



## **Guia de Migração**





## Guia de Migração

**Nota**

Antes de utilizar estas informações e o produto suportado por elas, leia as informações gerais em Avisos no final deste manual.

---

# Índice

<b>Migrando Aplicativos Utilizando o WebSphere Integration Developer . . .</b>	<b>1</b>
Versão do PDF. . . . .	1
Tutorial: Migrando para o WebSphere Integration Developer . . . . .	1
Migrando para o WebSphere Integration Developer . . . . .	1
Migrando para o WebSphere Process Server do WebSphere InterChange Server . . . . .	1

Migrando para o WebSphere Integration Developer a partir do WebSphere MQ Workflow . . . . .	30
Migrando Artefatos de Origem para o WebSphere Integration Developer do WebSphere Studio Application Developer Integration Edition . . . . .	36

<b>Avisos . . . . .</b>	<b>103</b>
-------------------------	------------



---

# Migrando Aplicativos Utilizando o WebSphere Integration Developer

O WebSphere Integration Developer Versão 6.0 fornece as ferramentas necessárias para migrar seu ambiente existente.

Os tópicos a seguir descrevem conceito, referência e instruções passo a passo de migração para o WebSphere Integration Developer:

---

## Versão do PDF

Estas informações de migração também estão disponíveis em PDF.

Este documento também está disponível como um arquivo PDF

É necessário o Adobe Acrobat para visualizá-lo. Uma versão livre deste software está disponível em [www.adobe.com](http://www.adobe.com).

---

## Tutorial: Migrando para o WebSphere Integration Developer

Este tutorial fornece informações sobre como migrar seu projeto de serviço existente e migrá-lo para um projeto de módulo utilizando o WebSphere Integration Developer.

Existe uma lição no tutorial e está em formato de um filme:

- Lição 1: Migrando um Projeto de Serviço mostra como migrar um projeto de serviço existente para um projeto de módulo.

Clique no link a seguir para ativar o tutorial:

**Nota:** O link a seguir não funciona neste Web site. O WebSphere Integration Developer deve ser instalado para que o link funcione corretamente.

Tutorial: Migrando para o WebSphere Integration Developer

---

## Migrando para o WebSphere Integration Developer

O WebSphere Integration Developer Versão 6.0 fornece as ferramentas necessárias para migrar seu ambiente existente.

Os tópicos a seguir descrevem conceito, referência e instruções passo a passo de migração para o WebSphere Integration Developer:

## Migrando para o WebSphere Process Server do WebSphere InterChange Server

A migração do WebSphere InterChange Server para o WebSphere Process Server é suportada por meio das seguintes funções:

**Nota:** Consulte o Release Notes para obter informações sobre as limitações relacionadas à migração neste release do WebSphere Process Server.

- Migração automática de artefatos de origem por meio de ferramentas de migração que podem ser chamadas a partir dos seguintes itens:

- Menu **Arquivo** → **Importar** do WebSphere Integration Developer
- Tela de Boas-vindas do WebSphere Integration Developer
- Assistente de Migração do WebSphere Process Server - Primeiras Etapas
- O utilitário de linha de comandos **reposMigrate**
- Suporte nativo no tempo de execução de muitas APIs do WebSphere InterChange Server
- Suporte para a tecnologia atual do WebSphere Business Integration Adapter para que os adaptadores existentes sejam compatíveis com o WebSphere Process Server

Embora a migração de artefatos de origem seja suportada, são recomendados análise e teste extensivos para determinar se os aplicativos resultantes funcionarão conforme o esperado no WebSphere Process Server, ou se precisarão de novo design pós-migração. Esta recomendação é baseada nas seguintes limitações em paridade funcional entre o WebSphere InterChange Server e esta versão do WebSphere Process Server. Não existe suporte nesta versão do WebSphere Process Server que seja equivalente a estas funções do WebSphere InterChange Server:

- Interface de Acesso
- Resposta de Pedido Síncrono Iniciada pelo Adaptador
- Chamada de Serviço de Envio Síncrono com Tempo Limite
- Chamada de Serviço Async\_in
- Isolamento
- Seqüência de Eventos
- Implementação Rápida/Atualização Dinâmica
- Segurança - Auditoria
- RBAC de Granularidade Fina
- Suporte a Grupo
- Planejador - Pausar Operação
- Os Descritores de Segurança não são migrados
- As transformações de Snippet XML customizadas não são suportadas durante a migração de Mapas e de Gabaritos de Colaboração

## **Caminhos de Migração Suportados para o WebSphere InterChange Server**

As ferramentas de migração do WebSphere Process Server suportam a migração do WebSphere InterChange Server versões 4.2.2, 4.2.3 e 4.3.

Qualquer release anterior à Versão 4.2.2 do WebSphere InterChange Server precisará primeiro migrar para a versão 4.2.2, 4.2.3 ou 4.3 antes de migrar para o WebSphere Process Server.

## **Preparando a Migração do WebSphere InterChange Server**

Antes de migrar para o WebSphere Process Server do WebSphere InterChange Server, primeiro, é necessário assegurar que você tenha preparado corretamente seu ambiente. O WebSphere Process Server fornece as ferramentas necessárias para migrar do WebSphere InterChange Server.

Estas ferramentas de migração podem ser chamadas de:

- Menu **Arquivo** → **Importar** do WebSphere Integration Developer
- Tela de Boas-vindas do WebSphere Integration Developer
- Assistente de Migração do WebSphere Process Server - Primeiras Etapas

A entrada para as ferramentas de migração é um arquivo jar de repositório exportado do WebSphere

InterChange Server. Portanto, antes de acessar as ferramentas de migração por meio de qualquer uma destas opções, primeiro é necessário:



1. Assegurar-se de que você esteja executando uma versão do WebSphere InterChange Server que possa ser migrada para o WebSphere Process Server. Consulte o tópico "Caminhos de Migração Suportados para o WebSphere InterChange Server".
2. Exportar os artefatos de origem do WebSphere InterChange Server para um arquivo jar do repositório utilizando o comando **repos\_copy** do WebSphere InterChange Server conforme descrito na documentação do WebSphere InterChange Server. Este arquivo jar será inserido nas ferramentas de migração.

## Migrando o WebSphere InterChange Server Utilizando o Assistente de Migração

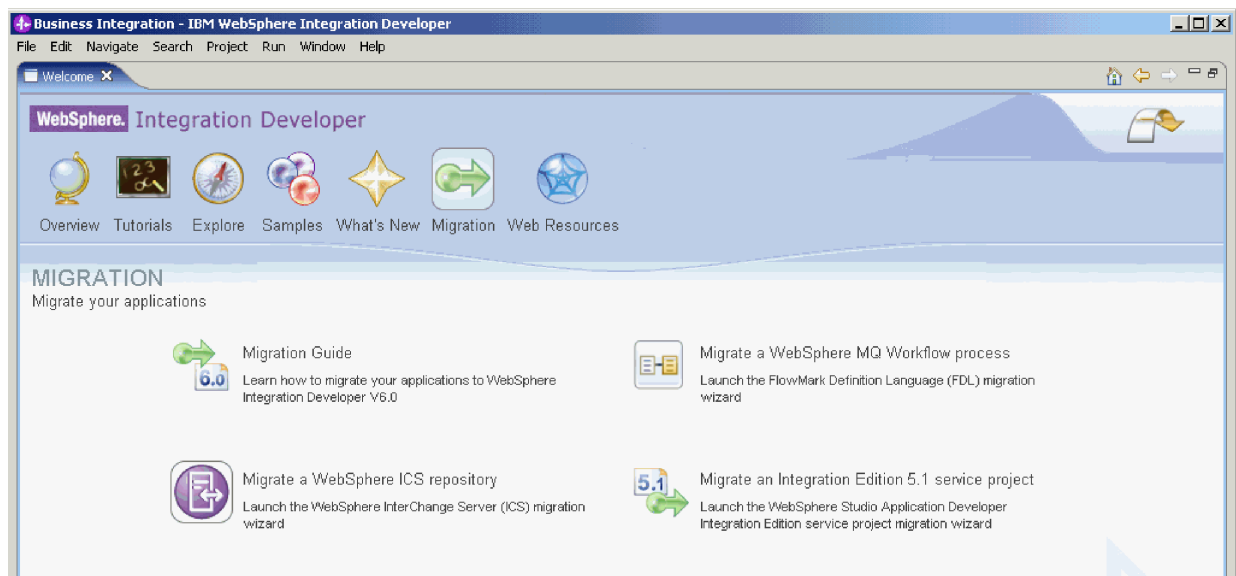
Você pode utilizar o Assistente de Migração do WebSphere Integration Developer para migrar seus artefatos existentes do WebSphere InterChange Server.

Siga estas etapas para utilizar o Assistente de Migração para migrar seus artefatos do WebSphere InterChange Server:

1. Na página de Boas-vindas, clique no  para abrir a página Migração:

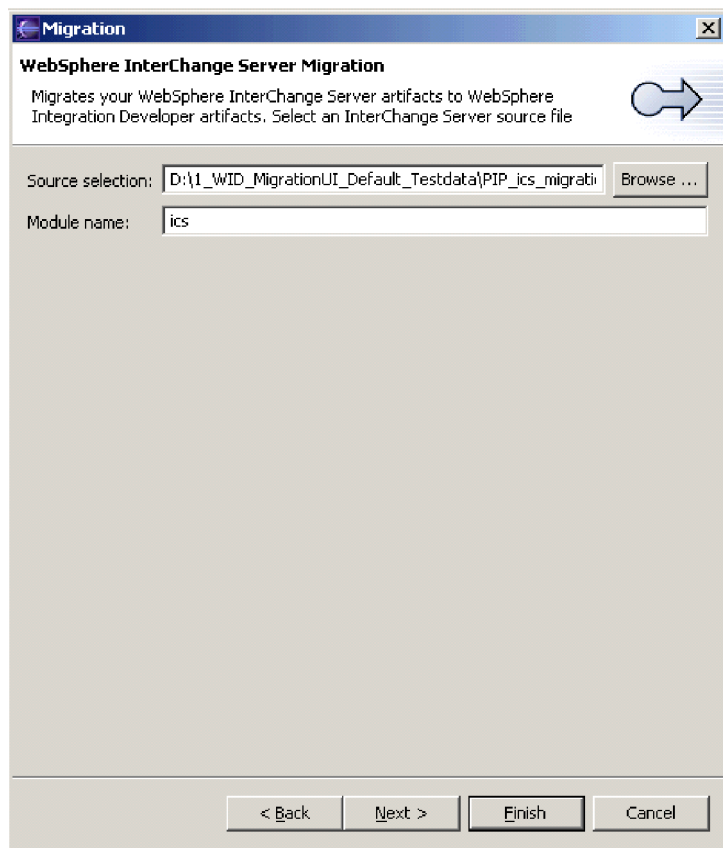


2. Na página Migração, selecione a opção Migrar um Repositório do WebSphere ICS:

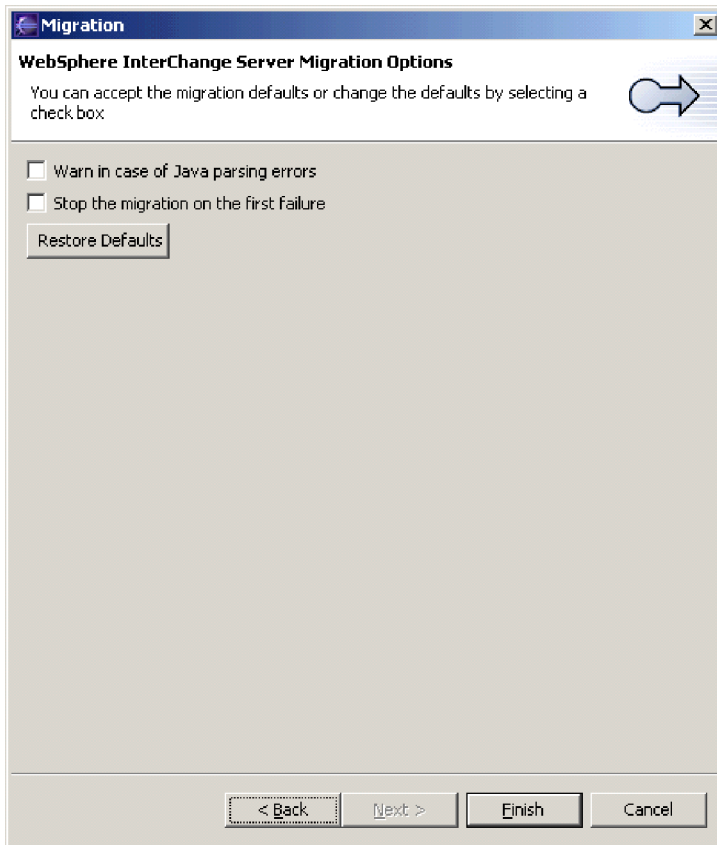


Você também pode ativar o assistente de migração a partir da opção de menu **Arquivo → Importar** e selecionar **Arquivo JAR do WebSphere InterChange Server** e clicar em **Avançar**.

3. É aberto o Assistente de Migração. Digite o nome do arquivo de origem no campo **Seleção de Origem** clicando no botão **Procurar** e navegando para o arquivo. Digite o nome do módulo no campo relevante. Clique em **Avançar**.



4. É aberta a janela **Opções de Migração**. Nesta página é possível aceitar os padrões de migração ou selecionar uma caixa de opções para alterar a opção.



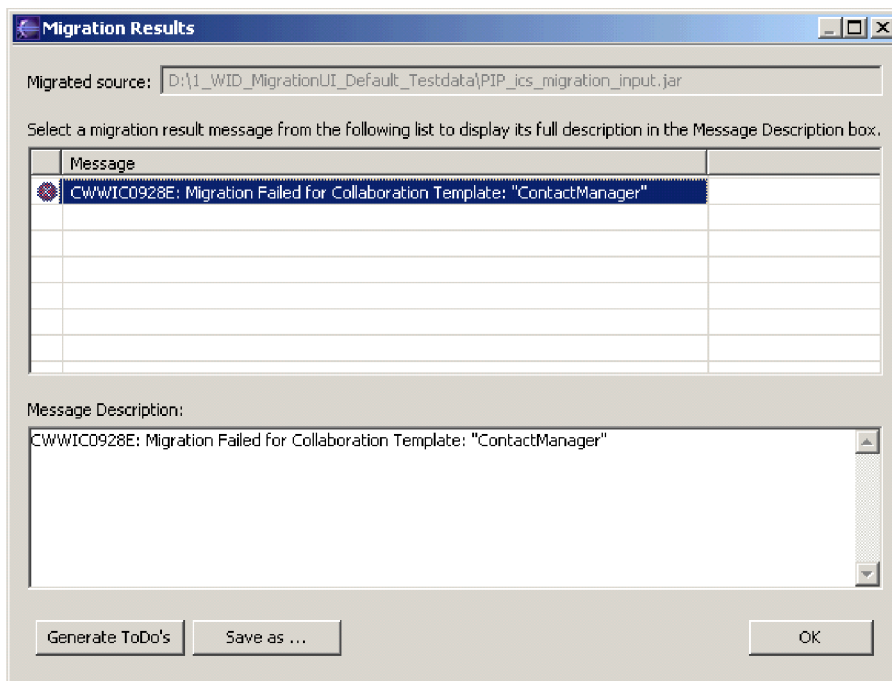
Clique em **Concluir**.

### Verificando a Migração do WebSphere InterChange Server

Se não forem relatados erros durante a migração do arquivo jar do WebSphere InterChange Server, isto indica que a migração dos artefatos foi bem-sucedida. Quando o Assistente de Migração não tiver sido concluído com êxito, será exibida uma lista de mensagens de erro, de aviso e/ou informativas. Estas mensagens podem ser utilizadas para verificar a migração do WebSphere InterChange Server.

**Nota:** Devido à complexidade de migração do WebSphere InterChange Server para o WebSphere Process Server, é altamente recomendável desempenhar um teste extensivo dos aplicativos resultantes em execução no WebSphere Process Server para assegurar que eles funcionem conforme o esperado, antes de colocá-los em produção.

A página a seguir aparece após a conclusão do Assistente de Migração:



Na janela Resultados da Migração, você pode ver uma lista de mensagens de migração. Clique na mensagem para ver uma descrição completa dos detalhes. Se necessário, nesta lista de mensagens, você pode criar uma lista de itens que precisam ser corrigidos, clicando no botão **Gerar Listas de Tarefas**.

## Trabalhando com Falhas de Migração do WebSphere InterChange Server

Se a migração do WebSphere InterChange Server falhar, existem duas maneiras para lidar com as falhas.

**Nota:** Você pode preferir a primeira opção por estar, inicialmente, mais familiarizado com o WebSphere InterChange Server. No entanto, à medida que se torna mais experiente com o WebSphere Process Server e seus novos artefatos, você pode optar por reparar os artefatos migrados no WebSphere Integration Developer.

1. Se a natureza do erro permitir, será possível ajustar os artefatos do WebSphere InterChange Server utilizando o conjunto de ferramentas do WebSphere InterChange Server e exportar o arquivo jar outra vez e tentar novamente a migração.
2. Você pode corrigir erros nos artefatos resultantes do WebSphere Process Server, editando os artefatos no WebSphere Integration Developer.

## Artefatos do WebSphere InterChange Server Manipulados por Ferramentas de Migração

As ferramentas de migração podem migrar automaticamente alguns dos artefatos do WebSphere InterChange Server.

Os seguintes artefatos podem ser migrados:

- **Objetos de Negócios** se tornam objetos de negócios do WebSphere Process Server
- **Mapas** se tornam mapas do WebSphere Process Server
- **Relacionamentos** se tornam relacionamentos e funções do WebSphere Process Server
- **Gabaritos de colaboração** se tornam BPEL e WSDL do WebSphere Process Server
- **Objetos de colaboração** se tornam artefatos SCA do WebSphere Process Server
- **Definições do conector** se tornam artefatos SCA do WebSphere Process Server

As ferramentas de migração podem configurar automaticamente recursos no WebSphere Process Server para os seguintes artefatos/recursos do WebSphere InterChange Server:

- Conjuntos DBConnection
- Bancos de Dados de Relacionamentos
- Entradas do Planejador

As ferramentas de migração *não* manipulam os seguintes artefatos do WebSphere InterChange Server:

- Artefatos de Avaliação de Desempenho

## APIs do WebSphere InterChange Server Suportadas

Além das ferramentas de migração de artefato de origem do WebSphere InterChange Server fornecidas no WebSphere Process Server e no WebSphere Integration Developer, há também suporte para várias das APIs que são fornecidas no WebSphere InterChange Server. As ferramentas de migração trabalham em conjunto com essas APIs do WebSphere InterChange Server, preservando seu código de snippet customizado o máximo possível durante a migração.

**Nota:** Essas APIs são fornecidas apenas para suportar aplicativos do WebSphere InterChange Server migrados até que possam ser modificados para utilizar novas APIs do Process Server. Todas essas APIs do WebSphere InterChange Server são reprovadas e serão removidas em um release futuro. As APIs do WebSphere InterChange Server suportadas no Process Server são listadas abaixo. Essas APIs oferecem no Process Server funções semelhantes às funções que oferecem no WebSphere InterChange Server. Consulte a documentação do WebSphere InterChange Server para obter uma descrição funcional dessas APIs.

### 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)
- executeStoredProcure(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

## Mapeando o WebSphere Process Server DataObject a partir do WebSphere InterChange Server XML

Se você utilizar Legacy Adapters para se conectar ao WebSphere Process Server, o algoritmo a seguir permitirá que você entenda melhor como o WebSphere Process Server DataObject foi criado a partir do WebSphere InterChange Server XML. Estas informações mostram onde os valores de dados foram colocados e também quais valores de dados foram escolhidos para substituir aqueles utilizados no WebSphere InterChange Server.

### Geral

- Para a definição do verbo no ChangeSummary, todas as definições serão feitas com as APIs **markCreate/Update/Delete**.
- Para a definição do verbo no ChangeSummary/EventSummary, os verbos **Create**, **Update**, e **Delete** serão definidos no ChangeSummary, enquanto todos os outros verbos serão definidos no EventSummary.

- Para obter o verbo a partir do ChangeSummary:
  - Se isDelete for verdadeiro, o verbo será **Delete**.

**Nota:** O valor de isCreate será ignorado se isDelete for verdadeiro. Isso pode ocorrer se a criação foi marcada antes do DataObject entrar no hub no ponto em que foi excluído, ou se tanto a criação quanto a exclusão ocorreram no hub.

- Se isDelete for falso e isCreate for verdadeiro, o verbo será **Create**.
  - Se isDelete for falso e isCreate for falso, o verbo será **Update**.
- Para impedir que um DataObject seja identificado como **Create** ao invés de **Update**, conforme pretendido, se o registro estiver ativado, você deve:
  - Suspender o registro durante a criação do DataObject.
  - Recomeçar o registro para a atualização do DataObject (ou utilizar a API **markUpdated**).

### Loading

Loading irá carregar uma XML de tempo de execução do WebSphere

InterChange Server em uma instância do WebSphere

Business Integration BusinessGraph AfterImage.

- Uma instância do BusinessGraph apropriado será criada.
- O Registro de ChangeSummary será ativado, para que sua ativação posterior não limpe as entradas.
- O Registro de ChangeSummary será pausado para impedir que informações não desejadas sejam incluídas no ChangeSummary.
- Os atributos de primeiro nível do BusinessObject serão criados no DataObject (consulte a seção "Processamento de Atributos" abaixo).
- Se o BusinessObject de primeiro nível tiver BusinessObjects filhos, eles serão processados recursivamente.
- Os atributos destes BusinessObjects filhos serão criados no DataObject (consulte a seção "Processamento de Atributos" abaixo).
- O verbo do BusinessObject de primeiro nível será definido no verbo de primeiro nível do BusinessGraph e definido nos resumos.
- O verbo dos BusinessObjects filhos será definido nos resumos.

## Saving

Saving irá salvar uma instância do WebSphere

Business Integration BusinessGraph AfterImage em um XML de tempo de execução WebSphere

InterChange Server. Uma Exceção será emitida se o BusinessGraph de entrada não for AfterImage.

- Uma instância do documento XML do WebSphere InterChange Server será criada.
- Todos os DataObjects excluídos no ChangeSummary serão tratados como se existissem em suas posições originais.

**Nota:** Este processo de cancelamento de exclusão não preservará a ordem da lista.

- O verbo do BusinessObject de primeiro nível será definido como o verbo de primeiro nível do BusinessGraph.
- Os atributos de primeiro nível do BusinessObject serão definidos a partir do DataObject (consulte a seção "Processamento de Atributos" abaixo).
- Se o DataObject tiver DataObjects filhos, eles serão processados recursivamente.
- O verbo dos DataObjects filhos serão manipulados da seguinte forma:
  - Se não houver um verbo para o DataObject filho no EventSummary ou no ChangeSummary, o verbo será definido para "" (cadeia vazia).
  - Se houver um verbo presente no EventSummary, mas não no ChangeSummary, esse verbo será utilizado.
  - Se houver um verbo presente no ChangeSummary, mas não no EventSummary, esse verbo será utilizado.
  - Se houver um verbo presente tanto no ChangeSummary quanto no EventSummary, ele será resolvido da seguinte forma: o valor no ChangeSummary será substituído em relação ao valor do EventSummary.
- Os atributos dos DataObjects filhos serão definidos no BusinessObject (consulte a seção "Processamento de Atributos" abaixo).

## Processamento de Atributos

- Todos os valores não apresentados abaixo serão carregados/salvos ASIS.
- ObjectEventId será carregado/salvo a partir do EventSummary.
- Para **CxBlank** e **CxIgnore**:
  - No lado do WebSphere Business Integration BusinessObject da conversão, **CxBlank** e **CxIgnore** serão definidos/identificados da seguinte forma:
    - **CxIgnore** - não definido ou definido com o valor Java null
    - **CxBlank** - valor dependente do tipo conforme mostrado na tabela abaixo
  - No lado XML do WebSphere InterChange Server da conversão, **CxBlank** e **CxIgnore** serão definidos/identificados da seguinte forma:

Tabela 1. Definindo CxBlank e 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
String/date/longtext	"CxIgnore"	""
Children BusinessObjects	(elemento vazio)	N/D

## Boas Práticas para o Processo de Migração do WebSphere InterChange Server

As diretrizes a seguir destinam-se a ajudá-lo no desenvolvimento de artefatos de integração para o WebSphere InterChange Server. Seguindo estas diretrizes, você pode facilitar a migração de artefatos do WebSphere InterChange Server para o WebSphere Process Server.

Estas recomendações destinam-se a ser utilizadas apenas como um guia para o desenvolvimento de novas soluções de integração. É reconhecido que o conteúdo existente pode não seguir estas diretrizes. Também é entendido que pode haver casos em que é necessário desviar-se destas diretrizes. Neste caso, é necessário ter cuidado para limitar o escopo do desvio para minimizar a quantidade de retrabalho requerido para migrar os artefatos. Observe que as diretrizes descritas aqui não incluem as boas práticas para o desenvolvimento de artefatos do WebSphere

InterChange Server em geral. Elas estão limitadas no escopo às considerações que podem afetar a facilidade em que os artefatos podem ser migrados futuramente.

### Desenvolvimento Geral

Existem várias considerações que se aplicam amplamente à maioria dos artefatos de integração. Em geral, os artefatos que alavancam os recursos fornecidos pelas ferramentas e estão de acordo com os modelos de metadados aplicados pelas ferramentas serão migrados de maneira mais uniforme. Além disso, os artefatos com extensões importantes e dependências externas provavelmente precisarão de mais intervenção manual durante a migração.

A lista a seguir resume as boas práticas para desenvolvimento geral de soluções baseadas no WebSphere InterChange Server para ajudar a facilitar a migração futura:

- Utilizar o WebSphere InterChange Server para soluções de integração de processos em tempo real e automatizadas
- Documentar o design do sistema e do componente
- Utilizar as ferramentas de desenvolvimento para editar artefatos de integração
- Alavancar boas práticas para definir regras com as ferramentas e snippets Java

Embora possa parecer óbvio, é importante que as soluções de integração sigam o modelo de programação e a arquitetura fornecida pelo WebSphere InterChange Server. É melhor ajustado para soluções de integração de processo em tempo real, automatizadas. Além disso, cada um dos componentes de integração no WebSphere InterChange Server desempenha uma função bem definida na arquitetura. Os desvios significativos deste modelo o tornarão mais desafiador para migrar o conteúdo para os artefatos apropriados no WebSphere Process Server.

Outra boa prática geral que aprimorará significativamente o êxito de futuros projetos de migração é documentar o design do sistema. Certifique-se de capturar a arquitetura e design de integração, incluindo os requisitos de design funcional e de qualidade de serviço, as interdependências de artefatos compartilhados entre projetos e também as decisões de design que foram tomadas durante a implementação. Isto ajudará na análise do sistema durante a migração e minimizará os esforços de retrabalho.

Para criar, configurar e modificar definições de artefatos, é importante que apenas as ferramentas de desenvolvimento fornecidas sejam utilizadas. Evite a manipulação manual de metadados de artefatos (ex. edição direta de arquivos XML), que pode danificar o artefato para migração.

Ao desenvolver código Java em gabaritos de colaboração, mapas, utilitários de códigos comuns e outros componentes, existem várias considerações a serem levadas em conta:

- Utilizar apenas as APIs publicadas
- Utilizar o Activity Editor
- Utilizar adaptadores para acessar EISs



- Evitar dependências externas no trecho de código Java
- Seguir as práticas de desenvolvimento de J2EE para portabilidade
- Não efetuar spawn de encadeamentos ou utilizar primitivas de sincronização de encadeamento. Se tiver que fazer isso, estas tarefas deverão ser convertidas para utilizar Beans Assíncronos durante a migração.
- Não fazer nenhuma E/S de disco utilizando java.io.\* Utilizar JDBC para armazenar dados.
- Não desempenhar funções que podem estar reservadas para um contêiner EJB, como E/S de soquete, carregamento de classe, carregamento de bibliotecas nativas, etc. Se tiver que fazer isso, estes snippets precisarão de conversão manual para utilizar as funções de contêiner EJB durante a migração.

Utilizar apenas as APIs publicadas na documentação do produto para os artefatos. Estas tarefas estão descritas detalhadamente nos guias de desenvolvimento do WebSphere InterChange Server. Embora em muitos casos as APIs de compatibilidade serão fornecidas no WebSphere Process Server, apenas as APIs publicadas serão incluídas. Embora o WebSphere InterChange Server possua muitas interfaces internas que um desenvolvedor pode querer utilizar, não há garantia de que elas serão suportadas durante o avanço.

Ao projetar a lógica de negócios e regras de transformação em mapas e gabaritos de colaboração, certifique-se de utilizar a ferramenta Activity Editor para a maior extensão possível. Isto assegurará que a lógica seja descrita por meio de metadados que podem ser mais prontamente convertidos para os novos artefatos. Para operações que você deseja reutilizar nas ferramentas, utilize o recurso “Minhas Coletas” do Activity Editor sempre que possível. Tente evitar bibliotecas de utilitário de código comum desenvolvidas em campo, incluídas como um arquivo Java archive (\*.jar) no caminho de classe do WebSphere InterChange Server, pois elas precisarão ser migradas manualmente.

Em quaisquer trechos de código Java que pode ser desenvolvido, é recomendável que o código seja o mais simples e atômico possível. O nível de sofisticação no código Java deve estar na ordem de script, envolvendo avaliações básicas, operações e cálculos, formatação de dados, conversões de tipos, etc. Se for requerida uma lógica de aplicativo mais extensiva ou sofisticada, será necessário utilizar os EJBs em execução no WebSphere Application Server para encapsular a lógica e utilizar chamadas de serviços da Web para chamá-la do WebSphere InterChange Server. Utilize bibliotecas JDK padrão em vez de bibliotecas de terceiros ou externas que precisariam ser migradas separadamente. Colete também toda a lógica relacionada em um único trecho de código e evite utilizar lógica onde os contextos de conexão e de transação estendem vários trechos de código. Com operações do banco de dados, por exemplo, o código relacionado à obtenção de uma conexão, ao início e fim de uma transação e à liberação da conexão deve estar em um trecho de código.

Em geral, certifique-se de que o código projetado para interagir com um EIS (Enterprise Information System) esteja colocado em adaptadores e não em mapas ou em gabaritos de colaboração. Geralmente, esta é a boa prática para design da arquitetura. Além disso, isto ajudará a evitar pré-requisitos para bibliotecas de terceiros e considerações relacionadas no código, como gerenciamento de conexões e possíveis implementações de JNI (Java Native Interface).

Torne o código o mais seguro possível utilizando a manipulação de exceção apropriada. Torne também o código compatível para execução em um ambiente do servidor de aplicativos J2EE, mesmo que ele esteja em execução em um ambiente J2SE. Siga as práticas de desenvolvimento de J2EE, como evitar variáveis estáticas, execução de spawn em encadeamentos e E/S de disco. Estas são as boas práticas recomendáveis que devem ser seguidas, em geral, mas serão pertinentes à portabilidade.

## Utilitários de Código Comuns

Conforme indicado anteriormente, é recomendável evitar o desenvolvimento de bibliotecas de utilitário de código comum para utilização nos artefatos de integração no ambiente do WebSphere



InterChange Server. Onde é necessária a reutilização de código nos artefatos de integração, é recomendável alavancar o recurso “Minhas Coletas” da ferramenta Activity Editor. Considere também a utilização de EJBs em execução no WebSphere

Application Server para encapsular a lógica e utilizar chamadas de serviços da Web para chamá-la a partir do WebSphere

InterChange Server. Embora seja possível que algumas bibliotecas de utilitários de código comuns sejam executadas apropriadamente no WebSphere

Process Server, o usuário será responsável pela migração dos utilitários customizados.

## **Conjuntos de Conexões com o Banco de Dados**

Os conjuntos de conexões com o banco de dados definidos pelo usuário são muito úteis em mapas e gabaritos de colaboração para consultas de dados simples e para gerenciamento de estados mais sofisticado através de instâncias de processo. Um conjunto de conexões com o banco de dados no WebSphere

InterChange Server será apresentado como um recurso JDBC padrão no WebSphere

Process Server e a função básica será a mesma. No entanto, a maneira que as conexões e transações são gerenciadas pode ser diferente.

