para invocar el servicio IMS

, siga estos pasos:

1. Cree un proyecto de módulo de integración de negocio nuevo para albergar este servicio IMS nuevo.
2. Para volver a crear el servicio EIS, vaya a **Archivo** → **Nuevo** → **Otro** → **Integración empresarial** → **Descubrimiento de servicios de empresa**.
3. Este asistente permite importar un servicio de un sistema EIS. Es muy parecido al asistente de WebSphere Studio Application Developer Integration Edition que creó el servicio EIS basado en WSIF en 5.1. Puede importar el adaptador de recursos IMS de J2C nuevo en este asistente. Sitúese en el directorio de instalación de WebSphere Integration Developer y vaya **Adaptadores de recurso** → **ims15** → **imsico9102.rar**.

Nota: Consulte el Centro de información para obtener más información acerca de cómo completar las propiedades de guardado y los paneles de operaciones. Durante el asistente Descubrimiento de servicios de empresa, cuando añada una operación, también podrá crear objetos de negocio para los tipos de datos de entrada o salida de la operación. Para ello es necesario tener el archivo fuente C o COBOL utilizado en el asistente de WebSphere Studio Application Developer Integration Edition. Estos archivos deben haberse copiado en el antiguo proyecto de servicio para que pueda señalar a los archivos origen que hay ahí. También puede importar los objetos de negocio utilizando el asistente **Archivo** → **Nuevo** → **Otro** → **Integración empresarial** → **Descubrimiento de datos de empresa**.

4. Una vez completado el asistente, abra la perspectiva Integración de negocio y expanda el módulo para poder ver el contenido correspondiente. Debe ver los objetos de negocio nuevos listados bajo los tipos de datos del módulo y las interfaces nuevas listadas bajo Interfaces.
5. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.) Debe ver que existe una Importación en el lienzo, esta Importación tiene un enlace EIS y representa el servicio que acaba de crear.

Ahora consulte la sección titulada “Crear exportaciones SCA para acceder al servicio migrado” para obtener instrucciones acerca de cómo exponer este servicio a los consumidores.

Crear un servicio Web alrededor del servicio J2C: opción 2:

Puede crear un servicio Web J2C y si el consumidor del servicio es un componente SCA, consuma el servicio como un Servicio Web IBM (SOAP/HTTP o SOAP/JMS).

Para crear un servicio Web alrededor del servicio J2C, siga estos pasos:

1. Cree el bean Java J2C pulsando **Archivo** → **Nuevo** → **J2C** → **Bean Java J2C**
2. Elija la versión 1.5 de **IMS Connector for Java** y pulse **Siguiente**.

3. Marque **Conexión gestionada** y especifique el nombre de búsqueda JNDI. Pulse **Siguiente**.
4. Especifique el proyecto, el paquete y el nombre para el bean Java nuevo. El bean consta de una interfaz y una clase de implementación. Pulse **Siguiente**.
5. Añada un método Java para cada función o servicio a la que desea acceder desde el EIS. Pueden añadirse métodos adicionales posteriormente en el editor fuente de Java a través de la vista Fragmentos de código. Cuando pulsa el botón **Añadir...**, elija el nombre para el método y pulse **Siguiente**.
6. Ahora puede elegir **Examinar...** para reutilizar tipos existentes o **Nuevo...** para lanzar el asistente Enlace de datos Java CICS/IMS (donde puede hacer referencia a un archivo fuente COBOL o C) para los tipos de datos de entrada y salida.
7. Una vez haya terminado de crear los métodos Java, pulse **Siguiente**.
8. Complete los pasos restantes de este asistente para crear el bean Java J2C.
9. Cree el servicio Web pulsando **Archivo → Nuevo → J2C → Página Web, Servicio Web o EJB a partir de bean Java J2C** para crear el servicio Web alrededor del bean Java J2C.
10. Complete el asistente.

Los consumidores de este servicio ahora pueden utilizar el servicio WSDL generado por este asistente para invocar el servicio IMS

Migrar un servicio ECI J2C-CICS:

Puede migrar un servicio ECI J2C-CICS a una Importación SCA con enlace EIS o a una Importación SCA con enlace de servicio Web.

Siga las instrucciones del tema "Migrar un proyecto de servicio J2C-IMS" pero asegúrese de importar el archivo RAR siguiente *en lugar* del archivo RAR IMS

:

- Sitúese en el directorio de instalación de WebSphere Integration Developer y vaya a **Resource Adapters → cics15 → cicseci.rar**.

Si sigue la segunda opción para crear un servicio Web J2C, elija el **ECIResourceAdapter** de de la v1.5 en el segundo panel del asistente de creación de bean Java

J2C.

Consulte también el tema "Migrar un servicio J2C-IMS".

Migrar un servicio EPI J2C-CICS:

No hay soporte directo para el servicio EPI J2C-CICS en WebSphere Integration Developer. Para acceder a este servicio desde un módulo SCA, necesitará migrar utilizando el *escenario de consumo*.

Consulte el tema "Escenario de consumo para la migración de servicios" para obtener instrucciones acerca de cómo migrar este tipo de servicio a WebSphere

Integration Developer.

Migrar un servicio J2C-HOD:

No hay soporte directo para el servicio J2C-HOD en WebSphere Integration Developer. Para acceder a este servicio desde un módulo SCA, necesitará migrar utilizando el *escenario de consumo*.

Consulte el tema "Escenario de consumo para la migración de servicios" para obtener instrucciones acerca de cómo migrar este tipo de servicio a WebSphere

Integration Developer.

Migrar un servicio transformador:

Puede migrar un servicio transformador para una Correlación de datos y una Correlación de interfaces de SCA cuando sea posible. También puede utilizar el *escenario de consumo* para acceder a este servicio desde un módulo SCA.

Los componentes de correlación de datos y de correlación de interfaces son nuevos en la versión 6.0. Ofrecen una función parecida al servicio transformador de 5.1 pero no tienen la posibilidad de transformación XSL completa. Si no puede sustituir el servicio transformador con uno de estos componentes, debe migrar utilizando el escenario de consumo ya que no hay soporte directo para el servicio transformador en WebSphere

Integration Developer. Siga los pasos documentados en la sección "Escenario de consumo para migración de servicios" para acceder a este servicio desde un módulo SCA.

Escenario de consumo para la migración de servicios:

En los casos en los que no hay una contrapartida directa para un tipo de servicio de WebSphere Studio Application Developer Integration Edition, se necesita un escenario de consumo para consumir el servicio antiguo de WebSphere Studio Application Developer Integration Edition tal cual al rediseñar la aplicación en WebSphere Integration Developer.

Estos son los pasos a seguir en WebSphere Studio Application Developer Integration Edition *antes* de invocar el asistente Migración:

1. Cree un proyecto Java nuevo para albergar este código proxy de cliente. No coloque este código proxy de cliente en el proyecto de servicio, ya que el asistente de migración automática que migra los proyectos de servicio pasará por alto los mensajes generados de tipo 5.1 y las clases de bean Java.
2. Abra WebSphere Studio Application Developer Integration Edition y pulse con el botón derecho sobre el archivo WSDL que contiene el servicio y el enlace de transformador y seleccione **Servicios de empresa** → **Generar proxy de servicio**. Se le preguntará que tipo de proxy desea crear, pero la única opción disponible será **Infraestructura de invocación de servicios Web (WSIF)**. Pulse **Siguiente**.
3. Ahora puede especificar el paquete y el nombre de la clase Java del proxy de servicio que se va a crear (creará el proxy en el proyecto de servicio actual.) Pulse **Siguiente**.
4. Ahora puede especificar el estilo del proxy, elija el **Apéndice de cliente**, seleccione las operaciones deseadas para incluir el proxy y pulse **Finalizar**. Esto crea una clase Java que expone los mismos métodos que el servicio de WebSphere Studio Application Developer Integration Edition, en los que los argumentos para los métodos Java son los componentes del mensaje WSDL origen.

Ahora puede migrar a WebSphere Integration Developer:

1. Copie el proyecto Java de proxy de cliente en el espacio de trabajo nuevo e impórtelo seleccionando **Archivo** → **Importar** → **Existing Proyecto existente en el espacio de trabajo**.
2. Importe el proyecto de servicio utilizando el asistente Migración. Con esta operación se creará un módulo de integración empresarial con los mensajes WSDL, los tipos de puerto, los enlaces y los servicios generados en WebSphere Studio Application Developer Integration Edition.
3. En la perspectiva Integración empresarial, expanda el módulo para poder ver su contenido. Abra el Editor de ensamblajes efectuando una doble pulsación sobre el primer elemento bajo el proyecto de módulo (tendrá el mismo nombre que el proyecto.)

4. Para crear el componente Java personalizado, en el proyecto del módulo, expanda **Interfaces** y seleccione la interfaz WSDL generada para este servicio transformador en WebSphere Studio Application Developer Integration Edition.
5. Arrastre y suelte esta interfaz en el Editor de ensamblaje. Aparecerá un diálogo en el que se le pedirá que seleccione el tipo de componente a crear. Seleccione **Componente sin tipo de implementación** y pulse **Aceptar**.
6. Aparecerá un componente en el diagrama de ensamblaje. Selecciónelo y vaya a la vista **Propiedades**.
7. En la pestaña **Descripción**, puede cambiar el nombre y el nombre de visualización del componente por algo más descriptivo (en este caso el nombre es algo como el nombre del EJB pero con un sufijo añadido como por ejemplo "JavaMed" ya que esto será un componente Java que medie entre la interfaz WSDL generada para el servicio transformador en WebSphere Studio Application Developer Integration Edition y la interfaz Java del proxy de cliente de transformador.)
8. En la pestaña **Detalles** verá que este componente tiene una interfaz, la interfaz que arrastró y soltó en el Editor de ensamblaje.
9. En el editor de ensamblaje, pulse con el botón derecho el componente que acaba de crear y seleccione **Generar implementación...** → **Java**. A continuación seleccione el paquete en el que se generará la implementación Java. Esto crea un servicio Java de esqueleto que se adhiere a la interfaz WSDL de acuerdo con el modelo de programación SCA, en el que los tipos complejos están representados por un objeto que es un `commonj.sdo.DataObject` y los tipos simples están representados por los equivalentes de objeto Java.

Ahora necesitará cumplimentar el código en el que verá los códigos "//TODO" en la clase de implementación Java generada. Hay dos opciones:

1. Mueva la lógica de la clase Java original a esta clase, adaptándola a la estructura de datos nueva.
2. Cree una instancia privada de la antigua clase Java dentro de esta clase Java generada y escriba código para:
 - a. Convertir todos los parámetros de la clase de implementación Java generada en parámetros esperados por la clase Java antigua
 - b. Invocar la instancia privada de la clase Java antigua con los parámetros convertidos
 - c. Convertir el valor de retorno de la clase Java antigua en el tipo de valor de retorno declarado por el método de implementación Java generado

Una vez completadas las opciones anteriores, debe volver a conectar el proxy de cliente. No debería haber "referencias", por lo tanto, necesita volver a conectar la interfaz del componente Java:

- Si un proceso de negocio invoca este servicio en el mismo módulo, se crea una conexión desde la referencia de proceso de negocio adecuada a esta interfaz del componente Java.
- Si un proceso de negocio invoca este servicio en otro módulo, cree una **Exportación con enlace SCA** y desde el otro módulo, arrastre y suelte esta exportación en el Editor de ensamblaje de ese módulo para crear la **Importación con enlace SCA** correspondiente. Conecte la referencia de proceso de negocio adecuada a esa Importación.
- Si este servicio se publicó en WebSphere Studio Application Developer Integration Edition para exponerlo externamente, consulte la sección "Crear exportaciones SCA para acceder al servicio migrado" para obtener instrucciones acerca de cómo volver a publicar el servicio.

Crear Exportaciones SCA para acceder al servicio migrado:

Debe crearse una Exportación SCA para poner el servicio migrado a disposición de los consumidores externos de acuerdo con el modelo SCA para todos los servicios para los que se generó el código de despliegue en el proyecto de servicio WebSphere Studio Application Developer Integration Edition. Esto incluye todos los servicios invocados por los clientes externos para la aplicación.

Si en WebSphere Studio Application Developer Integration Edition, pulsaba con el botón derecho el proceso BPML u otro WSDL de servicio y seleccionaba **Servicios de empresa** → **Generar código de**

despliegue, debe realizar los pasos manuales de migración siguientes. Tenga en cuenta que WebSphere Integration Developer se diferencia de WebSphere Studio Application Developer Integration Edition en que almacena todas las opciones de despliegue. Cuando se construye el proyecto, el código de despliegue se actualiza automáticamente en el EJB generado y los proyectos Web de modo que ya no hay ninguna opción para ejecutar manualmente **Generar código de despliegue**.

En la sección de interfaces para socios del asistente Generar código de despliegue BPEL se proporcionaban cinco opciones de enlace. La siguiente información de migración de servicio BPEL entrante ofrece más detalles sobre el tipo de exportación y las propiedades que se crearán según el tipo de enlace de despliegue seleccionado en WebSphere Studio Application Developer Integration Edition:

- EJB
- Servicio Web de IBM (SOAP/JMS)
- Servicio Web de IBM (SOAP/HTTP)
- Servicio Web de Apache (SOAP/HTTP)
- JMS

Migrar los enlaces de EJB y de proceso EJB:

Se puede migrar los enlaces de EJB y de proceso EJB a la construcción de SCA recomendada.

En WebSphere Studio Application Developer Integration Edition, este tipo de enlace permitía a los clientes comunicarse con un proceso BPEL u otro tipo de servicio invocando un EJB. Tenga presente que este tipo de enlace no era opcional para los microprocesos; siempre se seleccionaba ya que los demás tipos de enlace utilizaban internamente el EJB generado.

El nombre JNDI del EJB generado se generaba automáticamente como una combinación del nombre, el espacio de nombres destino y la indicación de la hora de inicio de validez del proceso BPEL. Por ejemplo, estos atributos se pueden encontrar examinando las propiedades del proceso BPEL en las pestañas de contenido Descripción y Servidor del editor BPEL:

Tabla 3. Espacio de nombres generado

Nombre de proceso	MyService
Espacio de nombres destino	http://www.example.com/process87787141/
Válido desde	Ene 01 2003 02:03:04

El espacio de nombres generado en este ejemplo es com/example/www/process87787141/MyService20030101T020304.

En WebSphere Studio Application Developer Integration Edition, al seleccionar el enlace de EJB como tipo de despliegue, no se proporcionaba ninguna opción.

Existen cuatro opciones para migrar el enlace de proceso de WebSphere Studio Application Developer Integration Edition. El tipo de cliente que accede al servicio determina cuáles de las opciones de migración siguientes se llevan a cabo:

Nota: Una vez efectuados los pasos manuales de migración, el cliente también debe migrarse al nuevo modelo de programación. Consulte el tema pertinente para los tipos de cliente siguientes:

Tabla 4. Información adicional sobre la migración de clientes

Tipo de cliente	Fuente de información adicional
Cliente EJB que invoca el bean de sesión generado. Este tipo de cliente invoca un método EJB correspondiente a la operación BPEL que se va a invocar.	Migrar el cliente EJB

Tabla 4. Información adicional sobre la migración de clientes (continuación)

Tipo de cliente	Fuente de información adicional
Cliente WSIF que utiliza el enlace de proceso de EJB	Migrar el cliente de enlace de proceso EJB
API de EJB del coreógrafo de procesos de negocio genérica	Migrar el cliente de la API de EJB del coreógrafo de procesos de negocio genérica
API de mensajería del coreógrafo de procesos de negocio genérica	Migrar el cliente de la API de mensajería del coreógrafo de procesos de negocio genérica
Otro proceso BPEL del mismo módulo	No disponible: conectar entre sí los componentes BPEL mediante el editor de ensamblajes
Otro proceso BPEL de otro módulo	No disponible: crear una importación con enlace de SCA en el módulo de referencia y configurar su enlace de modo que apunte a la exportación con enlace de SCA que cree a continuación en la opción 1

Es importante tener en cuenta que si el proceso de negocio se pasa una referencia a sí mismo fuera de su módulo (a través de una referencia de servicio), siempre deberá seguir la opción 1 situada más abajo (siempre puede realizar más de una de estas opciones) para crear una exportación con un enlace SCA para el proceso de negocio. Solo un proceso de negocio por módulo puede pasar su referencia de servicio fuera del módulo porque la exportación debe marcarse como la exportación predeterminada del módulo. Esto se hace especificando "true" para el atributo llamado "default" de una exportación como en:

Referencia de punto final predeterminada

Debe marcar manualmente esta exportación de proceso de negocio como el valor por omisión pulsando la exportación con el botón derecho del ratón en la vista Integración empresarial, seleccionando **Abrir con** y después **Editor de texto**.