Para maximizar a portabilidade futura, evite manter transações do banco de dados ativas em nós de snippet Java em um gabarito ou mapa de colaboração. Por exemplo, o código relacionado à obtenção de uma conexão, ao início e fim de uma transação e à liberação da conexão deve estar em um trecho de código.

## **Business Objects**

As considerações primárias para o desenvolvimento de objetos de negócios será utilizar apenas as ferramentas fornecidas para configurar artefatos, utilizar tipos de dados e comprimentos explícitos para atributos de dados e utilizar apenas as APIs documentadas.

Os objetos de negócios com o WebSphere Process Server são baseados em SDOs (Service Data Objects), que utilizam atributos de dados altamente especificados. Para objetos de negócios no WebSphere InterChange Server e adaptadores, os atributos de dados não são altamente especificados e, às vezes, os usuários especificam tipos de dados de cadeia para atributos de dados sem cadeia. Para evitar problemas no WebSphere Process Server, seja explícito na especificação de tipos de dados.

Como os objetos de negócios no WebSphere Process Server podem ser serializados no tempo de execução conforme são transmitidos entre componentes, é importante ser explícito com os comprimentos requeridos para atributos de dados, para minimizar a utilização de recursos do sistema. Por isso, não utilize o comprimento máximo de 255 caracteres para um atributo de cadeia, por exemplo. Além disso, não especifique atributos de comprimento zero que, no momento, têm como padrão 255 caracteres. Em vez disso, especifique o comprimento exato requerido para atributos.

As regras XSD NCName se aplicam a nomes de atributos de objeto de negócios no WebSphere Process Server, portanto, não utilize espaços ou ":" em nomes para atributos de objeto de negócios. Os nomes de atributos de objeto de negócios com espaços ou ":" são inválidos no WebSphere Process Server. Renomeie os atributos de objeto de negócios antes da migração.

Se estiver utilizando uma matriz em um objeto de negócios, não poderá depender da ordem da matriz durante a indexação na matriz em Mapas e/ou Relacionamentos. A construção migrada para o WebSphere Process Server não garante uma ordem de indexação, principalmente quando entradas são excluídas.

Mais uma vez, como afirmamos anteriormente, é importante utilizar apenas a ferramenta Business Object Designer para editar definições de objeto de negócios e utilizar apenas as APIs publicadas para objetos de negócios em artefatos de integração.

## **Gabaritos de Colaboração**

Muitas das diretrizes descritas anteriormente se aplicam ao desenvolvimento de gabaritos de colaboração.

Para assegurar que os processos sejam descritos apropriadamente com metadados, sempre utilize a ferramenta Process Designer para a criação e modificação de gabaritos de colaboração e evite editar os arquivos de metadados diretamente. Utilize a ferramenta Activity Editor sempre que possível para aumentar a utilização de metadados para descrever a lógica requerida.

Para minimizar a quantidade de retrabalho manual que pode ser requerido na migração, utilize apenas as APIs documentadas em gabaritos de colaboração. Evite utilizar variáveis estáticas. Em vez disso, utilize variáveis não estáticas e propriedades de colaboração para abordar os requisitos da lógica de negócios. Evite utilizar qualificadores Java finais, transientes e nativos em snippets Java. Eles não podem ser impostos em snippets Java de BPEL que são o resultado da migração de Gabaritos de Colaboração.

Para maximizar a portabilidade futura, evite utilizar chamadas de liberação de conexão explícitas e suporte a transações explícito (ou seja, confirmações explícitas & rollbacks explícitos) para Conjuntos de Conexões com o Banco de Dados Definidos pelo Usuário. Em vez disso, utilize a limpeza de conexão implícita gerenciada por contêiner e o suporte a transações implícito. Além disso, evite manter conexões e transações do sistema ativas em nós de snippet Java em um gabarito de colaboração. Isto se aplica a qualquer conexão com um sistema externo, bem como conjuntos de conexões com o banco de dados definidos pelo usuário. Conforme descrito anteriormente, as operações com um EIS externo devem ser gerenciadas em um adaptador e o código relacionado a uma operação do banco de dados deve estar contido em um trecho de código. Isto pode ser necessário em uma colaboração que, quando apresentada como um componente do processo de negócios de BPEL, pode ser seletivamente implementada como um fluxo que pode ser interrompido. Neste caso, o processo pode ser composto de várias transações separadas, apenas com informações de estados e de variáveis globais transmitidas entre as atividades. O contexto para qualquer conexão do sistema ou transação relacionada que estendeu estas transações de processos será perdido.

Não utilize caracteres especiais em nomes de propriedades do Gabarito de Colaboração. Estes caracteres especiais são inválidos em nomes de propriedades de BPEL para as quais eles serão migrados. Renomeie propriedades para remover estes caracteres especiais antes de migrar para evitar erros sintáticos no BPEL gerados pela migração.

Não faça referência a variáveis usando "this." Por exemplo, ao invés de "this.inputBusObj" use apenas "inputBusObj"

Utilize o escopo de nível de classe em variáveis em vez de variáveis com escopo definido pelo cenário. O escopo de cenário não é transportado durante a migração.

Inicialize todas as variáveis declaradas em snippets Java com um valor padrão. Por exemplo, "Object myObject = null;" Certifique-se de que todas as variáveis sejam inicializadas durante a declaração antes da migração.

Certifique-se de que não existam instruções de importação Java nas seções modificáveis pelo usuário de seus Gabaritos de Colaboração. Na definição do Gabarito de Colaboração, utilize os campos de

importação para especificar pacotes Java para importação.

## Mapas

Muitas das diretrizes descritas para gabaritos de colaboração também se aplicam a mapas.

Para assegurar que os mapas sejam descritos corretamente com metadados, sempre utilize a ferramenta Map Designer para a criação e modificação de mapas e evite editar os arquivos de metadados diretamente. Utilize a ferramenta Activity Editor sempre que possível para aumentar a utilização de metadados para descrever a lógica requerida.

Quando referir-se a um objeto de negócios filho em um mapa, utilize um submapa para os objetos de negócios filhos.

Evite utilizar o código Java como o "value" em um SET pois ele não é válido no WebSphere Process Server. Em vez disso, utilize constantes. Por exemplo, se o valor configurado for "xml version=" + "1.0" + " encoding=" + "UTF-8", isto não será válido no WebSphere Process Server. Em vez disso, altere-o para "xml version=1.0 encoding=UTF-8" antes da migração.

Para minimizar a quantidade de retrabalho manual que pode ser requerido na migração, utilize apenas as APIs documentadas em gabaritos de colaboração. Evite utilizar variáveis estáticas. Em vez disso, utilize variáveis não estáticas e propriedades de colaboração para abordar os requisitos da lógica de negócios. Evite utilizar qualificadores Java finais, transientes e nativos em snippets Java.

Se estiver utilizando uma matriz em um objeto de negócios, não poderá depender da ordem da matriz durante a indexação na matriz em Mapas. A construção migrada para o WebSphere Process Server não garante uma ordem de indexação, principalmente quando entradas são excluídas.

Para maximizar a portabilidade futura, evite utilizar chamadas de liberação de conexão explícitas e suporte a transações explícito (ou seja, confirmações explícitas & rollbacks explícitos) para Conjuntos de Conexões com o Banco de Dados Definidos pelo Usuário. Em vez disso, utilize a limpeza de conexão implícita gerenciada por contêiner e o suporte a transações implícito. Além disso, evite manter conexões e transações do sistema ativas em trecho de código Java em limites de nó de transformação. Isto se aplica a qualquer conexão com um sistema externo, bem como conjuntos de conexões com o banco de dados definidos pelo usuário. Conforme descrito anteriormente, as operações com um EIS externo devem ser gerenciadas em um adaptador e o código relacionado a uma operação do banco de dados deve estar contido em um trecho de código.

## Relacionamentos

Para relacionamentos, lembre-se de que embora as definições de relacionamentos poderão ser migradas para utilização no WebSphere Process Server, o esquema da tabela de relacionamentos e os dados da instância podem ser reutilizados pelo WebSphere Process Server e também compartilhados simultaneamente entre o WebSphere InterChange Server e o WebSphere Process Server.

As principais considerações para relacionamentos são utilizar apenas as ferramentas fornecidas para configurar os componentes relacionados e utilizar apenas as APIs publicadas para relacionamentos em artefatos de integração.

É importante utilizar apenas a ferramenta Relationship Designer para editar definições de relacionamentos. Além disso, permita que apenas o WebSphere InterChange Server configure o esquema de relacionamento, que é gerado automaticamente durante a implementação de definições de relacionamentos. Não altere o esquema da tabela de relacionamentos diretamente com ferramentas de banco de dados ou scripts SQL.

Além disso, se tiver que modificar manualmente os dados da instância de relacionamento no esquema da tabela de relacionamentos, certifique-se de utilizar os recursos fornecidos pelo Relationship Designer.

Conforme afirmamos anteriormente, é importante utilizar apenas as APIs publicadas para relacionamentos em artefatos de integração.

## Clientes da Estrutura de Acesso

Não desenvolva novos clientes adotando as APIs da interface IDL CORBA. Isto não será suportado no WebSphere

Process Server.

## Migrando para o WebSphere Integration Developer a partir do WebSphere MQ Workflow

O WebSphere Integration Developer fornece as ferramentas necessárias para migrar do WebSphere MQ Workflow.

O Assistente de Migração permite converter definições FDL de processos de negócios exportados do componente buildtime do WebSphere MQ Workflow para artefatos correspondentes do Business Process Choreographer. Os artefatos do Business Process Choreographer gerados incluem definições XMLSchema para objetos de negócios, definições WSDL, BPEL e definições TEL.

A ferramenta de conversão requer uma definição FDL semanticamente completa de um modelo de processo exportado do WebSphere MQ Workflow Buildtime com a opção **export deep**. Esta opção assegura que todas as especificações de dados necessários, de programa e de subprocesso estejam incluídas. Além disso, certifique-se de que qualquer definição do UPES (User Defined Process Execution Server) referida no modelo de processo do WebSphere MQ Workflow também seja selecionada quando o FDL for exportado do WebSphere MQ Workflow Buildtime.

**Nota:** O Assistente de Migração não cobre a migração de:

- Instâncias de tempo de execução do WebSphere MQ Workflow
- Aplicativos de programa que são chamados por um PEA (Program Execution Agent) do WebSphere MQ Workflow ou PES do WebSphere MQ Workflow (Process Execution Server para z/OS)

## Preparando a Migração do WebSphere MQ Workflow

Antes de migrar para o WebSphere Integration Developer do WebSphere MQ Workflow, primeiro é necessário assegurar que você tenha preparado corretamente seu ambiente.

O escopo e totalidade do mapeamento depende do grau de conformidade com as seguintes instruções para migração:

- Certifique-se de que as atividades do programa FDL estejam associadas a um UPES (User Defined Process Execution Server) se elas não forem atividades de **equipe** puras.
- Certifique-se de que as designações de equipe para as atividades do programa WebSphere MQ Workflow sejam compatíveis com os **verbos de equipe** padrão do TEL.
- Utilize nomes abreviados e simples para aprimorar a capacidade de leitura de modelos de processos migrados. Observe que os nomes de FDL podem ser nomes de BPEL inválidos. O Assistente de Migração ajuda a converter automaticamente nomes de FDL em nomes de BPEL válidos.

O Assistente de Migração produzirá construções do editor de processo de negócios sintaticamente corretas mesmo para construções do WebSphere MQ Workflow que não podem ser migradas (atividades do programa PEA ou PES, algumas designações de equipe dinâmicas e outros), que precisam ser adaptadas manualmente para artefatos do editor de processo de negócios executáveis.

A tabela a seguir esboça as regras de mapeamento aplicadas:

*Tabela 2. Regras de Mapeamento*

<b>WebSphere MQ Workflow</b>	<b>Business Process Choreographer</b>
Processo	<i>Processo com modo de execução: longRunning; Links do parceiro para interfaces de processo de entrada e de saída</i>
Origem e Depósito	<i>Variáveis para entrada e saída de processo; atividade de Recebimento e atividade de resposta</i>
Atividade do programa	<i>Atividade de chamada</i>
Atividade do processo	<i>Atividade de chamada</i>
Atividade vazia	<i>Atividade vazia</i>
Bloquear	<i>Escopo com atividades de BPEL incorporadas</i>
Condição de saída de atividade	<i>Atividade While (incluindo a atividade real)</i>
Condição inicial de atividade	<i>Condição de junção da atividade</i>
Designação de equipe de atividade	<i>Atividade de tarefa humana</i>
Contêiner de entrada e de saída de atividade	<i>Variáveis utilizadas para especificar a entrada/saída da atividade chamada</i>
Conector de controle; Condição de transição	<i>Link; Condição de transição</i>
Conector de dados	<i>Atividade de designação</i>
Contexto de dados global	<i>Variável</i>

Inicialmente, você deve tentar o processo de migração com pequenos projetos, se possível. O Assistente de Migração simplificará a conversão dos modelos de processos do WebSphere MQ Workflow em modelos de processos do editor de processos de negócios, mas é necessário estar ciente de que os processos não podem ser mapeados um para um, pois você está criando um novo modelo de programação. Os escopos semânticos das linguagens de especificação de processos subjacentes (FDL e BPEL) compartilham uma área de interseção, mas não se sobrepõem no total. Caso contrário, não será possível esperar novos benefícios do editor de processos de negócios. Os serviços da Web representam uma nova tecnologia promissora que requer a substituição de soluções reprovadas por novas.

Em geral, você deve sempre rever e, possivelmente, modificar os artefatos gerados. Pode ser necessário esforço adicional para possibilitar uma migração bem-sucedida ou para concluir a tarefa de migração.


### **Migrando o WebSphere MQ Workflow Utilizando o Assistente de Migração**

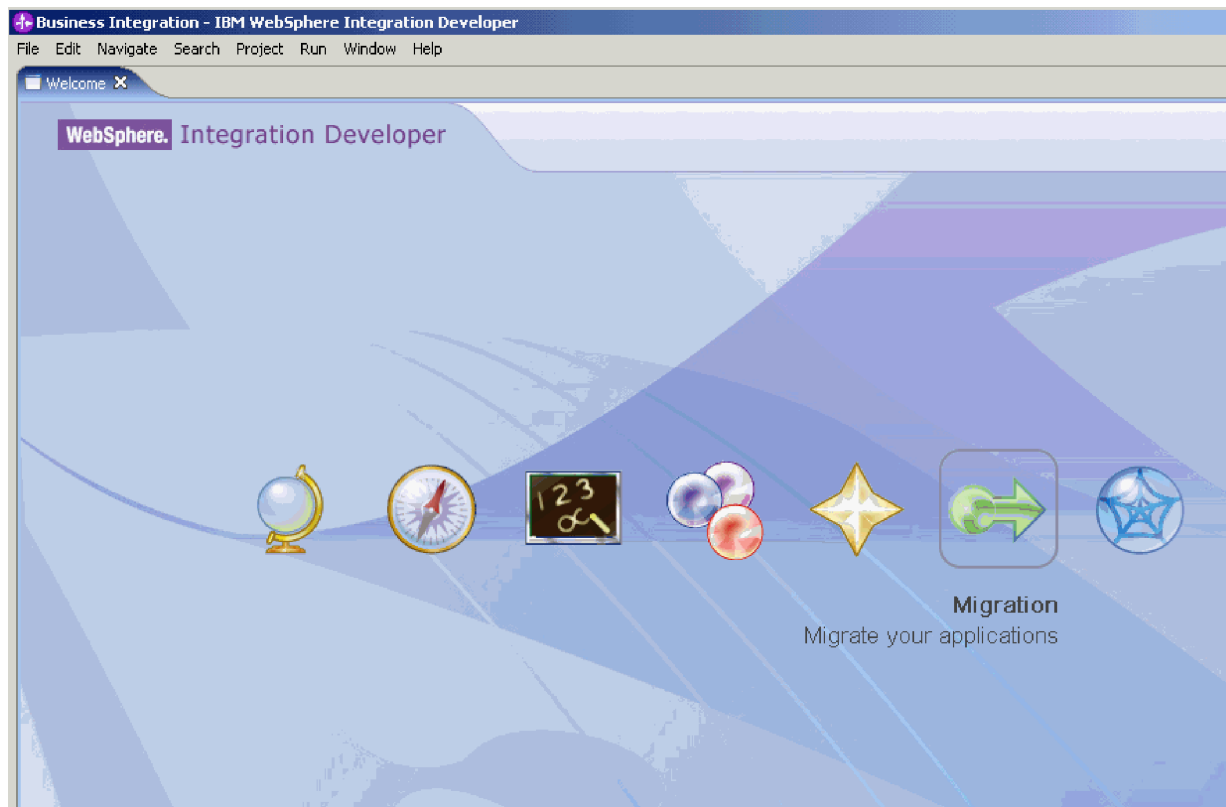
O Assistente de Migração permite converter definições FDL de processos de negócios exportados do componente buildtime do WebSphere MQ Workflow para artefatos correspondentes do Business Process Choreographer. Os artefatos do Business Process Choreographer gerados incluem definições XMLSchema para objetos de negócios, definições WSDL, BPEL e definições TEL.

**Nota:** O Assistente de Migração não cobre a migração de:

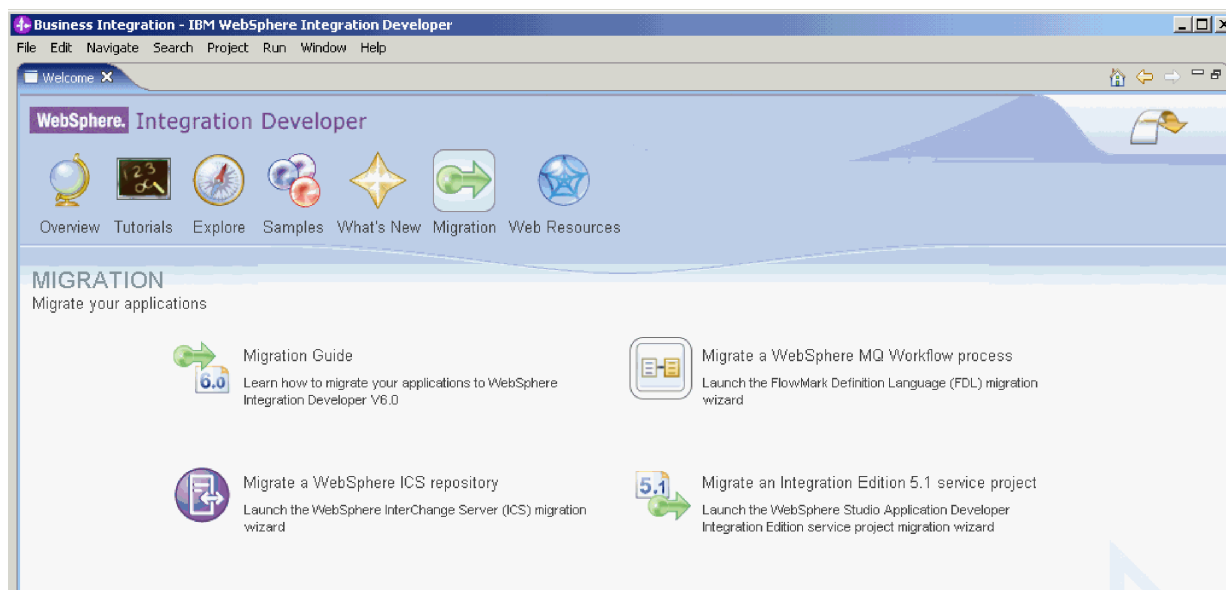
- Instâncias de tempo de execução do WebSphere MQ Workflow
- Aplicativos de programa que são chamados por um PEA (Program Execution Agent) do WebSphere MQ Workflow ou PES do WebSphere MQ Workflow (Process Execution Server para z/OS)

Siga estas etapas para utilizar o Assistente de Migração para migrar seus artefatos do WebSphere MQ Workflow:

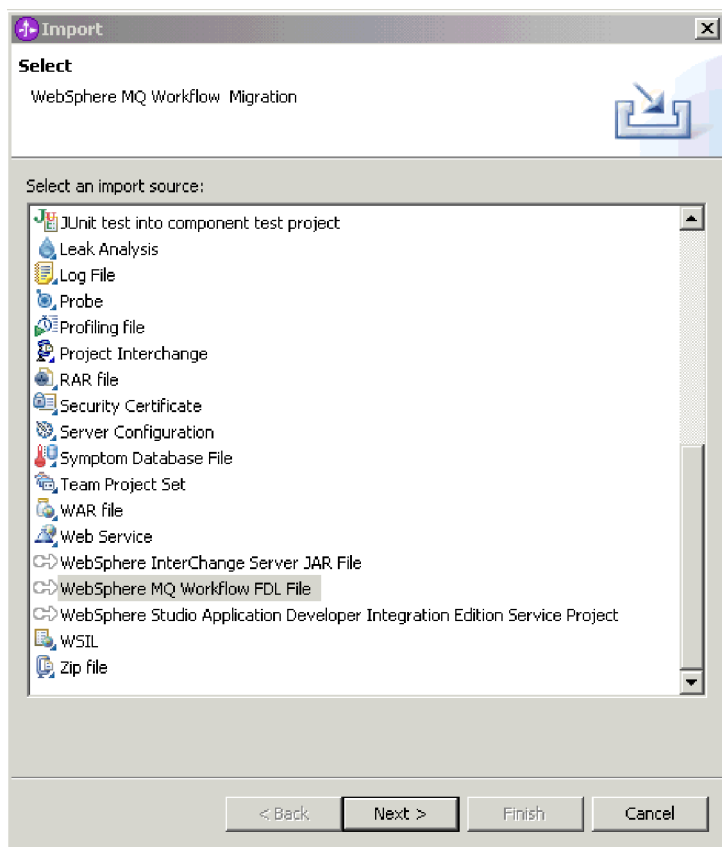
1. Na página de Boas-vindas, clique no  para abrir a página Migração.



2. Na página Migração, selecione a opção Migrar um Processo do WebSphere MQ Workflow:

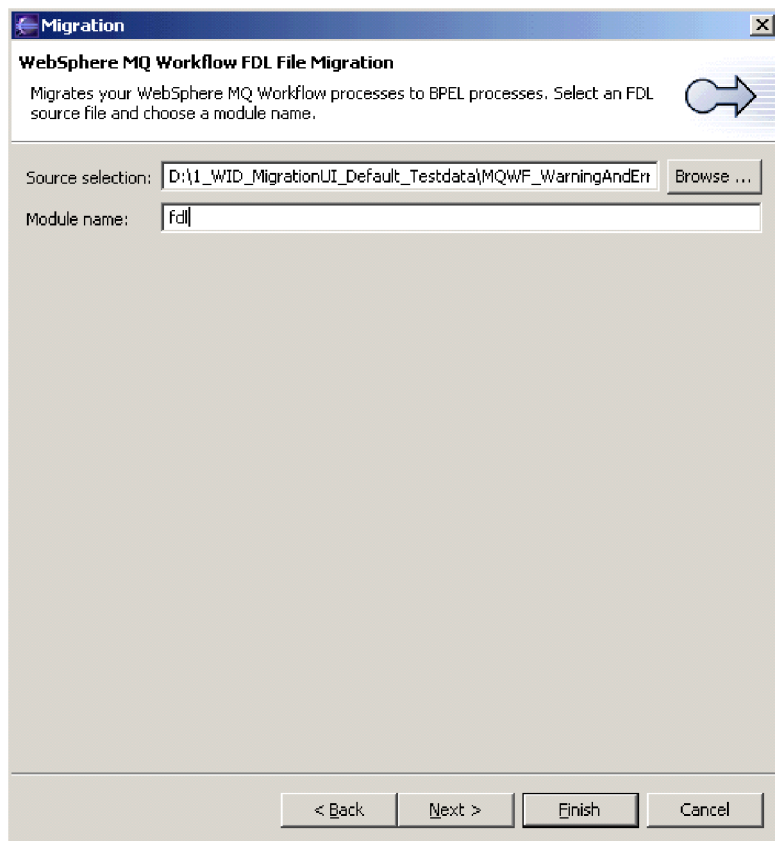


(Nota: Você também pode abrir o Assistente de Migração clicando em **Arquivo** → **Importar** → **Arquivo FDL do WebSphere MQ Workflow** :



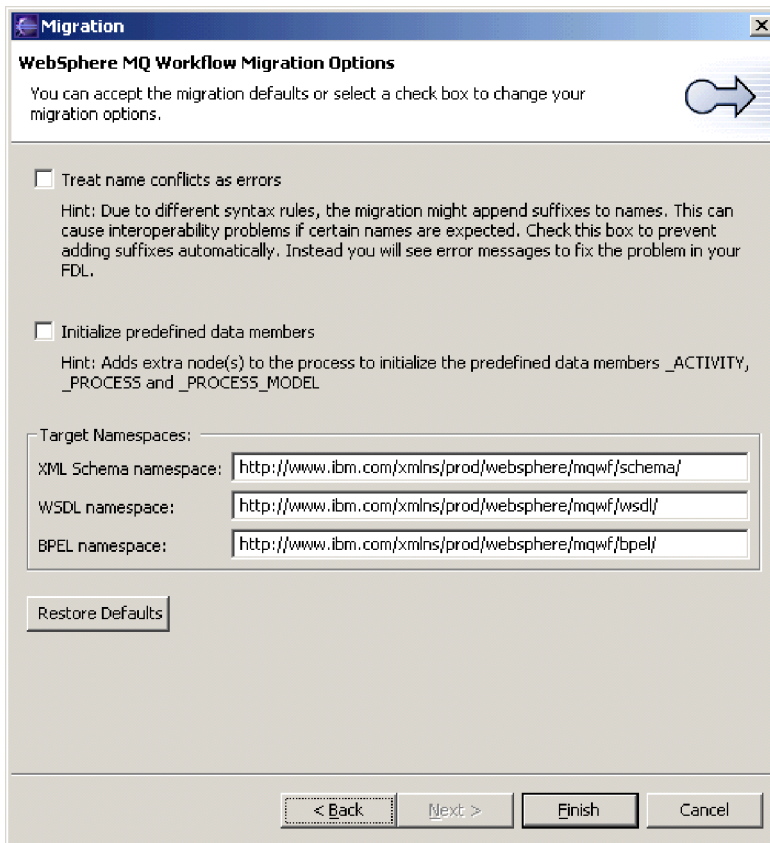
3. É aberto o Assistente de Migração. Digite o nome do arquivo .fdl no campo **Seleção de Origem** clicando no botão **Procurar** e navegando para o arquivo. Digite o nome do módulo no campo relevante. Clique em **Avançar**.





4. É aberta a página Opções de Migração. Nesta página é possível aceitar os padrões de migração ou selecionar uma caixa de opções para alterar a opção. Selecionando a caixa de opções **Tratar conflitos de nomes como erros**, você pode evitar a inclusão automática de sufixos que pode resultar em erros de interoperabilidade. A caixa de opções **Inicializar membros de dados predefinidos** inclui nós extras no processo para inicializar os membros de dados predefinidos.





**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:

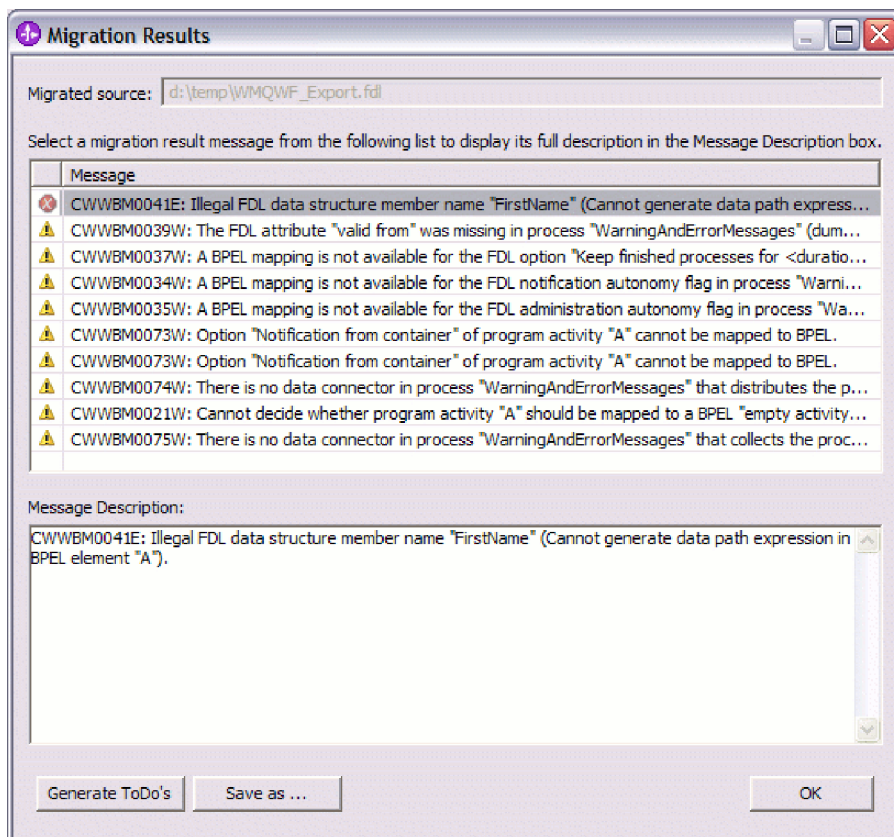
BPEL namespace:

Clique em **Concluir**.

## Verificando a Migração do WebSphere MQ Workflow

Quando o Assistente de Migração tiver sido concluído com êxito, será exibida uma lista de mensagens de erro, de aviso e/ou informativas. Estas mensagens podem ser utilizadas para verificar a migração do WebSphere MQ Workflow.

A página a seguir aparece após a conclusão do Assistente de Migração:



Na janela Resultados da Migração, você pode ver uma lista de mensagens de migração. Clique na mensagem para ver uma descrição completa dos detalhes. Se necessário, nesta lista de mensagens, você pode criar uma lista de itens que precisam ser corrigidos, clicando no botão **Gerar Listas de Tarefas**.

### Limitações do Processo de Migração (do WebSphere MQ Workflow)

Existem algumas limitações envolvidas no processo de migração do WebSphere MQ Workflow.

A migração do FDL gerará atividades de chamada para atividades UPES e os WSDLs correspondentes. No entanto, o ambiente de tempo de execução difere significativamente entre o IBM WebSphere MQ Workflow e o IBM WebSphere Process Server em termos das técnicas que são utilizadas para correlacionar mensagens de chamada e suas respostas. Portanto, o BPEL gerado não poderá ser executado com êxito se uma camada de compatibilidade do UPES não estiver disponível.

## Migrando Artefatos de Origem para o WebSphere Integration Developer do WebSphere Studio Application Developer Integration Edition

Os artefatos de origem podem ser migrados do WebSphere Studio Application Developer Integration Edition para o WebSphere Integration Developer. A migração dos artefatos de origem em um aplicativo envolve a migração deles para o novo modelo de programação do WebSphere Integration Developer para que a nova funcionalidade e os recursos possam ser utilizados. O aplicativo pode então ser reimplementado e instalado no servidor WebSphere Integration Developer Versão 6.0.

Muitos recursos disponíveis no WebSphere Business Integration Server Foundation 5.1 foram movidos para o WebSphere Application Server 6.0 base. Consulte este site para obter dicas sobre a migração desses recursos: [http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/rins\\_migratpme.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/rins_migratpme.html)

Para migrar totalmente um projeto de serviço do WebSphere Studio Application Developer Integration Edition existem duas etapas fundamentais a serem concluídas:

1. Utilize o assistente de Migração para migrar automaticamente os artefatos para o projeto de Módulo do Business Integration.
2. Utilize o WebSphere Integration Developer para completar manualmente a migração. Isso envolve corrigir qualquer código Java que não poderá ser automaticamente migrado e religar os artefatos migrados.

**Nota:** A migração do tempo de execução (caminho de upgrade) não será fornecida no WebSphere Process Server 6.0.0, portanto, esse caminho de migração do artefato de origem será a única opção para migrar projetos de serviço do WebSphere Studio Integration Edition na 6.0.0.

## **Caminhos de Migração Suportados para Migração de Artefatos de Origem**

Antes de iniciar a migração de artefatos de origem do WebSphere Studio Application Developer Integration Edition, é necessário rever os caminhos de migração suportados pelo WebSphere Integration Developer.

O assistente de Migração oferece a capacidade de migrar um projeto de serviço do WebSphere Studio Application Developer Integration Edition Versão 5.1 (ou superior) de cada vez. Ele *não* migrará um espaço de trabalho inteiro.

O assistente de Migração não migra binários de aplicativos – ele migra apenas artefatos de origem localizados em um projeto de serviço do WebSphere Studio Application Developer Integration Edition.

## **Preparando Artefatos de Origem para Migração**

Antes de migrar artefatos de origem para o WebSphere Integration Developer do WebSphere Studio Application Developer Integration Edition, primeiro, certifique-se de que tenha preparado corretamente seu ambiente.

As etapas a seguir descrevem como preparar seu ambiente antes de migrar artefatos de origem para o WebSphere

Integration Developer do WebSphere

Studio Application Developer Integration Edition:

1. Certifique-se de ter uma cópia de backup do espaço de trabalho inteiro da 5.1 antes de tentar migrar.
2. Reveja a seção de migração do Rational Application Developer Information Center para determinar a melhor maneira de migrar projetos específicos não-WBI no espaço de trabalho: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.rad.migration.doc/topics/tmigratefrom51x.html>
3. Reveja a seção Serviço da Web do Rational Application Developer Information Center para obter informações de segundo plano sobre a funcionalidade do serviço da Web fornecida pelo Rational Application Developer: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.webservices.rad.nav.doc/developingweb.html>
4. Certifique-se de que todos os recursos adequados do WebSphere Integration Developer estejam ativados. Se esses recursos não estiverem ativados, você não poderá ver as opções de menu que serão discutidas abaixo. Para ativar os recursos importantes:
  - No WebSphere Integration Developer, vá para o item de menu **Janela** e selecione **Preferências**.
  - Vá para **Workbench** e, em seguida, selecione a categoria **Recursos**.
  - Selecione todos os recursos nas seguintes categorias:
    - J2EE Avançado
    - Enterprise Java
    - Desenvolvedor de Integração

- Java Developer
  - Web Developer (típico)
  - Web Service Developer
  - XML Developer
- Clique em **OK**.
5. Utilize um novo diretório de espaço de trabalho para o WebSphere Integration Developer. Não é recomendável abrir o WebSphere Integration Developer em um espaço de trabalho antigo do WebSphere Studio Application Developer Integration Edition que contenha projetos de serviço, pois esses projetos primeiro devem ser migrados para um formato que possa ser lido pelo WebSphere Integration Developer. As etapas recomendadas para isso são:
    - a. Copie todos os projetos não de serviço do espaço de trabalho antigo para o novo. *Não* copie os projetos EJB, da Web e EAR 5.1 criados durante a geração do código de implementação para um projeto de serviço 5.1. O novo código de implementação da 6.0 será gerado outra vez automaticamente quando o módulo BI for construído.
    - b. Abra o WebSphere Integration Developer no espaço de trabalho vazio e importe todos os projetos não de serviço clicando em **Arquivo** → **Importar** → **Projeto Existente para o Espaço de Trabalho** e selecione os projetos copiados para o novo espaço de trabalho.
      - Se o projeto for um projeto J2EE, então, você deverá migrá-lo para o nível 1.4 utilizando o assistente Rational Application Developer Migration:
        - 1) Clique com o botão direito do mouse no projeto e selecione **Migração** → **Assistente de Migração do J2EE...**
        - 2) Reveja as instruções de aviso na primeira página e clique em **Avançar**.
        - 3) Certifique-se de que o **Projeto J2EE** esteja marcado na lista Projetos. Deixe marcados **Migrar estrutura do projeto** e **Migrar nível de especificação do J2EE**. Escolha J2EE versão **1.4** e **WebSphere Process Server v6.0 do Servidor de Destino**.
        - 4) Selecione quaisquer outras opções que possam ser adequadas para seu projeto J2EE e clique em **Concluir**. Se essa etapa for concluída com êxito, você verá uma mensagem **Migração concluída com êxito**.
        - 5) Se houver erros no projeto J2EE após a migração, você deverá remover todas as entradas do caminho de classe que se referem aos arquivos .jar ou bibliotecas da v5 e incluir a **Biblioteca do Sistema JRE** e as bibliotecas de **Destino do Servidor WPS** no caminho de classe (discutido abaixo). Isso deverá remover a maioria dos erros ou todos.
      - Para projetos EJB do WebSphere Business Integration com Extended Messaging (CMM) ou CMP/A (Container Managed Persistence over Anything), os arquivos do descritor de Extensão Jar EJB da IBM devem ser migrados depois da importação do projeto da 5.1 para o espaço de trabalho da 6.0. Consulte "Migrando Projetos EJB do WebSphere Business Integration" para obter informações adicionais.
      - Corrija o caminho de classe para cada projeto não de serviço importado no espaço de trabalho. Para incluir as bibliotecas do JRE e do WebSphere Process Server no caminho de classe, clique com o botão direito do mouse no projeto importado e selecione **Propriedades**. Vá para a entrada **Caminho de Construção Java** e selecione a guia **Bibliotecas**. Então, faça o seguinte:
        - 1) Selecione **Incluir biblioteca** → **Biblioteca do Sistema JRE** → **Alternar JRE - WPS Server v6.0 JRE** → **Concluir**.
        - 2) Em seguida, selecione **Incluir Biblioteca** → **Destino do Servidor WPS** → **Configurar Caminho de Classe do Servidor wps** → **Concluir**.
  6. Para melhores resultados, antes de executar o assistente de Migração, vá para o item de menu **Projeto** e certifique-se de que **Construir Automaticamente** NÃO esteja marcado. O WebSphere Integration Developer é diferente do WebSphere Studio Application Developer Integration Edition porque as opções de implementação de serviço são especificadas no tempo de design. Quando o projeto for

construído, o código de implementação será atualizado automaticamente nos projetos EJB e da Web gerados, portanto, não existe mais nenhuma opção para **Gerar Código de Implementação** manualmente.

7. Para migrar totalmente os arquivos .bpel em um projeto de serviço, é necessário assegurar que todos os arquivos .wsdl e .xsd referidos pelos arquivos .bpel possam ser resolvidos em um projeto de integração de negócios no novo espaço de trabalho:
  - Se os arquivos .wsdl e/ou .xsd estiverem no mesmo projeto de serviço que o arquivo .bpel, nenhuma ação adicional será requerida.
  - Se os arquivos .wsdl e/ou .xsd estiverem em um projeto de serviço diferente daquele que você está migrando, os artefatos da 5.1 deverão ser reorganizados utilizando o WebSphere Studio Application Developer Integration Edition antes da migração. A razão para isso é que os projetos de Módulo do Business Integration podem não compartilhar artefatos. A seguir estão as duas opções para reorganizar os artefatos da 5.1:
    - No WebSphere Studio Application Developer Integration Edition, crie um novo projeto Java que mantenha todos os artefatos comuns. Coloque todos os arquivos .wsdl e .xsd compartilhados por mais de um projeto de serviço nesse novo projeto Java. Inclua uma dependência nesse novo projeto Java para todos os projetos de serviço que utilizam esses artefatos comuns. No WebSphere Integration Developer, crie um novo projeto de Biblioteca do Business Integration com o mesmo nome do projeto Java compartilhado da 5.1 antes de migrar qualquer um dos projetos de serviço. Copie manualmente os arquivos .wsdl e .xsd antigos do projeto Java compartilhado da 5.1 para esta nova pasta de projetos da Biblioteca de BI. Isso deve ser feito antes da migração dos projetos de serviço do BPEL.
    - Outra opção é manter uma cópia local desses artefatos .wsdl e .xsd compartilhados em cada projeto de serviço de forma que não existam dependências entre projetos de serviço.
  - Se os arquivos .wsdl e/ou .xsd estiverem em qualquer outro tipo de projeto (geralmente outros Projetos Java), será necessário criar um projeto de Biblioteca de Business Integration com o mesmo nome que o projeto 5.1. Você também deve configurar o caminho de classe do novo projeto de biblioteca, incluindo as entradas do projeto Java 5.1, se houver. Este tipo de projeto é útil para armazenar artefatos compartilhados.

Agora você está pronto para iniciar o processo de migração.

## **Migrando Projetos de Serviço Utilizando o Assistente de Migração do WebSphere Integration Developer**

O Assistente de Migração do WebSphere Integration Developer permite a migração de projetos de serviço.

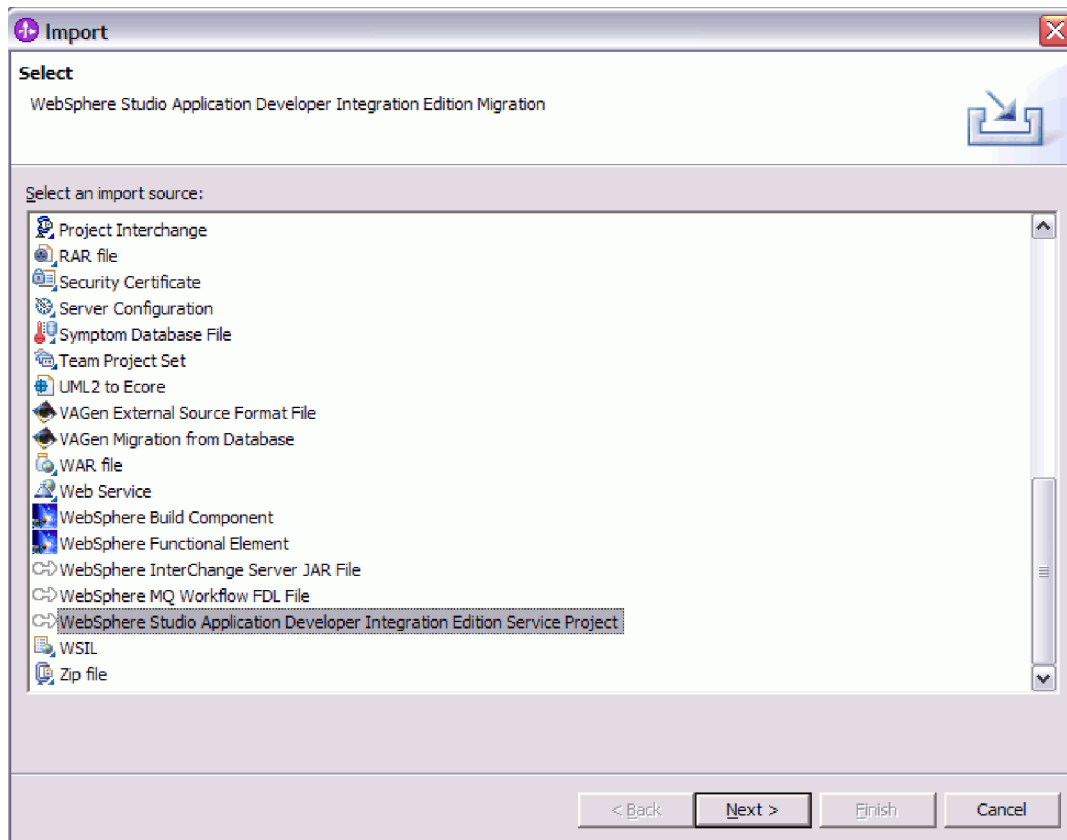
O Assistente de Migração faz o seguinte:

1. Cria um novo módulo do Business Integration (o nome do módulo é definido por você)
2. Migra as entradas de caminho de classe do projeto de serviço para o novo módulo
3. Copia todos os artefatos de origem do WebSphere Business Integration Server Foundation do projeto de origem selecionado para este módulo
4. Migra as extensões de BPEL em arquivos WSDL
5. Migra os processos de negócios (arquivos .bpel) do BPEL4WS versão 1.1 para o novo nível suportado pelo WebSphere Process Server, construído no BPEL4WS versão 1.1 com recursos importantes da especificação de lançamento do WS-BPEL versão 2.0
6. Cria um componente SCA para cada processo .bpel
7. Gera um arquivo de monitoramento .mon para cada processo BPEL para preservar o comportamento de monitoramento padrão do WebSphere Studio Application Developer Integration Edition (se necessário)

Siga estas etapas para migrar projetos de serviço utilizando o Assistente de Migração do WebSphere

Integration Developer:

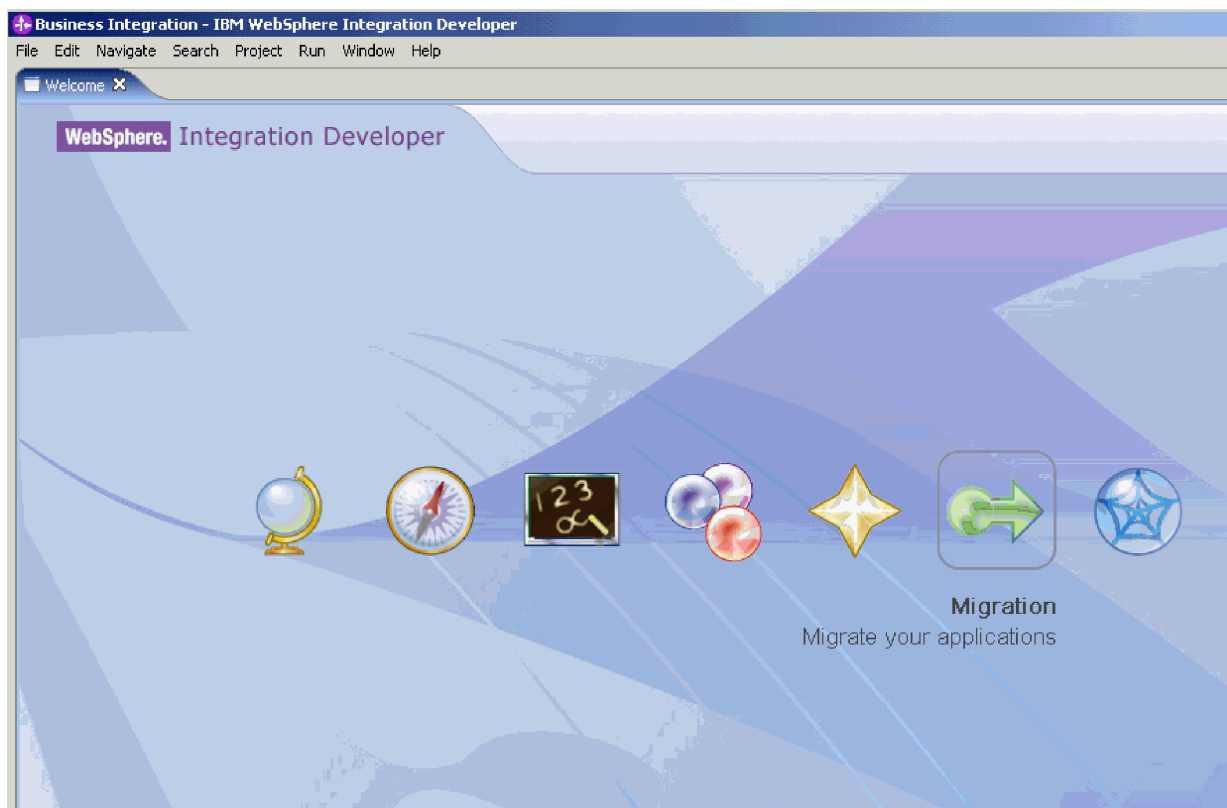
1. Chame o assistente selecionando **Arquivo → Importar → Projeto de Serviço do WebSphere Studio Application Developer Integration Edition**.



Você também pode abrir o assistente de Migração a partir da página de Boas-vindas clicando no



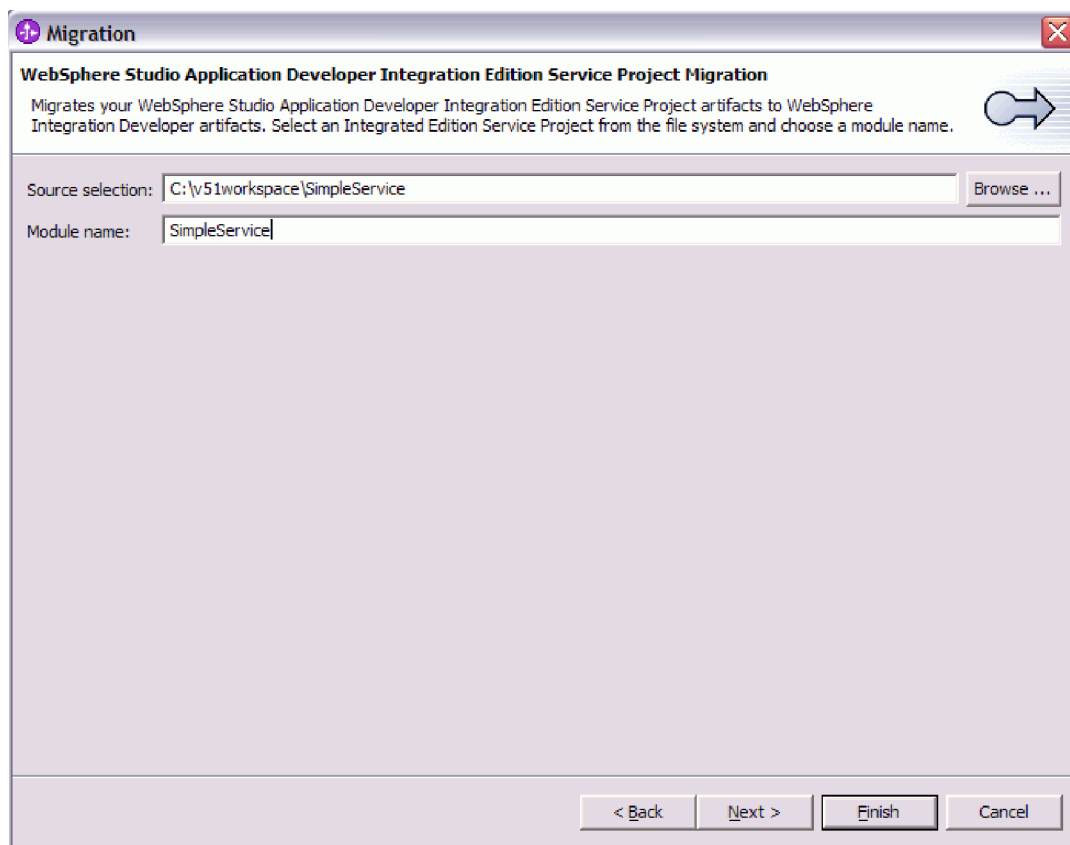




Na página Migração, selecione a opção Migrar um projeto de serviço do Integration Edition 5.1:



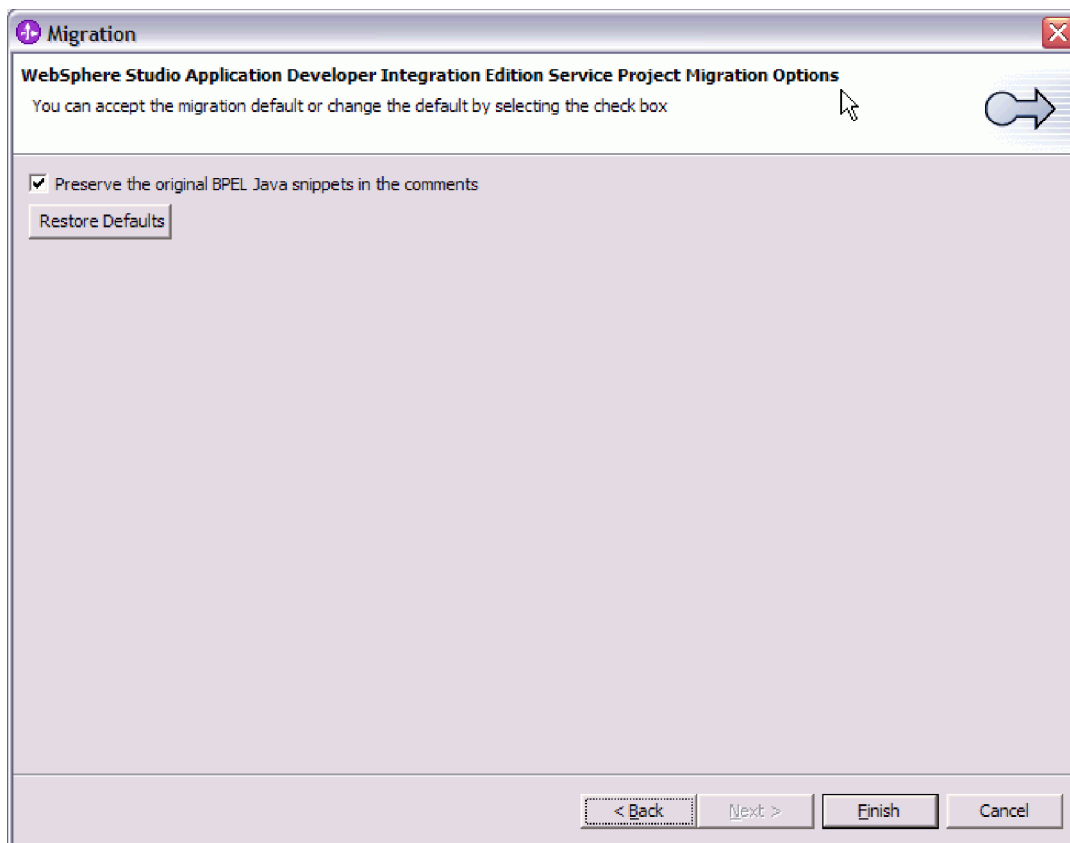
2. É aberto o Assistente de Migração. Digite o caminho para a Seleção de Origem ou clique no botão **Procurar** para localizá-lo. Digite também o nome do Módulo do local do Projeto de Serviço do WebSphere Studio Application Developer Integration Edition que irá migrar:



**Nota:** É recomendado que você escolha o nome do Projeto de Serviço como o nome do módulo, porque se houver outros projetos no espaço de trabalho do WebSphere Studio Application Developer Integration Edition que são dependentes desse projeto, você não terá que atualizar os caminhos de classe dos projetos dependentes depois de importá-los para o WebSphere Integration Developer.

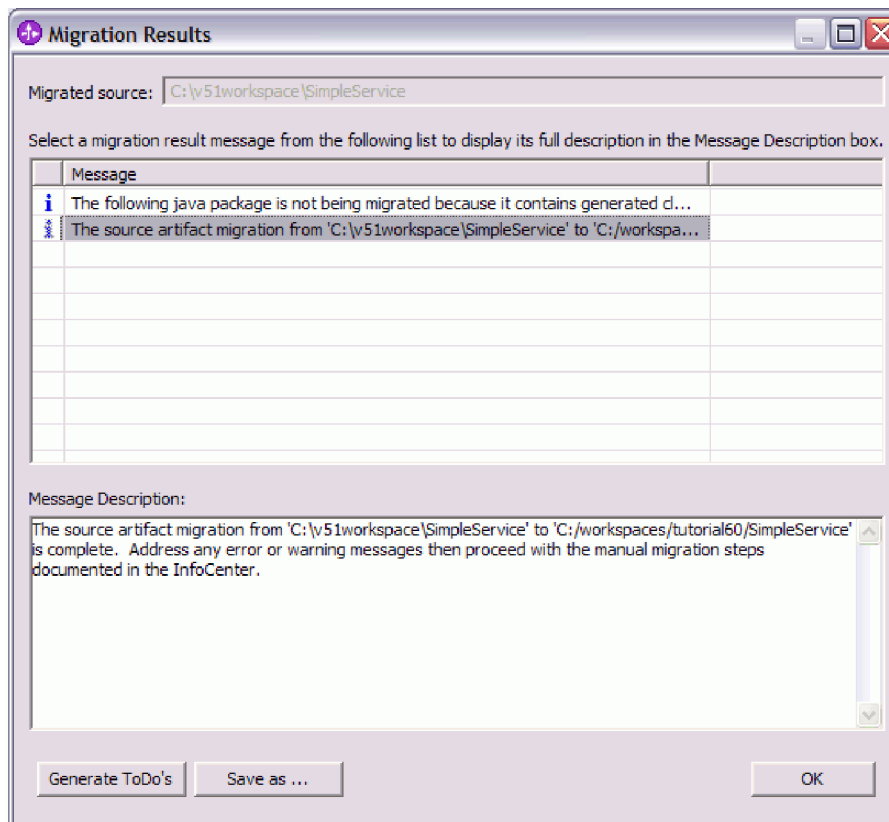
3. Nas Opções de migração, selecione Preservar snippets Java do BPEL original na caixa de opções de comentários:





Clique em **Concluir**.

4. Depois do processo de migração ser concluído, a janela Resultados da Migração é aberta:



Um arquivo de registro contendo essas mensagens de migração será automaticamente gerado para a pasta .metadata do espaço de trabalho da 6.0. O arquivo de registro deve ser nomeado ".log".

Após a conclusão do Assistente de Migração, construa o módulo de Business Integration criado e tente resolver erros de construção. Inspeção todos os arquivos .bpel migrados: certifique-se de que eles tenham sido totalmente migrados e possam ser abertos no Editor BPEL do WebSphere

Integration Developer. Existem alguns snippets Java

de BPEL que não podem ser automaticamente migrados. Se você vir erros nos snippets Java

do BPEL, consulte "Migrando para o Modelo de Programação SCA" para obter as etapas necessárias para corrigir os erros. Além disso, se você tiver utilizado o Assistente de Migração para migrar um projeto de serviço para um Módulo de BI, abra o editor de dependências de módulo para assegurar que as dependências estejam configuradas corretamente. Para isso, vá para a perspectiva Business Integration e dê um clique duplo no projeto do módulo de BI. Lá você pode incluir dependências em projetos de biblioteca de integração de negócios, em projetos Java

e em projetos J2EE.

## Migrando Manualmente o Aplicativo

Quando o Assistente de Migração tiver migrado com êxito os artefatos para o novo módulo de Business Integration, os artefatos deverão ser ligados para criar um aplicativo que esteja de acordo com o modelo SCA.

1. Abra o WebSphere Integration Developer e vá para a perspectiva Business Integration. Devem aparecer os módulos que foram criados pelo Assistente de Migração (um módulo para cada projeto de serviço migrado). O primeiro artefato listado no projeto do módulo é o arquivo de montagem do módulo (ele tem o mesmo nome que o módulo).

2. Dê um clique duplo no arquivo de montagem para abri-lo no Editor de Montagem no qual os componentes SCA podem ser criados e ligados para obter uma funcionalidade semelhante à do aplicativo da Versão 5.1. Se houver processos BPEL no projeto de serviço do WebSphere Studio Application Developer Integration Edition, o assistente de migração deve ter criado componentes SCA padrão para cada um desses processos e eles estarão no Editor de Montagem.
3. Selecione um componente e vá para a visualização Propriedades na qual as propriedades de Descrição, Detalhes e Implementação serão exibidas e podem ser editadas.

As informações a seguir descrevem mais detalhadamente como religar manualmente o aplicativo utilizando as ferramentas disponíveis no WebSphere Integration Developer:

### **Criando Componentes do SCA e Importações do SCA para os Serviços no Aplicativo de Reliação:**

Todos os projetos precisarão de alguma relação após a migração para reconectar os serviços na forma em que estavam na 5.1. Por exemplo, todos os processos de negócios migrados devem ser religados para seus parceiros de negócios. Um Componente ou Importação do SCA deve ser criado para todos os outros tipos de serviço. Para os projetos de serviço do WebSphere Studio Application Developer Integration Edition que interajam com os sistemas ou entidades externos ao projeto, uma Importação do SCA pode ser criada para que o projeto migrado acesse essas entidades como serviços de acordo com o modelo SCA.

Para os projetos de serviço do WebSphere Studio Application Developer Integration Edition que interajam com as entidades dentro projeto (por exemplo, um processo de negócios, um serviço de transformador ou uma classe Java), uma Importação do SCA pode ser criada para que o projeto migrado acesse essas entidades como serviços de acordo com o modelo SCA.

As seções a seguir fornecem detalhes sobre a Importação do SCA ou Componentes do SCA a serem criados com base no tipo de serviço que deve ser migrado:

#### *Migrando um Serviço Java:*

Você pode migrar um serviço Java para um Componente Java do SCA.

Se o Projeto de Serviço do WebSphere

Studio Application Developer Integration Edition era dependente de outros projetos Java

, copie os projetos existentes para o novo diretório de espaço de trabalho e importe-os para o WebSphere

Integration Developer utilizando o assistente **Arquivo → Importar → Projeto Existente para o Espaço de Trabalho**.

No WebSphere Studio Application Developer Integration Edition, ao gerar um novo serviço Java a partir de uma classe Java existente, as seguintes opções eram fornecidas:

- Criar esquemas XSD para tipos de dados complexos:
  - No arquivo WSDL da interface
  - Como um novo arquivo para cada tipo de dados
- Suportar capacidade de tratamento de erro:
  - Gerar falha
  - Não gerar falha
- Outros detalhes sobre o serviço a ser gerado, como nomes de ligação e de serviço

Existem muitos componentes novos na 6.0 que oferecem nova funcionalidade, como mapeamento de dados, mediação de interface, máquinas de estado de negócios, seletores, regras de negócios e muito

mais. Primeiro, você deve determinar se um desses novos tipos de componentes pode substituir o componente Java customizado. Se isso não for possível, siga o caminho de migração descrito abaixo.

Importe o projeto de serviço utilizando o assistente de Migração. Isto resultará na criação de um módulo de integração de negócios com Mensagens, PortTypes, Ligações e Serviços do WSDL gerados no WebSphere Studio Application Developer Integration Edition.

Na perspectiva Business Integration, expanda o módulo para ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).

Você tem as seguintes opções:

*Prós e Contras para Cada uma das Opções de Religação do Serviço Java:*

Existem prós e contras para cada uma das opções de religação do serviço Java.

A seguinte lista descreve as duas opções e os prós e contras de cada uma:

- A primeira opção provavelmente fornece melhor desempenho no tempo de execução porque a chamada de um serviço da Web é mais lenta do que a chamada de um componente Java.
- A primeira opção pode propagar contexto enquanto uma chamada de serviço da Web não propaga contexto da mesma maneira.
- A segunda opção não envolve a criação de nenhum código customizado.
- A segunda opção não poderá ser possível para algumas definições de interface Java, porque a geração de um serviço Java tem limitações. Consulte a documentação do Rational Application Developer aqui: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.doc/ref/rlimit.html>
- A segunda opção poderá resultar em uma alteração de interface e, então, em uma alteração no cliente SCA.
- A segunda opção requer que um servidor do WebSphere Process Server 6.0 seja instalado e configurado para funcionar com o WebSphere Integration Developer. Para ver os tempos de execução instalados que são configurados para trabalhar com o WebSphere Integration Developer, vá para **Janela** → **Preferências** → **Servidor** → **Tempos de Execução Instalados** e selecione a entrada **WebSphere Process Server v6.0**, se ela existir, e certifique-se de que ela aponta para o local onde o produto está instalado. Certifique-se de que essa entrada esteja marcada se o servidor não existir e desmarcada se esse servidor não estiver realmente instalado. Você também pode clicar no botão **Incluir...** se desejar incluir outro servidor.
- Se o componente Java foi construído no WebSphere Studio Application Developer Integration Edition utilizando a abordagem de cima para baixo, na qual o esqueleto do Java foi gerado a partir de um WSDL, então, os parâmetros dentro e fora dessa classe Java provavelmente serão a subclasse `WSIFFormatPartImpl`. Se for esse o caso, então, você escolherá a opção 1 para gerar um novo esqueleto Java de estilo SCA a partir do WSDL/XSDs original ou a opção 2 para gerar um novo esqueleto Java genérico (não dependente das APIs WSIF ou DataObject) a partir da interface WSDL original.

*Criando o Componente Java Customizado: Opção 1:*

A técnica de migração recomendada é utilizar o tipo Componente Java do WebSphere Integration Developer que permite representar o serviço Java como um componente SCA. Durante a migração, o código Java customizado deve ser escrito para conversão entre o estilo de interface Java do SCA e o estilo de interface do componente Java existente.