Opción 1 de migración para EJB y el enlace de procesos EJB:

La primera opción de migración del enlace de proceso de EJB de WebSphere Studio Application Developer Integration Edition permite a otro componente del mismo módulo acceder a los procesos de negocio.

En el editor de ensamblajes, conecte este otro componente con el componente BPEL:

1. Seleccione el elemento **Conexión** en la barra de herramientas.
2. Pulse el otro componente para seleccionarlo como origen de la conexión.
3. Pulse el componente **BPEL SCA** para seleccionarlo como destino de la conexión.
4. Guarde el diagrama de ensamblaje.

Opción 2 de migración para EJB y el enlace de procesos EJB:

La segunda opción de migración del enlace de proceso de EJB de WebSphere Studio Application Developer Integration Edition permite a otros clientes y módulos SCA acceder a los procesos de negocio.

Nota: Estos pasos son obligatorios si se utilizarán las API del coreógrafo de procesos de negocio genéricas para invocar el proceso de negocio.

La exportación con enlace de SCA hace que otros módulos SCA puedan acceder a un componente SCA. Para crear una exportación con enlace de SCA:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una exportación con enlace de SCA para cada interfaz de proceso BPEL para la que se haya generado un enlace de EJB en WebSphere Studio Application Developer Integration Edition:
 - a. Pulse con el botón derecho en el componente BPEL del editor de ensamblajes.

- b. Seleccione **Exportar...**
- c. Seleccione **Enlace de SCA**.
- d. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
- e. Una vez creada la exportación de SCA, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
- f. Guarde el diagrama de ensamblaje.

Opción 3 de migración para EJB y el enlace de procesos EJB:

La tercera opción de migración del enlace de proceso de EJB de WebSphere Studio Application Developer Integration Edition permite a una entidad no SCA (por ejemplo, un JSP o un cliente Java) acceder a los módulos.

La referencia autónoma hace que un cliente externo pueda acceder a un componente SCA. Para crear una referencia autónoma:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una referencia autónoma para cada interfaz de proceso BPEL para la que se haya generado un enlace de EJB en WebSphere Studio Application Developer Integration Edition:
 - a. Seleccione el elemento **Referencias autónomas** en la barra de herramientas.
 - b. Pulse el lienzo del editor de ensamblajes para crear una entidad SCA de referencias autónomas.
 - c. Seleccione el elemento **Conexión** en la barra de herramientas.
 - d. Pulse la entidad **Referencias autónomas** para seleccionarla como origen de la conexión.
 - e. Pulse el componente **BPEL SCA** para seleccionarlo como destino de la conexión.
 - f. Verá una alerta **Se creará una referencia coincidente en el nodo de origen. ¿Desea continuar?**; pulse **Aceptar**.
 - g. Seleccione la entidad **Referencias autónomas** que acaba de crear y seleccione el panel de contenido **Descripción** en la vista Propiedades.
 - h. Expanda el enlace **Referencias** y seleccione la referencia que acaba de crear. Se muestran el nombre y la descripción de la referencia, que pueden modificarse según convenga.
 - i. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
 - j. Guarde el diagrama de ensamblaje.

Opción 4 de migración para EJB y el enlace de procesos EJB:

La cuarta opción de migración del enlace de proceso de EJB de WebSphere Studio Application Developer Integration Edition permite a un cliente de servicios Web acceder a los procesos comerciales.

La exportación con enlace de servicio Web hace que un cliente de servicios Web externo pueda acceder a un componente SCA. Para crear una exportación con enlace de servicio Web:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una exportación con enlace de SCA para cada interfaz de proceso BPEL para la que se haya generado un enlace de EJB en WebSphere Studio Application Developer Integration Edition:
 - a. Pulse con el botón derecho en el componente BPEL del editor de ensamblajes.
 - b. Seleccione **Exportar...**
 - c. Seleccione **Enlace de servicio Web**.
 - d. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.

- e. Seleccione el transporte: **soap/http** o **soap/jms**.
- f. Una vez creada la exportación de servicios Web, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
- g. Guarde el diagrama de ensamblaje.

Migrar los enlaces de JMS y de proceso JMS:

Se puede migrar los enlaces de JMS y de proceso JMS a la construcción de SCA recomendada.

En WebSphere Studio Application Developer Integration Edition, este tipo de enlace permitía a los clientes comunicarse con un proceso BPEL u otro tipo de servicio enviando un mensaje a un MDB. Tenga presente que este tipo de enlace no era opcional para los procesos de larga ejecución y siempre se seleccionaba. En realidad, este tipo de enlace era el **único** permitido para las interfaces de petición-respuesta de los procesos de larga ejecución. Para los otros tipos de servicio, se generaría un MDB que invocaría el servicio adecuado.

El nombre JNDI empleado por el enlace de JMS era una combinación del nombre, el espacio de nombres destino y la indicación de la hora de inicio de validez del proceso BPEL.

En WebSphere Studio Application Developer Integration Edition, al seleccionar el enlace de JMS como tipo de despliegue para un proceso BPEL, se proporcionaban las opciones siguientes:

- **Fábrica de conexiones JNDI:** el valor predeterminado es `jms/BPECF` (este es el nombre JNDI de la fábrica de conexiones de cola del contenedor de procesos de negocio de destino).
- **Cola de destino JNDI:** el valor predeterminado es `jms/BPEIntQueue` (este es el nombre JNDI de la cola interna del contenedor de procesos de negocio de destino).
- **URL de proveedor JNDI: proporcionado por el servidor o personalizado:** especifique una dirección. El valor predeterminado es `iiop://localhost:2809`.

Existen cinco opciones para migrar el enlace de proceso de JMS de WebSphere Studio Application Developer Integration Edition. El tipo de cliente que accede al servicio determina cuáles de las opciones de migración siguientes se llevan a cabo:

Nota: Una vez efectuados los pasos manuales de migración, el cliente también debe migrarse al nuevo modelo de programación. Consulte el tema pertinente para los tipos de cliente siguientes:

Tabla 5. Información adicional sobre la migración de clientes

Tipo de cliente	Fuente de información adicional
Cliente WSIF que utiliza el enlace de proceso de JMS	Migrar el cliente de la API de mensajería del coreógrafo de procesos de negocio genérica y el cliente de enlace de proceso JMS
API de EJB del coreógrafo de procesos de negocio genérica	Migrar el cliente de la API de EJB del coreógrafo de procesos de negocio genérica
API de mensajería del coreógrafo de procesos comerciales genérica Migrar el negocio	Migrar el cliente de la API de mensajería del coreógrafo de procesos de negocio genérica
Otro proceso BPEL del mismo módulo	No disponible: conectar entre sí los componentes BPEL mediante el editor de ensamblajes
Otro proceso BPEL de otro módulo	No disponible: crear una importación con enlace de SCA en el módulo de referencia y configurar su enlace de modo que apunte a la exportación con enlace de SCA que cree a continuación en la opción 1

Es importante tener en cuenta que si el proceso de negocio se pasa una referencia a sí mismo fuera de su módulo (a través de una referencia de servicio), siempre deberá seguir la opción 1 situada

más abajo (siempre puede realizar más de una de estas opciones) para crear una exportación con un enlace SCA para el proceso de negocio. Solo un proceso de negocio por módulo puede pasar su referencia de servicio fuera del módulo porque la exportación debe marcarse como la exportación predeterminada del módulo. Esto se hace especificando "true" para el atributo llamado "default" de una exportación como en:

Referencia de punto final predeterminada

Debe marcar manualmente esta exportación de proceso de negocio como el valor por omisión pulsando la exportación con el botón derecho del ratón en la vista Integración empresarial, seleccionando **Abrir con** y después **Editor de texto**.

Opción 1 de migración para JMS y el enlace de procesos JMS:

La primera opción de migración del enlace de proceso de JMS de WebSphere Studio Application Developer Integration Edition permite a otro componente del mismo módulo acceder a los procesos de negocio.

En el editor de ensamblajes, conecte este otro componente con el componente BPEL:

1. Seleccione el elemento **Conexión** en la barra de herramientas.
2. Pulse el otro componente para seleccionarlo como origen de la conexión.
3. Pulse el componente **BPEL SCA** para seleccionarlo como destino de la conexión.
4. Guarde el diagrama de ensamblaje.

Opción 2 de migración para JMS y el enlace de procesos JMS:

La segunda opción de migración del enlace de proceso de JMS de WebSphere Studio Application Developer Integration Edition permite a otros clientes y módulos SCA acceder a los procesos de negocio.

La exportación con enlace de SCA hace que otros módulos SCA puedan acceder a un componente SCA. Para crear una exportación con enlace de SCA:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una exportación con enlace de SCA para cada interfaz de proceso BPEL para la que se haya generado un enlace de JMS en WebSphere Studio Application Developer Integration Edition:
 - a. Pulse con el botón derecho en el componente BPEL del editor de ensamblajes.
 - b. Seleccione **Exportar...**
 - c. Seleccione **Enlace de SCA**.
 - d. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
 - e. Una vez creada la exportación de SCA, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
 - f. Guarde el diagrama de ensamblaje.

Opción 3 de migración para JMS y el enlace de procesos JMS:

La tercera opción de migración del enlace de proceso de JMS de WebSphere Studio Application Developer Integration Edition permite a una entidad no SCA (por ejemplo, un JSP o un cliente Java) acceder a los procesos de negocio.

La referencia autónoma hace que un cliente externo pueda acceder a un componente SCA. Para crear una referencia autónoma:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.

2. Cree una referencia autónoma para cada interfaz de proceso BPEL para la que se haya generado un enlace de JMS en WebSphere Studio Application Developer Integration Edition:
 - a. Seleccione el elemento **Referencias autónomas** en la barra de herramientas.
 - b. Pulse el lienzo del editor de ensamblajes para crear una entidad SCA de referencias autónomas.
 - c. Seleccione el elemento **Conexión** en la barra de herramientas.
 - d. Pulse la entidad **Referencias autónomas** para seleccionarla como origen de la conexión.
 - e. Pulse el componente **BPEL SCA** para seleccionarlo como destino de la conexión.
 - f. Verá una alerta **Se creará una referencia coincidente en el nodo de origen. ¿Desea continuar?**; pulse **Aceptar**.
 - g. Seleccione la entidad **Referencias autónomas** que acaba de crear y seleccione el panel de contenido **Descripción** en la vista Propiedades.
 - h. Expanda el enlace **Referencias** y seleccione la referencia que acaba de crear. Se muestran el nombre y la descripción de la referencia, que pueden modificarse según convenga.
 - i. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
 - j. Guarde el diagrama de ensamblaje.

Opción 4 de migración para JMS y el enlace de procesos JMS:

La cuarta opción de migración del enlace de proceso de JMS de WebSphere Studio Application Developer Integration Edition permite a un cliente de servicios Web acceder a los procesos de negocio.

La exportación con enlace de servicio Web hace que un cliente de servicios Web externo pueda acceder a un componente SCA. Para crear una exportación con enlace de servicio Web:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una exportación con enlace de SCA para cada interfaz de proceso BPEL para la que se haya generado un enlace de JMS en WebSphere Studio Application Developer Integration Edition:
 - a. Pulse con el botón derecho en el componente BPEL del editor de ensamblajes.
 - b. Seleccione **Exportar....**
 - c. Seleccione **Enlace de servicio Web**.
 - d. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
 - e. Seleccione el transporte: **soap/http** o **soap/jms**.
 - f. Una vez creada la exportación de servicios Web, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
 - g. Guarde el diagrama de ensamblaje.

Opción 5 de migración para JMS y el enlace de procesos JMS:

La quinta opción de migración del enlace de proceso de JMS de WebSphere Studio Application Developer Integration Edition permite a un cliente JMS acceder a los procesos de negocio.

La exportación con enlace de JMS hace que un cliente JMS externo pueda acceder a un componente SCA. Para crear una exportación con enlace de JMS:

1. Para servicios BPEL, deberá crear y referenciar nuevos recursos de colas, ya que el enlace de procesos 5.1 JMS era bastante diferente del enlace 5.1 JMS estándar. Para servicios no BPEL, puede encontrar los valores que ha seleccionado para el código de despliegue JMS en WebSphere Studio Application Developer Integration Edition 5.1 buscando el archivo WSDL denominado **JMSBinding.wsdl** y **JMSService.wsdl** en el paquete adecuado bajo de la carpeta **ejbModule/META-INF** del proyecto EJB generado e inspeccionando la información de enlace y

servicio capturada allí. Desde el enlace, puede determinar si se han utilizado mensajes de texto u objeto y si se han utilizado enlaces de formato de datos de cliente. Si existen, debe considerar la posibilidad de escribir también un enlace de datos de cliente para la opción **Exportar con enlace JMS** de 6.0. Desde el servicio, puede buscar la fábrica de contexto inicial, el nombre de conexión JNDI, el nombre de destino JNDI y el estilo de destino (cola).

2. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
3. Cree una exportación con enlace de JMS para cada interfaz de proceso BPEL para la que se haya generado un enlace de JMS en WebSphere Studio Application Developer Integration Edition pulsando con el botón derecho del ratón en el componente BPEL del editor de ensamblajes.
4. Seleccione **Exportar...**
5. Seleccione **Enlace de JMS**.
6. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
7. En el siguiente panel (atributos de enlace de exportación de JMS), seleccione **Dominio de mensajería JMS**. Defina este atributo como **Punto a punto**.
8. Seleccione **Cómo se serializan los datos entre Objeto comercial y Mensaje JMS** y especifique los siguientes valores (es aconsejable seleccionar **Texto** en lugar de **Objeto**, ya que el texto, que generalmente es XML, es independiente del entorno de ejecución y permite la integración de servicios entre sistemas independientes):
 - a. En **Texto**, seleccione **Utilizar clase de selector de función JMS predeterminada** o indique el nombre totalmente calificado de la clase de implementación FunctionSelector.
 - b. En **Objeto**, seleccione **Utilizar clase de selector de función JMS predeterminada** o indique el nombre totalmente calificado de la clase de implementación FunctionSelector.
 - c. En **Proporcionado por el usuario**, indique el nombre totalmente calificado de la clase de implementación JMSDataBinding. Deberá seleccionar **Proporcionado por usuario** si la aplicación necesita acceso a propiedades de cabecera JMS que aún no están disponibles en el enlace de importación JMS. En este caso, a continuación deberá crear una clase de enlace de datos de cliente que amplíe el enlace de datos JMS estándar **com.ibm.websphere.sca.jms.data.JMSDataBinding** y añadir código personalizado para acceder a JMSMessage directamente. A continuación, suministrará el nombre de la clase personalizada para este campo. Consulte los ejemplos de JMS de la sección "Crear y modificar enlaces para componentes de importación y exportación" cuyo enlace figura más abajo.
 - d. En **Proporcionado por el usuario**, seleccione **Utilizar clase de selector de función JMS predeterminada** o indique el nombre totalmente calificado de la clase de implementación FunctionSelector.
9. Una vez creada la exportación con enlace de JMS, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
10. Seleccione el panel de contenido Enlace para ver muchas más opciones.
11. Guarde el diagrama de ensamblaje.

Migrar el enlace de servicios Web de IBM (SOAP/JMS):

Se puede migrar el enlace de servicios Web (SOAP/JMS) de IBM para procesos BPEL u otro tipo de servicio a la construcción SCA recomendada.

En WebSphere Studio Application Developer Integration Edition, este tipo de enlace permitía a los clientes comunicarse con un proceso BPEL invocando un servicio Web de IBM, en que el protocolo de comunicación era JMS y el mensaje seguía las reglas de codificación de SOAP.

A continuación figura un ejemplo de los convenios utilizados al generar un servicio Web de IBM (SOAP/JMS) para un servicio BPEL 5.1. El nombre JNDI del servicio Web de IBM generado era una combinación del nombre, el espacio de nombres destino y la indicación de la hora de inicio de validez del proceso BPEL, así como el nombre de la interfaz (tipo de puerto WSDL para el que se generó el

código de despliegue). Por ejemplo, estos atributos se pueden encontrar examinando las propiedades del proceso BPEL en las pestañas de contenido Descripción y Servidor del editor BPEL:

Tabla 6. Espacio de nombres generado

Nombre de proceso	MyService
Espacio de nombres destino	http://www.example.com/process87787141/
Válido desde	Ene 01 2003 02:03:04
Interfaz	ProcessPortType

El espacio de nombres generado en este ejemplo es com/example/www/process87787141/MyService20030101T020304_ProcessPortTypePT.

En WebSphere Studio Application Developer Integration Edition, al seleccionar el enlace de servicios Web de IBM (SOAP/JMS) como tipo de despliegue para un proceso BPEL u otro tipo de servicio, se proporcionaban las opciones siguientes:

- El valor predeterminado de estilo de documento era **DOCUMENT** / **otra opción: RPC**.
- El valor predeterminado de uso de documento era **LITERAL** / **otra opción: ENCODED**.
- El valor de URL de proveedor JNDI era **Proporcionado por el servidor** o **Personalizado** (debe especificarse una dirección, siendo el valor predeterminado iiop://localhost:2809).
- El valor predeterminado de estilo de destino era **queue** / **otra opción: topic**.
- El valor predeterminado de fábrica de conexiones JNDI era **jms/qcf** (este es el nombre JNDI de la fábrica de conexiones de cola de la cola MDB generada).
- El valor predeterminado de cola de destino JNDI era **jms/queue** (este es el nombre JNDI de la cola MDB generada).
- El valor predeterminado de puerto de escucha MDB era *<nombre de proyecto de servicio>***MdbListenerPort**.

En el proyecto EJB generado, pero no en el propio proyecto de servicio, se crea un archivo WSDL con la especificación del enlace SOAP/JMS de servicios Web de IBM. Esto significa que es preciso localizar manualmente este archivo y copiarlo en el proyecto de módulo de integración empresarial si es importante que el código de cliente de servicios Web de IBM no cambie. Por omisión, este archivo WSDL se creaba en el proyecto EJB en ejbModule/META-INF/wsdl/*<nombre proceso comercial>*_<nombre de tipo de puerto de interfaz de proceso comercial>_JMS.wsdl

El tipo de puerto WSDL y los mensajes de la interfaz de proceso de negocio se copian también en este archivo WSDL en lugar de hacer referencia al tipo de puerto WSDL y los mensajes definidos en el proyecto de servicio.

Si es importante que el código de cliente de servicios Web de IBM no se haya modificado tras la migración, la información de este archivo será necesaria para llevar a cabo los pasos manuales de migración.

Existen dos opciones para migrar el enlace de proceso de SOAP/JMS de WebSphere Studio Application Developer Integration Edition. Deberá elegirse entre migrar el cliente al modelo de programación de SCA o dejarlo como cliente de servicios Web:

Nota: Una vez efectuados los pasos manuales de migración, el cliente también debe migrarse al nuevo modelo de programación. Consulte el tema pertinente para los tipos de cliente siguientes:

Tabla 7. Información adicional sobre la migración de clientes

Tipo de cliente	Fuente de información adicional
Cliente de servicios Web de IBM	Migrar el cliente de servicios Web de IBM (SOAP/JMS)

Opción de migración 1 para el enlace de servicios Web (SOAP/JMS) de IBM:

La primera opción de migración del enlace de proceso de SOAP/JMS de WebSphere Studio Application Developer Integration Edition permite a un cliente de servicios Web acceder al servicio.

La exportación con enlace de servicio Web hace que un cliente de servicios Web externo pueda acceder a un componente SCA. Para crear una exportación con enlace de servicio Web:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una Exportación con enlace SCA para cada interfaz de servicio para la que se haya generado un enlace de servicio Web de IBM (SOAP/JMS) en WebSphere Studio Application Developer Integration Edition:
 - a. Pulse con el botón derecho en el componente SCA del editor de ensamblajes.
 - b. Seleccione **Exportar....**
 - c. Seleccione **Enlace de servicio Web**.
 - d. Si hay varias interfaces para el componente, seleccione la que le permita exportar con este tipo de enlace.
 - e. Seleccione el transporte **soap/jms**.
3. Una vez creada la exportación de servicios Web, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
4. Guarde el diagrama de ensamblaje.
5. Seleccione el panel de contenido de enlace y verá que se ha generado directamente en la carpeta de proyecto del módulo un enlace y un servicio WSDL de servicio Web de IBM. Se denominará *componente exportado* Export nombre de tipo de puerto WSDL Jms_Service.wsdl. Si examina ese archivo encontrará que se utiliza de manera predeterminada el enlace con envoltura de documento/literal, ya que es el estilo preferido en 6.0. Es el WSDL que los clientes de servicios Web de IBM utilizarán para invocar el servicio.
6. Siga estos pasos para generar un enlace de servicio Web y un servicio nuevos si desea preservar el código del cliente:
 - a. Copie el archivo WSDL 5.1 del proyecto EJB generado en 5.1 situado en `ejbModule/META-INF/wsdl/nombre de proceso de negocio/nombre de tipo de puerto de interfaz de proceso de negocioJMS.wsdl` en el proyecto de módulo de integración de negocio.
 - b. Después de copiar sobre el archivo y de volver a construir el módulo, puede ver mensajes de error ya que los tipos de esquema XML, los mensajes WSDL y los tipos de puerto WSDL utilizados por el servicio Web están duplicados en el archivo WSDL del servicio Web de IBM en 5.1. Para arreglar esto, suprima las definiciones duplicadas del enlace de servicio Web de IBM/WSDL de servicio y en su lugar añada una importación WSDL para el WSDL de interfaz real. **Nota:** es importante tener en cuenta que cuando WebSphere Studio Application Developer Integration Edition generó el código de despliegue de servicio Web de IBM modificó las definiciones de esquema en algunos casos. Esto puede originar incoherencias para los clientes existentes que utilizan el WSDL de servicio Web de IBM. Por ejemplo, el atributo del esquema "elementFormDefault" se estableció en "qualified" en el esquema en línea generado en el WSDL de servicio Web de IBM incluso aunque la definición de esquema original no estuviera calificada. Esto generaría el error siguiente en tiempo de ejecución: WSW3047E: Error: no se puede deserializar el elemento.
 - c. Pulse con el botón derecho sobre el archivo WSDL que acaba de copiar en el módulo de integración de negocio, seleccione **Abrir con** y después **Editor WSDL**.
 - d. Vaya a la pestaña Origen. Suprima todos los mensajes y tipos de puerto WSDL definidos en este archivo.
 - e. Ahora verá el error: El tipo de puerto '<tipo_de_puerto>' especificado para el enlace '<enlace>' no está definido. Para arreglar esto, en el Editor WSDL de la pestaña Gráfico, pulse con el botón derecho en la sección Importaciones y seleccione **Añadir importación**.

- f. En la vista de propiedades de la pestaña General, pulse el botón ... situado a la derecha del campo Ubicación. Busque la interfaz WSDL en la que están ubicados el mensaje WSDL y las definiciones de tipo de puerto y pulse **Aceptar** para importar el WSDL de interfaz en el WSDL de servicio/enlace.
- g. Guarde el archivo WSDL.
- h. Renueve/reconstruya el proyecto. Pase a la perspectiva Integración empresarial. Abra el Diagrama de ensamblaje en el Editor de ensamblaje.
- i. En la vista del explorador de proyectos, expanda el módulo que está migrando y expanda la categoría lógica **Puertos de servicio Web**. Debe ver el puerto que existe en el WSDL de enlace/servicio listado. Arrástrelo y suéltelo en el Editor de ensamblaje.
- j. Elija crear una **Exportación con enlace de servicio Web** y seleccione el nombre de puerto adecuado. Esto creará la Exportación que utiliza el enlace/servicio antiguo de forma que los clientes de servicios Web existentes no tienen que cambiar. Si selecciona la exportación que acaba de crear en el Editor de ensamblaje y va a la vista propiedades, en la pestaña Enlace debe ver que los nombres de servicio y el puerto 5.1 están cumplimentados.
- k. Guarde todos los cambios.
- l. Justo después de desplegar la aplicación puede cambiar la configuración del proyecto Web generado para que coincida con la dirección de servicio 5.1 (deberá hacer estos cambios cada vez que haga cambios en el módulo SCA que impliquen la regeneración de este archivo.) Si mira en la definición de *servicio* WSDL de servicio Web de IBM que está reutilizando de 5.1 verá la dirección de servicio a la que el cliente 5.1 está codificado: `<wsdl:soap:address location="http://localhost:9080/MyServiceWeb/services/MyServicePort"/>`
- m. Para que los artefactos de proyecto Web generados en 6.0 coincidan con esta dirección de servicio antigua, debe modificar el descriptor de despliegue del proyecto Web generado. Abra el descriptor de despliegue en WebSphere Integration Developer y, en la pestaña Servlets, añada una Correlación URL adicional que sea muy parecida a la correlación URL existente para esa exportación, con el mismo nombre de servlet pero con un patrón de URL distinto.
- n. Además, si necesita modificar la raíz de contexto de este proyecto Web para que coincida con la raíz de contexto en la dirección de servicio original (en este ejemplo la raíz de contexto es "MyServiceWeb"), puede abrir el descriptor de despliegue para la Aplicación de empresa J2EE Enterprise en la que está este proyecto Web y cambiar la raíz de contexto de ese módulo Web para que coincida con la raíz de contexto de la antigua dirección de servicio. Puede aparecer el error siguiente que puede pasar por alto: CHKJ3017E: el proyecto Web <NOM PROJ WEB> está correlacionado con una raíz de contexto no válida <RAÍZ CONTEXTO NUEVO> en el proyecto EAR <NOMBRE APL>.

Opción de migración 2 para el enlace de servicios Web (SOAP/JMS) de IBM:

La segunda opción de migración del enlace de proceso de SOAP/JMS de WebSphere Studio Application Developer Integration Edition permite a una entidad no SCA (por ejemplo, un JSP o un cliente Java) acceder a los procesos de negocio.

La referencia autónoma hace que un cliente externo pueda acceder a un componente SCA. Para crear una referencia autónoma:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una referencia autónoma para cada interfaz de proceso BPEL para la que se haya generado un enlace (SOAP/JMS) de servicio Web de IBM en WebSphere Studio Application Developer Integration Edition:
 - a. Seleccione el elemento **Referencias autónomas** en la barra de herramientas.
 - b. Pulse el lienzo del editor de ensamblajes para crear una entidad SCA de referencias autónomas.
 - c. Seleccione el elemento **Conexión** en la barra de herramientas.
 - d. Pulse la entidad **Referencias autónomas** para seleccionarla como origen de la conexión.

- e. Pulse el componente **BPEL SCA** para seleccionarlo como destino de la conexión.
- f. Verá una alerta **Se creará una referencia coincidente en el nodo de origen. ¿Desea continuar?;** pulse **Aceptar**.
- g. Seleccione la entidad **Referencias autónomas** que acaba de crear y seleccione el panel de contenido **Descripción** en la vista Propiedades.
- h. Expanda el enlace **Referencias** y seleccione la referencia que acaba de crear. Se muestran el nombre y la descripción de la referencia, que pueden modificarse según convenga.
- i. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
- j. Guarde el diagrama de ensamblaje.

Migrar el enlace de servicios Web de IBM (SOAP/HTTP):

Se puede migrar el enlace de servicios Web (SOAP/HTTP) de IBM para procesos BPEL u otro tipo de servicio a la construcción SCA recomendada.

En WebSphere Studio Application Developer Integration Edition, este tipo de enlace permitía a los clientes comunicarse con un proceso BPEL invocando un servicio Web de IBM, en que el protocolo de comunicación era HTTP y el mensaje seguía las reglas de codificación de SOAP.

A continuación figura un ejemplo de los convenios utilizados al generar un servicio Web de IBM (SOAP/HTTP) para un servicio BPEL 5.1. El nombre JNDI del servicio Web de IBM generado era una combinación del nombre, el espacio de nombres destino y la indicación de la hora de inicio de validez del proceso BPEL, así como el nombre de la interfaz (tipo de puerto WSDL para el que se generó el código de despliegue). Por ejemplo, estos atributos se pueden encontrar examinando las propiedades del proceso BPEL en las pestañas de contenido Descripción y Servidor del editor BPEL:

Tabla 8. Espacio de nombres generado

Nombre de proceso	MyService
Espacio de nombres destino	http://www.example.com/process87787141/
Válido desde	Ene 01 2003 02:03:04
Interfaz	ProcessPortType

El espacio de nombres generado en este ejemplo es com/example/www/process87787141/MyService20030101T020304_ProcessPortTypePT.

En WebSphere Studio Application Developer Integration Edition, al seleccionar el enlace de servicios Web de IBM (SOAP/HTTP) como tipo de despliegue para un proceso BPEL u otro tipo de servicio, se proporcionaban las opciones siguientes:

- El valor predeterminado de estilo de documento era **RPC / otra opción: DOCUMENT**.
- El valor predeterminado de uso de documento era **ENCODED / otra opción: LITERAL**.
- El valor predeterminado de dirección de direccionador era http://localhost:9080.

En los proyectos Web y EJB generados, pero no en el propio proyecto de servicio, se crea un archivo WSDL con la especificación del enlace y el servicio SOAP/HTTP de servicios Web de IBM. Esto significa que es preciso localizar manualmente este archivo y copiarlo en el proyecto de módulo de integración empresarial si es importante que el código de cliente de servicios Web de IBM no cambie. Por omisión, este archivo WSDL se creaba en el proyecto Web en WebContent/WEB-INF/wsd1/<nombre proceso comercial>_<nombre de tipo de puerto de interfaz de proceso comercial>_HTTP.wsd1

El tipo de puerto WSDL y los mensajes de la interfaz de proceso de negocio se copian también en este archivo WSDL en lugar de hacer referencia al tipo de puerto WSDL y los mensajes definidos en el proyecto de servicio.

Si es importante que el código de cliente de servicios Web de IBM no se haya modificado tras la migración, la información de este archivo será necesaria para llevar a cabo los pasos manuales de migración.

Existen dos opciones para migrar el enlace de proceso de SOAP/HTTP de WebSphere Studio Application Developer Integration Edition. Deberá elegirse entre migrar el cliente al modelo de programación de SCA o dejarlo como cliente de servicios Web:

Nota: Una vez efectuados los pasos manuales de migración, el cliente también debe migrarse al nuevo modelo de programación. Consulte el tema pertinente para los tipos de cliente siguientes:

Tabla 9. Información adicional sobre la migración de clientes

Tipo de cliente	Fuente de información adicional
Cliente de servicios Web de IBM	Migrar el cliente de servicios Web de IBM (SOAP/HTTP)

Opción de migración 1 para el enlace de servicios Web (SOAP/HTTP) de IBM:

La primera opción de migración del enlace de proceso de SOAP/HTTP de WebSphere Studio Application Developer Integration Edition permite a un cliente de servicios Web acceder a los procesos de negocio.

La exportación con enlace de servicio Web hace que un cliente de servicios Web externo pueda acceder a un componente SCA. Para crear una exportación con enlace de servicio Web:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una exportación con enlace de SCA para cada interfaz de proceso BPEL para la que se haya generado un enlace de servicio Web de IBM (SOAP/HTTP) en WebSphere Studio Application Developer Integration Edition pulsando con el botón derecho del ratón en el componente BPEL del editor de ensamblajes.
3. Seleccione **Exportar...**
4. Seleccione **Enlace de servicio Web**.
5. Si hay varias interfaces para el componente, seleccione la que le permita exportar con este tipo de enlace.
6. Seleccione el transporte **soap/http**.
7. Una vez creada la exportación de servicios Web, seleccione la exportación en el editor de ensamblajes y en la vista de propiedades seleccione el panel de contenido **Descripción**. Se muestran el nombre y la descripción de la exportación, que pueden modificarse según convenga.
8. Guarde el diagrama de ensamblaje.
9. Siga estos pasos para generar un enlace de servicio Web y un servicio nuevos si desea preservar el código del cliente:
 - a. Copie el archivo WSDL 5.1 del proyecto EJB generado en 5.1 situado en `ejbModule/META-INF/wsdl/nombre de proceso de negocio/nombre de tipo de puerto de interfaz de proceso de negocio_HTTP.wsdl` en el proyecto de módulo de integración de negocio.
 - b. Después de copiar sobre el archivo y de volver a construir el módulo, puede ver mensajes de error ya que los tipos de esquema XML, los mensajes WSDL y los tipos de puerto WSDL utilizados por el servicio Web están duplicados en el archivo WSDL del servicio Web de IBM en 5.1. Para arreglar esto, suprima las definiciones duplicadas del enlace de servicio Web de IBM/WSDL de servicio y en su lugar añada una importación WSDL para el WSDL de interfaz real. **Nota:** es importante tener en cuenta que cuando WebSphere Studio Application Developer Integration Edition generó el código de despliegue de servicio Web de IBM modificó las definiciones de esquema en algunos casos. Esto puede originar incoherencias para los clientes existentes que utilizan el WSDL de servicio Web de IBM. Por ejemplo, el atributo del esquema "elementFormDefault" se estableció en "qualified" en el esquema en línea generado en el WSDL de

servicio Web de IBM incluso aunque la definición de esquema original no estuviera calificada. Esto generaría el error siguiente en tiempo de ejecución: WSW3047E: Error: no se puede deserializar el elemento.