Para criar o componente Java

customizado:

1. No projeto de módulo, expanda **Interfaces** e selecione a interface WSDL que foi gerada para essa classe Java no WebSphere Studio Application Developer Integration.
2. Arraste e solte essa interface sobre o Editor de Montagem. Um diálogo aparecerá solicitando que você selecione o tipo de componente que irá criar. Selecione **Componente sem Tipo de Implementação** e clique em **OK**.
3. Um componente genérico aparecerá no diagrama de Montagem. Selecione-o e vá para a visualização **Propriedades**.
4. Na guia **Descrição**, você pode alterar o nome e o nome de exibição do componente para algo mais descritivo.
5. Na guia **Detalhes** você verá que esse componente tem uma interface - aquela que você arrastou e soltou sobre o Editor de Montagem.
6. Certifique-se de que a classe Java que você está tentando acessar esteja no caminho de classe do projeto de serviço, se ela não estiver contida dentro do próprio projeto de serviço.
7. Clique com o botão direito no projeto e selecione **Abrir Editor de Dependência...** Na seção **Java**, certifique-se de que o projeto que contém a classe Java antiga esteja listada. Se não estiver, inclua-a clicando no botão **Incluir...**
8. De volta ao Editor de Montagem, clique com o botão direito do mouse no componente que acabou de criar e selecione **Gerar Implementação...** → **Java** Em seguida, selecione o pacote no qual a implementação Java será gerada. Isso cria um esqueleto do serviço Java que adere à interface WSDL de acordo com o modelo de programação SCA, no qual os tipos complexos são representados por um objeto que é um `commonj.sdo.DataObject` e os tipos simples são representados por equivalentes de seus Objetos Java.

Os seguintes exemplos de código mostram:

1. Definições relevantes da interface WSDL 5.1.
2. Os métodos WebSphere Studio Application Developer Integration Edition 5.1 Java que correspondem ao WSDL
3. Os métodos WebSphere Integration Developer 6.0 Java para o mesmo WSDL

O seguinte código mostra as definições relevantes da interface WSDL 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>
```

O seguinte código mostra os métodos WebSphere

Studio Application Developer Integration Edition 5.1 Java

que correspondem ao WSDL:

```
public StockInfo getStockInfo(String symbol)
{
    return new StockInfo();
}
public void setStockPrice(String symbol, float newPrice)
{
    // definir algumas coisas
}
```

O seguinte código mostra os métodos do WebSphere

Integration Developer 6.0 Java

para o mesmo WSDL:

```
public DataObject getStockInfo(String aString) {
    //TODO Precisa ser implementado.
    return null;
}
public void setStockPrice(String symbol, Float newPrice) {
    //TODO Precisa ser implementado.
}
```

Agora, você precisará preencher o código no qual vê as tags “//TODO” na classe de implementação Java gerada. Existem duas opções:

1. Mova a lógica da classe Java original para essa classe, adaptando-a para utilizar DataObjects.
  - Essa será a opção recomendada se você tiver escolhido a abordagem de cima para baixo no WebSphere Studio Application Developer Integration Edition e desejar que o componente Java lide com parâmetros DataObject. Esse retrabalho é necessário porque as classes Java geradas a partir de definições do WSDL no WebSphere Studio Application Developer Integration Edition têm dependências do WSIF que devem ser eliminadas.
2. Crie uma instância privada da classe Java antiga dentro dessa classe Java gerada e escreva código para:
  - a. Converter todos os parâmetros da classe de implementação Java gerada em parâmetros que a classe Java antiga espera.
  - b. Chamar a instância privada da classe Java antiga com os parâmetros convertidos.
  - c. Converter o valor de retorno da classe Java antiga para o tipo de valor de retorno declarado pelo método de implementação Java gerado.
  - d. Essa opção é recomendada para cenários de consumo nos quais os proxies de serviço do WSIF devem ser consumidos pelos novos componentes Java de estilo da 6.0.

Depois de concluir uma das opções acima, você deverá religar o serviço Java. Não deve haver nenhuma referência, portanto, você precisa apenas religar a interface do componente Java:

- Se esse serviço for chamado por um processo de negócios no mesmo módulo, então, crie uma ligação da referência do processo de negócios adequada para essa interface do componente Java.
- Se esse serviço for chamado por um processo de negócios em outro módulo, crie uma **Exportação com Ligação SCA** e, a partir do outro módulo, arraste e solte essa exportação no Editor de Montagem desse módulo para criar a **Importação com Ligação SCA** correspondente. Ligue a referência do processo de negócios adequada àquela Importação.

- Se esse serviço tiver sido publicado no WebSphere Studio Application Developer Integration Edition para expô-lo externamente, então, consulte a seção "Criando Exportações do SCA para Acessar o Serviço Migrado" para obter instruções sobre como publicar novamente o serviço.

*Criando um Serviço da Web Java: Opção 2:*

Uma opção alternativa a considerar é a ferramenta de serviços da Web do Rational Application Developer que permite criar um serviço da Web em torno de uma classe Java.

**Nota:** Consulte as informações no seguinte site antes de tentar migrar utilizando esse método:  
<http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.was.creation.ui.doc/tasks/twsbeanw.html>

**Nota:** Esta opção requer que o tempo de execução de um serviço da Web seja configurado através do WebSphere Integration Developer antes de chamar o assistente de serviço da Web.

Se você tiver utilizado uma abordagem de baixo para cima no WebSphere

Studio Application Developer Integration Edition para gerar WSDL em torno da classe Java

, então, siga estas etapas:

1. Crie um novo projeto da Web e copie a classe Java em torno da qual gostaria de construir um serviço para a pasta de origem Java desse projeto da Web.
2. Clique com o botão direito do mouse no projeto do aplicativo corporativo que é o contêiner para a classe Java em torno do qual você está criando um serviço.
3. Selecione **Propriedades**, vá para as propriedades do **Servidor** e certifique-se de que **Tempo de execução de destino** esteja definido para **WebSphere Process Server v6.0** e **Servidor padrão** esteja definido para o **WebSphere Process Server v6.0** instalado.
4. Inicie o servidor de teste e implemente seu aplicativo para o servidor e certifique-se de que ele seja iniciado com êxito.
5. Em seguida, clique com o botão direito do mouse na classe Java em torno da qual gostaria de criar um serviço e selecione **Serviços da Web** → **Criar serviço da Web**.
6. Para **Tipo de Serviço da Web**, selecione **Serviço da Web Java bean** e desmarque a opção **Iniciar serviço da Web no projeto da Web** a menos que deseje implementar o serviço da Web diretamente. Você também pode opcionalmente selecionar a geração de um proxy de cliente. Clique em **Avançar**.
7. A classe Java na qual você clicou com o botão direito do mouse será mostrada, clique em **Avançar**.
8. Agora você deve configurar as opções de implementação de serviço. Clique no botão **Editar...** Para o tipo de servidor, escolha **WPS Server v6.0** e para o tempo de execução do serviço da Web, escolha **IBM WebSphere** e J2EE versão **1.4**. Se você não puder selecionar uma combinação válida fazendo isso, consulte a seção "Preparando-se para a Migração" para obter informações sobre a migração de projetos J2EE para o nível v1.4. Clique em **OK**.
9. Para o projeto Serviço, digite o nome do projeto da Web. Selecione também o projeto EAR apropriado. Clique em **Avançar**. Observe que você poderá precisar esperar vários minutos.
10. No painel Identidade do Java Bean de Serviço da Web, selecione o arquivo WSDL que conterá as definições do WSDL. Escolha os métodos que gostaria de expor no serviço da Web e escolha o estilo/codificação adequados (Document/Literal, RPC/Literal ou RPC/Encoded). Selecione a opção **Definir Mapeamento Customizado do Pacote para Espaço de Nomes** e selecione um espaço de nomes que seja exclusivo para a classe Java que está sendo migrada para todos os pacotes Java utilizados por esta interface da classe Java (o espaço de nomes padrão será exclusivo para o nome do pacote que pode causar conflitos se você criar outro Serviço da Web que utiliza as mesmas classes Java). Conclua os outros parâmetros, se necessário.
11. Clique em **Avançar** e, no painel, **Mapeamento de Pacote de Serviço da Web para Espaço de Nomes**, clique no botão **Incluir** e, na linha criada, digite o nome do pacote do Java bean e, em seguida,



inclua o espaço de nomes customizado que identifica exclusivamente esta classe Java. Continue incluindo mapeamentos para todos os pacotes Java utilizados pela interface do Java bean.

12. Clique em **Avançar**. Observe que você poderá precisar esperar vários minutos.
13. Clique em **Concluir**. Depois de concluir o assistente, você deverá copiar o arquivo WSDL gerado, que descreve o serviço Java para o projeto de módulo do Business Integration, se o projeto de serviço for um consumidor do serviço Java. Ele pode ser localizado no projeto da Web do roteador gerado na pasta WebContent/WEB-INF/wsdl. Atualize/reconstrua o projeto de módulo do Business Integration.
14. Mude para a perspectiva Business Integration e expanda o módulo e, em seguida, a categoria lógica **Portas do Serviço da Web**.
15. Selecione a porta que foi criada nas etapas anteriores e arraste-a e solte-a sobre o Editor de Montagem e selecione para criar uma **Importação com Ligação de Serviço da Web**. Selecione a interface WSDL da classe Java, se for solicitado. Agora o componente SCA que consumiu o componente Java na 5.1 pode ser ligado a essa Importação para completar as etapas manuais de migração da religação.

Observe que a interface poderá ser ligeiramente diferente da interface 5.1 e você poderá precisar inserir um componente Mediação de Interface entre o consumidor da 5.1 e a nova Importação. Para fazer isso, clique na ferramenta **wire** no Editor de Montagem e ligue o componente de origem SCA a essa nova **Importação com Ligação de Serviço da Web**. Como as interfaces são diferentes, você será avisado: **Os nós de origem e destino não têm interfaces correspondentes**. Escolha **criar um mapeamento de interface entre o nó de origem e de destino**. Dê um clique duplo no componente de mapeamento que foi criado no Editor de Montagem. Isso abrirá o editor de mapeamento. Consulte o Information Center para obter instruções sobre a criação de um mapeamento de interface.

Se você tiver utilizado uma abordagem de cima para baixo no WebSphere Studio Application Developer Integration Edition, gerando classes Java a partir de uma definição WSDL, siga estas etapas:

1. Crie um novo projeto da Web e copie o arquivo WSDL que gostaria para o esqueleto Java para a pasta de origem desse projeto da Web.
2. Clique com o botão direito do mouse no arquivo WSDL que contém o PortType do qual você deseja gerar um esqueleto Java e selecione **Serviços da Web → Gerar esqueleto de Java bean**.
3. Escolha o tipo de serviço da Web **Serviço da Web Java bean de Esqueleto** e complete o assistente.

Depois de concluir o assistente, você deve ter classes Java que implementem a interface de serviço e não sejam dependentes de APIs do WSIF.

#### *Migrando um Serviço EJB:*

Você pode migrar um serviço EJB para uma Importação do SCA com ligação de bean de sessão sem preservação de estado.

Se o Projeto de Serviço do WebSphere

Studio Application Developer Integration Edition era dependente de outro EJB, cliente EJB ou projeto Java

, importe esses projetos existentes utilizando o assistente **Arquivo → Importar → Projeto Existente para o Espaço de Trabalho**. Geralmente, este era o caso quando um EJB era referido a partir de um projeto de serviço. Se algum dos arquivos WSDL ou XSD referidos a partir do projeto de serviço existir em outro tipo de projeto, crie uma nova Biblioteca de Business Integration com o mesmo nome que o projeto de não-serviço antigo e copie todos os artefatos para a biblioteca.

Importe o projeto de serviço utilizando o assistente de Migração. Isto resultará na criação de um módulo de integração de negócios com Mensagens, PortTypes, Ligações e Serviços do WSDL gerados no WebSphere Studio Application Developer Integration Edition.



Na perspectiva Business Integration, expanda o módulo para ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).

Você tem as seguintes opções:

*Prós e Contras para Cada uma das Opções de Religação do Serviço EJB:*

Existem pros e contras para cada uma das opções de religação do serviço EJB.

A seguinte lista descreve as duas opções e os pros e contras de cada uma:

- A primeira opção provavelmente fornece melhor desempenho no tempo de execução, porque a chamada de um serviço da Web é mais lenta do que a chamada de um EJB.
- A primeira opção pode propagar contexto enquanto uma chamada de serviço da Web não propaga contexto da mesma maneira.
- A segunda opção não envolve a criação de nenhum código customizado.
- A segunda opção não poderá ser possível para algumas definições de interface EJB, porque a geração de um serviço EJB tem limitações. Consulte a documentação do Rational Application Developer aqui: <http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.doc/ref/rlimit.html>
- A segunda opção poderá resultar em uma alteração de interface e, então, em uma alteração no cliente SCA.
- A segunda opção requer que um servidor do WebSphere Process Server 6.0 seja instalado e configurado para funcionar com o WebSphere Integration Developer. Para ver os tempos de execução instalados que são configurados para trabalhar com o WebSphere Integration Developer, vá para **Janela** → **Preferências** → **Servidor** → **Tempos de Execução Instalados** e selecione a entrada **WebSphere Process Server v6.0**, se ela existir, e certifique-se de que ela aponta para o local onde o produto está instalado. Certifique-se de que essa entrada esteja marcada se o servidor não existir e desmarcada se esse servidor não estiver realmente instalado. Você também pode clicar no botão **Incluir...** se desejar incluir outro servidor.
- Se o componente Java foi construído no WebSphere Studio Application Developer Integration Edition utilizando a abordagem de cima para baixo, na qual o esqueleto do EJB foi gerado a partir de um WSDL, então, os parâmetros dentro e fora dessa classe Java provavelmente serão a subclasse `WSIFFormatPartImpl`. Se for esse o caso, então, você escolherá a opção 2 para gerar um novo esqueleto EJB genérico (não dependente das APIs WSIF ou DataObject) a partir da interface WSDL.

*Criando o Componente EJB Customizado: Opção 1:*

A técnica de migração recomendada é utilizar o tipo Importar com Ligação de Sessão sem Preservação de Estado do WebSphere Integration Developer que permite chamar um EJB de sessão sem preservação de estado como um componente SCA. Durante a migração, o código Java customizado deve ser escrito para conversão entre o estilo de interface Java do SCA e o estilo de interface EJB existente.

Para criar o componente EJB customizado:

1. No projeto de módulo, expanda **Interfaces** e selecione a interface WSDL que foi gerada para esse EJB no WebSphere Studio Application Developer Integration.
2. Arraste e solte essa interface sobre o Editor de Montagem. Um diálogo aparecerá solicitando que você selecione o tipo de componente que irá criar. Selecione **Componente sem Tipo de Implementação** e clique em **OK**.
3. Um componente genérico aparecerá no diagrama de Montagem. Selecione-o e vá para a visualização **Propriedades**.
4. Na guia **Descrição**, você pode alterar o nome e o nome de exibição do componente para algo mais descritivo. Escolha um nome como seu nome do EJB, mas anexe um sufixo, como "JavaMed", porque

ele será um componente Java que faz mediação entre a interface WSDL gerada para o EJB no WebSphere Studio Application Developer Integration e a interface Java do EJB.

5. Na guia **Detalhes** você verá que esse componente tem uma interface - aquela que você arrastou e soltou sobre o Editor de Montagem.
6. De volta ao Editor de Montagem, clique com o botão direito do mouse no componente que acabou de criar e selecione **Gerar Implementação... → Java**. Em seguida, selecione o pacote no qual a implementação Java será gerada. Isso cria um esqueleto do serviço Java que adere à interface WSDL de acordo com o modelo de programação SCA, no qual os tipos complexos são representados por um objeto que é um `commonj.sdo.DataObject` e os tipos simples são representados por equivalentes de seus Objetos Java.

Os seguintes exemplos de código mostram:

1. Definições relevantes da interface WSDL 5.1.
2. Os métodos WebSphere Studio Application Developer Integration Edition 5.1 Java que correspondem ao WSDL
3. Os métodos WebSphere Integration Developer 6.0 Java para o mesmo WSDL

O seguinte código mostra as definições relevantes da interface WSDL 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>
```

O seguinte código mostra os métodos WebSphere

Studio Application Developer Integration Edition 5.1 Java

que correspondem ao WSDL:

```
public StockInfo getStockInfo(String symbol)
{
  return new StockInfo();
}
public void setStockPrice(String symbol, float newPrice)
{
  // definir algumas coisas
}
```

O seguinte código mostra os métodos do WebSphere

Integration Developer 6.0 Java

para o mesmo WSDL:

```
public DataObject getStockInfo(String aString) {
    //TODO Precisa ser implementado.
    return null;
}

public void setStockPrice(String symbol, Float newPrice) {
    //TODO Precisa ser implementado.
}
```

Eventualmente, você precisa preencher o código real no qual vê as tags “//TODO” na classe de implementação Java

gerada. Primeiro, é necessário criar uma referência a partir desse componente Java

para o EJB real a fim de que ele possa acessar o EJB de acordo com o modelo de programação SCA:

1. Mantenha o Editor de Montagem aberto e mude para a perspectiva J2EE. Localize o projeto EJB que contém o EJB para o qual você está criando um serviço.
2. Expanda seu item **Descritor de Implementação: <nome\_do\_projeto>** e localize o EJB. Arraste-o e solte-o sobre o Editor de Montagem. Se for avisado sobre dependências do projeto que precisam ser atualizadas, selecione a caixa de opções **Abrir o editor de dependência do módulo...** e clique em **OK**.
3. Na seção J2EE, certifique-se de que o projeto EJB esteja listado e, se não estiver, inclua-o clicando no botão **Incluir...**
4. Salve as dependências do módulo e feche esse editor. Você verá que uma nova Importação foi criada no Editor de Montagem. Você pode selecioná-la e ir para a visualização Propriedades na guia Descrição para alterar o nome da importação e o nome de exibição para algo mais significativo. Na guia Ligação, você verá que o tipo de importação é automaticamente definido para **Ligação de Bean de Sessão Sem Preservação de Estado** e o nome JNDI do EJB já está definido adequadamente.
5. Selecione a ferramenta de Ligação a partir da paleta no Editor de Montagem.
6. Clique no componente Java e solte o mouse.
7. Em seguida, clique em Importação EJB e solte o mouse.
8. Quando for perguntado **A referência correspondente será criada no nó de origem. Deseja continuar?**, clique em **OK**. Isso cria uma ligação entre os dois componentes.
9. Selecione o componente Java no Editor de Montagem e a visualização Propriedades na guia Detalhes, expanda Referências e selecione a referência para o EJB que acabou de ser criado. Você pode atualizar o nome da referência se o nome gerado não for muito descritivo ou adequado. Lembre-se do nome dessa referência para uso futuro.
10. Salve o diagrama de Montagem.

Você deve utilizar o modelo de programação SCA para chamar o EJB a partir da classe Java

gerada. Abra a classe Java

gerada e siga estas etapas para escrever o código que chamará o serviço EJB. Para a classe de implementação Java

gerada:

1. Crie uma variável privada (cujo tipo é aquele de sua interface EJB remota):  
private YourEJBInterface ejbService = null;

2. Se houver tipos complexos na interface EJB, então, crie também uma variável privada para o BOFactory:

```
private BOFactory boFactory = (BOFactory)
    ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo
    /BOFactory");
```

3. No construtor da classe de implementação Java, utilize as APIs do SCA para resolver a referência EJB (lembre-se de preencher o nome da referência EJB que você escreveu há algumas etapas atrás) e defina a variável privada igual a essa referência:

```
// Localize o serviço EJB
this.ejbService = (YourEJBInterface)
    ServiceManager.INSTANCE.locateService("name-of-your-ejb-reference");
```

Para cada “//TODO” na classe de implementação Java

gerada:

1. Converta todos os parâmetros nos tipos de parâmetros que o EJB espera.
2. Chame o método adequado na referência EJB utilizando o modelo de programação SCA, enviando os parâmetros convertidos.
3. Converta o valor de retorno do EJB no tipo de valor de retorno declarado pelo método de implementação Java gerado.

```
/**
 * Método gerado para suportar o tipo de porta WSDL de implementação nomeada
 * "interface.MyBean".
 */
public BusObjImpl getStockInfo(String aString) {
    BusObjImpl boImpl = null;
    try {
        // chame o método EJB
        StockInfo stockInfo = this.ejbService.getStockInfo(aString);
        // formule o objeto de dados SCA a ser retornado.
        boImpl = (BusObjImpl)
            this.boFactory.createByClass(StockInfo.class);
        // converta manualmente todos os dados do tipo de retorno EJB para o
        // objeto de dados SCA a ser retornado
        boImpl.setInt("index", stockInfo.getIndex());
        boImpl.setString("symbol", stockInfo.getSymbol());
        boImpl.setDouble("price", stockInfo.getPrice());
    } catch (RemoteException e) {
        e.printStackTrace();
    }
    return boImpl;
}

/**
 * Método gerado para suportar o tipo de porta WSDL de implementação nomeada
 * "interface.MyBean".
 */
public void setStockPrice(String symbol, Float newPrice) {
    try {
        this.ejbService.setStockPrice(symbol, newPrice.floatValue());
    } catch (RemoteException e) {
        e.printStackTrace();
    }
}
```

*Criando um Serviço da Web EJB: Opção 2:*

Uma opção alternativa a considerar é a ferramenta de serviços da Web do Rational Application Developer que permite criar um serviço da Web em torno de um EJB.

**Nota:** Consulte as informações no seguinte site antes de tentar migrar utilizando esse método:  
<http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.was.creation.ejb.ui.doc/tasks/twsejbw.html>

**Nota:** Esta opção requer que o tempo de execução de um serviço da Web seja configurado através do WebSphere Integration Developer antes de chamar o assistente de serviço da Web.

Para criar um serviço da Web em torno de um EJB, siga estas etapas:

1. Clique com o botão direito do mouse no projeto do aplicativo corporativo que é o contêiner para o EJB em torno do qual você está criando um serviço.
2. Selecione **Propriedades**, vá para as propriedades do **Servidor** e certifique-se de que **Tempo de execução de destino** esteja definido para **WebSphere Process Server v6.0** e **Servidor padrão** esteja definido para o **WebSphere Process Server v6.0** instalado.
3. Inicie o servidor de teste e implemente seu aplicativo para o servidor e certifique-se de que ele seja iniciado com êxito.
4. Na perspectiva J2EE, expanda o **Projeto EJB** na visualização Explorador de Projeto. Expanda o **Descrição de Implementação** e, em seguida, a categoria **Beans de Sessão**. Selecione o bean em torno do qual deseja gerar o serviço da Web.
5. Clique com o botão direito do mouse e selecione **Serviços da Web** → **Criar serviço da Web**.
6. Para **Tipo de Serviço da Web**, selecione **Serviço da Web EJB** e desmarque a opção **Iniciar serviço da Web no projeto da Web**, a menos que deseje implementar o serviço da Web diretamente. Clique em **Avançar**.
7. Certifique-se de que o EJB no qual clicou com o botão direito do mouse esteja selecionado aqui e clique em **Avançar**.
8. Agora você deve configurar as opções de implementação de serviço. Clique no botão **Editar...** Para o tipo de servidor, escolha **WPS Server v6.0** e para o tempo de execução do serviço da Web, escolha **IBM WebSphere** e J2EE versão **1.4**. Se você não puder selecionar uma combinação válida fazendo isso, consulte a seção "Preparando-se para a Migração" para obter informações sobre a migração de projetos J2EE para o nível v1.4. Clique em **OK**.
9. Para o projeto Serviço, digite o nome do projeto EJB que contém o EJB. Selecione também o projeto EAR apropriado. Clique em **Avançar**. Observe que você poderá precisar esperar vários minutos.
10. No painel Configuração EJB do Serviço da Web, selecione o projeto do roteador adequado a ser utilizado (escolha o nome do projeto da Web do roteador que gostaria que fosse criado e esse projeto será incluído no mesmo aplicativo corporativo do EJB original). Selecione o transporte desejado (**SOAP sobre HTTP** ou **SOAP sobre JMS**). Clique em **Avançar**.
11. Selecione o arquivo WSDL que conterá as definições do WSDL. Escolha os métodos que gostaria de expor no serviço da Web e escolha o estilo/codificação adequados (Document/Literal, RPC/Literal ou RPC/Encoded). Selecione a opção **Definir Mapeamento Customizado do Pacote para Espaço de Nomes** e selecione um espaço de nomes que seja exclusivo para o EJB que está sendo migrado para todos os pacotes Java utilizados pelo EJB (o espaço de nomes padrão será exclusivo para o nome do pacote que pode causar conflitos se você criar outro Serviço da Web que utiliza as mesmas classes Java). Conclua os outros parâmetros, se necessário. Existem limitações para cada uma das combinações de estilo/codificação. Consulte as limitações para obter informações adicionais:  
<http://publib.boulder.ibm.com/infocenter/rtnl0600/topic/com.ibm.etools.webservice.doc/ref/rlimit.html>
12. Clique em **Avançar** e, no painel **Mapeamento de Pacote de Serviço da Web para Espaço de Nomes**, clique no botão **Incluir** e, na linha criada, digite o nome do pacote do EJB, em seguida, o espaço de nomes customizado que identifica exclusivamente este EJB. Continue incluindo mapeamentos para todos os pacotes Java utilizados pela interface EJB.
13. Clique em **Avançar**. Observe que você poderá precisar esperar vários minutos.
14. Clique em **Concluir**. Depois de concluir o assistente, você deverá copiar o arquivo WSDL gerado, que descreve o serviço EJB para o projeto de módulo do Business Integration, se o projeto de serviço for

um consumidor do serviço EJB. Ele pode ser localizado no projeto da Web do roteador gerado na pasta WebContent/WEB-INF/wsdl. Atualize/reconstrua o projeto de módulo do Business Integration.

15. Mude para a perspectiva Business Integration e expanda o módulo migrado e, em seguida, a categoria lógica **Portas do Serviço da Web**.
16. Selecione a porta que foi gerada nas etapas anteriores e arraste-a e solte-a sobre o Editor de Montagem e selecione para criar uma **Importação com Ligação de Serviço da Web**. Selecione a interface WSDL do EJB, se for solicitado. Agora o componente SCA que consumiu o EJB na 5.1 pode ser ligado a essa Importação para completar as etapas manuais de migração da relação.

Se você tiver utilizado uma abordagem de cima para baixo no WebSphere Studio Application Developer Integration Edition, gere um esqueleto EJB a partir de uma definição WSDL e, em seguida, siga estas etapas:

1. Crie um novo projeto da Web e copie o arquivo WSDL do qual gostaria de gerar o esqueleto EJB para a pasta de origem desse projeto da Web.
2. Clique com o botão direito do mouse no arquivo WSDL que contém o PortType do qual você deseja gerar um esqueleto EJB e selecione **Serviços da Web → Gerar esqueleto de Java bean**.
3. Escolha o tipo de serviço da Web **Serviço da Web EJB de Esqueleto** e complete o assistente.

Depois de concluir o assistente, você deve ter um EJB que implemente a interface de serviço e não seja dependente de APIs do WSIF.

Observe que a interface poderá ser ligeiramente diferente da interface 5.1 e você poderá precisar inserir um componente Mediação de Interface entre o consumidor da 5.1 e a nova Importação. Para fazer isso, clique na ferramenta **wire** no Editor de Montagem e ligue o componente de origem SCA a essa nova **Importação com Ligação de Serviço da Web**. Como as interfaces são diferentes, você será avisado: **Os nós de origem e destino não têm interfaces correspondentes**. Escolha **criar um mapeamento de interface entre o nó de origem e de destino**. Dê um clique duplo no componente de mapeamento que foi criado no Editor de Montagem. Isso abrirá o editor de mapeamento. Consulte o Information Center para obter instruções sobre a criação de um mapeamento de interface.

Depois de concluir isso, você deverá religar o serviço EJB. Não deve haver nenhuma referência, portanto, você precisa apenas religar a interface do componente Java:

- Se esse serviço for chamado por um processo de negócios no mesmo módulo, então, crie uma ligação da referência do processo de negócios adequada para esse componente EJB.
- Se esse serviço for chamado por um processo de negócios em outro módulo, crie uma **Exportação com Ligação SCA** e, a partir do outro módulo, arraste e solte essa exportação no Editor de Montagem desse módulo para criar a **Importação com Ligação SCA** correspondente. Ligue a referência do processo de negócios adequada àquela Importação.
- Se esse serviço tiver sido publicado no WebSphere Studio Application Developer Integration Edition para expô-lo externamente, então, consulte a seção "Migração de Serviço Não-BPEL de Entrada" para obter instruções sobre como publicar novamente o serviço.

*Migrando um Processo de Negócios para a Chamada do Serviço de Processo de Negócios:*

Este cenário aplica-se a um processo de negócios que chama outro processo de negócios, no qual o segundo processo de negócios é chamado utilizando uma Ligação do Processo WSIF. Esta seção mostra como migrar um BPEL para uma chamada de serviço BPEL utilizando uma ligação ou uma Importação/Exportação com Ligação SCA.

Para migrar um projeto de serviço de ligação de processo (BPEL) para um serviço de saída, siga estas etapas:



1. Na perspectiva Business Integration, expanda o módulo para ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).
2. Existem vários cenários no qual um processo BPEL pode chamar outro processo BPEL. Localize o cenário abaixo que se aplica ao seu aplicativo:
  - Se o BPEL que está sendo chamado estiver no mesmo módulo, crie uma ligação da referência apropriada no primeiro componente BPEL para a interface apropriada no componente BPEL de destino.
  - Se o BPEL que está sendo chamado estiver em outro módulo (em que o outro módulo é um projeto de serviço migrado):
    - a. Crie uma **Exportação com Ligação SCA** para o segundo processo de negócios em seu diagrama de montagem de módulo.
    - b. Expanda o ícone de montagem do segundo módulo no navegador na visualização Business Integration. Você deve ver a exportação que acabou de criar.
    - c. Arraste e solte a exportação a partir da visualização Business Integration no segundo módulo sobre o editor de montagem aberto do primeiro módulo. Isso criará uma Importação com Ligação SCA no primeiro módulo. Se esse serviço tiver sido publicado no WebSphere Studio Application Developer Integration Edition para expô-lo externamente, então, consulte a seção "Criando Exportações do SCA para Acessar o Serviço Migrado".
    - d. Ligue a referência adequada no primeiro processo de negócios para a importação que você acabou de criar no módulo.
    - e. Salve o diagrama de Montagem.
  - Para obter a ligação posterior quando chamar o segundo processo de negócios:
    - a. Deixe a referência do componente do primeiro processo de negócios desligada. Abra o primeiro processo no editor BPEL e, na seção **Parceiros de Referência**, selecione o parceiro que corresponde ao segundo processo BPEL a ser chamado utilizando a ligação posterior.
    - b. Na visualização Propriedades na guia **Descrição**, digite o nome do segundo processo de negócios no campo **Gabarito de Processo**.
    - c. Salve o processo de negócios. Agora você concluiu a configuração da chamada de limite posterior.

#### *Migrando um Serviço da Web (SOAP/JMS):*

Você pode migrar um Serviço da Web (SOAP/JMS) para uma Importação do SCA com Ligação de Serviço da Web.

Para migrar um projeto de serviço SOAP/JMS para uma migração de serviço de saída, siga estas etapas:

1. Primeiro, você precisará importar o projeto de serviço utilizando o assistente de Migração. Isto resultará na criação de um módulo do Business Integration com Mensagens, PortTypes, Ligações e Serviços do WSDL gerados no WebSphere Studio Application Developer Integration Edition. Observe que, se o IBM Web Service (SOAP/JMS) que este aplicativo chamará também for um serviço da Web do WebSphere Studio Application Developer Integration Edition que será migrado, pode ter havido atualizações desse serviço da Web durante a migração. Se este for o caso, será necessário utilizar os arquivos WSDL migrados desse serviço da Web aqui.
2. Na perspectiva Business Integration, expanda o módulo para que seja possível ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).
3. Em seguida, inclua uma Importação que permitirá que o aplicativo interaja com o IBM Web Service (via SOAP/JMS) de acordo com o modelo de programação SCA. Certifique-se de que as definições de interface, ligação e serviço WSDL estejam presentes no módulo migrado ou em uma biblioteca da qual o módulo migrado depende.

4. Na perspectiva Business Integration, expanda o módulo migrado e abra seu Diagrama de Montagem no Editor de Montagem.
5. Expanda a categoria lógica Portas de Serviço da Web e arraste e solte a porta correspondente ao serviço que você deseja chamar no Editor de Montagem.
6. Escolha a criação de uma **Importação com Ligação de Serviço da Web**.
7. Depois de criar a importação, selecione-a no Editor de Montagem e vá para a visualização Propriedades. Na guia Ligação, você verá a porta e o serviço aos quais a importação está ligada.
8. Salve o diagrama de montagem.