- c. Pulse con el botón derecho sobre el archivo WSDL que acaba de copiar en el módulo de integración de negocio, seleccione **Abrir con** y después **Editor WSDL**.
- d. Vaya a la pestaña Origen. Suprima todos los mensajes y tipos de puerto WSDL definidos en este archivo.
- e. Ahora verá el error: El tipo de puerto '<tipo_de_puerto>' especificado para el enlace '<enlace>' no está definido. Para arreglar esto, en el Editor WSDL de la pestaña Gráfico, pulse con el botón derecho en la sección Importaciones y seleccione **Añadir importación**.
- f. En la vista de propiedades de la pestaña General, pulse el botón ... situado a la derecha del campo Ubicación. Busque la interfaz WSDL en la que están ubicados el mensaje WSDL y las definiciones de tipo de puerto y pulse **Aceptar** para importar el WSDL de interfaz en el WSDL de servicio/enlace.
- g. Guarde el archivo WSDL.
- h. Renueve/reconstruya el proyecto. Pase a la perspectiva Integración empresarial. Abra el Diagrama de ensamblaje en el Editor de ensamblaje.
- i. En la vista del explorador de proyectos, expanda el módulo que está migrando y expanda la categoría lógica **Puertos de servicio Web**. Debe ver el puerto que existe en el WSDL de enlace/servicio listado. Arrástrelo y suéltelo en el Editor de ensamblaje.
- j. Elija crear una **Exportación con enlace de servicio Web** y seleccione el nombre de puerto adecuado. Esto creará la Exportación que utiliza el enlace/servicio antiguo de forma que los clientes de servicios Web existentes no tienen que cambiar. Si selecciona la exportación que acaba de crear en el Editor de ensamblaje y va a la vista propiedades, en la pestaña Enlace debe ver que los nombres de servicio y el puerto 5.1 están cumplimentados.
- k. Guarde todos los cambios.
- l. Justo después de desplegar la aplicación puede cambiar la configuración del proyecto Web generado para que coincida con la dirección de servicio 5.1 (deberá hacer estos cambios cada vez que haga cambios en el módulo SCA que impliquen la regeneración de este archivo.) Si mira en la definición de servicio WSDL de servicio Web de IBM que está reutilizando de 5.1, verá la dirección de servicio a la que el cliente 5.1 está codificado: `<wsdlsoap:address location="http://localhost:9080/MyServiceWeb/services/MyServicePort"/>`
- m. Para que los artefactos de proyecto Web generados en 6.0 coincidan con esta dirección de servicio antigua, debe modificar el descriptor de despliegue del proyecto Web generado. Abra el descriptor de despliegue en WebSphere Integration Developer y, en la pestaña Servlets, añada una Correlación URL adicional que sea muy parecida a la correlación URL existente para esa exportación, con el mismo nombre de servlet pero con un patrón de URL distinto.
- n. Además, si necesita modificar la raíz de contexto de este proyecto Web para que coincida con la raíz de contexto en la dirección de servicio original (en este ejemplo la raíz de contexto es "MyServiceWeb"), puede abrir el descriptor de despliegue para la Aplicación de empresa J2EE Enterprise en la que está este proyecto Web y cambiar la raíz de contexto de ese módulo Web para que coincida con la raíz de contexto de la antigua dirección de servicio. Puede aparecer el error siguiente que puede pasar por alto: CHKJ3017E: el proyecto Web <NOM PROJ WEB> está correlacionado con una raíz de contexto no válida <RAÍZ CONTEXTO NUEVO> en el proyecto EAR <NOMBRE APL>.

Opción de migración 2 para el enlace de servicios Web (SOAP/HTTP) de IBM:

La segunda opción de migración del enlace de proceso de SOAP/HTTP de WebSphere Studio Application Developer Integration Edition permite a una entidad no SCA (por ejemplo, un JSP o un cliente Java) acceder a los procesos de negocio.

La referencia autónoma hace que un cliente externo pueda acceder a un componente SCA. Para crear una referencia autónoma:

1. Abra el editor de ensamblajes para el módulo creado por el asistente de migración.
2. Cree una referencia autónoma para cada interfaz para la que se haya generado un enlace de servicio Web de IBM (SOAP/HTTP) en WebSphere Studio Application Developer Integration Edition:
 - a. Seleccione el elemento **Referencias autónomas** en la barra de herramientas.
 - b. Pulse el lienzo del editor de ensamblajes para crear una entidad SCA de referencias autónomas.
 - c. Seleccione el elemento **Conexión** en la barra de herramientas.
 - d. Pulse la entidad **Referencias autónomas** para seleccionarla como origen de la conexión.
 - e. Pulse el componente **SCA** para seleccionarlo como destino de la conexión.
 - f. Verá una alerta **Se creará una referencia coincidente en el nodo de origen. ¿Desea continuar?**; pulse **Aceptar**.
 - g. Seleccione la entidad **Referencias autónomas** que acaba de crear y seleccione el panel de contenido **Descripción** en la vista Propiedades.
 - h. Expanda el enlace **Referencias** y seleccione la referencia que acaba de crear. Se muestran el nombre y la descripción de la referencia, que pueden modificarse según convenga.
 - i. Si hay varias interfaces para el proceso, seleccione la que le permita exportar con este tipo de enlace.
 - j. Guarde el diagrama de ensamblaje.

Migrar el enlace de servicios Web de Apache (SOAP/HTTP):

Se puede migrar el enlace de servicios Web de Apache (SOAP/HTTP) para procesos BPEL u otro tipo de servicio a la construcción SCA recomendada.

En WebSphere Studio Application Developer Integration Edition, este tipo de enlace permitía a los clientes comunicarse con un proceso BPEL o con otro tipo de servicio invocando un servicio Web de Apache.

En WebSphere Studio Application Developer Integration Edition, al seleccionar el enlace de servicios Web de Apache como tipo de despliegue para un proceso BPEL u otro tipo de servicio, se proporcionaban las opciones siguientes:

- El valor de estilo de documento era **RPC** (única opción disponible).
- El valor de acción SOAP era **URN:nombre de tipo de puerto WSDL**.
- El valor de dirección era **http://localhost:9080/nombre de proyecto de servicioWeb/servlet/rpcrouter**.
- El valor predeterminado de uso de codificación era **yes** (al establecerse **yes**, el valor de estilo de codificación era **http://schemas.xmlsoap.org/soap/encoding/**).

En el proyecto de servicio se crea un archivo WSDL con la especificación del enlace y el servicio SOAP de Apache. Por omisión, se crea en el mismo directorio que el servicio al que envuelve con el nombre `<nombre de proceso comercial>_<nombre de tipo de puerto de interfaz de proceso comercial>SOAP.wsdl`. El enlace y el servicio utilizan directamente el tipo de puerto WSDL y los mensajes de la interfaz de proceso de negocio. Tras la migración, *no* debe utilizar este WSDL para ninguna otra tarea salvo tal vez utilizar los mismos nombres de espacio de nombres, puerto y servicio en el nuevo WSDL que se generará en la versión 6.

Existen dos opciones para migrar el enlace de proceso de servicios Web de WebSphere Studio Application Developer Integration Edition. Deberá elegirse entre migrar el cliente al modelo de programación de SCA o dejarlo según el modelo de programación de servicios Web de IBM. No hay ningún enlace que sea equivalente al tipo de enlace de servicios Web de Apache (SOAP/HTTP) en el modelo de programación de SCA de V6.

Debe migrar este servicio Web de Apache para utilizar el motor de servicios Web de IBM. Consulte el tema "Migrar el enlace (SOAP/HTTP) de Servicios Web de IBM" para obtener instrucciones acerca de cómo realizar esta migración y de cómo crear un servicio Web de IBM (SOAP/HTTP).

Migrar al modelo de programación de SCA:

Para cualquier código Java de formato libre que interactúe con un servicio WebSphere Studio Application Developer Integration Edition, esta sección mostrará cómo llevar a cabo la migración del modelo de programación de WSIF al nuevo modelo de programación de SCA, en que los datos que fluyen por la aplicación se almacenan en objetos de datos de servicio (SDO) de Eclipse. En esta sección aprenderá a migrar manualmente los tipos de cliente más comunes al modelo de programación nuevo.

Para cualquier proceso BPEL que contenga fragmentos de código Java, en esta sección se describe cómo llevar a cabo la migración de la antigua API de fragmentos de código Java a la nueva API de fragmentos de código Java en que los datos que fluyen por la aplicación se almacenan en objetos de datos de servicio (SDO) de Eclipse. Siempre que es posible el asistente de migración migra automáticamente los fragmentos de código, pero hay fragmentos de código que el asistente de migración no puede migrar por completo, con lo que es preciso realizar una serie de pasos manuales para completar la migración.

A continuación se proporciona un resumen de los cambios en el modelo de programación:

Modelo de programación V5.1

1. Basado en WSIF y WSDL
2. Proxys generados para servicios
3. Manejadores de formato y beans para tipos

Modelo de programación V6.0 (más centrado en Java)

1. Servicios SCA basados en SDO con códigos de doclet
2. Enlaces de interfaz para servicios
3. SDO y enlaces de datos para tipos

Migrar las llamadas de API WSIFMessage a las API SDO:

En la sección siguiente se facilita información detallada sobre cómo realizar la migración del anterior modelo de programación de WebSphere Business Integration Server Foundation Versión 5.1, en que los datos que fluyen por la aplicación se representan como objetos WSIFMessage y se genera una interfaz muy tipificada, al nuevo modelo de programación de WebSphere Process Server Versión 6.0, en que los datos se representan como objetos de datos de servicio (SDO) y no se genera una interfaz muy tipificada.

Tabla 10. Cambios y soluciones para migrar las llamadas de API WSIFMessage a las API SDO

Cambio	Solución
Las clases de envoltura basadas en WSIFMessage ya no se generan para tipos de mensaje WSDL ni tampoco las clases de ayuda de bean Java para los tipos de esquema complejos.	<p>Al escribir código que interactúa con servicios SCA, las API de SDO genéricas deben utilizarse para manipular los mensajes de <code>commonj.sdo.DataObject</code> que albergan los datos que fluyen a través de la aplicación.</p> <p>Las definiciones de mensaje WSDL que tienen un componente con tipo simple ahora estarán representadas por un tipo Java simple que representa directamente el componente en lugar de tener una envoltura alrededor de los datos reales. Si el componente de mensaje único es un tipo complejo, los datos se representan como un <code>DataObject</code> que se adhiere a la definición de tipo complejo.</p> <p>Las definiciones de mensaje WSDL que tienen varios componentes se corresponden a un <code>DataObject</code> con propiedades para todos los componentes de mensaje, donde los <code>complexType</code>s están representados como propiedades 'reference-type' del <code>DataObject</code> padre, accesible a través de los métodos <code>getDataObject</code> y <code>setDataObject</code>.</p>
No deben utilizarse los métodos de obtención rigurosos para los componentes WSIFMessage no los beans Java generados.	Debe utilizarse una API SDO permisiva para obtener las propiedades <code>DataObject</code> .
Los métodos de establecimiento rigurosos para las variables BPEL ya no están disponibles.	Debe utilizarse una API SDO permisiva para establecer las propiedades <code>DataObject</code> .
Ya no deben utilizarse los métodos de obtención permisivos para las propiedades WSIFMessage.	Debe utilizarse una API SDO permisiva para establecer las propiedades <code>DataObject</code> .
Ya no deben utilizarse los métodos de establecimiento permisivos para las propiedades WSIFMessage.	Debe utilizarse una API SDO permisiva para establecer las propiedades <code>DataObject</code> .
Todas las llamadas de API de WSIFMessage deben migrarse a la API de SDO si es posible.	Migre la llamada a una llamada de API de SDO si es posible. Vuelva a diseñar la lógica si no es posible.

Migrar el código de cliente de WebSphere Business Integration Server Foundation:

En esta sección se muestra cómo migrar los distintos tipos de cliente posibles para los tipos de servicio de WebSphere Business Integration Server Foundation 5.1.

Migrar el cliente EJB:

Este tema muestra cómo migrar clientes que utilizan una interfaz genérica EJB para invocar un servicio.

1. Arrastre la exportación con enlace de SCA del módulo migrado y suéltela en el editor de ensamblajes de este nuevo módulo. Se creará una importación con enlace de SCA. Para que un cliente pueda obtener una referencia a esta importación, debe crearse una referencia autónoma.
2. En la paleta, seleccione el elemento Referencias autónomas. Pulse una vez sobre el lienzo del editor de ensamblajes a fin de crear una nueva referencia autónoma para este nuevo módulo.
3. Seleccione la herramienta de conexión y pulse sobre la referencia de servicio; a continuación, pulse **Importar**.
4. Pulse **Aceptar** cuando se le muestre una alerta que le indica que se creará una referencia coincidente en el nodo de origen.
5. Se le preguntará: **Es más sencillo para un cliente Java utilizar una interfaz Java con esta referencia. ¿Desea convertir la referencia WSDL en una referencia Java compatible?**

- a. Responda **Sí** si desea que el cliente busque este servicio y lo convierta temporalmente en una clase Java para invocarlo utilizando una interfaz Java. Esta nueva interfaz Java toma el nombre del tipo de puerto WSDL, en que el paquete de la interfaz se deriva del espacio de nombres del tipo de puerto WSDL. Hay un método definido para cada operación definida en el tipo de puerto WSDL, y cada componente de mensaje WSDL se representa como un argumento para los métodos de interfaz.
 - b. Responda **No** si desea que el cliente busque este servicio y utilice la interfaz `com.ibm.websphere.sca.Service` genérica para invocarlo utilizando la operación de invocación como un servicio SCA genérico.
6. Si lo desea, cambie el nombre de la referencia autónoma por otro más significativo seleccionando el componente Referencias autónomas en el editor de ensamblajes. Vaya a la vista Propiedades, acceda a la pestaña Detalles, desplácese por la información detallada, seleccione la referencia que acaba de crear y modifique el nombre. Recuerde el nombre elegido para esta referencia, porque el cliente necesitará utilizar este nombre al invocar el método `locateService` de la instancia `com.ibm.websphere.sca.ServiceManager`.
 7. Pulse **Guardar** para guardar el diagrama de ensamblaje.

El cliente debe tener este nuevo módulo en la vía de acceso de clases local para poder acceder al módulo EJB migrado en ejecución en el servidor.

A continuación se muestra el aspecto del código de cliente para un servicio de tipo "CustomerInfo":

```
// Crear un ServiceManager nuevo
ServiceManager serviceManager = ServiceManager.INSTANCE;
// Localizar el servicio CustomerInfo
CustomerInfo customerInfoService = (CustomerInfo) serviceManager.locateService ("<name-of-standalone-reference-from-prev
// Invocar el servicio CustomerInfo
System.out.println(" [getMyValue] getting customer info...");
DataObject customer = customerInfoService.getCustomerInfo(customerID);
```

El cliente debe cambiar la construcción del mensaje. Los mensajes se basaban en la clase `WSIFMessage`, pero ahora deben basarse en la clase `commonj.sdo.DataObject`.

Migrar el cliente de enlace de proceso EJB:

Este tema muestra cómo migrar clientes que utilizan el enlace de proceso EJB WSIF para acceder a un servicio BPEL.

Los clientes que utilizó el enlace de proceso EJB para invocar un proceso de negocio deben utilizar ahora la API SCA para invocar el servicio (el proceso de negocio migrado debe tener una Exportación con enlace SCA) o la API de cliente de servicio Web de IBM

para invocar el servicio (el proceso de negocio migrado debe tener una Exportación con enlace de servicio Web.)

Consulte los temas "Migrar el cliente EJB", "Cliente de servicios Web IBM (SOAP/JMS)" o "Cliente de servicios Web IBM (SOAP/HTTP)" para obtener más información acerca de cómo generar esos clientes.

Migrar el cliente de servicios Web de IBM (SOAP/JMS):

Este tema muestra cómo migrar clientes que utilizan las API de servicio Web (SOAP/JMS) para invocar un servicio.

No se necesita migración para los clientes existentes durante la migración. Tenga en cuenta que debe modificar manualmente el proyecto Web generado (crear una correlación de servlet) y que a veces tendrá que modificar la raíz de contexto del proyecto en el descriptor de despliegue de aplicaciones para publicar el servicio en la misma dirección en la que se publicó en WebSphere

Business Integration Server Foundation. Consulte el tema "Migrar el enlace de servicios Web de IBM (SOAP/JMS)".

Es importante tener en cuenta que al contrario que en 5.1 donde se podía generar un proxy de cliente RPC o WSIF, en 6.0 las herramientas solo soportan la generación de clientes RPC porque RPC es la API preferida por 6.0 por encima de la API de WSIF.

Nota: para generar nuevos proxys de cliente desde WebSphere Integration Developer, debe tener instalado un WebSphere Process Server o WebSphere Application Server.

1. Asegúrese de tener instalado un WebSphere Process Server o WebSphere Application Server.
2. En la perspectiva Recursos o Java, localice el archivo WSDL correspondiente a la **Exportación con enlace de servicio Web**, pulse con el botón derecho y seleccione **Servicios Web → Generar cliente** (fíjese en que este asistente es muy parecido al asistente de la versión 5.1.)
3. Para el Tipo de proxy de cliente, elija **Proxy Java** y pulse **Siguiente**.
4. La ubicación del archivo WSDL debería estar cumplimentada. Pulse **Siguiente**.
5. A continuación debe seleccionar las opciones adecuadas para especificar la configuración del entorno de cliente incluyendo el servidor y el tiempo de ejecución del servicio Web, la versión J2EE, el tipo de cliente (Java, EJB, Web, Cliente de aplicaciones.) Pulse **Siguiente**.
6. Concluya los pasos restantes para generar el proxy de cliente.

Migrar el cliente de servicios Web de IBM (SOAP/HTTP):

Este tema muestra cómo migrar clientes que utilizan las API de servicio Web (SOAP/HTTP) para invocar un servicio.

No se necesita migración para los clientes existentes durante la migración. Tenga en cuenta que debe modificar manualmente el proyecto Web generado (crear una correlación de servlet) y que a veces tendrá que modificar la raíz de contexto del proyecto en el descriptor de despliegue de aplicaciones para publicar el servicio en la misma dirección en la que se publicó en WebSphere

Business Integration Server Foundation. Consulte el tema "Migrar el enlace de servicios Web de IBM (SOAP/HTTP)".

Si se han producido cambios en el diseño y desea generar un cliente proxy nuevo, los pasos siguientes le mostrarán cómo hacerlo. Es importante tener en cuenta que al contrario que en 5.1 donde se podía generar un proxy de cliente RPC o WSIF, en 6.0 las herramientas solo soportan la generación de clientes RPC porque RPC es la API preferida por 6.0 por encima de la API de WSIF.

Nota: para generar nuevos proxys de cliente desde WebSphere Integration Developer, debe tener instalado un WebSphere Process Server o WebSphere Application Server.

1. Asegúrese de tener instalado un WebSphere Process Server o WebSphere Application Server.
2. Seleccione el archivo WSDL correspondiente a la **Exportación con enlace de servicio Web**, pulse con el botón derecho y seleccione **Servicios Web → Generar cliente** (fíjese en que este asistente es muy parecido al asistente de la versión 5.1.)
3. Para el Tipo de proxy de cliente, elija **Proxy Java** y pulse **Siguiente**.
4. La ubicación del archivo WSDL debería estar cumplimentada. Pulse **Siguiente**.
5. A continuación debe seleccionar las opciones adecuadas para especificar la configuración del entorno de cliente incluyendo el servidor y el tiempo de ejecución del servicio Web, la versión J2EE, el tipo de cliente (Java, EJB, Web, Cliente de aplicaciones.) Pulse **Siguiente**.
6. Concluya los pasos restantes para generar el proxy de cliente.

Migrar el cliente de servicios Web de Apache (SOAP/HTTP):

Las API de cliente de servicios Web de Apache no son apropiadas para invocar un servicio de WebSphere Integration Developer. Debe migrarse el código de cliente para utilizar las API de cliente de servicios Web de IBM (SOAP/HTTP).

Consulte la sección "Cliente de servicios Web de IBM

(SOAP/HTTP)" para obtener más información.

En 5.1 si se generaba automáticamente un proxy de cliente, el proxy utilizaba las API de WSIF para interactuar con el servicio. En 6.0 las herramientas solo soportan la generación de clientes RPC porque RPC es la API preferida por 6.0 por encima de la API de WSIF.

Nota: para generar nuevos proxys de cliente desde WebSphere Integration Developer, debe tener instalado un WebSphere Process Server o WebSphere Application Server.

1. Asegúrese de tener instalado un WebSphere Process Server o WebSphere Application Server.
2. Seleccione el archivo WSDL correspondiente a la **Exportación con enlace de servicio Web**, pulse con el botón derecho y seleccione **Servicios Web** → **Generar cliente** (fíjese en que este asistente es muy parecido al asistente de la versión 5.1.)
3. Para el Tipo de proxy de cliente, elija **Proxy Java** y pulse **Siguiente**.
4. La ubicación del archivo WSDL debería estar cumplimentada. Pulse **Siguiente**.
5. A continuación debe seleccionar las opciones adecuadas para especificar la configuración del entorno de cliente incluyendo el servidor y el tiempo de ejecución del servicio Web, la versión J2EE, el tipo de cliente (Java, EJB, Web, Cliente de aplicaciones.) Pulse **Siguiente**.
6. Concluya los pasos restantes para generar el proxy de cliente.

Migrar el cliente JMS:

Los clientes que se comunicaban con un servicio 5.1 por medio de la API JMS (enviando un mensaje JMS a una cola) pueden requerir migración manual. Este tema muestra cómo migrar clientes que utilizan las API JMS (enviando un mensaje JMS a una cola) para invocar un servicio.

Debe asegurarse de que la opción **Exportar con enlace JMS** que ha creado en un paso anterior podrá aceptar este mensaje de texto u objeto sin cambios. Puede que sea necesario escribir un enlace de datos personalizado para conseguirlo. Consulte la sección "Migrar los enlaces de JMS y de proceso JMS" para obtener más información.

El cliente debe cambiar la construcción del mensaje. Antes los mensajes se basaban en la clase WSIFMessage, pero ahora deben basarse en la clase `commonj.sdo.DataObject`. Consulte la sección "Migrar llamadas API de WSIFMessage a de SDO" para obtener más detalles acerca de cómo realizar esta migración manual.

Migrar el cliente de la API de EJB del coreógrafo de procesos de negocio genérica:

Este tema muestra cómo migrar clientes que utilizan la API genérica EJB del coreógrafo de procesos 5.1 para invocar un servicio BPEL.

Hay una nueva versión de la API de EJB genérica que utiliza DataObjects como formato de mensaje. El cliente debe cambiar la construcción del mensaje. Antes los mensajes se basaban en la clase WSIFMessage, pero ahora deben basarse en la clase `commonj.sdo.DataObject`. Tenga presente que la API de EJB genérica no ha cambiado de modo significativo, dado que ClientObjectWrapper sigue proporcionando una envoltura de mensaje en torno al formato de mensaje concreto.

Ejemplo: `DataObject dobj = myClientObjectWrapper.getObject();`
`Serie resultante = dobj.getInt("resultInt");`

El nombre JNDI del EJB genérico antiguo que toma objetos WSIFMessage es:

GenericProcessChoreographerEJB

Nombre JNDI: com/ibm/bpe/api/BusinessProcessHome

Interfaz: com.ibm.bpe.api.BusinessProcess

Hay dos EJB genéricos en 6.0 ya que las operaciones de tareas manuales están ahora disponibles como EJB aparte. Los nombres JNDI 6.0 de estos EJB genéricos son:

GenericBusinessFlowManagerEJB

Nombre JNDI: com/ibm/bpe/api/BusinessFlowManagerHome

Interfaz: com.ibm.bpe.api.BusinessFlowManager

HumanTaskManagerEJB

Nombre JNDI: com/ibm/task/api/TaskManagerHome

Interfaz: com.ibm.task.api.TaskManager

Migrar el cliente de la API de mensajería del coreógrafo de procesos de negocio genérica y el cliente de enlace de proceso JMS:

No existe ninguna API de mensajería genérica en WebSphere Process Server 6.0. Consulte la sección "Migrar el JMS y los enlaces de proceso JMS" para elegir un modo diferente de exponer los procesos comerciales a los clientes y reescribir el cliente de acuerdo con el enlace elegido.

Migrar el cliente Web del coreógrafo de procesos de negocio:

Este tema muestra cómo migrar los valores de cliente Web y los JSP personalizados del coreógrafo de procesos 5.1.

El asistente de migración conserva los valores de cliente Web 5.1 y, en el editor Human Task Editor, no puede editarlos. Debe crear nuevos valores de cliente Web y JSP mediante WebSphere

Integration Developer 6.0.

Migrar modificaciones de cliente Web

En 5.1 el usuario podía modificar el aspecto del cliente Web basado en Struts modificando el JSP **Header.jsp** y la hoja de estilo **dwc.css**.

Puesto que el cliente Web 6.0 (redenominado como **Explorador de coreógrafo de procesos de negocio**) está basado en Java Server Faces (JSF) en lugar de en Struts, no es posible migrar automáticamente las modificaciones del cliente Web. Por lo tanto, es recomendable consultar la documentación del "Explorador de coreógrafo de procesos de negocio" para conocer los detalles acerca de la personalización de la versión 6.0 de esta aplicación.

Los JSP definidos por el usuario pueden definirse para procesos de negocio y para actividades de personal. El cliente Web utiliza estos JSP para visualizar los mensajes de entrada y de salida para el proceso y las actividades.

Estos JSP son particularmente útiles cuando:

1. Los mensajes tienen componentes no primitivos para mejorar la capacidad de utilización de la estructura de datos del mensaje.
2. Desea ampliar las posibilidades del cliente Web.

Hay más opciones diferentes disponibles al especificar los valores de cliente Web para un proceso 6.0 por lo que deberá utilizar WebSphere Integration Developer para rediseñar los valores de cliente Web para los procesos y las actividades migrados:

1. Seleccione el lienzo del proceso o una actividad del proceso.
2. En la vista Propiedades, seleccione la pestaña **Cliente** para volver a diseñar los valores de cliente Web.
3. Migrar manualmente los JSP definidos por el usuario:

- a. Consulte la sección "Migrar al modelo de programación SCA" para conocer los cambios en el modelo de programación.
 - b. El cliente Web utiliza las API genéricas para interactuar con procesos de negocio. Consulte las secciones que muestran cómo migrar llamadas a estas API genéricas.
4. Especifique el nombre del JSP nuevo en los valores de cliente Web 6.0 para el proceso

Nota: La correlación de JSP no es necesaria con el Explorador de Coreógrafo de procesos de negocio (BPC) 6.0 porque los DataObjects no necesitan ninguna correlación personalizada.

Migrar los fragmentos de código Java BPEL de WebSphere Business Integration Server Foundation:

Para cualquier proceso BPEL que contenga fragmentos de código Java, en esta sección se describe cómo llevar a cabo la migración de la antigua API de fragmentos de código Java a la nueva API de fragmentos de código Java en que los datos que fluyen por la aplicación se almacenan como objetos de datos de servicio (SDO) de Eclipse.

Consulte la sección "Migrar desde las llamadas API de WSIFMessage a las API de SDO" para conocer los pasos de migración específicos de la transición de WSIFMessage a SDO a realizar.

Siempre que es posible el asistente de migración migra automáticamente los fragmentos de código, pero hay fragmentos de código que el asistente de migración no puede migrar por completo. Por consiguiente, es preciso realizar una serie de pasos manuales adicionales para completar la migración. Consulte el tema Limitaciones para conocer más detalles acerca de los tipos de fragmentos de código Java que deben migrarse manualmente. Siempre que se encuentre uno de estos fragmentos de código, el asistente Migración explicará por qué no puede migrarse automáticamente y emitirá un aviso o un mensaje de error.

En la tabla siguiente se detallan los cambios en el modelo de programación de fragmentos de código Java de BPEL y la API de la coreografía de procesos de la versión 5.1 a la 6.0:

Tabla 11. Cambios y soluciones para migrar fragmentos de código Java de BPEL de WebSphere Business Integration Server Foundation

Cambio	Solución
<p>Las clases de envoltura basadas en WSIFMessage ya no se generan para tipos de mensaje WSDL ni tampoco las clases de ayuda de bean Java para los tipos de esquema complejos.</p>	<p>Es posible acceder a las variables de BPEL directamente por nombre. Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, ahora estas variables representarán directamente el componente en lugar de tener una envoltura alrededor de los datos reales. Las variables cuyo tipo de mensaje tenga varios componentes tendrán una envoltura DataObject alrededor de los componentes (donde la envoltura en WebSphere Application Developer Integration Edition era un WSIFMessage.)</p> <p>Puesto que las variables BPEL pueden utilizarse directamente en fragmentos de código de 6.0, son menos necesarias de lo que lo eran en 5.1.</p> <p>Los métodos de obtención rigurosos para las variables BPEL inicializaban implícitamente el objeto de envoltura WSIFMessage alrededor de los componentes de mensaje. No hay ningún objeto de 'envoltura' para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente: en este caso, las variables de BPEL representan directamente el componente. (en el caso en que el único componente sea un tipo simple XSD, la variable BPEL estará representada como el tipo de envoltura de objeto Java como por ejemplo java.lang.String, java.lang.Integer, etc.) Las variables BPEL con definiciones de mensaje WSDL de varios componentes se tratan de forma distinta: todavía hay una envoltura alrededor de los componentes y esta envoltura DataObject debe inicializarse explícitamente en el fragmento de código Java 6.0 si es que no la ha establecido una operación anterior.</p> <p>Si las variables locales de los fragmentos de código 5.1 tenían el mismo nombre que la variable BPEL, puede haber conflictos, de modo que intente remediar la situación si es posible.</p>
<p>Los objetos WSIFMessage ya no se utilizan para representar variables BPEL.</p>	<p>Si las clases Java personalizadas invocadas desde los fragmentos de código Java tienen un parámetro WSIFMessage, será necesario migrarlo de forma que acepte/devuelva un DataObject.</p>
<p>Los métodos de obtención rigurosos para las variables BPEL ya no están disponibles.</p>	<p>Es posible acceder a las variables directamente por nombre. Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, ahora directamente representará el componente en lugar de tener una envoltura alrededor de los datos reales. Las variables cuyo tipo de mensaje tenga varios componentes tendrán una envoltura DataObject alrededor de los componentes (donde la envoltura en WebSphere Application Developer Integration Edition era un WSIFMessage.)</p>

Tabla 11. Cambios y soluciones para migrar fragmentos de código Java de BPEL de WebSphere Business Integration Server Foundation (continuación)

Cambio	Solución
Los métodos de establecimiento rigurosos para las variables BPEL ya no están disponibles.	Es posible acceder a las variables directamente por nombre. Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, ahora estas variables representarán directamente el componente en lugar de tener una envoltura alrededor de los datos reales. Las variables cuyo tipo de mensaje tenga varios componentes tendrán una envoltura DataObject alrededor de los componentes (donde la envoltura en WebSphere Application Developer Integration Edition era un WSIFMessage.)
Los métodos de obtención permisivos para las variables de BPEL que devuelven un WSIFMessage ya no están disponibles.	<p>Es posible acceder a las variables directamente por nombre. Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, ahora estas variables representarán directamente el componente en lugar de tener una envoltura alrededor de los datos reales. Las variables cuyo tipo de mensaje tenga varios componentes tendrán una envoltura DataObject alrededor de los componentes (donde la envoltura en WebSphere Application Developer Integration Edition era un WSIFMessage.)</p> <p>Tenga en cuenta que había dos variaciones del método <code>getVariableAsWSIFMessage</code>:</p> <pre>getVariableAsWSIFMessage(String variableName)getVariableAsWSIFMessage(String variableName)</pre> <p>Para una actividad de fragmento de código Java, el acceso por omisión es de lectura-escritura. Puede cambiarlo a sólo lectura especificando @bpe.readOnlyVariables con la lista de nombres de las variables en un comentario en el fragmento de código. Por ejemplo, puede establecer las variables B y D a sólo lectura del siguiente modo:</p> <pre>variableB.setString("/x/y/z", variableA.getString("/a/b/c")); // @bpe.readOnlyVariables variableD.setInt("/x/y/z", variableC.getInt("/a/b/c")); // @bpe.readOnlyVariables</pre> <p>Además, si tiene un fragmento de código Java en una condición, las variables son de sólo lectura por omisión, pero puede cambiarlas a lectura-escritura especificando @bpe.readWriteVariables...</p>
Los métodos de establecimiento permisivos para las variables BPEL ya no están disponibles.	Es posible acceder a las variables directamente por nombre. Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, ahora estas variables representarán directamente el componente en lugar de tener una envoltura alrededor de los datos reales. Las variables cuyo tipo de mensaje tenga varios componentes tendrán una envoltura DataObject alrededor de los componentes (donde la envoltura en WebSphere Application Developer Integration Edition era un WSIFMessage.)