Depois de concluir isso, você deverá religar o serviço:

- Se esse serviço for chamado por um processo de negócios no mesmo módulo, então, crie uma ligação da referência do processo de negócios adequada para essa Importação.
- Se esse serviço for chamado por um processo de negócios em outro módulo, crie uma **Exportação com Ligação SCA** e, a partir do outro módulo, arraste e solte essa exportação no Editor de Montagem desse módulo para criar a **Importação com Ligação SCA** correspondente. Ligue a referência do processo de negócios adequada àquela Importação.
- Salve o diagrama de montagem.

*Migrando um Serviço da Web (SOAP/HTTP):*

Você pode migrar um Serviço da Web (SOAP/HTTP) para uma Importação do SCA com Ligação de Serviço da Web.

Para migrar um projeto de serviço SOAP/HTTP para uma migração de serviço de saída, siga estas etapas:

1. Primeiro, você precisará importar o projeto de serviço utilizando o assistente de Migração. Isto resultará na criação de um módulo do Business Integration com Mensagens, PortTypes, Ligações e Serviços do WSDL gerados no WebSphere Studio Application Developer Integration Edition. Observe que, se o IBM Web Service (SOAP/HTTP) que este aplicativo chamará também for um serviço da Web do WebSphere Studio Application Developer Integration Edition que será migrado, pode ter havido atualizações desse serviço da Web durante a migração. Se este for o caso, será necessário utilizar os arquivos WSDL migrados desse serviço da Web aqui.
2. Na perspectiva Business Integration, expanda o módulo para que seja possível ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).
3. Em seguida, inclua uma Importação que permitirá que o aplicativo interaja com o IBM Web Service (via SOAP/HTTP) de acordo com o modelo de programação SCA. Certifique-se de que as definições de interface, ligação e serviço WSDL estejam presentes no módulo migrado ou em uma biblioteca da qual o módulo migrado depende.
4. Na perspectiva Business Integration, expanda o módulo migrado e abra seu Diagrama de Montagem no Editor de Montagem.
5. Expanda a categoria lógica Portas de Serviço da Web e arraste e solte a porta correspondente ao serviço que você deseja chamar no Editor de Montagem.
6. Escolha a criação de uma **Importação com Ligação de Serviço da Web**.
7. Depois de criar a importação, selecione-a no Editor de Montagem e vá para a visualização Propriedades. Na guia Ligação, você verá a porta e o serviço aos quais a importação está ligada.
8. Salve o diagrama de montagem.

Depois de concluir isso, você deverá religar o serviço:

- Se esse serviço for chamado por um processo de negócios no mesmo módulo, então, crie uma ligação da referência do processo de negócios adequada para essa Importação.



- Se esse serviço for chamado por um processo de negócios em outro módulo, crie uma **Exportação com Ligação SCA** e, a partir do outro módulo, arraste e solte essa exportação no Editor de Montagem desse módulo para criar a **Importação com Ligação SCA** correspondente. Ligue a referência do processo de negócios adequada àquela Importação.
- Salve o diagrama de montagem.

#### *Migrando um Serviço JMS:*

Você pode migrar um serviço JMS para uma Importação do SCA com Ligação JMS.

**Nota:** Se a mensagem JMS estiver sendo enviada para um WebSphere Business Integration Adapter, consulte a seção "Migrando Interações com o WebSphere Business Integration Adapter".

Para migrar um projeto de serviço JMS para uma migração de serviço de saída, siga estas etapas:

1. Primeiro, você precisará importar o projeto de serviço utilizando o assistente de Migração. Isto resultará na criação de um módulo do Business Integration com Mensagens, PortTypes, Ligações e Serviços do WSDL gerados no WebSphere Studio Application Developer Integration Edition.
2. Na perspectiva Business Integration, expanda o módulo para que seja possível ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).
3. Em seguida, inclua uma Importação que permitirá que o aplicativo interaja com uma fila JMS de acordo com o modelo de programação SCA.
4. No Editor de Montagem, expanda o projeto de módulo migrado, a categoria **Interfaces** e localize PortType do WSDL que descreve o serviço da Web que o aplicativo chamará. Arraste-o e solte-o sobre o Editor de Montagem.
5. Um diálogo **Criação de Componente** permitirá que você selecione o tipo de componente que irá criar. Escolha **Importar sem Ligação**.
6. Você verá que uma nova Importação foi criada no Editor de Montagem e se selecioná-la e for para a visualização Propriedades na guia Descrição, poderá alterar o nome da importação e o nome de exibição para algo mais significativo.
7. Você pode consultar a ligação WSDL 5.1 e arquivos de serviço para obter detalhes sobre o serviço JMS que está sendo migrado e utilizá-los para preencher os detalhes da "Importação com Ligação JMS" 6.0. Localize a ligação JMS 5.1 e os arquivos WSDL de serviço no projeto de serviço 5.1 (geralmente, eles são denominados \*JMSBinding.wsdl e \*JMSService.wsdl). Examine as informações de ligação e de serviço capturadas lá. A partir da ligação, você pode determinar se as mensagens de texto ou de objeto foram utilizadas e se as ligações de formato de dados customizadas foram utilizadas. Se houver alguma, também será necessário considerar a gravação de uma ligação de dados customizada para "Importação com Ligação JMS" 6.0. A partir do serviço, é possível localizar o depósito de informações do provedor de contexto inicial, o nome da connection factory JNDI, o nome do destino JNDI e o estilo do destino (fila).
8. Clique com o botão direito do mouse na importação e selecione **Gerar Ligação** e, em seguida, **Ligação do JMS**. Você será solicitado a digitar os seguintes parâmetros:

#### **Selecionar domínio do sistema de mensagens JMS:**

- Ponto-a-Ponto
- Publicação-Assinatura
- Independente do Domínio

#### **Selecione como os dados são serializados entre o Objeto de Negócios e a Mensagem JMS:**

- Texto
- Objeto
- Fornecido pelo usuário

**Se Fornecido pelo Usuário for selecionado:**

Especifique o nome completo da classe de implementação com.ibm.websphere.sca.jms.data.JMSDataBinding. Será necessário especificar uma ligação de dados definida pelo usuário se seu aplicativo precisar configurar quaisquer propriedades de cabeçalho JMS que, normalmente, não estão disponíveis na Ligação de Importação JMS. Neste caso, é possível criar uma classe de ligação de dados customizada que estende a ligação de dados JMS padrão "com.ibm.websphere.sca.jms.data.JMSDataBinding" e incluir código customizado para acessar JMSMessage diretamente. Consulte exemplos de JMS em "Criando e Modificando Ligações para Componentes de Importação e Exportação" no link abaixo.

**A conectividade de entrada está utilizando a classe do seletor de função JMS padrão:**

<selecionado> ou <seleção cancelada>

9. Selecione a importação que acabou de criar. Na visualização Propriedades, vá para a guia Ligação. Você pode preencher manualmente todas as informações sobre ligação listadas lá para os mesmos valores que especificou antes no WebSphere Studio Application Developer Integration Edition. A informações de ligação que você poderá especificar são:
- Ligação de Importação do JMS (a mais importante)
  - Conexão
  - Adaptador de Recursos
  - Destinos de JMS
  - Ligações de Método

Depois de concluir isso, você deverá religar o serviço:

- Se esse serviço for chamado por um processo de negócios no mesmo módulo, então, crie uma ligação da referência do processo de negócios adequada para essa Importação.
- Se esse serviço for chamado por um processo de negócios em outro módulo, crie uma **Exportação com Ligação SCA** e, a partir do outro módulo, arraste e solte essa exportação no Editor de Montagem desse módulo para criar a **Importação com Ligação SCA** correspondente. Ligue a referência do processo de negócios adequada àquela Importação.
- Salve o diagrama de montagem.

*Migrando um Serviço J2C-IMS:*

Você pode migrar um serviço J2C-CICS para uma Importação do SCA com Ligação EIS ou Importação do SCA com Ligação de Serviço da Web.

Não utilize nenhum dos artefatos do WebSphere

Studio Application Developer Integration Edition gerados para esse serviço IMS

. Você precisará recriar o serviço utilizando os assistentes disponíveis no WebSphere

Integration Developer e religar manualmente o aplicativo.

**Nota:** Ative a Auto-construção ou construa o módulo manualmente.

Você tem as seguintes opções:

**Nota:** Para ambas as opções, observe que se um serviço BPEL chamar esse serviço IMS, o BPEL precisará ser alterado ligeiramente, porque a interface exposta pelo serviço EIS será ligeiramente diferente da antiga interface 5.1. Para fazer isso, abra o editor BPEL e ajuste o link de parceiro correspondente ao serviço EIS e utilize a nova interface (arquivo WSDL) gerada ao executar as etapas acima. Faça todas as alterações necessárias nas atividades do BPEL para a nova interface WSDL do serviço EIS.

### *Prós e Contras para Cada uma das Opções de Religação do Serviço J2C-IMS:*

Existem prós e contras para cada uma das opções de religação do serviço J2C-IMS.

A seguinte lista descreve as duas opções e os prós e contras de cada uma:

- A primeira opção utiliza o componente SCA padrão para chamar o serviço IMS.
- A primeira opção tem algumas limitações:
  - A API de especificação do SDO versão 1 não fornece acesso à matriz de byte COBOL ou C – isso terá impacto nos clientes que trabalham com vários segmentos IMS.
  - A especificação do SDO versão 1 para serialização não suporta redefinições COBOL ou uniões C.
- A segunda opção utiliza a abordagem JSR 109 padrão para conectar-se ao serviço IMS. Essa funcionalidade está disponível como parte do Rational Application Developer.

### *Criar uma Importação do SCA para Chamar o Serviço IMS: Opção 1:*

Você pode criar uma Importação do SCA com Ligação EIS que utilizará DataObjects para armazenar a mensagem/dados para se comunicar com o sistema IMS.

Para criar uma Importação do SCA para chamar o serviço IMS

, siga estas etapas:

1. Crie um novo projeto de módulo do Business Integration para hospedar esse novo serviço IMS.
2. Para recriar o serviço EIS, vá para **Arquivo** → **Novo** → **Outro** → **Business Integration** → **Enterprise Service Discovery**.
3. Esse assistente permite importar um serviço a partir de um sistema EIS. Ele é muito similar ao assistente do WebSphere Studio Application Developer Integration Edition que criou o serviço EIS baseado no WSIF na 5.1. Você pode importar o novo adaptador de recursos IMS do J2C nesse assistente. Você deve procurar o diretório no qual o WebSphere Integration Developer está instalado e pesquisar detalhadamente **Adaptadores de Recursos** → **ims15** → **imsico9102.rar**.

**Nota:** Consulte o Information Center para obter informações adicionais sobre a conclusão das propriedades de salvamento e dos painéis de operação. Durante o assistente Enterprise Service Discovery, quando você incluir uma operação também poderá criar objetos de negócios para o tipo de dados de entrada ou saída da operação. Isso requer que você tenha o arquivo de origem C ou COBOL utilizado no assistente do WebSphere Studio Application Developer Integration Edition. Esses arquivos devem ter sido copiados para o projeto de serviço antigo, portanto, você poderá apontar para os arquivos de origem contidos nele. Você também pode importar os objetos de negócios utilizando o assistente separado **Arquivo** → **Novo** → **Outro** → **Business Integration** → **Enterprise Data Discovery**.

4. Depois de concluir o assistente, abra a perspectiva Business Integration e expanda o módulo para que possa ver seu conteúdo. Você deve ver novos objetos de negócios listados nos Tipos de Dados do módulo e novas interfaces listadas em Interfaces.
5. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto). Você deve ver que existe uma Importação no canvas, essa Importação tem uma Ligação EIS e representa o serviço que você acabou de criar.

Agora consulte a seção intitulada “Criando Exportações do SCA para Acessar o Serviço Migrado” para obter instruções sobre como expor esse serviço para os consumidores.

### *Criar um Serviço da Web em torno do Serviço J2C: Opção:*

Você pode criar um serviço da Web J2C e se o consumidor do serviço for um componente do SCA, consuma o serviço como um IBM Web Service (SOAP/HTTP ou SOAP/JMS).

Para criar um serviço da Web em torno do serviço J2C, siga estas etapas:

1. Crie o Java Bean do J2C clicando em **Arquivo** → **Novo** → **J2C** → **Java Bean do J2C**
2. Escolha a versão 1.5 do **Conector IMS para Java** e clique em **Avançar**.
3. Marque **Conexão Gerenciada** e digite o nome de consulta do JNDI. Clique em **Avançar**.
4. Especifique o projeto, o pacote e o nome para o novo Java bean. O bean consiste em uma interface e em uma classe de implementação. Clique em **Avançar**.
5. Inclua um método Java para cada função ou serviço que você deseja acessar a partir do EIS. Métodos adicionais podem ser incluídos posteriormente no editor de origem Java através da visualização Snippets. Ao clicar no botão **Incluir...**, escolha o nome para o método e clique em **Avançar**.
6. Agora você pode escolher **Procurar...** para reutilizar tipos existentes ou **Novo...** para ativar o Assistente de Ligação de Dados Java do CICS/IMS (no qual você pode referir-se a um arquivo de origem COBOL ou C) para os tipos de dados de entrada e saída.
7. Depois de concluir a criação de métodos Java, clique em **Avançar**.
8. Complete as etapas restantes nesse assistente para criar o Java Bean do J2C.
9. Crie o Serviço da Web clicando em **Arquivo** → **Novo** → **J2C** → **Página da Web, Serviço da Web ou EJB em Java Bean do J2C** para criar o serviço da Web em torno do Java Bean do J2C.
10. Complete o assistente.

Os consumidores desse serviço agora podem utilizar o serviço WSDL gerado por esse assistente para chamar o serviço IMS

.

*Migrando um Serviço ECI do J2C-CICS:*

Você pode migrar um serviço ECI do J2C-CICS para uma Importação do SCA com Ligação EIS ou Importação do SCA com Ligação de Serviço da Web.

Siga as instruções no tópico "Migrando um Projeto de Serviço J2C-IMS", mas certifique-se de importar o seguinte arquivo RAR *instead* do arquivo RAR do IMS

:

- Procure o diretório no qual o WebSphere Integration Developer está instalado e pesquise detalhadamente **Adaptadores de Recurso** → **cics15** → **cicseci.rar**.

Se você seguir a segunda opção para criar um serviço da Web do J2C, escolha o **ECIResourceAdapter** da v1.5 no segundo painel do assistente de criação do Java

Bean do J2C.

Consulte também o tópico "Migrando um Serviço J2C-IMS".

*Migrando um Serviço EPI do J2C-CICS:*

Não há suporte direto para o serviço EPI do J2C-CICS no WebSphere Integration Developer. Para acessar este serviço a partir de um módulo SCA, será necessário fazer a migração utilizando o *cenário de consumo*.

Consulte o tópico "O Cenário de Consumo para Migração de Serviço" para obter instruções sobre como migrar este tipo de serviço para o WebSphere

Integration Developer.

### *Migrando um Serviço J2C-HOD:*

Não há suporte direto para o serviço J2C-HOD no WebSphere Integration Developer. Para acessar este serviço a partir de um módulo SCA, será necessário fazer a migração utilizando o *cenário de consumo*.

Consulte o tópico "O Cenário de Consumo para Migração de Serviço" para obter instruções sobre como migrar este tipo de serviço para o WebSphere

Integration Developer.

### *Migrando um Serviço do Transformador:*

Você pode migrar um serviço de transformador para um Mapa de Dados e Mapa de Interface do SCA, quando possível. Você também pode utilizar o *cenário de consumo* para acessar este serviço a partir de um módulo SCA.

Os componentes do mapa de dados e do mapa de interface são novos na versão 6.0. Eles oferecem função similar para o serviço de transformador da 5.1, mas não têm a capacidade total de transformação do XSL. Se você não puder substituir seu serviço de transformador por um desses componentes, então, deverá migrar utilizando o cenário de consumo, pois não há suporte direto para o serviço de transformador no WebSphere

Integration Developer. Siga as etapas documentadas na seção "O Cenário de Consumo para Migração de Serviço" para acessar este serviço a partir de um módulo SCA.

### *O Cenário de Consumo para Migração de Serviço:*

Nos casos em que não há nenhuma contraparte direta para um tipo de serviço do WebSphere Studio Application Developer Integration Edition, um cenário de consumo é necessário para consumir o serviço antigo do WebSphere Studio Application Developer Integration Edition como está ao reprojeter o aplicativo no WebSphere Integration Developer.

A seguir há algumas etapas para executar no WebSphere Studio Application Developer Integration Edition *antes* de chamar o assistente de Migração:

1. Crie um novo projeto Java para manter seu código do proxy de cliente. Não coloque esse código do proxy de cliente no projeto de serviço, porque as mensagens geradas no estilo 5.1 e as classes de Java bean serão ignoradas pelo assistente de Migração automática que migra projetos de serviço.
2. Abra o WebSphere Studio Application Developer Integration Edition e clique com o botão direito do mouse no arquivo WSDL que contém a ligação e o serviço do transformador e selecione **Serviços Corporativos** → **Gerar Proxy de Serviço**. Você será solicitado sobre o tipo de proxy que irá criar, mas apenas o **WSIF (Web Services Invocation Framework)** estará disponível. Clique em **Avançar**.
3. Você pode agora especificar o pacote e o nome da classe Java do proxy de serviço que irá criar (você criará o proxy no projeto de serviço atual). Clique em **Avançar**.
4. Agora você pode especificar o estilo de proxy, escolher **Stub de Cliente**, selecionar as operações desejadas que irá incluir no proxy e clicar em **Concluir**. Isto cria uma classe Java que expõe os mesmos métodos que o serviço do WebSphere Studio Application Developer Integration Edition, em que os argumentos dos métodos Java são as partes da mensagem WSDL de origem.

Agora você pode migrar para o WebSphere Integration Developer:

1. Copie o projeto Java do proxy do cliente para o novo espaço de trabalho e importe-o indo para **Arquivo** → **Importar** → **Projeto Existente para Espaço de Trabalho**.
2. Importe o projeto de serviço utilizando o assistente de Migração. Isto resultará na criação de um módulo do Business Integration com Mensagens, PortTypes, Ligações e Serviços do WSDL gerados no WebSphere Studio Application Developer Integration Edition.

3. Na perspectiva Business Integration, expanda o módulo para que seja possível ver seu conteúdo. Abra o Editor de Montagem dando um clique duplo no primeiro item no projeto do módulo (ele terá o mesmo nome que o projeto).
4. Para criar o componente Java customizado, sob o projeto do módulo, expanda **Interfaces** e selecione a interface WSDL que foi gerada para esse serviço de transformador no WebSphere Studio Application Developer Integration Edition.
5. Arraste e solte essa interface sobre o Editor de Montagem. Um diálogo aparecerá solicitando que você selecione o tipo de componente que irá criar. Selecione **Componente sem Tipo de Implementação** e clique em **OK**.
6. Um componente genérico aparecerá no diagrama de Montagem. Selecione-o e vá para a visualização **Propriedades**.
7. Na guia **Descrição**, você pode alterar o nome e o nome de exibição do componente para algo mais descritivo (nesse caso, nomeie-o para algo como seu nome do EJB, mas anexe um sufixo, como "JavaMed", porque ele será um componente Java que faz mediação entre a interface WSDL gerada para o serviço de transformador no WebSphere Studio Application Developer Integration Edition e a interface Java do proxy de cliente de transformador).
8. Na guia **Detalhes** você verá que esse componente tem uma interface - aquela que você arrastou e soltou sobre o Editor de Montagem.
9. De volta ao Editor de Montagem, clique com o botão direito do mouse no componente que acabou de criar e selecione **Gerar Implementação... → Java**. Em seguida, selecione o pacote no qual a implementação Java será gerada. Isso cria um esqueleto do serviço Java que adere à interface WSDL de acordo com o modelo de programação SCA, no qual os tipos complexos são representados por um objeto que é um `commonj.sdo.DataObject` e os tipos simples são representados por equivalentes de seus Objetos Java.

Agora, você precisará preencher o código no qual vê as tags `//TODO` na classe de implementação Java gerada. Existem duas opções:

1. Mova a lógica da classe Java original para essa classe, adaptando-a para a nova estrutura de dados.
2. Crie uma instância privada da classe Java antiga dentro dessa classe Java gerada e escreva código para:
  - a. Converter todos os parâmetros da classe de implementação Java gerada em parâmetros que a classe Java antiga espera.
  - b. Chamar a instância privada da classe Java antiga com os parâmetros convertidos.
  - c. Converter o valor de retorno da classe Java antiga para o tipo de valor de retorno declarado pelo método de implementação Java gerado.

Depois de concluir as opções acima, você deverá religar o proxy de cliente. Não deve haver nenhuma "referência", portanto, você precisa apenas religar a interface do componente Java:

- Se esse serviço for chamado por um processo de negócios no mesmo módulo, então, crie uma ligação da referência do processo de negócios adequada para essa interface do componente Java.
- Se esse serviço for chamado por um processo de negócios em outro módulo, crie uma **Exportação com Ligação SCA** e, a partir do outro módulo, arraste e solte essa exportação no Editor de Montagem desse módulo para criar a **Importação com Ligação SCA** correspondente. Ligue a referência do processo de negócios adequada àquela Importação.
- Se esse serviço tiver sido publicado no WebSphere Studio Application Developer Integration Edition para expô-lo externamente, então, consulte a seção "Criando Exportações do SCA para Acessar o Serviço Migrado" para obter instruções sobre como publicar novamente o serviço.

### Criando Exportações SCA para Acessar o Serviço Migrado:

Uma Exportação SCA deve ser criada para disponibilizar o serviço migrado para consumidores externos, de acordo com o modelo SCA para todos os serviços para os quais o código de implementação foi gerado



no projeto de serviço do WebSphere Studio Application Developer Integration Edition. Isto inclui todos os serviços chamados por clientes externos para o aplicativo.

Se no WebSphere Studio Application Developer Integration Edition, você clicava com o botão direito do mouse no processo BPEL ou em outro WSDL de Serviço e selecionava **Serviços Corporativos** → **Gerar Código de Implementação**, será necessário desempenhar as etapas de migração manuais abaixo. Observe que o WebSphere Integration Developer é diferente do WebSphere Studio Application Developer Integration Edition porque ele armazena todas as opções de implementação. Quando o projeto for construído, o código de implementação será atualizado automaticamente nos projetos EJB e da Web gerados, portanto, não existe mais nenhuma opção para **Gerar Código de Implementação** manualmente.

Foram especificadas cinco opções de ligação na seção Interfaces para Parceiros do assistente Gerar Código de Implementação de BPEL. As seguintes informações de migração de serviço do BPEL de entrada fornecem detalhes adicionais sobre o tipo e propriedades da Exportação a ser criada com base nos tipos de ligação de implementação que foram selecionados no WebSphere Studio Application Developer Integration Edition:

- EJB
- IBM Web Service (SOAP/JMS)
- IBM Web Service (SOAP/HTTP)
- Apache Web Service (SOAP/HTTP)
- JMS

*Migrando o EJB e as Ligações do Processo EJB:*

O EJB e as ligações do processo EJB podem ser migrados para a construção SCA recomendada.

No WebSphere Studio Application Developer Integration Edition, este tipo de ligação permite que clientes se comuniquem com um processo BPEL ou outro tipo de serviço chamando um EJB. Observe que este tipo de ligação não era opcional para microprocessos – ele era sempre selecionado como EJB gerado utilizado internamente por outros tipos de ligação.

O nome JNDI do EJB gerado era gerado automaticamente como uma combinação do nome, espaço de nomes de destino e time stamp valid-from do BPEL. Por exemplo, estes atributos podem ser localizados examinando as propriedades do processo BPEL no editor BPEL nas guias de conteúdo Descrição e Servidor:

*Tabela 3. Espaço de Nomes Gerado*

Nome do Processo	MyService
Espaço de Nomes de Destino	http://www.example.com/process87787141/
Válido a partir de	01 de Jan de 2003 02:03:04

O espaço de nomes gerado para este exemplo é com/example/www/process87787141/MyService20030101T020304.

No WebSphere Studio Application Developer Integration Edition, quando a ligação EJB era selecionada como o tipo de implementação, nenhuma opção era especificada.

Existem quatro opções para migrar a ligação do processo do WebSphere Studio Application Developer Integration Edition. O tipo de cliente(s) que acessa(m) o serviço determinará qual(is) opção(ões) de migração abaixo será(ão) desempenhada(s):

**Nota:** Após a conclusão das etapas de migração manuais, o cliente também deverá ser migrado para o novo modelo de programação. Consulte o tópico apropriado para os seguintes tipos de clientes:

Tabela 4. Informações Adicionais para Migrar Clientes

Tipo de Cliente	Para obter informações adicionais, consulte
Cliente EJB que chama o bean de sessão gerado. Tal cliente chamaria um método EJB correspondente à operação BPEL a ser chamada	Migrando o Cliente EJB
Cliente WSIF que utiliza a ligação do processo EJB	Migrando o Cliente de Ligação do Processo EJB
API EJB Genérica do Business Process Choreographer	Migrando o cliente da API EJB genérica do Business Process Choreographer
API do Sistema de Mensagens Genérica do Business Process Choreographer	Migrando o Cliente da API do Sistema de Mensagens Genérica do Business Process Choreographer
Outro processo BPEL no mesmo módulo	N/D: Ligar componentes BPEL utilizando o Editor de Montagem
Outro processo BPEL em um módulo diferente	N/D: Criar uma <b>Importação com Ligação SCA</b> no módulo de referência e configurar sua ligação para apontar para a <b>Exportação com Ligação SCA</b> criada abaixo na Opção 1

É importante observar que se o processo de negócios transmitir uma referência para si mesmo fora de seu módulo (através de uma referência de serviço), você deve sempre seguir a opção 1 abaixo (você pode sempre executar mais de uma dessas opções) para criar uma Exportação com Ligação SCA para o processo de negócios. Apenas um processo de negócios por módulo poderá transmitir sua referência de serviço fora do módulo, porque sua exportação deve ser marcada como a exportação padrão do módulo. Isso é feito especificando "true" para o atributo chamado "default" de uma exportação, como em:

Referência do nó de extremidade padrão

Você deve marcar manualmente essa exportação do processo de negócios como o padrão clicando com o botão direito do mouse na visualização Exportar no Business Integration e selecionando **Abrir Com** e, em seguida, **Editor de Texto**.

#### Opção de Migração 1 para o EJB e a Ligação de Processo EJB:

A primeira opção de migração para a ligação do processo EJB do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis para outro componente no mesmo módulo.

No Editor de Montagem, ligue este outro componente ao componente BPEL:

1. Selecione o item **Ligação** na barra de ferramentas.
2. Clique no outro componente para selecioná-lo como a origem da ligação.
3. Clique no componente **BPEL SCA** para selecioná-lo como o destino da ligação.
4. Salve o diagrama de montagem.

#### Opção de Migração 2 para o EJB e a Ligação de Processo EJB:

A segunda opção de migração para a ligação do processo EJB do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis para outros módulos e clientes SCA.

**Nota:** Estas etapas serão obrigatórias se as APIs genéricas do Business Process Choreographer forem utilizadas para chamar o processo de negócios.

A Exportação com Ligação SCA torna um componente SCA acessível por outros módulos SCA. Para criar uma Exportação com uma ligação SCA:



1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Exportação com Ligação SCA para cada interface de processo BPEL que teve uma ligação EJB gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Clique com o botão direito do mouse no componente BPEL no Editor de Montagem.
  - b. Selecione **Exportar...**
  - c. Selecione **Ligação SCA**.
  - d. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
  - e. Quando a Exportação de SCA for criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
  - f. Salve o diagrama de montagem.

*Opção de Migração 3 para o EJB e a Ligação de Processo EJB:*

A terceira opção de migração para a ligação do processo EJB do WebSphere Studio Application Developer Integration Edition é tornar os módulos acessíveis por uma entidade não-SCA (por exemplo, um cliente JSP ou Java).

A Referência Independente torna um componente SCA acessível por qualquer cliente externo. Para criar uma Referência Independente:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de Migração.
2. Crie uma Referência Independente para cada interface de processo BPEL que teve uma ligação EJB gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Selecione o item **Referências Independentes** na barra de ferramentas.
  - b. Clique no canvas do Editor de Montagem para criar uma entidade SCA de Referências Independentes.
  - c. Selecione o item **Ligação** na barra de ferramentas.
  - d. Clique na entidade **Referências Independentes** para selecioná-la como a origem da ligação.
  - e. Clique no componente **BPEL SCA** para selecioná-lo como o destino da ligação.
  - f. Você verá um alerta **A referência correspondente será criada no nó de origem. Deseja continuar?**, clique em **OK**.
  - g. Selecione a entidade **Referências Independentes** recém-criada na visualização Propriedades e selecione a área de janela de conteúdo **Descrição**.
  - h. Expanda o link **Referências** e selecione a referência recém-criada. O nome e a descrição da referência são listados e podem ser modificados se necessário.
  - i. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
  - j. Salve o diagrama de montagem.

*Opção de Migração 4 para o EJB e a Ligação de Processo EJB:*

A quarta opção de migração para a ligação de processo EJB do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis por um cliente de Serviços da Web.

A Exportação com ligação de serviço da Web torna um componente SCA acessível por um cliente de serviços da Web externo. Para criar uma Exportação com Ligação de Serviço da Web:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Exportação com Ligação SCA para cada interface de processo BPEL que teve uma ligação EJB gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Clique com o botão direito do mouse no componente BPEL no Editor de Montagem.

- b. Selecione **Exportar...**
- c. Selecione **Ligação de Serviço da Web**.
- d. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
- e. Selecione o transporte: **soap/http** ou **soap/jms**.
- f. Quando a Exportação de Serviços da Web for criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
- g. Salve o diagrama de montagem.

*Migrando o JMS e as Ligações do Processo JMS:*

O JMS e as ligações do processo JMS podem ser migrados para a construção SCA recomendada.

No WebSphere Studio Application Developer Integration Edition, este tipo de ligação permite que os clientes se comuniquem com um processo BPEL ou outro tipo de serviço enviando uma mensagem para um MDB. Observe que este tipo de ligação não era opcional para processos de execução longa e ele era sempre selecionado. De fato, este tipo de ligação era o **único** tipo de ligação permitido para interfaces de pedido-resposta de processos de execução longa. Para outros tipos de serviço, um MDB será gerado e chamará o serviço adequado.

O nome JNDI utilizado pela ligação JMS era uma combinação do nome, do espaço de nomes de destino e de time stamp valid-from de BPEL.

No WebSphere Studio Application Developer Integration Edition, quando a ligação JMS era selecionada como o tipo de implementação para um processo BPEL, as seguintes opções eram especificadas:

- **Connection Factory JNDI** - o padrão é `jms/BPECF` (este é o nome JNDI da connection factory de fila do contêiner do processo de negócios de destino)
- **Fila de Destino JNDI** - o padrão é `jms/BPEIntQueue` (este é o nome JNDI da fila interna do contêiner de processo de negócios de destino)
- **URL do Provedor JNDI: Fornecido pelo Servidor ou Customizado** - é necessário digitar um endereço. O padrão é `iiop://localhost:2809`

Existem cinco opções para migrar a ligação do processo JMS do WebSphere Studio Application Developer Integration Edition. O tipo de cliente(s) que acessa(m) o serviço determinará qual(is) opção(ções) de migração abaixo será(ão) desempenhada(s):

**Nota:** Após a conclusão das etapas de migração manuais, o cliente também deverá ser migrado para o novo modelo de programação. Consulte o tópico apropriado para os seguintes tipos de clientes:

*Tabela 5. Informações Adicionais para Migrar Clientes*

Tipo de Cliente	Para obter informações adicionais, consulte
Cliente WSIF que utiliza a ligação de processo JMS	Migrando o Cliente da API do Sistema de Mensagens Genérica do Business Process Choreographer e o Cliente de Ligação de Processo JMS
API EJB Genérica do Business Process Choreographer	Migrando o Cliente da API EJB Genérica do Business Process Choreographer
API do Sistema de Mensagens Genérica do Business Process Choreographer - Migrando Negócios	Migrando o Cliente da API do Sistema de Mensagens Genérica do Business Process Choreographer
Outro processo BPEL no mesmo módulo	N/D: Ligar componentes BPEL utilizando o Editor de Montagem

Tabela 5. Informações Adicionais para Migrar Clientes (continuação)

Tipo de Cliente	Para obter informações adicionais, consulte
Outro processo BPEL em um módulo diferente	N/D: Criar uma 'Importação com Ligação SCA' no módulo de referência e configurar sua ligação para apontar para a 'Exportação com Ligação SCA' criada abaixo na Opção 1.

É importante observar que se o processo de negócios transmitir uma referência para si mesmo fora de seu módulo (através de uma referência de serviço), você deve sempre seguir a opção 1 abaixo (você pode sempre executar mais de uma dessas opções) para criar uma Exportação com Ligação SCA para o processo de negócios. Apenas um processo de negócios por módulo poderá transmitir sua referência de serviço fora do módulo, porque sua exportação deve ser marcada como a exportação padrão do módulo. Isso é feito especificando "true" para o atributo chamado "default" de uma exportação, como em:

Referência do nó de extremidade padrão

Você deve marcar manualmente essa exportação do processo de negócios como o padrão clicando com o botão direito do mouse na visualização Exportar no Business Integration e selecionando **Abrir Com** e, em seguida, **Editor de Texto**.

#### *Opção de Migração 1 para o JMS e a Ligação de Processo JMS:*

A primeira opção de migração para a ligação do processo JMS do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis para outro componente no mesmo módulo.

No Editor de Montagem, ligue este outro componente ao componente BPEL:

1. Selecione o item **Ligação** na barra de ferramentas.
2. Clique no outro componente para selecioná-lo como a origem da ligação.
3. Clique no componente **BPEL SCA** para selecioná-lo como o destino da ligação.
4. Salve o diagrama de montagem.

#### *Opção de Migração 2 para o JMS e a Ligação de Processo JMS:*

A segunda opção de migração para a ligação do processo JMS do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis para outros módulos e clientes SCA.

A Exportação com Ligação SCA torna um componente SCA acessível por outros módulos SCA. Para criar uma Exportação com uma ligação SCA:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Exportação com Ligação SCA para cada interface de processo BPEL que teve uma ligação JMS gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Clique com o botão direito do mouse no componente BPEL no Editor de Montagem.
  - b. Selecione **Exportar...**
  - c. Selecione **Ligação SCA**.
  - d. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
  - e. Quando a Exportação de SCA for criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
  - f. Salve o diagrama de montagem.

### *Opção de Migração 3 para o JMS e a Ligação de Processo JMS:*

A terceira opção de migração para a ligação do processo JMS do WebSphere Studio Application Developer Integration Edition é tornar os processos de negócios acessíveis por uma entidade não-SCA (por exemplo, um cliente JSP ou Java).

A Referência Independente torna um componente SCA acessível por qualquer cliente externo. Para criar uma Referência Independente:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Referência Independente para cada interface de processo BPEL que teve uma ligação JMS gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Selecione o item **Referências Independentes** na barra de ferramentas.
  - b. Clique no canvas do Editor de Montagem para criar uma entidade SCA de Referências Independentes.
  - c. Selecione o item **Ligação** na barra de ferramentas.
  - d. Clique na entidade **Referências Independentes** para selecioná-la como a origem da ligação.
  - e. Clique no componente **BPEL SCA** para selecioná-lo como o destino da ligação.
  - f. Você verá um alerta **A referência correspondente será criada no nó de origem. Deseja continuar?**, clique em **OK**.
  - g. Selecione a entidade **Referências Independentes** recém-criada na visualização Propriedades e selecione a área de janela de conteúdo **Descrição**.
  - h. Expanda o link **Referências** e selecione a referência recém-criada. O nome e a descrição da referência são listados e podem ser modificados se necessário.
  - i. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
  - j. Salve o diagrama de montagem.

### *Opção de Migração 4 para o JMS e a Ligação de Processo JMS:*

A quarta opção de migração para a ligação do processo JMS do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis por um cliente de serviços da Web.

A Exportação com ligação de serviço da Web torna um componente SCA acessível por um cliente de serviços da Web externo. Para criar uma Exportação com ligação de serviço da Web:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Exportação com Ligação SCA para cada interface de processo BPEL que teve uma ligação JMS gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Clique com o botão direito do mouse no componente BPEL no Editor de Montagem.
  - b. Selecione **Exportar...**
  - c. Selecione **Ligação de Serviço da Web**.
  - d. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
  - e. Selecione o transporte: **soap/http** ou **soap/jms**.
  - f. Quando a Exportação de Serviços da Web for criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
  - g. Salve o diagrama de montagem.

### *Opção de Migração 5 para o JMS e a Ligação de Processo JMS:*

A quinta opção de migração para a ligação do processo JMS do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis por um cliente JMS.

A Exportação com ligação JMS torna um componente SCA acessível por um cliente JMS externo. Para criar uma Exportação com Ligação JMS:

1. Para serviços BPEL, será necessário criar e fazer referência a novos recursos de fila, pois a ligação de processo JMS 5.1 era muito diferente da ligação JMS 5.1 padrão. Para serviços não-BPEL, é possível localizar os valores selecionados para o código de implementação JMS no WebSphere Studio Application Developer Integration Edition 5.1, localizando o arquivo WSDL denominado **JMSBinding.wsdl** e **JMSService.wsdl** no pacote apropriado, sob a pasta **ejbModule/META-INF** do projeto EJB gerado e inspecionando as informações de ligação e de serviço capturadas lá. A partir da ligação, você pode determinar se as mensagens de texto ou de objeto foram utilizadas e se as ligações de formato de dados customizadas foram utilizadas. Se houver alguma, também será necessário considerar a gravação de uma ligação de dados customizada para **Exportação com Ligação JMS** de 6.0. A partir do serviço, é possível localizar o depósito de informações do provedor de contexto inicial, o nome da connection factory JNDI, o nome do destino JNDI e o estilo do destino (fila).
2. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
3. Crie uma Exportação com Ligação JMS para cada interface de processo BPEL que teve uma ligação JMS gerada para ela no WebSphere Studio Application Developer Integration Edition, clicando com o botão direito do mouse no componente BPEL no Editor de Montagem.
4. Selecione **Exportar...**
5. Selecione **Ligação JMS**.
6. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
7. No próximo painel (atributos de Ligação de Exportação JMS), selecione **Domínio do Sistema de Mensagens JMS**. Defina este atributo como **Ponto-a-Ponto**.
8. Selecione **como os dados são serializados entre o Objeto de Negócios e a Mensagem JMS** e digite os seguintes valores (é recomendável selecionar **Texto** em vez de **Objeto** porque o texto, que geralmente é XML, é independente do tempo de execução e permite a integração de serviço entre sistemas diferentes):
  - a. Para **Texto**, selecione para utilizar o **Seletor de função JMS padrão** ou digite o nome completo da classe de implementação FunctionSelector.
  - b. Para **Objeto**, selecione para utilizar o **Seletor de função JMS padrão** ou digite o nome completo da classe de implementação FunctionSelector.
  - c. Para **Fornecido pelo Usuário**, digite o nome completo da classe de implementação JMSDataBinding. Será necessário selecionar **Fornecido pelo Usuário** se seu aplicativo precisar acessar propriedades do cabeçalho JMS que não estão prontamente disponíveis na Ligação de Importação JMS. Neste caso, é necessário criar uma classe de ligação de dados customizada que estende a ligação de dados JMS padrão **com.ibm.websphere.sca.jms.data.JMSDataBinding** e incluir o código customizado para acessar JMSMessage diretamente. Em seguida, você fornecerá o nome de sua classe customizada para este campo. Consulte exemplos de JMS em "Criando e Modificando Ligações para Componentes de Importação e Exportação" no link abaixo.
  - d. Para **Fornecido pelo Usuário**, selecione para utilizar o **Seletor de função JMS padrão** ou digite o nome completo da classe de implementação FunctionSelector.
9. Quando a Exportação com Ligação JMS tiver sido criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
10. Selecione a área de janela de conteúdo Ligação para ver muito mais opções.
11. Salve o diagrama de montagem.

*Migrando a Ligação do IBM Web Service (SOAP/JMS):*



A ligação do IBM Web Service (SOAP/JMS) para um processo BPEL ou outro tipo de serviço pode ser migrada para a construção SCA recomendada.

No WebSphere Studio Application Developer Integration Edition, este tipo de ligação permitia que os clientes se comunicassem com um processo BPEL ou outro tipo de serviço, chamando um IBM Web Service, no qual o protocolo de comunicação era JMS e a mensagem seguia as regras de codificação SOAP.

A seguir está um exemplo das convenções utilizadas durante a geração de um IBM Web Service (SOAP/JMS) para um serviço BPEL 5.1. O nome JNDI do IBM Web Service gerado era uma combinação do nome, espaço de nomes de destino e time stamp valid-from do BPEL, bem como o nome da interface (tipo de porta WSDL para o qual o código de implementação foi gerado). Por exemplo, estes atributos podem ser localizados examinando as propriedades do processo BPEL no editor BPEL nas guias de conteúdo Descrição e Servidor:

*Tabela 6. Espaço de Nomes Gerado*

Nome do Processo	MyService
Espaço de Nomes de Destino	http://www.example.com/process87787141/
Válido a partir de	01 de Jan de 2003 02:03:04
Interface	ProcessPortType

O espaço de nomes gerado para este exemplo é com/example/www/process87787141/MyService20030101T020304\_ProcessPortTypePT.

No WebSphere Studio Application Developer Integration Edition, quando a ligação do IBM Web Service (SOAP/JMS) foi selecionada como o tipo de implementação para um processo BPEL ou outro tipo de serviço, foram especificadas as seguintes opções:

- Para Estilo de Documento, o padrão era **DOCUMENT** / **outra opção: RPC**
- Para Utilização do Documento, o padrão era **LITERAL** / **outra opção: ENCODED**
- Para URL do Provedor JNDI, ela era **Fornecida pelo Servidor** ou **Customizada** (deve ser digitado um endereço, o padrão é `iiop://localhost:2809`)
- Para Estilo de Destino, o padrão era **queue** / **outra opção era topic**
- Para Connection Factory JNDI, o padrão era **jms/qcf** (este é o nome JNDI da connection factory de fila para a fila do MDB gerada)
- Para Fila de Destino JNDI, o padrão era **jms/queue** (este é o nome JNDI da fila do MDB gerada)
- Para Porta Listener MDB, o padrão era **<Nome do Projeto de Serviço>MdbListenerPort**

Um arquivo WSDL que especifica a ligação e o serviço SOAP/JMS do IBM Web Service é criado no projeto EJB gerado, mas não no próprio projeto de serviço. Isto significa que você deve localizar manualmente esse arquivo e copiá-lo para seu projeto de módulo de integração de negócios, se for importante que o código de cliente IBM Web Service não seja alterado. Por padrão, este arquivo WSDL foi criado no projeto EJB em `ejbModule/META-INF/wsdl/<nome do processo de negócios>_<nome do tipo de porta da interface do processo de negócios>_JMS.wsdl`

O PortType e Mensagens WSDL da interface do processo de negócios são realmente copiados também para este arquivo WSDL, em vez de referir-se ao PortType e Mensagens WSDL existentes no projeto de serviço.

Se for importante que o código de cliente IBM Web Service permaneça inalterado após a migração, as informações neste arquivo serão requeridas para as etapas de migração manuais abaixo.

Existem duas opções para migrar a ligação do processo SOAP/JMS do WebSphere Studio Application Developer Integration Edition. A opção terá que ser feita, se migrar o cliente para o modelo de programação SCA ou se deixá-lo como um cliente de serviços da Web:

**Nota:** Após a conclusão das etapas de migração manuais, o cliente também deverá ser migrado para o novo modelo de programação. Consulte o tópico apropriado para os seguintes tipos de clientes:

*Tabela 7. Informações Adicionais para Migrar Clientes*

Tipo de Cliente	Para obter informações adicionais, consulte
Cliente IBM Web Service	Migrando o Cliente IBM Web Service (SOAP/JMS)

*Opção de Migração 1 para a Ligação do IBM Web Service (SOAP/JMS):*

A primeira opção de migração para a ligação SOAP/JMS do WebSphere Studio Application Developer Integration Edition é tornar o serviço acessível para um cliente de serviços da Web.

A Exportação com Ligação de Serviço da Web torna um componente SCA acessível por um cliente de serviços da Web externo. Para criar uma Exportação com Ligação de Serviço da Web:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Exportação com Ligação SCA para cada interface de serviço que teve uma ligação do IBM Web Service (SOAP/JMS) gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Clique com o botão direito do mouse no componente SCA no Editor de Montagem.
  - b. Selecione **Exportar...**
  - c. Selecione **Ligação de Serviço da Web**.
  - d. Se houver várias interfaces para o componente, selecione as interfaces a serem exportadas com este tipo de ligação.
  - e. Selecione o transporte **soap/jms**.
3. Quando a Exportação de Serviços da Web for criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
4. Salve o diagrama de montagem.
5. Selecione a área de janela de conteúdo Ligação e verá que uma Ligação e o Serviço de Ligação WSDL do IBM Web Service foram gerados diretamente na pasta do projeto do módulo. Ele será denominado *componente que foi exportado* Exportar Nome de PortType WSDL Jms\_Service.wsdl. Se você inspecionar este arquivo, perceberá que a ligação agrupada de Document/Literal é utilizada por padrão, pois é o estilo preferido na 6.0. Este é o WSDL que os clientes IBM Web Service utilizarão para chamar o serviço.
6. Siga estas etapas para gerar um novo serviço e uma nova ligação de serviço da Web, se a preservação do código do cliente for desejada:
  - a. Copie o arquivo WSDL da 5.1 do projeto EJB gerado da 5.1 em `ejbModule/META-INF/wsdl/nome do processo de negócios/nome do tipo de porta da interface do processo de negóciosJMS.wsdl` para o projeto de módulo do Business Integration.
  - b. Depois de copiar sobre o arquivo e reconstruir o módulo, você poderá ver mensagens de erro porque os tipos de esquema XML, as mensagens do WSDL e os tipos de porta do WSDL utilizados pelo serviço da Web estão duplicados no arquivo WSDL do IBM Web Service na 5.1. Para corrigir isso, exclua essas definições duplicadas do WSDL de ligação/serviço do IBM Web Service e em seu lugar inclua uma importação do WSDL para o WSDL da interface real. **Nota:** É importante observar que quando o WebSphere Studio Application Developer Integration Edition gerou o código de implementação do IBM Web Service, ele modificou as definições de esquema em alguns casos. Isso poderá causar inconsistências para clientes existentes que utilizam o WSDL do IBM Web Service. Por exemplo, o atributo de esquema "elementFormDefault" foi definido para "qualified" no esquema sequencial gerado no WSDL do IBM Web Service mesmo que a definição de esquema original não tenha sido qualificada. Isso causará a geração do seguinte erro durante o tempo de execução: WSW3047E: Erro: Não é possível desserializar o elemento.

- c. Clique com o botão direito do mouse neste arquivo WSDL recém-copiado para o módulo do Business Integration e selecione **Abrir com e**, em seguida, **Editor WSDL**.
- d. Vá para a guia Origem. Exclua todos os PortTypes e Mensagens WSDL definidos neste arquivo.
- e. Agora você verá o erro: 0 tipo de porta '<Tipo\_de\_porta>' especificado para a ligação '<ligação>' está indefinido. Para corrigir isso, no editor WSDL da guia Gráfico, clique com o botão direito do mouse na seção Importações e selecione **Incluir Importação**.
- f. Na visualização Propriedades na guia Geral, clique no botão ... à direita do campo Local. Procure o WSDL da interface no qual as definições de mensagem e de tipo de porta do WSDL estão localizadas e clique em **OK** para importar o WSDL da interface para o WSDL de serviço/ligação.
- g. Salve o arquivo WSDL.
- h. Atualize/reconstrua o projeto. Vá para a perspectiva Business Integration. Abra o Diagrama de Montagem do módulo no Editor de Montagem.
- i. Na visualização Explorador de Projeto, expanda o módulo que você está migrando e expanda a categoria lógica **Portas de Serviço da Web**. Você deve ver a porta existente no WSDL de ligação/serviço listado. Arraste-a e solte-a sobre o Editor de Montagem.
- j. Escolha a criação de uma **Exportação com Ligação de Serviço da Web** e selecione o nome de porta adequado. Isso criará a Exportação que utiliza a ligação/serviço antigo de forma que clientes de serviço da Web existentes não tenham de fazer alterações. Se você selecionar a exportação que acabou de criar no Editor de Montagem e for para a visualização Propriedades, na guia Ligação deverá ver que os nomes de porta e de serviço da 5.1 foram preenchidos.
- k. Salve todas as alterações.
- l. Pouco antes de implementar o aplicativo, você pode alterar a configuração do projeto da Web gerado para que corresponda ao endereço de serviço da 5.1 (você precisa fazer essas alterações toda vez que fizer alterações no módulo SCA, o que faz com que esse arquivo seja gerado novamente). Se você examinar a definição de *Serviço* do IBM Web Service WSDL que está sendo reutilizada da 5.1, verá o endereço de serviço para o qual o cliente 5.1 foi codificado: `<wsdl:soap:address location="http://localhost:9080/MyServiceWeb/services/MyServicePort"/>`
- m. Para fazer com que os artefatos do projeto da Web gerado da 6.0 correspondam a esse endereço de serviço antigo, você deve modificar o descritor de implementação do projeto da Web gerado. Abra o descritor de implementação no WebSphere Integration Developer e na guia Servlets, inclua um Mapeamento de URL adicional que é muito similar ao mapeamento de URL existente para aquela exportação, com o mesmo nome de servlet, mas um padrão de URL diferente.
- n. Além disso, se precisar modificar a raiz de contexto desse projeto da Web de forma que ela corresponda à raiz de contexto no endereço de serviço original (nesse exemplo, a raiz de contexto é "MyServiceWeb"), então, você poderá abrir o descritor de implementação para o Aplicativo Corporativo J2EE no qual está esse projeto da Web e alterar a raiz de contexto desse módulo da Web para que corresponda à raiz de contexto do endereço de serviço antigo. Você poderá ver o seguinte erro que pode ser ignorado: CHKJ3017E: Projeto da Web: <NOME DO PROJ DA WEB> está mapeado para uma raiz de Contexto inválida: <RAIZ DE CONTEXTO NOVO> no Projeto EAR: <NOME DO APLICATIVO>.

#### *Opção de Migração 2 para a Ligação do IBM Web Service (SOAP/JMS):*

A segunda opção para a ligação do processo SOAP/JMS do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis a uma entidade não-SCA (por exemplo, um cliente JSP ou Java).

A Referência Independente torna um componente SCA acessível por qualquer cliente externo. Para criar uma Referência Independente:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Referência Independente para cada interface de processo BPEL que teve uma ligação IBM Web Service (SOAP/JMS) gerada para ela no WebSphere Studio Application Developer Integration Edition:



- a. Selecione o item **Referências Independentes** na barra de ferramentas.
- b. Clique no canvas do Editor de Montagem para criar uma entidade SCA de Referências Independentes.
- c. Selecione o item **Ligação** na barra de ferramentas.
- d. Clique na entidade **Referências Independentes** para selecioná-la como a origem da ligação.
- e. Clique no componente **BPEL SCA** para selecioná-lo como o destino da ligação.
- f. Você verá um alerta **A referência correspondente será criada no nó de origem. Deseja continuar?**, clique em **OK**.
- g. Selecione a entidade **Referências Independentes** recém-criada na visualização Propriedades e selecione a área de janela de conteúdo **Descrição**.
- h. Expanda o link **Referências** e selecione a referência recém-criada. O nome e a descrição da referência são listados e podem ser modificados se necessário.
- i. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
- j. Salve o diagrama de montagem.

#### *Migrando a Ligação do IBM Web Service (SOAP/HTTP):*

A ligação do IBM Web Service (SOAP/HTTP) para um processo BPEL ou outro tipo de serviço pode ser migrada para a construção SCA recomendada.

No WebSphere Studio Application Developer Integration Edition, este tipo de ligação permitia que os clientes se comunicassem com um processo BPEL ou outro tipo de serviço, chamando um IBM Web Service, no qual o protocolo de comunicação era HTTP e a mensagem seguia as regras de codificação SOAP.

A seguir está um exemplo das convenções utilizadas durante a geração de um IBM Web Service (SOAP/HTTP) para um serviço BPEL 5.1. O nome JNDI do IBM Web Service gerado era uma combinação do nome, espaço de nomes de destino e time stamp valid-from do BPEL, bem como o nome da interface (tipo de porta WSDL para o qual o código de implementação foi gerado). Por exemplo, estes atributos podem ser localizados examinando as propriedades do processo BPEL no editor BPEL nas guias de conteúdo Descrição e Servidor:

*Tabela 8. Espaço de Nomes Gerado*

Nome do Processo	MyService
Espaço de Nomes de Destino	http://www.example.com/process87787141/
Válido a partir de	01 de Jan de 2003 02:03:04
Interface	ProcessPortType

O espaço de nomes gerado para este exemplo é com/example/www/process87787141/MyService20030101T020304\_ProcessPortTypePT.

No WebSphere Studio Application Developer Integration Edition, quando a ligação do IBM Web Service (SOAP/HTTP) foi selecionada como o tipo de implementação para um processo BPEL ou outro tipo de serviço, foram especificadas as seguintes opções:

- Para Estilo de Documento, o padrão era **RPC** / **outra opção: DOCUMENT**
- Para Utilização do Documento, o padrão era **ENCODED** / **outra opção: LITERAL**
- Para Endereço do Roteador, o padrão era http://localhost:9080

Um arquivo WSDL que especifica a ligação e o serviço SOAP/HTTP IBM Web Service é criado nos projetos da Web e EJB gerados, mas não no próprio projeto de serviço. Isto significa que você deve localizar manualmente esse arquivo e copiá-lo para seu projeto de módulo de integração de negócios, se

for importante que o código de cliente IBM Web Service não seja alterado. Por padrão, este arquivo WSDL foi criado no projeto Web em WebContent/WEB-INF/wsdl/<nome do processo de negócios>\_<nome do tipo de porta da interface do processo de negócios>\_HTTP.wsdl

O PortType e Mensagens WSDL da interface do processo de negócios são realmente copiados também para este arquivo WSDL, em vez de referir-se ao PortType e Mensagens WSDL existentes no projeto de serviço.

Se for importante que o código de cliente IBM Web Service permaneça inalterado após a migração, as informações neste arquivo serão requeridas para as etapas de migração manuais abaixo.

Existem duas opções para migrar a ligação do processo SOAP/HTTP do WebSphere Studio Application Developer Integration Edition. A opção terá que ser feita, se migrar o cliente para o modelo de programação SCA ou se deixá-lo como um cliente de serviços da Web:

**Nota:** Após a conclusão das etapas de migração manuais, o cliente também deverá ser migrado para o novo modelo de programação. Consulte o tópico apropriado para os seguintes tipos de clientes:

*Tabela 9. Informações Adicionais para Migrar Clientes*

Tipo de Cliente	Para obter informações adicionais, consulte
Cliente IBM Web Service	Migrando o Cliente IBM Web Service (SOAP/HTTP)

*Opção de Migração 1 para a Ligação do IBM Web Service (SOAP/HTTP):*

A primeira opção de migração para a ligação do processo SOAP/HTTP do WebSphere Studio Application Developer Integration Edition é tornar os processos de negócios acessíveis a um cliente de serviços da Web.

A Exportação com Ligação de Serviço da Web torna um componente SCA acessível por um cliente de serviços da Web externo. Para criar uma Exportação com Ligação de Serviço da Web:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de Migração.
2. Crie uma Exportação com Ligação SCA para cada interface de processo BPEL que teve uma ligação IBM Web Service (SOAP/HTTP) gerada para ela no WebSphere Studio Application Developer Integration Edition, clicando com o botão direito do mouse no componente BPEL no Editor de Montagem.
3. Selecione **Exportar...**
4. Selecione **Ligação de Serviço da Web**.
5. Se houver várias interfaces para o componente, selecione as interfaces a serem exportadas com este tipo de ligação.
6. Selecione o transporte **soap/http**.
7. Quando a Exportação de Serviços da Web for criada, selecione a exportação no Editor de Montagem e, na visualização Propriedades, selecione a área de janela de conteúdo **Descrição**. O nome e a descrição da Exportação são listados e podem ser modificados se necessário.
8. Salve o diagrama de montagem.
9. Siga estas etapas para gerar um novo serviço e uma nova ligação de serviço da Web, se a preservação do código do cliente for desejada:
  - a. Copie o arquivo WSDL 5.1 do projeto EJB gerado da 5.1 em ejbModule/META-INF/wsdl/nome do processo de negócios/nome do tipo de porta da interface do processo de negócios\_HTTP.wsdl para o projeto de módulo do Business Integration.
  - b. Depois de copiar sobre o arquivo e reconstruir o módulo, você poderá ver mensagens de erro porque os tipos de esquema XML, as mensagens do WSDL e os tipos de porta do WSDL utilizados pelo serviço da Web estão duplicados no arquivo WSDL do IBM Web Service na 5.1.

Para corrigir isso, exclua essas definições duplicadas do WSDL de ligação/serviço do IBM Web Service e em seu lugar inclua uma importação do WSDL para o WSDL da interface real. **Nota:** É importante observar que quando o WebSphere Studio Application Developer Integration Edition gerou o código de implementação do IBM Web Service, ele modificou as definições de esquema em alguns casos. Isso poderá causar inconsistências para clientes existentes que utilizam o WSDL do IBM Web Service. Por exemplo, o atributo de esquema "elementFormDefault" foi definido para "qualified" no esquema seqüencial gerado no WSDL do IBM Web Service mesmo que a definição de esquema original não tenha sido qualificada. Isso causará a geração do seguinte erro durante o tempo de execução: WWS3047E: Erro: Não é possível desserializar o elemento.

- c. Clique com o botão direito do mouse neste arquivo WSDL recém-copiado para o módulo do Business Integration e selecione **Abrir com** e, em seguida, **Editor WSDL**.
- d. Vá para a guia Origem. Exclua todos os PortTypes e Mensagens WSDL definidos neste arquivo.
- e. Agora você verá o erro: O tipo de porta '<Tipo\_de\_porta>' especificado para a ligação '<ligação>' está indefinido. Para corrigir isso, no editor WSDL da guia Gráfico, clique com o botão direito do mouse na seção Importações e selecione **Incluir Importação**.
- f. Na visualização Propriedades na guia Geral, clique no botão ... à direita do campo Local. Procure o WSDL da interface no qual as definições de mensagem e de tipo de porta do WSDL estão localizadas e clique em **OK** para importar o WSDL da interface para o WSDL de serviço/ligação.
- g. Salve o arquivo WSDL.
- h. Atualize/reconstrua o projeto. Vá para a perspectiva Business Integration. Abra o Diagrama de Montagem do módulo no Editor de Montagem.
- i. Na visualização Explorador de Projeto, expanda o módulo que você está migrando e expanda a categoria lógica **Portas de Serviço da Web**. Você deve ver a porta existente no WSDL de ligação/serviço listado. Arraste-a e solte-a sobre o Editor de Montagem.
- j. Escolha a criação de uma **Exportação com Ligação de Serviço da Web** e selecione o nome de porta adequado. Isso criará a Exportação que utiliza a ligação/serviço antigo de forma que clientes de serviço da Web existentes não tenham de fazer alterações. Se você selecionar a exportação que acabou de criar no Editor de Montagem e for para a visualização Propriedades, na guia Ligação deverá ver que os nomes de porta e de serviço da 5.1 foram preenchidos.
- k. Salve todas as alterações.
- l. Pouco antes de implementar o aplicativo, você pode alterar a configuração do projeto da Web gerado para que corresponda ao endereço de serviço da 5.1 (você precisa fazer essas alterações toda vez que fizer alterações no módulo SCA, o que faz com que esse arquivo seja regenerado). Se você examinar a definição de serviço do IBM Web Service WSDL que está sendo reutilizada da 5.1, verá o endereço de serviço para o qual o cliente 5.1 foi codificado: <wsdl:soap:address location="http://localhost:9080/MyServiceWeb/services/MyServicePort"/>
- m. Para fazer com que os artefatos do projeto da Web gerado da 6.0 correspondam a esse endereço de serviço antigo, você deve modificar o descritor de implementação do projeto da Web gerado. Abra o descritor de implementação no WebSphere Integration Developer e na guia Servlets, inclua um Mapeamento de URL adicional que é muito similar ao mapeamento de URL existente para aquela exportação, com o mesmo nome de servlet, mas um padrão de URL diferente.
- n. Além disso, se precisar modificar a raiz de contexto desse projeto da Web de forma que ela corresponda à raiz de contexto no endereço de serviço original (nesse exemplo, a raiz de contexto é "MyServiceWeb"), então, você poderá abrir o descritor de implementação para o Aplicativo Corporativo J2EE no qual está esse projeto da Web e alterar a raiz de contexto desse módulo da Web para que corresponda à raiz de contexto do endereço de serviço antigo. Você poderá ver o seguinte erro que pode ser ignorado: CHKJ3017E: Projeto da Web: <NOME DO PROJ DA WEB> está mapeado para uma raiz de Contexto inválida: <RAIZ DE CONTEXTO NOVO> no Projeto EAR: <NOME DO APLICATIVO>.

*Opção de Migração 2 para a Ligação do IBM Web Service (SOAP/HTTP):*

A segunda opção para a ligação do processo SOAP/HTTP do WebSphere Studio Application Developer Integration Edition é tornar processos de negócios acessíveis a uma entidade não-SCA (por exemplo, um cliente JSP ou Java).

A Referência Independente torna um componente SCA acessível por qualquer cliente externo. Para criar uma Referência Independente:

1. Abra o Editor de Montagem para o módulo criado pelo assistente de migração.
2. Crie uma Referência Independente para cada interface que teve uma ligação IBM Web Service (SOAP/HTTP) gerada para ela no WebSphere Studio Application Developer Integration Edition:
  - a. Selecione o item **Referências Independentes** na barra de ferramentas.
  - b. Clique no canvas do Editor de Montagem para criar uma entidade SCA de Referências Independentes.
  - c. Selecione o item **Ligação** na barra de ferramentas.
  - d. Clique na entidade **Referências Independentes** para selecioná-la como a origem da ligação.
  - e. Clique no componente **SCA** para selecioná-lo como o destino da ligação.
  - f. Você verá um alerta **A referência correspondente será criada no nó de origem. Deseja continuar?**, clique em **OK**.
  - g. Selecione a entidade **Referências Independentes** recém-criada na visualização Propriedades e selecione a área de janela de conteúdo **Descrição**.
  - h. Expanda o link **Referências** e selecione a referência recém-criada. O nome e a descrição da referência são listados e podem ser modificados se necessário.
  - i. Se houver várias interfaces para o processo, selecione as interfaces a serem exportadas com este tipo de ligação.
  - j. Salve o diagrama de montagem.

*Migrando a Ligação do Apache Web Service (SOAP/HTTP):*

A ligação do Apache Web Service (SOAP/HTTP) para um processo BPEL ou outro tipo de serviço pode ser migrada para a construção SCA recomendada.

No WebSphere Studio Application Developer Integration Edition, este tipo de ligação permite que os clientes se comuniquem com um processo BPEL ou outro tipo de serviço, chamando um Apache Web Service.

No WebSphere Studio Application Developer Integration Edition, quando a ligação do Apache Web Service foi selecionada como o tipo de implementação para um processo BPEL ou outro tipo de serviço, foram especificadas as seguintes opções:

- Para Estilo de Documento, era **RPC** (nenhuma outra opção disponível)
- Para ação do SOAP, era **URN:WSDL PortType name**
- Para Endereço, era **http://localhost:9080/Service Project NameWeb/servlet/rpcrouter**
- Para Utilizar Codificação, o padrão era **yes** (Se **yes**, o Estilo de Codificação era configurado como: **http://schemas.xmlsoap.org/soap/encoding/**)

Um arquivo WSDL que especifica a ligação do Apache SOAP e o serviço é criado no projeto de serviço. Por padrão, ele é criado no mesmo diretório que o serviço que está agrupando com o nome *<nome do processo de negócios>\_<nome do tipo de porta de interface do processo de negócios>\_SOAP.wsdl*. O PortType e Mensagens WSDL da interface do processo de negócios são utilizados por esta ligação e serviço diretamente. Após a migração, você *não* deve utilizar este WSDL para nada além de talvez utilizar o mesmo espaço de nomes, porta e nomes de serviços no novo WSDL que será gerado na Versão 6.