Tabla 11. Cambios y soluciones para migrar fragmentos de código Java de BPEL de WebSphere Business Integration Server Foundation (continuación)

Cambio	Solución
Los métodos de obtención permisivos para los componentes de mensaje de variables BPEL no son adecuados para los mensajes de un solo componente y se han cambiado por mensajes de varios componentes.	<p>Migre al método de obtención permisivos para las propiedades de las variables BPEL (de DataObject).</p> <p>Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, la variable BPEL representa directamente el componente y debería accederse directamente a la variable sin utilizar un método de obtención.</p> <p>Había dos variaciones del método <code>getVariablePartAsObject</code>:</p> <pre>getVariablePartAsObject(String variableName, String partName) getVariablePartAsObject(String variableName, String partName,boolean</pre> <p>Para los mensajes de varios componentes, este método proporciona funcionalidad equivalente en 6.0:</p> <pre>getVariableProperty(String variableName, QName propertyName);</pre> <p>En 6.0 no existe el concepto de utilización de una variable para el acceso solo de lectura (lo que sí existía en 5.1 para el primer método que figura más arriba, así como para el segundo método con <code>forUpdate='false'</code>.) La variable se utiliza directamente en el fragmento de código 6.0 y siempre puede actualizarse.</p>
Los métodos de establecimiento permisivos para los componentes de mensaje de variables BPEL no son adecuados para los mensajes de un solo componente y se han cambiado por mensajes de varios componentes.	<p>Migre al método de establecimiento permisivo para las propiedades de las variables BPEL (de DataObject).</p> <p>Tenga en cuenta que para las variables BPEL cuya definición de mensaje WSDL tenga un solo componente, la variable BPEL representa directamente el componente y debería accederse directamente a la variable sin utilizar un método de establecimiento.</p> <p>Es necesario migrar las llamadas al método siguiente:</p> <pre>setVariableObjectPart(String variableName, String partName, Object da</pre> <p>Para los mensajes de varios componentes, este método proporciona funcionalidad equivalente en 6.0:</p> <pre>setVariableProperty(String variableName, QName propertyName,Serializa</pre>
Los métodos de obtención rigurosos para los enlaces de socio de BPEL ya no están disponibles.	Migre a los métodos de obtención permisivos para enlaces de socio de BPEL.
Los métodos de establecimiento rigurosos para los enlaces de socio de BPEL ya no están disponibles.	Migre a los métodos de establecimiento permisivos para enlaces de socio de BPEL.
Los métodos de obtención rigurosos para los conjuntos de correlación de BPEL ya no están disponibles.	<p>Fragmento de código V5.1:</p> <pre>String corrSetPropStr = getCorrelationSetCorrSetAPropertyCust int corrSetPropInt = getCorrelationSetCorrSetBPropertyCustome</pre> <p>Fragmento de código V6.0:</p> <pre>String corrSetPropStr = (String) getCorrelationSetProperty("C int corrSetPropInt = ((Integer) getCorrelationSetProperty ("C</pre>

Tabla 11. Cambios y soluciones para migrar fragmentos de código Java de BPEL de WebSphere Business Integration Server Foundation (continuación)

Cambio	Solución
Se necesita un parámetro adicional para los métodos de obtención con tipo difuminado para las propiedades personalizadas de actividad BPEL.	<p>Fragmento de código V5.1: String val = getActivityCustomProperty("propName");</p> <p>Fragmento de código V6.0: String val = getActivityCustomProperty("name-of-current-act</p>
Se necesita un parámetro adicional para los métodos de establecimiento con tipo difuminado para las propiedades personalizadas de actividad BPEL.	<p>Fragmento de código V5.1: String newVal = "new value"; setActivityCustomProperty("propName", newVal);</p> <p>Fragmento de código V6.0: String newVal = "new value"; setActivityCustomProperty("name-of-current-activity", "prop</p>
El método raiseFault(QName faultQName, Serializable message) ya no existe.	Migre raiseFault(QName faultQName, String variableName) cuando sea posible, de lo contrario migre al método raiseFault(QName faultQName) o cree una variable BPEL nueva para el objeto serializable.

Migrar interacciones con WebSphere Business Integration Adapters:

Si el cliente JMS es un WebSphere Business Integration Adapter, necesitará utilizar las herramientas de Descubrimiento de servicios de empresa para crear la Importación con enlace JMS. Esta Importación utiliza un enlace de datos especial para serializar el SDO con el formato exacto esperado por el WebSphere Business Integration Adapter.

Para acceder a las herramientas de Descubrimiento de servicios de empresa:

1. Vaya a **Archivo** → **Nuevo** → **Otros** → **Integración empresarial** y seleccione **Descubrimiento de servicios de empresa**. Pulse **Siguiente**.
2. Elija **Importador de artefacto WebSphere Business Integration Adapter**. Pulse **Siguiente**.
3. Especifique la vía de acceso del archivo de configuración de WebSphere Business Integration Adapter (.cfg) y el directorio que contiene el esquema XML de los objetos de negocio que utiliza el adaptador. Pulse **Siguiente**.
4. Examine la consulta generada y si es correcta, pulse **Ejecutar consulta**. En la lista **Objetos descubiertos por consulta**, seleccione los objetos que desea añadir (uno por uno) y pulse el botón **>> Añadir**.
5. Acepte los parámetros de configuración para el objeto de negocio y pulse **Aceptar**.
6. Repita el proceso para cada objeto de negocio.
7. Pulse **Siguiente**.
8. Como **Formato de objeto de negocio de tiempo de ejecución**, seleccione **SDO**. Como **Proyecto destino**, seleccione el módulo que acaba de migrar. Deje el campo **Carpeta** en blanco.
9. Pulse **Finalizar**.

Esta herramienta migrará los XSD antiguos al formato esperado por el enlace de datos especial, por lo que debe eliminar los XSD de WebSphere

Business Integration Adapter antiguos y utilice los XSD nuevos. Si el módulo no recibirá mensajes del adaptador, suprima las exportaciones generadas por esta herramienta. Si el módulo no enviará mensajes al adaptador, suprima la Importación. Consulte el centro de información para obtener más información acerca de esta característica.

Migrar interfaces WSDL que tienen tipos de matriz codificados para SOAP:

En esta sección se muestra cómo migrar o manejar esquemas XML que tienen tipos de matriz codificados para SOAP.

Los tipos de matriz codificados para Soap que tienen el estilo RPC se tratarán como secuencias no enlazadas de un tipo concreto en 6.0. No es recomendable crear tipos XSD que hagan referencia a los tipos soapend:Array de ninguna forma ya que el modelo de programación se mueve hacia el estilo documento/literal acomodado en lugar de hacia el estilo RPC.

Habrán casos en los que una aplicación SCA deba invocar un servicio externo que utilice el tipo soapend:Array. No hay forma de evitar esto en algunos casos, así que esta es la manera de manejar esta situación:

Código WSDL de ejemplo:

```
<xsd:complexType name="Vendor">
<xsd:all>
<xsd:element name="name" type="xsd:string" />
<xsd:element name="phoneNumber" type="xsd:string" />
</xsd:all>
</xsd:complexType>
</xsd:schema>
<xsd:complexType name="Vendors">
<xsd:complexContent mixed="false">
<xsd:restriction base="soapenc:Array">
<xsd:attribute wsdl:arrayType="tns:Vendor[]" ref="soapenc:arrayType" xmlnsxsd="http://schemas.xmlsoap.org/wsdl/" />
</xsd:restriction>
</xsd:complexContent>
<xsd:complexType name="VendorsForProduct">
<xsd:all>
<xsd:element name="productId" type="xsd:string" />
<xsd:element name="vendorList" type="tns:Vendors" />
</xsd:all>
</xsd:complexType>
<xsd:complexType name="Product">
<xsd:all>
<xsd:element name="productId" type="xsd:string" />
<xsd:element name="productName" type="xsd:string" />
</xsd:all>
</xsd:complexType>
<message name="doFindVendorResponse">
<part name="returnVal" type="tns:VendorsForProduct" />
</message>
<operation name="doFindVendor">
<input message="tns:doFindVendor" />
<output message="tns:doFindVendorResponse" />
</operation>
```

Código de ejemplo para un cliente de este servicio Web:

```
// Localizar el servicio de proveedor (vendedor) y buscar la operación doFindVendor
Service findVendor=(Service)ServiceManager.INSTANCE.locateService("vendorSearch");
OperationType doFindVendorOperationType=findVendor.getReference().getOperationType("doGoogleSearch");
// Crear el DataObject de entrada
DataObject doFindVendor=DataFactory.INSTANCE.create(doFindVendorOperationType.getInputType());
doFindVendor.setString("productId", "12345");
doFindVendor.setString("productName", "Refrigerator");
// Invocar el servicio FindVendor
```

```

    DataObject findVendorResult = (DataObject)findVendor.invoke(doFindVendorOperationType, doFindVendor);
// Visualizar el resultado
int resultProductId=findVendorResult.getString("productId");
DataObject resultElements=findVendorResult.getDataObject("vendorList");
Sequence results=resultElements.getSequence(0);
for (int i=0, n=results.size(); i
for (int i=0, n=results.size(); i

```

A continuación se proporciona otro ejemplo en el que el tipo root del objeto de datos es soapenc:Array. Observe cómo se crea el DataObject sampleElements utilizando el segundo esquema listado anteriormente. Primero se obtiene el tipo del DataObject y, a continuación, la propiedad de sampleStructElement. En realidad se trata de una propiedad de sustitución y solo se utiliza para obtener una propiedad válida para utilizarla al añadir los DataObjects a la secuencia. En su caso puede utilizarse un patrón como este:

Código WSDL de ejemplo:

```

<s:schema elementFormDefault="qualified" targetNamespace="http://soapinterop.org/xsd">
<s:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
<s:import namespace="http://schemas.xmlsoap.org/wsdl/" />
<s:complexType name="SOAPStruct">
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" form="unqualified" name="varInt" type="s:int" />
<s:element minOccurs="1" maxOccurs="1" form="unqualified" name="varString" type="s:string" />
<s:element minOccurs="1" maxOccurs="1" form="unqualified" name="varFloat" type="s:float" />
</s:sequence>
</s:complexType>
<s:complexType name="ArrayOfSOAPStruct">
<s:complexContent mixed="false">
<s:restriction base="soapenc:Array">
<s:attribute wsdl:arrayType="s0:SOAPStruct[]" ref="soapenc:arrayType" />
</s:restriction>
</s:complexContent>
</s:complexType>
</s:schema>
<wsdl:message name="echoStructArraySoapIn">
<wsdl:part name="inputStructArray" type="s0:ArrayOfSOAPStruct" />
</wsdl:message>
<wsdl:message name="echoStructArraySoapOut">
<wsdl:part name="return" type="s0:ArrayOfSOAPStruct" />
</wsdl:message>
<wsdl:operation name="echoStructArray">
<wsdl:input message="tns:echoStructArraySoapIn" />
<wsdl:output message="tns:echoStructArraySoapOut" />
</wsdl:operation>
<schema targetNamespace="http://sample/elements"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://sample/elements">
<element name="sampleStringElement" type="string"/>
<element name="sampleStructElement" type="any"/>
</schema>

```

Código de ejemplo para un cliente de este servicio Web:

```

//
Crear el DataObject de entrada y obtener la secuencia SDO para el elemento
// any
DataFactory dataFactory=DataFactory.INSTANCE;
DataObject arrayOfStruct = dataFactory.create("http://soapinterop.org/xsd", "ArrayOfSOAPStruct");
Sequence sequence=arrayOfStruct.getSequence("any");
// Obtener la propiedad SDO para el elemento sample que deseamos utilizar
// aquí para llenar la secuencia
// Hemos definido este elemento en un archivo XSD, consulte SampleElements.xsd
DataObject sampleElements=dataFactory.create("http://sample/elements",
"DocumentRoot");
Property property = sampleElements.getType().getProperty("sampleStructElement");

```

```
// Añadir los elementos a la secuencia
DataObject item=dataFactory.create("http://soapinterop.org/xsd", "SOAPStruct");
item.setInt("varInt", 1);
item.setString("varString", "Hello");
item.setFloat("varFloat", 1.0f);
sequence.add(property, item);
item=dataFactory.create("http://soapinterop.org/xsd", "SOAPStruct");
item.setInt("varInt", 2);
item.setString("varString", "World");
item.setFloat("varFloat", 2.0f);
sequence.add(property, item);
// Invocar la operación echoStructArray
System.out.println("[client] invoking echoStructArray operation");
DataObject echoArrayOfStruct = (DataObject)interopTest.invoke("echoStructArray", arrayOfStruct);
// Visualizar el resultado
if (echoArrayOfStruct!=null) {
sequence=echoArrayOfStruct.getSequence("any");
for (int i=0, n=sequence.size(); i<n; i++) {
item=(DataObject)sequence.getValue(i);
System.out.println("[client] item varInt = "+
item.getInt("varInt")+"
varString="+item.getString("varString")+"
varFloat="+item.getFloat("varFloat"));
}
```

Migrar proyectos EJB de WebSphere Business Integration:

In WebSphere Studio Application Developer Integration Edition, los proyectos EJB pueden tener características especiales de WebSphere Business Integration como por ejemplo Mensajería ampliada (CMM) y CMP/A (persistencia gestionada por componente en cualquier parte.) Los descriptores de despliegue para esos proyectos deben migrarse y esta sección muestra cómo realizar esa migración.

Para realizar esta migración, siga estos pasos:

1. Copie el proyecto EJB de WebSphere Business Integration EJB en el área de trabajo 6.0 nueva e impórtela de WebSphere Integration Developer utilizando el asistente **Archivo → Importar → Proyecto existente en el espacio de trabajo**. También puede ejecutar el asistente Migración de J2EE.
2. Cierre todas las instancias de WebSphere Integration Developer que estén ejecutándose en el área de trabajo 6.0.
3. Ejecute el script siguiente que migrará los descriptores de despliegue de Integración empresarial de WebSphere en el proyecto EJB:

En Windows:

```
%WID_HOME%\wstools\eclipse\plugins\com.ibm.wbit.migration.wsadie_6.0.0\WSADIEEJBProjectMigration.bat
```

En Linux:

```
$WID_HOME/wstools/eclipse/plugins/com.ibm.wbit.migration.wsadie_6.0.0/WSADIEEJBProjectMigration.sh
```

Los parámetros siguientes están soportados; el área de trabajo y el nombre de proyecto son obligatorios:

```
Uso:      WSADIEEJBProjectMigration.bat
[-e eclipse-folder] -d workspace -p project
carpeta-eclipse: La ubicación de la carpeta eclipse; normalmente es el 'eclipse'
que se encuentra bajo la carpeta de instalación del producto.
área_trabajo: El área de trabajo que contiene el proyecto WSADIE EJB a migrar.
proyecto: El nombre del proyecto a migrar.
```

Por ejemplo,

```
WSADIEEJBProjectMigration.bat -e "C:\IBM\WID6\eclipse" -d "d:\my60workspace" -p "MyWBIEJBProject"
```

4. Cuando abra WebSphere Integration Developer deberá renovar el proyecto EJB para obtener los archivos actualizados.

5. Busque el archivo **ibm-web-ext.xmi** en el proyecto EJB. Si encuentra uno, asegúrese de que la línea siguiente figure en el archivo bajo el elemento:

```
<webappext:WebAppExtension> element:  
<webApp href="WEB-INF/web.xml#WebApp"/>
```
6. Elimine el código de despliegue antiguo generado en 5.1. Vuelva a generar el código de despliegue siguiendo las directrices de WebSphere Application Server.

Realizar un arreglo manual de colisiones de espacio de nombres:

En WebSphere Studio Application Developer Integration Edition 5.1, la definición de dos tipos XSD o WSDL diferentes con el mismo nombre y espacio de nombres destino estaba soportada. En WebSphere Integration Developer 6.0, no está soportada. Es necesario realizar una migración manual si se encuentran errores de definiciones duplicadas después de construir los proyectos migrados.