Existem duas opções para migrar a ligação do processo do WebSphere Studio Application Developer Integration Edition Web Service. A opção terá que ser feita, se migrar o cliente para o modelo de

programação SCA ou se deixá-lo como um modelo de programação IBM Web Services. Não existe mais nenhuma ligação que seja equivalente ao tipo de ligação do Apache Web Service (SOAP/HTTP) no modelo de programação SCA da 6.

Você deve migrar esse Apache Web Service para utilizar o mecanismo do IBM Web Service. Consulte o tópico "Migrando a Ligação do IBM Web Service (SOAP/HTTP)" para obter instruções sobre como executar essa migração e criar um IBM Web Service (SOAP/HTTP).

### **Migrando para o Modelo de Programação SCA:**

Para qualquer código Java de formato livre que interaja com um serviço do WebSphere Studio Application Developer Integration Edition, esta seção mostrará como migrar do modelo de programação do WSIF para o novo modelo de programação SCA no qual os dados que fluem pelo aplicativo são armazenados nos SDOs (Service Data Objects) do Eclipse. Esta seção também mostrará como migrar manualmente os tipos de cliente mais comuns para o novo modelo de programação.

Para qualquer processo BPEL que contenha snippets Java, esta seção explica como migrar da API de snippet Java antiga para a nova API de snippet Java na qual os dados que fluem pelo aplicativo são armazenados nos SDOs (Service Data Objects) do Eclipse. Sempre que possível, os snippets são migrados automaticamente pelo assistente de migração, mas existem snippets que o assistente de migração não pode migrar totalmente, indicando que são requeridas etapas manuais para concluir a migração.

A seguir há um resumo das alterações no modelo de programação:

#### **Modelo de Programação da V5.1**

1. Baseado no WSIF e no WSDL
2. Proxies gerados para serviços
3. Rotinas de tratamento de beans e formatos para tipos

#### **Modelo de Programação da V6.0 (Mais Centralizado em Java)**

1. Serviços SCA baseados em SDOs com tags doclet
2. Ligações de interface para serviços
3. SDOs e Databindings para tipos

#### *Migrando Chamadas de API WSIFMessage para APIs SDO:*

A seção a seguir detalha como migrar do modelo de programação antigo do WebSphere Business Integration Server Foundation Versão 5.1 no qual os dados que fluem por meio do aplicativo são representados como objetos WSIFMessage com uma interface gerada que foi altamente especificada para o novo modelo de programação do WebSphere Process Server Versão 6.0 no qual os dados são representados como SDOs (Service Data Objects) e não é gerada nenhuma interface altamente especificada.



*Tabela 10. Alterações e Soluções para Migrar Chamadas da API WSIFMessage para APIs do SDO*

Alterar	Solução
As classes do wrapper baseadas no WSIFMessage não são mais geradas para tipos de mensagens do WSDL, nem são as classes do assistente do Java bean geradas para os tipos de esquema complexos.	<p>Ao escrever código que interage com serviços do SCA, as APIs genéricas do SDO devem ser utilizadas para manipular as mensagens <code>commonj.sdo.DataObject</code> que mantêm os dados que fluem através do aplicativo.</p> <p>As definições de mensagem WSDL que têm uma única parte digitada simples agora serão representadas por um tipo Java simples que representa diretamente a parte em vez de ter um wrapper em torno dos dados reais. Se a única parte da mensagem for um tipo complexo, então, os dados serão representados como um <code>DataObject</code> que adere à definição do tipo complexo.</p> <p>As definições de mensagens do WSDL que têm várias partes agora correspondem a um <code>DataObject</code> que tem propriedades para todas as partes da mensagem, em que <code>complexType</code>s são representados como propriedades 'reference-type' do <code>DataObject</code> pai, acessíveis através dos métodos <code>getDataObject</code> e <code>setDataObject</code>.</p>
Os métodos getter altamente especificados para partes WSIFMessage e os Java gerados não devem ser utilizados.	A API SDO pouco especificada deve ser utilizada para obter as propriedades de <code>DataObject</code> .
Os métodos setter altamente especificados para partes de mensagem das variáveis BPEL não estão mais disponíveis.	A API SDO pouco especificada deve ser utilizada para definir as propriedades de <code>DataObject</code> .
Os métodos getter pouco especificados para as propriedades de WSIFMessage não devem mais ser utilizados.	A API SDO pouco especificada deve ser utilizada para definir as propriedades de <code>DataObject</code> .
Os métodos setter pouco especificados para as propriedades de WSIFMessage não devem mais ser utilizados.	A API SDO pouco especificada deve ser utilizada para definir as propriedades de <code>DataObject</code> .
Todas as chamadas da API WSIFMessage devem ser migradas para a API SDO, onde possível.	Migre a chamada para uma chamada de API SDO equivalente, onde possível. Se não for possível, projete novamente a lógica.

#### *Migrando o Código de Cliente do WebSphere Business Integration Server Foundation:*

Esta seção mostra como migrar os vários tipos de clientes que eram possíveis para os tipos de serviço do WebSphere Business Integration Server Foundation 5.1.

#### *Migrando o Cliente EJB:*

Este tópico mostra como migrar clientes que utilizam uma interface EJB para chamar um serviço.

1. Arraste e solte a Exportação com Ligação SCA do módulo migrado para o Editor de Montagem deste novo módulo. Isto criará uma Importação com Ligação SCA. Para que um cliente obtenha uma referência a esta importação, deve ser criada uma Referência Independente.
2. Na paleta, selecione o item Referência Independente. Clique no canvas do Editor de Montagem uma vez, para criar uma nova referência independente para este novo módulo.
3. Selecione a ferramenta de ligação e clique na referência de serviço e, em seguida, clique em **Importar**.
4. Clique em **OK** quando receber um alerta de que será criada uma referência correspondente no nó de origem.

5. Será perguntado: **É mais fácil para um cliente Java utilizar uma interface Java com esta referência – deseja converter a referência WSDL em uma referência Java compatível?:**
  - a. Responda **Sim** se desejar que o cliente examine esse serviço e lance-o como uma classe Java para chamá-lo utilizando uma interface Java. Esta nova interface Java utiliza o nome do PortType WSDL, no qual o pacote da interface é derivado do espaço de nomes do PortType WSDL. Existe um método definido para cada operação definida no PortType WSDL, e cada parte da mensagem WSDL é representada como um argumento para os métodos de interface.
  - b. Responda **Não** se desejar que o cliente examine esse serviço e utilize a interface genérica `com.ibm.websphere.sca.Service` para chamá-lo utilizando a operação de chamada como um serviço SCA genérico.
6. Renomeie a referência independente para um nome mais significativo, se apropriado, selecionado o componente Referências Independentes no Editor de Montagem. Vá para a visualização Propriedades, para a guia Detalhes, fazendo uma pesquisa detalhada e selecionando a referência recém-criada e modificando o nome. Lembre-se do nome escolhido para essa referência porque o cliente precisará utilizar esse nome ao chamar o método `locateService` da instância `com.ibm.websphere.sca.ServiceManager`.
7. Clique em **Salvar** para salvar o diagrama de Montagem.

O cliente deve ter este novo módulo em seu caminho de classe local para acessar o módulo EJB migrado em execução no servidor.

A seguir é mostrada a aparência do código do cliente para um serviço do tipo "CustomerInfo":

```
// Crie um novo ServiceManager
ServiceManager serviceManager = ServiceManager.INSTANCE;
// Localize o serviço CustomerInfo
CustomerInfo customerInfoService = (CustomerInfo) serviceManager.locateService
("<nome-da-referência-independente-da-etapa-anterior");
// Chame o serviço CustomerInfo
System.out.println(" [getMyValue] getting customer info...");
DataObject customer = customerInfoService.getCustomerInfo(customerID);
```

O cliente deve alterar a forma que a mensagem é construída. As mensagens eram baseadas na classe `WSIFMessage`, mas agora elas devem ser baseadas na classe `commonj.sdo.DataObject`.

#### *Migrando o Cliente de Ligação do Processo EJB:*

Este tópico mostra como migrar clientes que utilizam a ligação de processo EJB de WSIF para acessar um serviço BPEL.

Clientes que utilizaram a Ligação do Processo EJB para chamar um processo de negócios devem agora utilizar a API SCA para chamar o serviço (o processo de negócios migrado deve ter uma Exportação com Ligação SCA) ou a API do IBM

Web Service Client para chamar o serviço (o processo de negócios migrado deve ter uma Exportação com Ligação de Serviço da Web).

Consulte os tópicos "Migrando Cliente EJB", "Cliente IBM Web Service (SOAP/JMS)" ou "Cliente IBM Web Service (SOAP/HTTP)" para obter informações adicionais sobre a geração de tais clientes.

#### *Migrando o Cliente IBM Web Service (SOAP/JMS):*

Este tópico mostra como migrar clientes que utilizam APIs do Web Service (SOAP/JMS) para chamar um serviço.

Nenhuma migração é necessária para clientes existentes durante a migração. Observe que você deve modificar manualmente o projeto da Web gerado (criar um novo mapeamento de servlet) e, às vezes,

modificar a raiz de contexto do projeto da Web no descritor de implementação do aplicativo corporativo para publicar o serviço para o mesmo endereço exato que foi publicado no WebSphere

Business Integration Server Foundation. Consulte o tópico "Migrando a Ligação IBM

Web Service (SOAP/JMS)".

É importante observar que ao contrário da 5.1, na qual um proxy de cliente do WSIF ou RPC pode ser gerado, na 6.0 as ferramentas suportam apenas a geração de cliente RPC, porque o RPC é a API preferida da 6.0 em relação a API do WSIF.

**Nota:** Para gerar um novo proxy de cliente a partir do WebSphere Integration Developer, é necessário ter um WebSphere Process Server ou um WebSphere Application Server instalado.

1. Certifique-se de que tenha um WebSphere Process Server ou um WebSphere Application Server instalado.
2. Na perspectiva Recursos ou Java, localize o arquivo do WSDL correspondente à **Exportação com Ligação de Serviço da Web** e, em seguida, clique com o botão direito do mouse e selecione **Serviços da Web → Gerar Cliente** (Observe que esse assistente é muito similar ao assistente da 5.1).
3. Para o Tipo de Proxy de Cliente, escolha **Proxy Java** e clique em **Avançar**.
4. O local do WSDL deve ser preenchido. Clique em **Avançar**.
5. Em seguida, você deve selecionar as opções adequadas para especificar a configuração do ambiente do cliente incluindo o tempo de execução e o servidor do serviço da Web, a versão do J2EE e o tipo de cliente (Java, EJB, Web, Application Client). Clique em **Avançar**.
6. Finalize as etapas restantes para gerar o proxy de cliente.

*Migrando o Cliente IBM Web Service (SOAP/HTTP):*

Este tópico mostra como migrar clientes que utilizam APIs do Web Service (SOAP/HTTP) para chamar um serviço.

Nenhuma migração é necessária para clientes existentes durante a migração. Observe que você deve modificar manualmente o projeto da Web gerado (criar um novo mapeamento de servlet) e, às vezes, modificar a raiz de contexto do projeto da Web no descritor de implementação do aplicativo corporativo para publicar o serviço para o mesmo endereço exato que foi publicado no WebSphere

Business Integration Server Foundation. Consulte o tópico "Migrando a Ligação IBM

Web Service (SOAP/HTTP)".

Se tiverem ocorrido alterações de design e você desejar gerar um novo proxy de cliente, as seguintes etapas mostrarão como fazer isso. É importante observar que ao contrário da 5.1, na qual um proxy de cliente do WSIF ou RPC pode ser gerado, na 6.0 as ferramentas suportam apenas a geração de cliente RPC, porque o RPC é a API preferida da 6.0 em relação a API do WSIF.

**Nota:** Para gerar um novo proxy de cliente a partir do WebSphere Integration Developer, é necessário ter um WebSphere Process Server ou um WebSphere Application Server instalado.

1. Certifique-se de que tenha um WebSphere Process Server ou um WebSphere Application Server instalado.
2. Selecione o arquivo do WSDL correspondente à **Exportação com Ligação de Serviço da Web** e, em seguida, clique com o botão direito do mouse e selecione **Serviços da Web → Gerar Cliente** (Observe que esse assistente é muito similar ao assistente da 5.1).
3. Para o Tipo de Proxy de Cliente, escolha **Proxy Java** e clique em **Avançar**.
4. O local do WSDL deve ser preenchido. Clique em **Avançar**.



5. Em seguida, você deve selecionar as opções adequadas para especificar a configuração do ambiente do cliente incluindo o tempo de execução e o servidor do serviço da Web, a versão do J2EE e o tipo de cliente (Java, EJB, Web, Application Client). Clique em **Avançar**.
6. Finalize as etapas restantes para gerar o proxy de cliente.

*Migrando o Cliente Apache Web Service (SOAP/HTTP):*

As APIs do cliente Apache Web Service não são apropriadas para chamar um serviço do WebSphere Integration Developer. O código do cliente deve ser migrado para utilizar as APIs do cliente IBM Web Service (SOAP/HTTP).

Consulte a seção " IBM

Web Service (SOAP/HTTP) Client" para obter informações adicionais.

Na 5.1, se um proxy de cliente fosse automaticamente gerado, esse proxy utilizaria APIs do WSIF para interagir com o serviço. Na 6.0, as ferramentas suportam apenas a geração de cliente RPC porque o RPC é a API preferida da 6.0 em relação a API do WSIF.

**Nota:** Para gerar um novo proxy de cliente a partir do WebSphere Integration Developer, é necessário ter um WebSphere Process Server ou um WebSphere Application Server instalado.

1. Certifique-se de que tenha um WebSphere Process Server ou um WebSphere Application Server instalado.
2. Selecione o arquivo do WSDL correspondente à **Exportação com Ligação de Serviço da Web** e, em seguida, clique com o botão direito do mouse e selecione **Serviços da Web → Gerar Cliente** (Observe que esse assistente é muito similar ao assistente da 5.1).
3. Para o Tipo de Proxy de Cliente, escolha **Proxy Java** e clique em **Avançar**.
4. O local do WSDL deve ser preenchido. Clique em **Avançar**.
5. Em seguida, você deve selecionar as opções adequadas para especificar a configuração do ambiente do cliente incluindo o tempo de execução e o servidor do serviço da Web, a versão do J2EE e o tipo de cliente (Java, EJB, Web, Application Client). Clique em **Avançar**.
6. Finalize as etapas restantes para gerar o proxy de cliente.

*Migrando o Cliente JMS:*

Os clientes que se comunicaram com um serviço 5.1 por meio da API JMS (enviando uma mensagem JMS para uma fila) podem requerer migração manual. Este tópico mostra como migrar clientes que utilizam APIs JMS (enviando uma mensagem JMS para uma fila) para chamar um serviço.

É necessário assegurar que a **Exportação com Ligação JMS** criada em uma etapa anterior possa aceitar esta mensagem de texto ou de objeto sem alterações. Pode ser necessário gravar uma ligação de dados customizada para fazer isso. Consulte a seção "Migrando o JMS e as Ligações de Processo JMS" para obter informações adicionais.

O cliente deve alterar a forma que a mensagem é construída. Anteriormente, as mensagens eram baseadas na classe WSIFMessage mas agora elas devem ser baseadas na classe commonj.sdo.DataObject. Consulte a seção "Migrando as Chamadas da API WSIFMessage para APIS do SDO" para obter detalhes adicionais como fazer essa migração manual.

*Migrando o Cliente da API EJB Genérica do Business Process Choreographer:*

Este tópico mostra como migrar clientes que utilizam a API de EJB genérico do process choreographer 5.1 para chamar um serviço BPEL.

Existe uma nova versão da API de EJB Genérica que utiliza DataObjects como seu formato de mensagem. O cliente deve alterar a forma que a mensagem é construída. Anteriormente, as mensagens eram baseadas na classe WSIFMessage mas agora elas devem ser baseadas na classe `commonj.sdo.DataObject`. Observe que a API de EJB Genérica não foi alterada significativamente, pois o `ClientObjectWrapper` ainda fornece um wrapper de mensagem no formato de mensagem específico.

```
Ex: DataObject dobj = myClientObjectWrapper.getObject();  
String result = dobj.getInt("resultInt");
```

O nome JNDI do EJB Genérico antigo que obtém o objeto `WSIFMessage` é:

```
GenericProcessChoreographerEJB  
Nome JNDI: com/ibm/bpe/api/BusinessProcessHome  
Interface: com.ibm.bpe.api.BusinessProcess
```

Existem dois EJBs genéricos na 6.0 porque as operações de tarefa humana agora estão disponíveis como um EJB separado. Os nomes JNDI da 6.0 desses EJBs Genéricos são:

```
GenericBusinessFlowManagerEJB  
JNDI Name: com/ibm/bpe/api/BusinessFlowManagerHome  
Interface: com.ibm.bpe.api.BusinessFlowManager  
HumanTaskManagerEJB  
JNDI Name: com/ibm/task/api/TaskManagerHome  
Interface: com.ibm.task.api.TaskManager
```

*Migrando o Cliente da API do Sistema de Mensagens Genérica do Business Process Choreographer e o Cliente de Ligação de Processo JMS:*

Não existe uma API do sistema de mensagens genérica no WebSphere Process Server 6.0. Consulte a seção "Migrando o JMS e Ligações de Processo JMS" para escolher uma maneira diferente de expor o processo de negócios para consumidores e registrar o cliente de acordo com a ligação escolhida.

*Migrando o Cliente da Web do Business Process Choreographer:*

Este tópico mostra como migrar as configurações do cliente Web do Process Choreographer 5.1 e JSPs customizados.

O assistente de Migração preserva as configurações do cliente Web 5.1 e, no Human Task Editor, não é possível editar estas configurações. Você deve criar novas configurações do cliente Web e JSPs utilizando o WebSphere

Integration Developer 6.0.

### **Migrando Modificações do Cliente Web**

Na 5.1, você pode modificar a aparência e comportamento do cliente Web baseado em Struts modificando seu `JSP Header.jsp` e folha de estilo `dwc.css`.

Como o cliente Web 6.0 (renomeado para **Business Process Choreographer Explorer**) é baseado em JSF (Java Server Faces) em vez de Struts, a migração automática de modificações do cliente Web não é possível. Portanto, recomenda-se que você consulte a documentação do "Business Process Choreographer Explorer" para obter detalhes sobre a customização da versão 6.0 desse aplicativo.

JSPs definidos pelo usuário poderão ser definidos para processos de negócios e para atividades de equipe. O cliente Web utiliza essas JSPs para exibir mensagens de entrada e saída para o processo e as atividades.

Essas JSPs são particularmente úteis quando:

1. Mensagens têm partes não primitivas para aprimorar a usabilidade da estrutura de dados da mensagem.
2. Você deseja estender as capacidades do cliente Web.

Existem opções adicionais e diferentes disponíveis ao especificar configurações do cliente Web para um processo 6.0, portanto, você terá de utilizar o WebSphere Integration Developer para reprojeter as configurações do cliente Web para processos e atividades migrados:

1. Selecione o canvas do processo ou uma atividade no processo.
2. Na visualização Propriedades, selecione a guia **Cliente** para reprojeter as configurações do cliente Web.
3. Migre manualmente qualquer JSP definida pelo usuário:
  - a. Consulte a seção "Migrando para o Modelo de Programação SCA" para obter as alterações do modelo de migração.
  - b. O cliente Web utiliza as APIs Genéricas para interagir com processos de negócios. Consulte as seções que mostram como migrar chamadas para essas APIs genéricas.
4. Especifique o nome da nova JSP nas configurações do cliente Web 6.0 para o processo.

**Nota:** O mapeamento de JSPs não é necessário com o Business Process Choreographer Explorer 6.0 porque os DataObjects não precisam de nenhum mapeamento customizado.

#### *Migrando Snippets Java do BPEL do WebSphere Business Integration Server Foundation:*

Para processos BPEL que contenham snippets Java, esta seção detalha como migrar da API de snippet Java antiga para a nova API de snippet Java na qual os dados que fluem pelo aplicativo são armazenados como SDOs (Service Data Objects) do Eclipse.

Consulte a seção "Migrando das Chamadas da API WSIFMessage para APIs do SDO" para obter as etapas de migração para executar ações específicas à transição do WSIFMessage para o SDO.

Sempre que possível, os snippets são migrados automaticamente pelo assistente de migração, mas existem snippets que o assistente de migração não pode migrar totalmente. Isto requer etapas manuais extras para concluir a migração. Consulte o tópico Limitações para obter detalhes sobre os tipos de snippets Java que devem ser migrados manualmente. Sempre que um desses snippets for encontrado, o assistente de Migração explicará por que ele não pode ser migrado automaticamente e emitirá uma mensagem de aviso ou de erro.

A seguinte tabela detalha as alterações no modelo de programação e na API do snippet Java do BPEL do Process Choreographer versão 5.1 para 6.0:

*Tabela 11. Alterações e Soluções para Migrar os Snippets Java do BPEL do WebSphere Business Integration Server Foundation*

Alterar	Solução
<p>As classes do wrapper baseadas no WSIFMessage não são mais geradas para tipos de mensagens do WSDL, nem são as classes do assistente do Java bean geradas para os tipos de esquema complexos.</p>	<p>As variáveis BPEL podem ser acessadas diretamente pelo nome. Observe que, para variáveis BPEL cuja definição de mensagem WSDL possui uma única parte, estas variáveis agora representarão diretamente a parte em vez de ter um wrapper ao redor dos dados reais. As variáveis cujo tipo de mensagem tem várias partes terão um wrapper DataObject em torno das partes (em que o wrapper no WebSphere Application Developer Integration Edition era um WSIFMessage).</p> <p>Como as variáveis BPEL podem ser utilizadas diretamente nos snippets 6.0, há menos necessidade de variáveis locais do que havia na 5.1.</p> <p>Os getters altamente especificados para as variáveis BPEL inicializavam implicitamente o objeto do wrapper WSIFMessage em torno das partes da mensagem. Não há nenhum objeto 'wrapper' para as variáveis BPEL cuja definição de mensagem do WSDL tem apenas uma única parte: nesse caso, as variáveis BPEL representam diretamente a parte (No caso em que a única parte é um tipo simples XSD, a variável BPEL será representada como o tipo de wrapper do objeto Java, como <code>java.lang.String</code>, <code>java.lang.Integer</code>, etc). As variáveis BPEL com definições de mensagens do WSDL de várias partes são manipuladas de forma diferente: ainda há um wrapper em torno das partes e esse wrapper DataObject deve ser explicitamente inicializado no trecho de código Java 6.0 se ainda não tiver sido definido por uma operação anterior.</p> <p>Se qualquer variável local dos snippets da 5.1 tiver o mesmo nome da variável BPEL, poderão ocorrer conflitos; portanto, tente remediar essa situação, se possível.</p>
<p>Os objetos WSIFMessage não são mais utilizados para representar variáveis BPEL.</p>	<p>Se qualquer classe Java customizada chamada a partir dos snippets Java tiver um parâmetro WSIFMessage, ela precisará ser migrada de forma que aceite/retorne um DataObject.</p>
<p>Os métodos getter altamente especificados para variáveis BPEL não estão mais disponíveis.</p>	<p>As variáveis podem ser acessadas diretamente pelo nome. Observe que para as variáveis BPEL, cuja definição de mensagem do WSDL tem uma única parte, agora representarão diretamente a parte em vez de ter um wrapper em torno dos dados reais. As variáveis cujo tipo de mensagem tem várias partes terão um wrapper DataObject em torno das partes (em que o wrapper no WebSphere Application Developer Integration Edition era um WSIFMessage).</p>

Tabela 11. Alterações e Soluções para Migrar os Snippets Java do BPEL do WebSphere Business Integration Server Foundation (continuação)

Alterar	Solução
Os métodos setter altamente especificados para variáveis BPEL não estão mais disponíveis.	As variáveis podem ser acessadas diretamente pelo nome. Observe que, para variáveis BPEL cuja definição de mensagem WSDL possui uma única parte, estas variáveis agora representarão diretamente a parte em vez de ter um wrapper ao redor dos dados reais. As variáveis cujo tipo de mensagem tem várias partes terão um wrapper DataObject em torno das partes (em que o wrapper no WebSphere Application Developer Integration Edition era um WSIFMessage).
Os métodos getter pouco especificados para variáveis BPEL que retornam um WSIFMessage não estão mais disponíveis.	<p>As variáveis podem ser acessadas diretamente pelo nome. Observe que, para variáveis BPEL cuja definição de mensagem WSDL possui uma única parte, estas variáveis agora representarão diretamente a parte em vez de ter um wrapper ao redor dos dados reais. As variáveis cujo tipo de mensagem tem várias partes terão um wrapper DataObject em torno das partes (em que o wrapper no WebSphere Application Developer Integration Edition era um WSIFMessage).</p> <p>Observe que há duas variações do método <code>getVariableAsWSIFMessage</code>:</p> <pre>getVariableAsWSIFMessage(String variableName) getVariableAsWSIFMessage(String variableName,     boolean forUpdate)</pre> <p>Para uma atividade de snippet Java, o acesso padrão é de leitura/gravação. É possível alterá-lo para de leitura especificando <b>@bpe.readOnlyVariables</b> com a lista de nomes das variáveis em um comentário no snippet. Por exemplo, você pode configurar a variável B e a variável D como de leitura da seguinte forma:</p> <pre>variableB.setString("/x/y/z", variableA.getString("/a/b/c")); // @bpe.readOnlyVariables names="variableA" variableD.setInt("/x/y/z", variableC.getInt("/a/b/c")); // @bpe.readOnlyVariables names="variableC"</pre> <p>Além disso, se você tiver um snippet Java em uma condição, as variáveis serão de leitura por padrão, mas será possível torná-las de leitura/gravação especificando <b>@bpe.readWriteVariables...</b></p>
Os métodos setter pouco especificados para variáveis BPEL não estão mais disponíveis.	As variáveis podem ser acessadas diretamente pelo nome. Observe que, para variáveis BPEL cuja definição de mensagem WSDL possui uma única parte, estas variáveis agora representarão diretamente a parte em vez de ter um wrapper ao redor dos dados reais. As variáveis cujo tipo de mensagem tem várias partes terão um wrapper DataObject em torno das partes (em que o wrapper no WebSphere Application Developer Integration Edition era um WSIFMessage).

*Tabela 11. Alterações e Soluções para Migrar os Snippets Java do BPEL do WebSphere Business Integration Server Foundation (continuação)*

Alterar	Solução
Os métodos getter pouco especificados para partes de mensagem das variáveis BPEL não são adequados para mensagens de uma única parte e foram alterados para mensagens de várias partes.	<p>Migre para o método getter pouco especificado para as propriedades das variáveis BPEL (DataObject's).</p> <p>Observe que para as variáveis BPEL, cuja definição de mensagem do WSDL tem uma única parte, a variável BPEL representa diretamente a parte e deve ser acessada diretamente sem utilizar um método getter.</p> <p>Há duas variações do método <code>getVariablePartAsObject</code>:</p> <pre>getVariablePartAsObject(String variableName, String partName) getVariablePartAsObject(String variableName, String partName, boolean forUpdate)</pre> <p>Para mensagens de várias partes, a funcionalidade equivalente é fornecida para esse método na 6.0:</p> <pre>getVariableProperty(String variableName, QName propertyName);</pre> <p>Na 6.0, não há nenhuma noção da utilização de uma variável para acesso de leitura (que era o caso na 5.1 para o primeiro método acima, bem como o segundo método com <code>forUpdate='false'</code>). A variável é diretamente utilizada no snippet da 6.0 e está sempre apta para ser atualizada.</p>
Os métodos setter pouco especificados para partes de mensagem das variáveis BPEL não são adequados para mensagens de uma única parte e foram alterados para mensagens de várias partes.	<p>Migre para o método setter pouco especificado para as propriedades das variáveis BPEL (DataObject's).</p> <p>Observe que para as variáveis BPEL, cuja definição de mensagem do WSDL tem uma única parte, a variável BPEL representa diretamente a parte e deve ser acessada diretamente sem utilizar um método setter.</p> <p>As chamadas para o seguinte método devem ser migradas:</p> <pre>setVariableObjectPart(String variableName, String partName, Object data)</pre> <p>Para mensagens de várias partes, a funcionalidade equivalente é fornecida para esse método na 6.0:</p> <pre>setVariableProperty(String variableName, QName propertyName, Serializable value);</pre>
Os métodos getter altamente especificados para links de parceiros BPEL não estão mais disponíveis.	Migre para os métodos getter pouco especificados para links de parceiros BPEL.
Os métodos setter altamente especificados para links de parceiros BPEL não estão mais disponíveis.	Migre para os métodos setter pouco especificados para links de parceiros BPEL.

*Tabela 11. Alterações e Soluções para Migrar os Snippets Java do BPEL do WebSphere Business Integration Server Foundation (continuação)*

Alterar	Solução
Os métodos getter altamente especificados para conjuntos de correlações BPEL não estão mais disponíveis.	<p><b>Snippet da V5.1:</b></p> <pre>String corrSetPropStr =     getCorrelationSetCorrSetAProperty     CustomerName(); int corrSetPropInt =     getCorrelationSetCorrSetBProperty     CustomerId();</pre> <p><b>Snippet da V6.0:</b></p> <pre>String corrSetPropStr =     (String) getCorrelationSetProperty     ("CorrSetA", new     QName("CustomerName")); int corrSetPropInt =     ((Integer) getCorrelationSetProperty     ("CorrSetB", new     QName("CustomerId"))).intValue();</pre>
Parâmetros adicionais necessários para os métodos getter pouco especificados para propriedades customizadas da atividade BPEL.	<p><b>Snippet da V5.1:</b></p> <pre>String val =     getActivityCustomProperty("propName");</pre> <p><b>Snippet da V6.0:</b></p> <pre>String val =     getActivityCustomProperty     ("name-of-current-activity", "propName");</pre>
Parâmetros adicionais necessários para os métodos setter pouco especificados para propriedades customizadas da atividade BPEL.	<p><b>Snippet da V5.1:</b></p> <pre>String newVal = "new value"; setActivityCustomProperty     ("propName", newVal);</pre> <p><b>Snippet da V6.0:</b></p> <pre>String newVal = "new value"; setActivityCustomProperty     ("name-of-current-activity", "propName",     newVal);</pre>
O método <code>raiseFault(QName faultQName, Serializable message)</code> não existe mais.	Migre para o <code>raiseFault(QName faultQName, String variableName)</code> onde possível; caso contrário, migre para o método <code>raiseFault(QName faultQName)</code> ou crie uma nova variável BPEL para o objeto <code>Serializable</code> .

*Migrando Interações com o WebSphere Business Integration Adapters:*

Se o JMS Client for um WebSphere Business Integration Adapter, poderá ser necessário utilizar as ferramentas Enterprise Service Discovery para criar a Importação com Ligação JMS. Esta Importação utiliza uma ligação de dados especial para serializar o SDO para o formato exato esperado pelo WebSphere Business Integration Adapter.

Para acessar a ferramenta Enterprise Service Discovery:



1. Vá para **Arquivo** → **Novo** → **Outro** → **Business Integration** e selecione **Enterprise Service Discovery**. Clique em **Avançar**.
2. Escolha **Importador de Artefatos do WebSphere Business Integration Adapter**. Clique em **Avançar**.
3. Digite o caminho do arquivo de configuração do WebSphere Business Integration Adapter (.cfg) e o diretório que contém o esquema XML dos objetos de negócios que o adaptador utiliza. Clique em **Avançar**.
4. Examine a consulta que é gerada e, se estiver correta, clique em **Executar Consulta**. Na lista **Objetos descobertos por consulta**, selecione os objetos que você deseja incluir (um a um) e clique no botão **>> Incluir**.
5. Aceite os parâmetros de configuração para o objeto de negócios e clique em **OK**.
6. Repita essa operação para cada objeto de negócios.
7. Clique em **Avançar**.
8. Para o **Formato do objeto de negócios de tempo de execução** selecione **SDO**. Para o **Projeto de Destino** selecione o módulo que acabou de migrar. Deixe o campo **Pasta** em branco.
9. Clique em **Concluir**.