Para solucionar este problema, siga estos pasos:

1. Si las definiciones son las mismas, suprima una de ellas y luego borre y reconstruya el proyecto. Corrija los errores que puedan producirse haciendo que los archivos WSDL/XSD existentes señalen al archivo que contiene la definición que *no* ha suprimido.
2. Si las definiciones *no* son las mismas y debe utilizar ambas definiciones en el servicio migrado, redenomine la definición o el espacio de nombres destino. Si en todo el archivo sólo hay unas pocas definiciones duplicadas, es aconsejable cambiar sus nombres. Si todas las definiciones del archivo están duplicadas, es aconsejable cambiar el espacio de nombres destino de todas las definiciones. Borre y reconstruya el proyecto, asegurándose de que los artefactos que desee utilizar en las definiciones que ha modificado hagan referencia al nombre de definición o espacio de nombres nuevo.
3. En caso de tener dos sentencias import para el mismo espacio de nombres en un archivo WSDL, puede solucionar este problema encadenando las importaciones de modo que uno de estos WSDL importe el otro, que a su vez importará el siguiente, etc., a fin de que sólo exista una importación para este espacio de nombres por cada archivo WSDL. A continuación, borre y reconstruya el proyecto.

Suprimir manualmente definiciones de WSIF (Infraestructura de invocación de servicios Web) 5.1:

Una vez realizada la migración de los artefactos origen, debe suprimir todas las definiciones WSDL de enlace y servicio WSIF 5.1 de los proyectos 6.0 que ya no se utilicen. El escenario de consumo de la migración de servicios es el único caso en que se seguirá utilizando un enlace o servicio WSIF.

Los siguientes espacios de nombres WSDL indican que una definición de enlace o servicio es un servicio WSIF 5.1 y que puede descartarse si ya no se utiliza:

Espacio de nombres WSIF EJB:

<http://schemas.xmlsoap.org/wsdl/ejb/>

Espacio de nombres WSIF Java:

<http://schemas.xmlsoap.org/wsdl/java/>

Espacio de nombres WSIF JMS:

<http://schemas.xmlsoap.org/soap/jms/>

Espacio de nombres WSIF de proceso comercial:

<http://schemas.xmlsoap.org/wsdl/process/>

Espacio de nombres WSIF de transformador:

<http://schemas.xmlsoap.org/wsdl/transformer/>

Espacio de nombres WSIF IMS:

<http://schemas.xmlsoap.org/wsdl/ims/>

Espacio de nombres WSIF CICS_ECI:

<http://schemas.xmlsoap.org/wsdl/cicseci/>

Espacio de nombres WSIF CICS-EPI:

<http://schemas.xmlsoap.org/wsdl/cicsepi/>

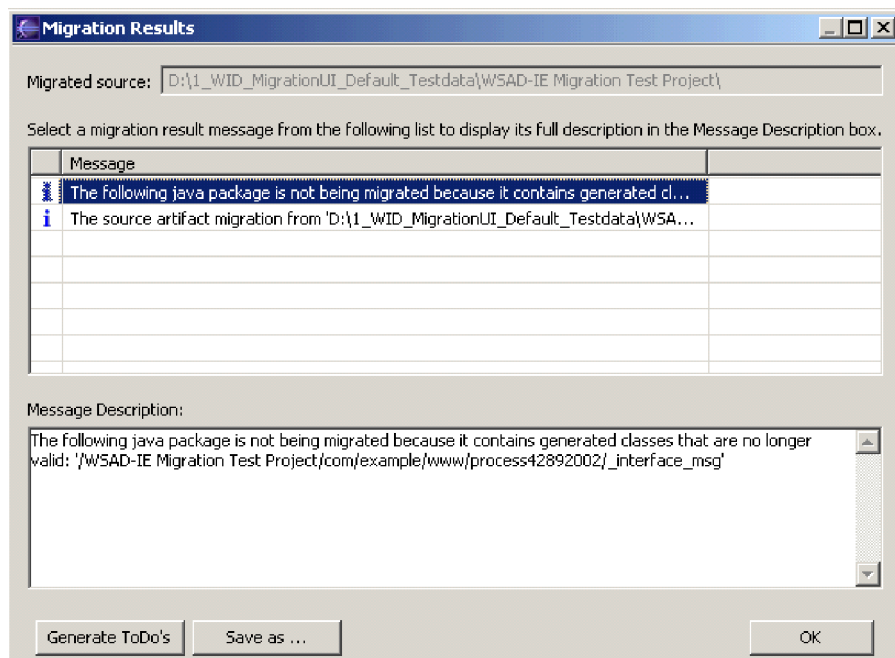
Espacio de nombres WSIF HOD:

<http://schemas.xmlsoap.org/wsdl/hod3270/>

Verificar la migración de artefactos origen

Una vez finalizado correctamente el asistente de migración, se visualiza una lista de mensajes de error, aviso e informativos. Puede emplear estos mensajes para verificar la migración de artefactos origen.

Al finalizar el asistente de migración se muestra la página siguiente:



Examine cada uno de los mensajes para ver si debe realizar alguna acción para corregir de inmediato un artefacto que no se ha migrado por completo.

Para verificar que una parte de la migración está completa, vaya a la perspectiva Integración empresarial y asegúrese de que todos los procesos y las interfaces WSDL del anterior proyecto de servicio aparecen en el nuevo módulo. Construya el proyecto y corrija los errores que impidan su construcción.

Tras realizar los pasos manuales de migración necesarios para finalizar la migración de la aplicación de integración empresarial, exporte la aplicación como un archivo EAR e instálela en un servidor WebSphere Process Server, además de configurar los recursos adecuados.

Efectúe los pasos manuales de migración necesarios para migrar el código de cliente o generar nuevo código de cliente con WebSphere Integration Developer. Asegúrese de que el cliente puede acceder a la aplicación y que la aplicación presenta el mismo comportamiento que en el entorno de ejecución anterior.

Trabajar con los errores de migración de artefactos origen

Si se producen anomalías en la migración de artefactos origen de WebSphere Studio Application Developer Integration Edition, existen diversos métodos para manejar los errores.

A continuación se muestran algunos de los errores de migración de artefactos origen posibles:

- Si recibe el mensaje siguiente:
"Mensaje de error de migración"
Razón: Anomalía de migración muy grave
Mensaje: Póngase en contacto con el representante de IBM

Examine el archivo de anotaciones de WebSphere Integration Developer en la carpeta .metadata de la nueva área de trabajo para ver información detallada sobre el error. Si es posible, resuelva la causa del error, suprima el módulo creado en la nueva área de trabajo y vuelva a intentar efectuar la migración.

Si el asistente de migración finaliza sin mostrar este mensaje, se visualizará una lista de mensajes informativos, de aviso y de error. Estos indican que alguna parte del proyecto de servicio no se ha podido migrar automáticamente y que deben realizarse cambios manuales para completar la migración.

Procedimientos recomendados para el proceso de migración de artefactos origen

Existe una serie de procedimientos recomendados para el proceso de migración de artefactos origen de WebSphere Studio Application Developer Integration Edition.

Las recomendaciones siguientes muestran cómo diseñar los servicios de WebSphere Studio Application Developer Integration Edition para asegurarse de que se migrarán correctamente al nuevo modelo de programación:

- Intente utilizar la actividad **Asignar** en todas las operaciones posibles (en lugar del servicio de transformador, que sólo es necesario cuando se precisa una transformación avanzada). Debe utilizar este procedimiento porque es necesario construir componentes intermedios para que un módulo SCA invoque un servicio de transformador. Además, no hay soporte de herramientas especiales en WebSphere Integration Developer para los servicios de transformador creados en 5.1 (debe utilizar el editor WSDL o XML para modificar el XSLT incorporado en el archivo WSDL si necesita cambiar el comportamiento del servicio de transformador.)
- Especifique un componente por mensaje WSDL igual que por la especificación de interoperatividad de servicios Web (WS-I) y el estilo preferido en 6.0.
- Utilice el estilo de literal de documentos WSDL ya que es el estilo preferido en 6.0.
- Asegúrese de dar un nombre a todos los tipos complejos y de que cada tipo complejo pueda identificarse de forma exclusiva por su espacio de nombres destino y su nombre. A continuación se muestra el modo recomendado de definir tipos complejos y elementos de ese tipo (definición del tipo complejo seguida de una definición de elemento que la utiliza):

```
<schema attributeFormDefault="qualified"
elementFormDefault="unqualified"
targetNamespace="http://util.claimshandling.bpe.samples.websphere.ibm.com"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://util.claimshandling.bpe.samples.websphere.ibm.com">
<complexType name="Duration">
<all>
<element name="hours" type="int"/>
<element name="minutes" type="int"/>
<element name="days" type="int"/>
</all>
</complexType>
<element name="DurationElement" type="tns:Duration"/>
</schema>
```

El ejemplo siguiente representa un tipo complejo anónimo que debe evitarse, ya que puede provocar problemas cuando se serializa un SDO en XML (elemento que contiene una definición de tipo complejo anónimo):

```
<schema attributeFormDefault="qualified"
elementFormDefault="unqualified"
targetNamespace="http://util.claimshandling.bpe.samples.websphere.ibm.com"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://util.claimshandling.bpe.samples.websphere.ibm.com">
<element name="DurationElement">
<complexType>
<all>
<element name="hours" type="int"/>
<element name="minutes" type="int"/>
<element name="days" type="int"/>

```



```

</all>
</complexType>
</element>
</schema>

```

- Si publica un servicio para consumidores externos, genere el código de despliegue del servicio utilizando los servicios Web de IBM (en lugar de emplear Apache SOAP/HTTP) ya que los servicios Web de IBM reciben soporte directo en 6.0 y los servicios Web de Apache no.
- Hay dos modos de organizar los archivos WSDL y XSD en 5.1 para minimizar la cantidad de reorganización necesaria durante la migración. En 6.0, los artefactos compartidos como, por ejemplo, archivos WSDL y XSD deben localizarse en proyectos de integración empresarial (*módulos y bibliotecas de integración empresarial*) para que un servicio de integración empresarial les haga referencia:
 - Guarde todos los archivos WSDL compartidos por varios proyectos de servicio en un proyecto Java al que los proyectos de servicio puedan hacer referencia. Durante la migración a 6.0 creará una biblioteca de Integración empresarial nueva con el mismo nombre que el proyecto Java compartido de 5.1. Copie todos los artefactos del proyecto Java compartido 5.1 en la biblioteca para que el asistente para la migración pueda resolverlos cuando migre los proyectos de servicio que utilizan esos artefactos
 - Guarde una copia local de todos los archivos WSDL/XSD a los que hace referencia un proyecto de servicio en el propio proyecto de servicio. Los proyectos de servicio de WebSphere Studio Application Developer Integration Edition se migrarán a un módulo de integración empresarial en WebSphere Integration Developer y un módulo no puede tener dependencias sobre otros módulos (un proyecto de servicio con dependencias sobre otro proyecto de servicio para compartir archivos WSDL o XSD no se migrará correctamente).
- Evite utilizar la API de mensajería genérica (Generic MDB) del coreógrafo de procesos de negocio (BPC) ya que no se proporcionará en 6.0. En 6.0 *no* habrá disponible una interfaz MDB de enlace tardío.
- Utilice la API de EJB genérica Business Process Choreographer en lugar de invocar los beans de sesión generados que son específicos de una versión determinada de un proceso. Estos beans de sesión no se generarán en V6.0.0.0.
- Si tiene un proceso de negocio con varias respuestas para la misma operación, asegúrese de que si cualquiera de ellas tiene valores de cliente, todas las respuestas para esa operación tengan los mismos valores de cliente ya que en 6.0 solo se da soporte un conjunto de valores de cliente por respuesta de operación.
- Diseñe fragmentos de código Java BPEL según las directrices siguientes:
 - Evite enviar parámetros WSIFMessage a clases Java personalizadas; intente no depender del formato de datos WSIFMessage en la medida de lo posible.
 - Evite utilizar las API de metadatos WSIF si es posible.
- Evite declarar variables locales en fragmentos de código Java de BPEL que tengan el mismo nombre que las variables BPEL. En 6.0 las variables BPEL son directamente accesibles en los fragmentos de código por lo que las variables locales con el mismo nombre pueden crear un conflicto.
- En lo posible, evite crear servicios Java o EJB de arriba abajo ya que el esqueleto Java/EJB que se genera a partir de los mensajes/tipos de puerto de WSDL dependerá de las clases WSIF (por ejemplo: WSIFFormatPartImpl). En su lugar, cree primero las interfaces Java/EJB y genere un servicio alrededor de la clase Java o el EJB (procedimiento de abajo arriba.)
- Evite crear o utilizar interfaces WSDL que hagan referencia al tipo soapenc:Array porque este tipo de interfaz no está soportado de forma nativa en el modelo de programación SCA
- Evite crear tipos de mensajes cuyo elemento de alto nivel sea un tipo de matriz (el atributo maxOccurs es mayor que uno) porque este tipo de interfaz no está soportado de forma nativa en el modelos de programación SCA.
- Defina las interfaces WSDL de forma precisa; en lo posible, evite complexTypes de XSD que tengan referencias al tipo xsd:anyType.
- Para cualquier WSDL y los XSD que genere a partir de un bean EJB o Java, asegúrese de que el espacio de nombres destino sea exclusivo (el nombre de clase y de paquete Java están representados por el

espacio de nombres destino) para evitar colisiones al migrar a WebSphere Process Server V6. En WebSphere Process Server V6, no se permiten dos definiciones WSDL/XSD diferentes que tengan el mismo nombre y el mismo espacio de nombres destino. Esta situación se produce con frecuencia cuando se utilizan el asistente Servicio Web o el mandato Java2WSDL sin especificar explícitamente el espacio de nombres destino (éste será exclusivo con respecto al nombre de paquete del bean EJB o Java, pero no con respecto a la propia clase, por lo que se producirán problemas cuando se genere un servicio Web para dos o más beans EJB o Java en el mismo paquete). La solución consiste en especificar un paquete personalizado para la correlación de espacio de nombres en el asistente Servicio Web o utilizar la opción de línea de mandatos **-namespace** de Java2WSDL para asegurarse de que el espacio de nombres de los archivos generados sea exclusivo para la clase dada.

- Utilice espacios de nombres exclusivos para cada archivo WSDL siempre que sea posible. Existen limitaciones con respecto a la importación de dos archivos WSDL diferentes con el mismo espacio de nombres de acuerdo con la especificación WSDL 1.1, y en WebSphere Integration Developer 6.0 estas limitaciones deben cumplirse estrictamente.

Limitaciones del proceso de migración (para la migración de artefactos origen)

Existe una serie de limitaciones en el proceso de migración de artefactos origen de WebSphere Studio Application Developer Integration Edition.

A continuación se indican las limitaciones del proceso de migración de artefactos origen:

Limitaciones generales

- Este asistente de migración no puede manejar áreas de trabajo de WebSphere Studio Application Developer Integration Edition enteras. Está pensado para migrar un proyecto de servicio de WebSphere Studio Application Developer Integration Edition cada vez.
- El asistente de migración no migra binarios de aplicaciones; solamente migra los artefactos origen que se encuentran en un proyecto de servicio de WebSphere Studio Application Developer Integration Edition.
- Los beans de reglas de negocio están obsoletos en WebSphere Process Server 6.0 pero hay una opción durante la instalación de WebSphere Process Server para instalar el soporte de beans de reglas de negocio obsoletos Business Rule Beans de forma que se ejecutarán "tal cual" en un servidor WebSphere Process Server 6.0. No hay soporte de herramientas para los beans de reglas comerciales, y si desea que los artefactos de bean de reglas comerciales se compilen en las herramientas, debe seguir la documentación de WebSphere Integration Developer para instalar las características obsoletas sobre el servidor de pruebas WebSphere Process Server 6.0 incorporado y después añadir manualmente los archivos jar obsoletos a la vía de acceso de clases del proyecto como jar externos. Debe utilizar las herramientas de reglas comerciales nuevas disponibles en WebSphere Integration Developer para volver a crear las reglas comerciales nuevas correspondientes de acuerdo a la especificación 6.0.
- El soporte de Mensajería ampliada está obsoleto en WebSphere Process Server 6.0, pero hay una opción durante la instalación de WebSphere Process Server para instalar el soporte para la Mensajería ampliada obsoleta de forma que se ejecute "tal cual" en un servidor WebSphere Process Server 6.0. No hay soporte de herramientas para las funciones de Mensajería ampliada antigua, y si desea que los artefactos de la Mensajería ampliada antigua se compilen en las herramientas, debe seguir la documentación de WebSphere Integration Developer para instalar las características obsoletas sobre el servidor de pruebas WebSphere Process Server 6.0 incorporado y después añadir manualmente los archivos jar obsoletos a la vía de acceso de clases del proyecto como "jar externos".
- El enlace de datos JMS estándar suministrado no proporciona acceso a propiedades de cabecera JMS personalizadas. Debe escribirse un enlace de datos personalizado para que los servicios SCA obtengan acceso a las propiedades de cabecera JMS personalizadas.

Limitaciones del modelo de programación SCA

- La especificación de SDO versión 1 no proporciona acceso a la matriz de bytes COBOL o C; esto afectará a los que trabajan con segmentos múltiples de IMS.

- La especificación de SDO versión 1 para la serialización no soporta redefiniciones COBOL ni uniones C.
- Al rediseñar los artefactos origen de acuerdo con el modelo de programación SCA, tenga en cuenta que el estilo WSDL de documento/literal acomodado (que es el estilo predeterminado para los artefactos nuevos creados utilizando las herramientas de WebSphere Integration Developer) no soporta la sobrecarga de métodos. Los otros estilos WSDL todavía están soportados por lo que es recomendable utilizar otro estilo/codificación de WSDL que no sea documento/literal acomodado para estos casos.
- El soporte nativo para las matrices es limitado. Para invocar un servicio externo que expone una interfaz WSDL con tipos soapenc:Array, debe crear una interfaz WSDL que defina un elemento cuyo atributo "maxOccurs" sea mayor que uno (este es el procedimiento recomendado para diseñar un tipo de matriz.)

Limitaciones técnicas del proceso de migración de BPEL

- **Varias respuestas para una operación BPEL:** en WebSphere Business Integration Server Foundation un proceso de negocio podía tener una actividad de recepción y varias actividades de respuesta para la misma operación.
- **Limitaciones de la migración de fragmentos de código Java BPEL:** el modelo de programación ha cambiado de modo significativo de WebSphere Studio Application Developer Integration Edition a WebSphere Integration Developer y no todas las API de WebSphere Studio Application Developer Integration Edition soportadas se pueden migrar directamente a las API de WebSphere Integration Developer correspondientes. Los fragmentos de código Java BPEL pueden contener cualquier lógica Java por lo que tal vez la herramienta de migración automática no pueda convertir todos los fragmentos de código Java al nuevo modelo de programación. La mayoría de las llamadas de API de fragmentos de código estándar se migrarán automáticamente del modelo de programación de los fragmentos de código Java de la 5.1 al modelo de programación de los fragmentos de código Java de la 6.0. Las llamadas de API WSIF se migran a llamadas de API DataObject cuando es posible. Las clases Java personalizadas que aceptan objetos WSIFMessage deberán migrarse manualmente para que acepten y devuelvan objetos `commonj.sdo.DataObject`:
 - **API de metadatos WSIFMessage:** el uso de las API WSIF debe eliminarse siempre que sea posible. El asistente de migración no puede migrar automáticamente los metadatos WSIFMessage y otras API WSIF al SDO equivalente por lo que se necesita la migración manual.
 - **API de EndpointReference/EndpointReferenceType:** estas clases no se migran automáticamente. Se necesita la migración manual ya que los métodos de obtención/establecimiento de enlace de socio utilizan objetos `commonj.sdo.DataObject` en lugar de los objetos `com.ibm.websphere.srm.bpel.wsaddressing.EndpointReferenceType` de 5.1.
 - **Tipos complejos con nombres duplicados:** si una aplicación declara tipos complejos (en WSDL o XSD) con espacios de nombres y nombres locales idénticos, o espacios de nombres distintos pero nombres locales idénticos, es posible que los fragmentos de código Java que utilizan estos tipos no se migren correctamente. Verifique los fragmentos de código para ver que son correctos tras finalizar el asistente de migración.
 - **Tipos complejos con nombres locales idénticos a las clases Java del paquete java.lang:** si una aplicación declara tipos complejos (en WSDL o XSD) con nombres locales idénticos a las clases del paquete `java.lang` de J2SE 1.4.2, es posible que los fragmentos de código Java que utilizan la clase `java.lang` correspondiente no se migren correctamente. Verifique los fragmentos de código para ver que son correctos tras finalizar el asistente de migración.
 - **Variables BPEL con el mismo nombre que variables locales en el fragmento de código Java BPEL:** si definió variables locales con el mismo nombre que las variables BPEL en los fragmentos de código Java BPEL, necesitará arreglar esto manualmente ya que ahora puede acceder a las variables BPEL por el nombre en los fragmentos de código Java y puede producirse un conflicto.
 - **Variables BPEL de sólo lectura y lectura-escritura** - En los fragmentos de código Java, era posible establecer una variable BPEL en "sólo lectura", indicando que los cambios efectuados en ese objeto no afectarían en absoluto al valor de la variable BPEL. También era posible establecer una variable BPEL en "lectura-escritura", indicando que los cambios efectuados en el objeto se reflejarían en la

propia variable BPEL. A continuación se muestran cuatro formas de acceder a un fragmento de código Java como de "sólo lectura" en un fragmento de código Java de BPEL 5-1:

```
getMyInputVariable()  
getMyInputVariable(false)  
getVariableAsWSIFMessage("MyInputVariable")  
getVariableAsWSIFMessage("MyInputVariable", false)
```

A continuación se indican las dos formas de acceder a una variable BPEL como de "lectura-escritura" en cualquier fragmento de código Java de BPEL 5.1:

```
getMyInputVariable(true)  
getVariableAsWSIFMessage("MyInputVariable", true)
```

En la versión 6.0, el acceso de sólo lectura y de lectura-escritura a las variables BPEL se maneja "en función del fragmento de código", lo que significa que puede añadir un comentario especial al fragmento de código Java de BPEL para especificar si las actualizaciones de la variable BPEL deben descartarse o conservarse una vez finalizada la ejecución del fragmento de código. Estos son los valores de acceso por omisión para los tipos de fragmentos de código Java de BPEL 6.0:

```
Actividad de fragmento de código Java de BPEL  
Acceso por omisión: lectura-escritura  
Alterar temporalmente el acceso por omisión con un comentario que contenga:  
@bpe.readOnlyVariables names="variableA,variableB"  
Expresión de fragmento de código Java de BPEL (utilizada en un tiempo de espera,  
condición, etc.)  
Acceso por omisión: sólo lectura  
Alterar temporalmente el acceso por omisión con un comentario que contenga:  
@bpe.readWriteVariables names="variableA,variableB"
```

Durante la migración, estos comentarios se crearán automáticamente cuando se haya accedido a una variable mediante un procedimiento que no sea por omisión en 6.0. En caso de que exista un conflicto (es decir, que se haya accedido a la variable BPEL como "sólo lectura" y como "lectura-escritura" en el mismo fragmento de código), se emitirá un aviso y el acceso se establecerá en "lectura-escritura". Si recibe avisos de este tipo, asegúrese de que establecer el acceso de la variable BPEL en "lectura-escritura" sea correcto en su situación. Si no es correcto, debe corregirlo manualmente mediante el editor BPEL de WebSphere Integration Developer.

- **Propiedades primitivas con múltiples valores en tipos complejos:** en 5.1, las propiedades con múltiples valores se representan mediante matrices del tipo de propiedad. Como tal, las llamadas para obtener y establecer la propiedad utilizan matrices. En 6.0, se utiliza `java.util.List` para esta representación. La migración automática gestionará todos los casos en que la propiedad con múltiples valores es algún tipo de objeto Java, pero si la propiedad es de tipo primitivo Java (`int`, `long`, `short`, `byte`, `char`, `float`, `double` y `boolean`), las llamadas para obtener y establecer toda la matriz no se convierten. En este caso la migración manual puede requerir la adición de un bucle para envolver/desenvolver los primitivos en/de su clase de envoltura Java correspondiente (`Integer`, `Long`, `Short`, `Byte`, `Character`, `Float`, `Double` y `Boolean`) para su uso en el resto del fragmento de código.
- **Creación de instancias de las clases generadas que representan tipos complejos:** en 5.1, las clases generadas de tipos complejos definidos en una aplicación se podían instanciar fácilmente en un fragmento de código Java mediante el constructor sin argumentos predeterminado. A continuación se muestra un ejemplo:

```
MyProperty myProp = new MyProperty();  
InputMessage myMsg = new InputMessageMessage();  
myMsg.setMyProperty(myProp);
```

En 6.0, debe utilizarse una clase de fábrica especial para instanciar estos tipos, o puede emplearse una instancia del tipo continente para crear el subtipo. Si se ha definido una variable de proceso BPEL `InputVariable` con el tipo `InputMessage`, la versión 6.0 del fragmento de código anterior sería:

```

com.ibm.websphere.bo.BOFactory boFactory=
    (com.ibm.websphere.bo.BOFactory)
com.ibm.websphere.sca.ServiceManager.INSTANCE.locateService(
    "com/ibm/websphere/bo/BOFactory");
commonj.sdo.DataObject myMsg =
    boFactory.createByType(getVariableType("InputVariable"));
commonj.sdo.DataObject myProp =
    myMsg.createDataObject("MyProperty");

```

El conversor de fragmentos de código intenta realizar este cambio, pero si el orden en que se realizan las instanciaciones originales no sigue el patrón de primero el padre y después el hijo deberá realizarse la migración manual (es decir, el conversor no intenta reordenar de manera inteligente las sentencias de instanciación del fragmento de código).

- En WebSphere Business Integration Server Foundation 5.1, las referencias dinámicas se representan como componentes de mensaje WSDL de tipo EndpointReferenceType o elemento EndpointReference del espacio de nombres:

```
http://wsaddressing.bpel.srm.websphere.ibm.com
```

Tales referencias se migran al tipo de elemento service-ref estándar desde el espacio de nombres de proceso de negocio estándar:

```
http://schemas.xmlsoap.org/ws/2004/03/business-process/
```

```
http://schemas.xmlsoap.org/ws/2004/08/addressing
```

Consulte la documentación del Editor BPEL para obtener instrucciones acerca de cómo importar manualmente estas definiciones de esquema en el proyecto para que todas las referencias se resuelvan adecuadamente.

- **Tipo de mensaje de variable BPEL** - Debe especificarse un tipo de mensaje WSDL para todas las variables BPEL utilizadas en fragmentos de código Java. Los fragmentos de código Java que acceden a variables BPEL sin el atributo "messageType" especificado no pueden migrarse.

Avisos

La documentación XDoclet incluida en este producto de IBM se utiliza con permiso y está cubierta por la declaración de atribución de copyright siguiente: Copyright (c) 2000-2004, XDoclet Team. Reservados todos los derechos.

Partes basadas en *Design Patterns: Elements of Reusable Object-Oriented Software*, por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, Copyright (c) 1995 por Addison-Wesley Publishing Company, Inc. Reservados todos los derechos.

Derechos restringidos de los usuarios del gobierno de los EE.UU.: utilización, duplicación o revelación restringidas por el contrato GSA ADP Schedule Contract con IBM Corp.

Esta información se ha desarrollado para productos y servicios ofrecidos en los EE.UU, IBM puede no ofrecer los productos, servicios o características tratados en esta documentación en otros países. El representante local de IBM le puede informar acerca de los productos y servicios que actualmente están disponibles en su localidad. Las referencias hechas a productos, programas o servicios de IBM no pretenden afirmar ni dar a entender que únicamente puedan utilizarse dichos productos, programas o servicios de IBM. Puede utilizarse en su lugar cualquier otro producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patente pendientes de aprobación que cubran temas descritos en esta documentación. La entrega de esta documentación no le otorga ninguna licencia sobre dichas patentes. Puede enviar las consultas sobre licencias, por escrito, a la siguiente dirección:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
Estados Unidos de América

Para consultas sobre licencias relativas a la información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM en su país o envíe las consultas, por escrito, a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no se aplica en el Reino Unido ni en ningún otro país en el que tales disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION SUMINISTRA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS O CONDICIONES IMPLÍCITAS DE NO VULNERACIÓN, DE COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunas legislaciones no contemplan la declaración de limitación de responsabilidad, ni implícitas ni explícitas, en determinadas transacciones, por lo que cabe la posibilidad de que esta declaración no se aplique en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente, se efectúan cambios en la información incluida en este documento; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Cualquier referencia hecha en esta información a sitios Web no de IBM se proporciona únicamente para su comodidad y no debe considerarse en modo alguno como promoción de dichos sitios Web. Los materiales de estos sitios web no forman parte de los materiales de IBM para este producto y el uso que se haga de estos sitios web es de la entera responsabilidad del usuario.

IBM puede utilizar o distribuir la información que usted le suministre del modo que IBM considere conveniente sin incurrir por ello en ninguna obligación para con usted.

Los licenciatarios de este programa que deseen obtener información acerca de él con el fin de: (i) intercambiar la información entre los programas creados independientemente y otros programas (incluido este) y (ii) utilizar mutuamente la información que se ha intercambiado, deben ponerse en contacto con:

*Intellectual Property Dept. for Rational Software
IBM Corporation
20 Maguire Road
Lexington, Massachusetts 02421-3112
Estados Unidos de América*

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluyendo en algunos casos el pago de una cantidad.

El programa bajo licencia descrito en esta documentación y todo el material bajo licencia disponible para el mismo, lo proporciona IBM bajo los términos del Acuerdo de Cliente IBM, el Acuerdo de Licencia de Programa Internacional IBM o cualquier otro acuerdo equivalente entre ambas partes.

Los datos de rendimiento incluidos aquí se determinaron en un entorno controlado. Por lo tanto, los resultados que se obtengan en otros entornos operativos pueden variar significativamente. Tal vez se hayan realizado mediciones en sistemas que estén en fase de desarrollo y no existe ninguna garantía de que esas mediciones vayan a ser iguales en los sistemas disponibles en el mercado. Además, es posible que algunas mediciones se hayan estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información concerniente a productos no IBM se ha obtenido de los suministradores de dichos productos, de sus anuncios publicados o de otras fuentes de información pública disponibles. IBM no ha comprobado dichos productos y no puede afirmar la exactitud en cuanto a rendimiento, compatibilidad u otras características relativas a productos no IBM. Las consultas acerca de las posibilidades de los productos no de IBM deben dirigirse a las personas que los suministran.

Todas las declaraciones relativas a la dirección o la intención futura de IBM están sujetas a cambios o anulación sin previo aviso y tan solo representan metas y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlas de la forma más completa posible, los ejemplos pueden incluir nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustra las técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir los programas de ejemplo de cualquier forma, sin tener que pagar a IBM, con intención de desarrollar, utilizar, comercializar o distribuir programas de aplicación que estén en conformidad con la interfaz de programación de aplicaciones (API) de la plataforma operativa para la que están escritos los programas de ejemplo. Los ejemplos no se han probado minuciosamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni dar por sentada la fiabilidad, la facilidad de mantenimiento ni el funcionamiento de los programas. Usted puede copiar, modificar y distribuir los programas de ejemplo

de cualquier forma, sin tener que pagar a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que estén en conformidad con las interfaces de programación de aplicaciones (API) de IBM.

Cada copia o cada parte de los programas de ejemplo o de los trabajos que se deriven de ellos debe incluir un aviso de copyright como se indica a continuación:

(C) (el nombre de su empresa) (el año). Algunas partes de este código se derivan de programas de ejemplo de IBM Corp. (C) Copyright IBM Corp. 2000, 2005. Reservados todos los derechos.

Si está viendo esta información en copia software, es posible que las fotografías y las ilustraciones en color no aparezcan.

Información de interfaces de programación

La información de las interfaces de programación está destinada a ayudarle a crear software de aplicaciones mediante este programa.

Las interfaces de programación de uso general le permiten escribir software de aplicaciones que obtenga los servicios de las herramientas de este programa.

Sin embargo, aquí también puede haber información de diagnóstico, modificación y ajuste. La información de diagnóstico, modificación y ajuste que se proporciona está destinada a ayudarle a depurar el software de las aplicaciones.

Aviso: no utilice la información de diagnóstico, modificación y ajuste como interfaz de programación porque está sujeta a cambios.

Marcas registradas y marcas de servicio

Consulte el sitio <http://www.ibm.com/legal/copytrade.shtml>.



Impreso en España

SC11-3119-01