Esta ferramenta migrará os XSDs antigos para o formato esperado pela ligação de dados especial, portanto, remova os XSDs antigos do WebSphere

Business Integration Adapter do módulo e utilize os novos XSDs. Se o módulo não for receber mensagens do adaptador, exclua as Exportações geradas por essa ferramenta. Se o módulo não enviar nenhuma mensagem para o adaptador, exclua a Importação. Consulte o centro de informações para obter informações adicionais sobre este recurso.

#### *Migrando Interfaces WSDL com Tipos de Matrizes Codificadas pelo SOAP:*

Esta seção mostra como migrar ou manipular esquemas XML que têm tipos de matrizes codificadas pelo SOAP.

Os tipos de matrizes codificados pelo SOAP que possuem o estilo RPC serão tratados como seqüências não ligadas de um tipo concreto na 6.0. Não é recomendado que você crie nenhum tipo XSD que faça referência aos tipos soapenc:Array de nenhuma maneira, porque o modelo de programação move-se em direção ao estilo Document/Literal agrupado em vez de ao estilo RPC.

Há casos em que um aplicativo SCA deve chamar um serviço externo que não utiliza o tipo soapenc:Array. Em alguns casos, não existe uma maneira de evitar isto, portanto, a seguir é mostrado como lidar com esta situação:

Código WSDL de amostra:

```
<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:wsdl="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 amostra para um cliente deste Serviço da Web:

```

// Localize o serviço do fornecedor e a operação doFindVendor
Service findVendor=(Service)ServiceManager.INSTANCE.locateService("vendorSearch");
OperationType doFindVendorOperationType=findVendor.getReference().getOperationType("doGoogleSearch");
// Crie o DataObject de entrada
DataObject doFindVendor=DataFactory.INSTANCE.create(doFindVendorOperationType.getInputType());
doFindVendor.setString("productId", "12345");
doFindVendor.setString("productName", "Refrigerator");
// Chame o serviço FindVendor
DataObject findVendorResult = (DataObject)findVendor.invoke(doFindVendorOperationType, doFindVendor);
// Exiba os resultados
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

```

Aqui há outro exemplo em que o tipo de raiz do objeto de dados é um `soapenc:Array`. Observe como o `sampleElements DataObject` é criado utilizando o segundo esquema listado acima. O tipo do `DataObject` é obtido primeiro e, em seguida, é obtida a propriedade para `sampleStructElement`. Isso é realmente uma propriedade do sinalizador de substituição e é utilizada apenas para obter uma propriedade válida a ser utilizada ao incluir o `DataObjects` na sequência. Um padrão como esse pode ser utilizado em seu cenário:

Código WSDL de amostra:

```

<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 amostra para um cliente deste Serviço da Web:

```

// Crie o DataObject de entrada e obtenha a sequência de SDO para qualquer
// elemento
DataFactory dataFactory=DataFactory.INSTANCE;
DataObject arrayOfStruct = dataFactory.create("http://soapinterop.org/xsd","ArrayOfSOAPStruct");
Sequence sequence=arrayOfStruct.getSequence("any");
// Obtenha a propriedade de SDO para o elemento de amostra que desejamos utilizar
// aqui para ocupar a sequência
// Definimos este elemento em um arquivo XSD, consulte SampleElements.xsd
DataObject sampleElements=dataFactory.create("http://sample/elements",
"DocumentRoot");
Property property = sampleElements.getType().getProperty("sampleStructElement");
// Inclua os elementos na sequência
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);
// Chame a operação echoStructArray
System.out.println("[client] invoking echoStructArray operation");
DataObject echoArrayOfStruct = (DataObject)interopTest.invoke("echoStructArray", arrayOfStruct);
// Exiba os resultados
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"));

```

## Migrando Projetos EJB do WebSphere Business Integration:

No WebSphere Studio Application Developer Integration Edition, os projetos EJB podem ter recursos especiais do WebSphere Business Integration como o Extended Messaging (CMM) e CMP/A (Component-managed Persistence Anywhere). Os descritores de implementação para tais projetos devem ser migrados e esta seção mostra como executar essa migração.

Para executar essa migração, complete estas etapas:

1. Copie o projeto EJB do WebSphere Business Integration para o novo espaço de trabalho da 6.0 e importe-o a partir do WebSphere Integration Developer utilizando o assistente **Arquivo → Importar → Projeto Existente para Espaço de Trabalho**. Opcionalmente, você também pode executar o assistente de Migração do J2EE.
2. Feche todas as instâncias do WebSphere Integration Developer em execução no espaço de trabalho da 6.0.

3. Execute o seguinte script que migrará os descritores de implementação do WebSphere Business Integration no projeto EJB:

**No Windows:**

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

**No Linux:**

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

Os seguintes parâmetros são suportados, em que o nome do espaço de trabalho e do projeto são obrigatórios:

Uso:       WSADIEEJBProjectMigration.bat  
[-e eclipse-folder] -d workspace -p project  
eclipse-folder: O local da pasta do eclipse -- normalmente é 'eclipse'  
localizado na pasta de instalação do produto.  
workspace: O espaço de trabalho que contém o projeto EJB do WSADIE a ser migrado.  
project: O nome do projeto a ser migrado.

Por exemplo:

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

4. Ao abrir o WebSphere Integration Developer, você precisará atualizar o projeto EJB para obter os arquivos atualizados.
5. Procure o arquivo **ibm-web-ext.xmi** no projeto EJB. Se um for localizado, certifique-se de que a seguinte linha esteja presente no arquivo sob o elemento:  

```
<webappext:WebAppExtension> element:  
<webApp href="WEB-INF/web.xml#WebApp"/>
```
6. Remova o código de implementação antigo que foi gerado na 5.1. Gere novamente o código de implementação seguindo as diretrizes do WebSphere Application Server.

### Desempenhando uma Correção Manual para Conflitos de Espaços de Nomes:

No WebSphere Studio Application Developer Integration Edition 5.1, a definição de dois tipos XSD ou WSDL diferentes com o mesmo nome e espaço de nomes de destino era suportada. No WebSphere Integration Developer 6.0, ela não é suportada. Será requerida a migração manual se você encontrar erros de definição duplicada após a construção de seus projetos migrados.

Para resolver este problema, conclua as seguintes etapas:

1. Se as definições forem iguais, exclua uma delas e, em seguida, limpe e reconstrua seu projeto. Corrija os erros que podem surgir apontando arquivos WSDL/XSD existentes para o arquivo que contém a definição que *não* foi excluída.
2. Se as definições *não* forem iguais e for necessário utilizar as duas definições em seu serviço migrado, renomeie a definição ou o espaço de nomes de destino. Se houver apenas algumas definições duplicadas em todo o arquivo, será recomendável alterar seus nomes. Se todas as definições no arquivo forem duplicadas, será recomendável alterar o espaço de nomes de destino de todas as definições. Limpe e reconstrua o projeto, certificando-se de que os artefatos que você deseja utilizar na(s) definição(ões) modificada(s) façam referência ao novo nome da definição ou espaço de nomes.
3. Caso existam duas instruções de importação para o mesmo espaço de nomes em um arquivo WSDL, será possível resolver este problema encadeando as importações de forma que um destes WSDLs importe o outro, que importa o próximo, etc., para que exista apenas uma importação para este espaço de nomes por arquivo WSDL. Em seguida, limpe e reconstrua o projeto.

### Excluindo Manualmente Definições do WSIF (Web Services Invocation Framework) 5.1:

Depois de concluir a migração de seus artefatos de origem, é necessário excluir todas as definições WSDL de Ligação e Serviço do WSIF 5.1 de seus projetos de 6.0 que não estão sendo mais utilizados. O cenário de consumo para migração de serviço é o único caso em que uma Ligação ou Serviço do WSIF ainda estaria sendo utilizado.

Os espaços de nomes WSDL a seguir indicam que uma definição de ligação ou de serviço é um Serviço do WSIF 5.1 e pode ser descartado se não for mais utilizado:

**Espaço de Nomes de WSIF EJB:**

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

**Espaço de Nomes de WSIF Java:**

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

**Espaço de Nomes de WSIF JMS:**

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

**Espaço de Nomes de WSIF do Processo de Negócios:**

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

**Espaço de Nomes de WSIF do Transformador:**

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

**Espaço de Nomes de WSIF do IMS:**

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

**Espaço de Nomes de WSIF CICS-ECI:**

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

**Espaço de Nomes de WSIF CICS-ECI:**

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

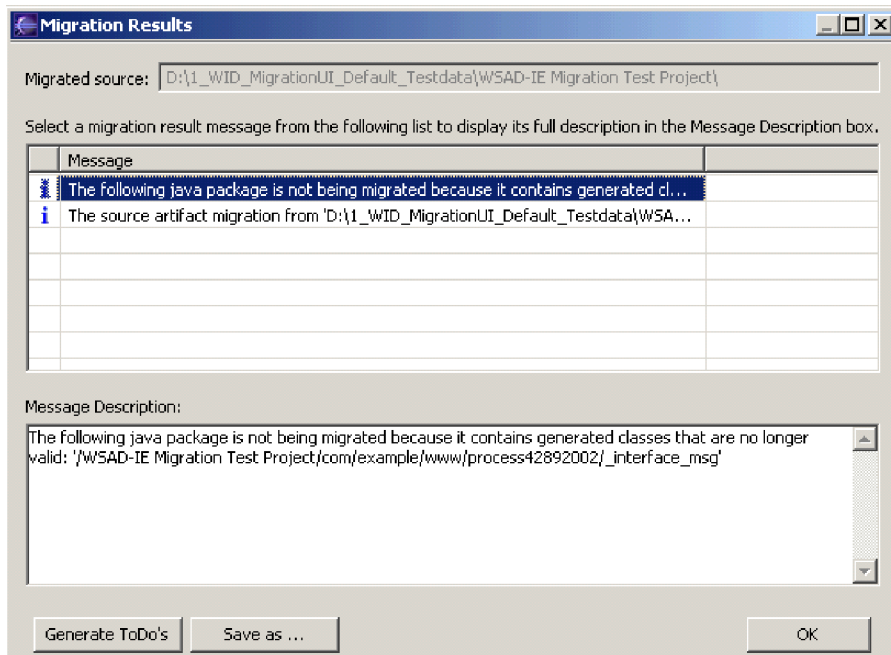
**Espaço de Nomes de WSIF do HOD:**

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

## **Verificando a Migração de Artefatos de Origem**

Quando o Assistente de Migração tiver sido concluído com êxito, será exibida uma lista de mensagens de erro, de aviso e/ou informativas. Estas mensagens podem ser utilizadas para verificar a migração de artefatos de origem.

A página a seguir aparece após a conclusão do Assistente de Migração:



Examine cada mensagem para verificar se alguma ação precisa ser executada para corrigir imediatamente um artefato que não pôde ser totalmente migrado.

Para verificar se uma parte da migração foi concluída, vá para a perspectiva Business Integration e certifique-se de que todos os processos e interfaces WSDL do projeto de serviço antigo apareçam no novo módulo. Construa o projeto e corrija quaisquer erros que impedem a construção do projeto.

Depois de executar as etapas de migração manuais requeridas para concluir a migração do aplicativo Business Integration, exporte o aplicativo como um arquivo EAR e instale-o em um WebSphere Process Server, configurando os recursos apropriados.

Desempenhe as etapas de migração manuais requeridas para migrar qualquer código de cliente ou gerar novo código de cliente utilizando o WebSphere Integration Developer. Certifique-se de que o cliente possa acessar o aplicativo e que o aplicativo tenha o mesmo comportamento existente no ambiente de tempo de execução anterior.

## Trabalhando com Falhas de Migração de Artefatos de Origem

Se a migração de artefatos de origem do WebSphere Studio Application Developer Integration Edition falhar, existem várias maneiras de lidar com as falhas.

A seguir estão algumas das possíveis falhas de migração de artefatos de origem:

- Se você receber a seguinte mensagem:  
 "Mensagem de Erro de Migração"  
 Razão: Falha de Migração Fatal  
 Mensagem: Entre em Contato com o Representante IBM

será necessário consultar o arquivo de registro do WebSphere Integration Developer na pasta .metadata do novo espaço de trabalho para ver os detalhes do erro. Se possível, resolva a causa do erro, exclua o módulo criado no novo espaço de trabalho e tente a migração novamente.

Se o assistente de Migração for concluído sem esta mensagem, será exibida uma lista de mensagens informativas, de aviso e de erro. Elas significam que alguma parte do projeto de serviço não pôde ser migrada automaticamente e que as alterações manuais devem ser executadas para completar a migração.

## Boas Práticas para o Processo de Migração de Artefatos de Origem

Existem diversas boas práticas para o processo de migração de artefatos de origem do WebSphere Studio Application Developer Integration Edition.

As práticas a seguir mostram como projetar serviços do WebSphere Studio Application Developer Integration Edition para assegurar que eles sejam migrados com êxito para o novo modelo de programação:

- Tente utilizar a atividade de **Designação** onde for possível (em oposição ao serviço de transformador que é requerido apenas quando é necessária uma transformação avançada). Você deve utilizar essa prática porque o componente intermediário deve ser construído para que o módulo SCA chame um serviço de transformador. Adicionalmente, não há nenhum suporte de ferramenta especial no WebSphere Integration Developer para os serviços de transformador criados na 5.1 (você deve utilizar o editor WSDL ou XML para modificar o XSLT incorporado no arquivo WSDL se precisar alterar o comportamento do serviço de transformador).
- Especifique uma parte por mensagem WSDL utilizando a especificação WS-I (Web Services Interoperability) e o estilo preferido da 6.0.
- Utilize o estilo WSDL doc-literal pois este é o estilo preferido na 6.0.
- Certifique-se de que todos os tipos complexos tenham recebido um nome e que cada tipo complexo possa ser identificado exclusivamente por seu espaço de nomes e nome de destino. A seguir é mostrada a maneira recomendada para definir tipos complexos e elementos desse tipo (definição de tipo complexo seguida por uma definição de elemento que a 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>
```

O exemplo a seguir é um tipo complexo anônimo que deve ser evitado, pois pode causar problemas quando um SDO for serializado para XML (elemento contendo uma definição de tipo complexo anônima):

```
<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>
```

- Se estiver publicando um serviço para consumidores externos, gere o código de implementação do serviço utilizando os IBM Web Services (em oposição ao Apache SOAP/HTTP) porque os IBM Web Services são diretamente suportados na 6.0 e os Apache Web Services não são.



- Existem duas maneiras de organizar arquivos WSDL e XSD na 5.1 para minimizar a quantidade de reorganização que deve ser feita durante a migração. Na 6.0, os artefatos compartilhados, como arquivos WSDL e XSD, devem ser localizados em projetos do BI (*módulos e bibliotecas* do Business Integration) para serem referidos por um serviço do BI:
  - Mantenha todos os arquivos WSDL compartilhados por mais de um projeto de serviço em um projeto Java ao qual os projetos de serviço podem fazer referência. Durante a migração para a 6.0, você criará uma nova Biblioteca do Business Integration com o mesmo nome do projeto Java compartilhado pela 5.1. Copie todos os artefatos do projeto Java 5.1 compartilhado para a biblioteca para que o assistente de migração possa resolver os artefatos quando ele migrar os projetos de serviços que utilizam esses artefatos
  - Mantenha uma cópia local de todos os arquivos WSDL/XSD aos quais um projeto de serviço se refere no próprio projeto de serviço. Os projetos de serviço do WebSphere Studio Application Developer Integration Edition serão migrados para um Módulo do Business Integration no WebSphere Integration Developer e um módulo não pode ter dependências de outros módulos (um projeto de serviço com dependências de outro projeto de serviço em relação ao compartilhamento de arquivos WSDL ou XSD não será migrado corretamente).
- Evite utilizar a API do Sistema de Mensagens Genérico (MDBs Genéricos) do Business Process Choreographer, pois ela não será fornecida na 6.0. Uma ligação posterior de oferta da interface MDB *não* estará disponível na 6.0.
- Utilize a API de EJB Genérico do Business Process Choreographer em oposição à chamada de beans de sessão gerados específicos de uma determinada versão de um processo. Esses beans de sessão não serão gerados na V6.0.0.0.
- Se você tiver um processo de negócios com várias respostas para a mesma operação, certifique-se de que se qualquer uma delas tiver configurações do cliente e que todas as respostas para essa operação tiverem as mesmas configurações do cliente que a 6.0, apenas um conjunto de configurações do cliente será suportado por resposta de operação.
- Projete snippets Java de BPEL de acordo com as seguintes instruções:
  - Evite enviar parâmetros WSIFMessage para quaisquer classes Java customizadas - tente não depender do formato de dados de WSIFMessage, se possível.
  - Evite utilizar as APIs de metadados do WSIF, se possível.
- Evite declarar variáveis locais em snippets Java do BPEL que têm o mesmo nome das variáveis BPEL. Na 6.0, as variáveis BPEL estão diretamente acessíveis nos snippets, portanto, as variáveis locais com o mesmo nome poderão criar um conflito.
- Evite criar serviços EJB ou Java de cima para baixo onde possível, porque a estrutura Java/EJB gerada a partir de PortTypes/Mensagens WSDL será dependente de classes WSIF (por exemplo, WSIFFormatPartImpl). Em vez disso, crie primeiro as interfaces Java/EJB e gere um serviço em torno da classe/EJB Java (abordagem de baixo para cima).
- Evite criar ou utilizar interfaces WSDL que façam referência ao tipo soapenc:Array, porque esse tipo de interface não é suportado de forma nativa no modelo de programação SCA.
- Evite criar tipos de mensagens cujo elemento de alto nível seja um tipo de matriz (atributo maxOccurs é maior que um), porque esse tipo de interface não é suportado de forma nativa no modelo de programação SCA.
- Defina suas interfaces WSDL precisamente - evite, onde possível, complexTypes de XSD que tenham referências ao tipo xsd:anyType.
- Para quaisquer WSDL e XSDs gerados a partir de um EJB ou Java bean, certifique-se de que o espaço de nomes de destino seja exclusivo (o nome da classe e o nome do pacote Java são representados pelo espaço de nomes de destino) para evitar conflitos durante a migração para o WebSphere Process Server V6. No WebSphere Process Server V6, duas diferentes definições de WSDL/XSD que possuem o mesmo nome e espaço de nomes de destino não são permitidas. Esta situação geralmente ocorre quando o assistente de Serviço da Web ou o comando Java2WSDL é utilizado sem especificar o espaço de nomes de destino explicitamente (o espaço de nomes de destino será exclusivo para o nome do pacote do EJB ou Java bean, mas não para a própria classe, portanto, ocorrerão problemas quando um



serviço da Web for gerado para dois ou mais EJB ou Java beans no mesmo pacote). A solução é especificar um mapeamento de pacote customizado para espaço de nomes no assistente de Serviço da Web ou utilizar a opção da linha de comandos **-namespace** Java2WSDL para assegurar que o espaço de nomes dos arquivos gerados seja exclusivo para a classe especificada.

- Utilize espaços de nomes exclusivos para cada arquivo WSDL sempre que possível. Existem limitações na importação de dois arquivos WSDL diferentes com o mesmo espaço de nomes, de acordo com a especificação WSDL 1.1 e, no WebSphere Integration Developer 6.0, estas limitações são estritamente aplicadas.

## **Limitações do Processo de Migração (Para Migração de Artefatos de Origem)**

Existem algumas limitações envolvidas no processo de migração de artefatos de origem do WebSphere Studio Application Developer Integration Edition.

As listas a seguir detalham algumas das limitações do processo de migração para migração de artefatos de origem:

### **Limitações Gerais**

- Este assistente de Migração não pode manipular espaços de trabalho inteiros do WebSphere Studio Application Developer Integration Edition. Ele destina-se a migrar um projeto de serviço do WebSphere Studio Application Developer Integration Edition de cada vez.
- O assistente de Migração não migra binários de aplicativos, ele migra apenas artefatos de origem localizados em um projeto de serviço do WebSphere Studio Application Developer Integration Edition.
- O Business Rule Beans foi reprovado no WebSphere Process Server 6.0 mas existe uma opção durante a instalação do WebSphere Process Server para instalar o suporte para o Business Rule Beans reprovado de forma que ele será executado “como está” em um servidor do WebSphere Process Server 6.0. No entanto, não existe nenhum suporte a ferramentas para o Business Rule Beans antigo e, se você desejar que os artefatos do Business Rule Bean antigo sejam compilados nas ferramentas, deverá seguir a documentação do WebSphere Integration Developer para instalar esses recursos reprovados sobre o servidor de teste do WebSphere Process Server 6.0 incorporado e, em seguida, incluir manualmente os arquivos jar reprovados no caminho de classe do projeto como jars externos. Você deve utilizar as novas ferramentas do Business Rule disponíveis no WebSphere Integration Developer para recriar suas regras de negócios de acordo com a especificação de 6.0.
- O suporte ao Extended Messaging foi reprovado no WebSphere Process Server 6.0 mas existe uma opção durante a instalação do WebSphere Process Server para instalar suporte para os recursos reprovados do Extended Messaging de forma que os aplicativos existentes possam ser executados “como estão” em um servidor do WebSphere Process Server 6.0. No entanto, não existe nenhum suporte de ferramentas para os recursos do Extended Messaging antigo e, se você desejar que os artefatos do Extended Messaging antigo sejam compilados nas ferramentas, deverá seguir a documentação do WebSphere Integration Developer para instalar esses recursos reprovados sobre o servidor de teste do WebSphere Process Server 6.0 incorporado e, em seguida, incluir manualmente os arquivos jar reprovados no caminho de classe do projeto como “jars externos”.
- A ligação de dados JMS padrão fornecida não oferece acesso a propriedades de cabeçalho JMS customizadas. Uma ligação de dados customizada deve ser gravada para que os serviços SCA obtenham acesso a quaisquer propriedades de cabeçalho JMS customizadas.

### **Limitações do Modelo de Programação SCA**

- A especificação do SDO versão 1 não fornece acesso à matriz de byte COBOL ou C – isso terá impacto nos que trabalham com vários segmentos IMS.
- A especificação do SDO versão 1 para serialização não suporta redefinições COBOL ou uniões C.
- Ao reprojetar os artefatos de origem de acordo com o modelo de programação SCA, observe que o estilo document/literal agrupado do WSDL (que é o estilo padrão para os novos artefatos criados utilizando as ferramentas do WebSphere Integration Developer) não suporta a sobrecarga de métodos. Os outros estilos do WSDL ainda são suportados, portanto, é recomendado que outro estilo/codificação do WSDL diferente de document/literal agrupado seja utilizado para esses casos.

- O suporte nativo para matrizes é limitado. Para chamar um serviço externo que expõe uma interface WSDL com tipos soapenc:Array, será necessário criar uma interface WSDL que define um elemento cujo atributo "maxOccurs" seja maior que um (esta é a abordagem recomendada para projetar um tipo de matriz).

## Limitações Técnicas do Processo de Migração de BPEL

- **Várias Respostas por Operação de BPEL** - No WebSphere Business Integration Server Foundation, um processo de negócios poderia ter uma atividade de recebimento e várias atividades de resposta para a mesma operação.
- **Limitações de Migração do Snippet Java BPEL** - O modelo de programação foi alterado significativamente do WebSphere Studio Application Developer Integration Edition para o WebSphere Integration Developer e nem todas as APIs suportadas do WebSphere Studio Application Developer Integration Edition podem ser migradas diretamente para as APIs correspondentes do WebSphere Integration Developer. Qualquer lógica Java pode ser localizada nos snippets Java BPEL de modo que a ferramenta de migração automática pode não estar apta a converter cada snippet Java para o novo modelo de programação. A maior parte das chamadas de API do snippet padrão será migrada automaticamente do modelo de programação do snippet Java 5.1 para o modelo de programação do snippet Java 6.0. As chamadas de API do WSIF são migradas para as chamadas da API DataObject onde for possível. As classes Java customizadas que aceitam objetos WSIFMessage precisarão de migração manual de forma que então aceitem e retornem objetos commonj.sdo.DataObject:
  - **APIs de Metadados WSIFMessage** - Todo o uso da API do WSIF deve ser eliminado onde for possível. O assistente de migração não pode migrar automaticamente os metadados WSIFMessage e outras APIs do WSIF para o SDO equivalente, portanto, a migração manual é requerida.
  - **APIs EndpointReference/EndpointReferenceType** - Estas classes não são migradas automaticamente. A migração manual é necessária porque os métodos getter/setter de link do parceiro lidam com objetos commonj.sdo.DataObject em vez dos objetos com.ibm.websphere.srm.bpel.wsaddressing.EndpointReferenceType da 5.1.
  - **Tipos Complexos com Nomes Duplicados** - Se um aplicativo declarar tipos complexos (em WSDLs ou XSDs) com espaços de nomes e nomes locais idênticos ou espaços de nomes diferentes mas nomes locais idênticos, os snippets Java que utilizam estes tipos poderão não ser migrados corretamente. Verifique se os snippets estão corretos após a conclusão do assistente de migração.
  - **Tipos complexos com nomes locais idênticos a classes Java no pacote java.lang** - Se um aplicativo declarar tipos complexos (em WSDLs ou XSDs) com nomes locais idênticos a classes no pacote java.lang do J2SE 1.4.2, os snippets Java que utilizam a classe de java.lang correspondente poderão não ser migrados corretamente. Verifique se os snippets estão corretos após a conclusão do assistente de migração.
  - **Variáveis BPEL com o mesmo nome de variáveis locais no snippet Java do BPEL** - Se você definiu variáveis locais com o mesmo nome das variáveis BPEL nos snippets Java do BPEL, precisará corrigir isso manualmente, porque as variáveis BPEL agora estão acessíveis pelo nome nos snippets Java e poderá existir um conflito.
  - **Variáveis BPEL de leitura e de leitura/gravação** - Em qualquer trecho de código Java 5.1, era possível configurar uma variável BPEL como "somente leitura", indicando que as alterações feitas neste objeto não afetarão o valor da variável BPEL. Também era possível configurar uma variável BPEL como de "leitura/gravação", indicando que as alterações feitas no objeto seriam refletidas para a própria variável BPEL. A seguir são mostradas quatro maneiras em que um snippet Java poderia ser acessado como "de leitura" em qualquer snippetJava de BPEL 5.1:

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

A seguir estão duas maneiras em que uma variável BPEL poderia ser acessada como de "leitura/gravação" em qualquer snippet Java de BPEL 5.1:

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

Na 6.0, o acesso de leitura e de leitura/gravação a variáveis BPEL é manipulado em uma "base por snippet", indicando que é possível incluir um comentário especial no snippet Java de BPEL para especificar se as atualizações na variável BPEL devem ser descartadas ou mantidas após a conclusão da execução do snippet. A seguir estão as configurações de acesso padrão para os tipos de snippet Java de BPEL 6.0:

Atividade de Snippet Java de BPEL

Acesso Padrão: leitura/gravação

Substituir Acesso Padrão pelo comentário contendo:

```
@bpe.readOnlyVariables names="variableA,variableB"
```

Expressão de Snippet Java de BPEL (Utilizada em um Tempo Limite, Condição, etc)

Acesso Padrão: de leitura

Substituir Acesso Padrão pelo comentário contendo:

```
@bpe.readWriteVariables names="variableA,variableB"
```

Durante a migração, estes comentários serão criados automaticamente quando uma variável tiver sido acessada de uma maneira que não seja o padrão no 6.0. No caso de haver um conflito (indicando que a variável BPEL foi acessada como "de leitura" e como "leitura/gravação" no mesmo snippet), será emitido um aviso e o acesso será configurado como "leitura/gravação". Se você receber algum desses avisos, certifique-se de que a configuração do acesso à variável BPEL como de "leitura/gravação" esteja correta para sua situação. Se ela não estiver correta, será necessário corrigi-la manualmente utilizando o editor BPEL do WebSphere Integration Developer.

- **Propriedades primitivas com diversos valores em tipos complexos** - Na 5.1, as propriedades com diversos valores são representadas por matrizes do tipo da propriedade. Por isso, as chamadas para obter e configurar a propriedade utilizam matrizes. Na 6.0, `java.util.List` é utilizado para esta representação. A migração automática manipulará todos os casos em que a propriedade com diversos valores é algum tipo de objeto Java, mas no caso de o tipo da propriedade ser um primitivo Java (`int`, `long`, `short`, `byte`, `char`, `float`, `double` e `boolean`), as chamadas para obter e configurar toda a matriz não são convertidas. A migração manual neste caso pode exigir a inclusão de um loop para agrupar/desagrupar os primitivos em/de sua classe de wrapper Java (`Integer`, `Long`, `Short`, `Byte`, `Character`, `Float`, `Double` e `Boolean`) para utilização no restante do snippet.
- **Instanciação de classes geradas que representam tipos complexos** - Na 5.1, as classes geradas de tipos complexos definidos em um aplicativo podem ser facilmente instanciadas em um snippet Java utilizando o construtor sem argumento padrão. Um exemplo disto é:

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

Na 6.0, uma classe de depósito de informações do provedor especial deve ser utilizada para instanciar estes tipos ou uma instância do tipo de abrangência pode ser utilizada para criar o subtipo. Se uma variável de processo BPEL `InputVariable` tiver sido definida como tendo o tipo `InputMessage`, a versão 6.0 do snippet precedente será:

```
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");
```

O conversor de snippet tenta fazer esta alteração mas, se a ordem em que ocorrem as instanciações originais não seguir o padrão parent-then-child, a migração manual será requerida (ou seja, o conversor não tenta reordenar de maneira inteligente as instruções de instanciação no snippet).

- No WebSphere Business Integration Server Foundation 5.1, referências dinâmicas são representadas como partes de mensagens do WSDL do tipo EndpointReferenceType ou elemento EndpointReference a partir do espaço de nomes:

<http://wsaddressing.bpel.srm.websphere.ibm.com>

Tais referências serão migradas para o tipo de elemento service-ref padrão a partir do espaço de nomes do processo de negócios padrão:

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

<http://schemas.xmlsoap.org/ws/2004/08/addressing>

Consulte a documentação do BPEL Editor para obter instruções sobre a importação manual dessas definições de esquema para o seu projeto a fim de que todas as referências sejam resolvidas adequadamente.

- **Tipo de Mensagem de Variável BPEL** - Um tipo de mensagem WSDL deve ser especificado para todas as variáveis BPEL utilizadas em snippets Java. Não é possível migrar snippets Java que acessam variáveis BPEL sem o atributo "messageType" especificado.



---

## Avisos

A publicação XDoclet incluída neste produto IBM é utilizada com permissão e é coberta pela seguinte declaração de atribuição de direitos autorais: Direitos Autorais (c) 2000-2004, XDoclet Team. Todos os direitos reservados.

Partes baseadas em *Design Patterns: Elements of Reusable Object-Oriented Software*, de Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, Direitos Autorais (c) 1995 da Addison-Wesley Publishing Company, Inc. Todos os direitos reservados.

Direitos Restritos para Usuários do Governo dos Estados Unidos - Uso, duplicação e divulgação restritos pelo documento GSA ADP Schedule Contract com a IBM Corporation.

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos. É possível que a IBM não ofereça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte um representante IBM local para obter informações sobre os produtos e serviços disponíveis atualmente em sua área. Qualquer referência a produtos, programas ou serviços IBM, não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM, poderá ser utilizado em substituição a este produto, programa ou serviço. Entretanto, a avaliação e verificação da operação de qualquer produto, programa ou serviço não-IBM são de responsabilidade do Cliente.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não garante ao Cliente nenhum direito sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil  
Av. Pasteur, 138-146  
Botafogo  
Rio de Janeiro, RJ  
CEP 22290-240

Para pedidos de licença relacionados a informações de DBCS (Conjunto de Caracteres de Byte Duplo), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie pedidos de licença, por escrito, para:

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

O parágrafo a seguir não se aplica a nenhum país em que tais disposições não estejam de acordo com a legislação local: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS (OU CONDIÇÕES) DE NÃO-INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, esta disposição pode não se aplicar ao Cliente.

Estas informações podem conter imprecisões técnicas ou erros tipográficos. Periodicamente são feitas alterações nas informações aqui contidas; tais alterações serão incorporadas em futuras edições desta publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar os produtos e/ou programas descritos nesta publicação, sem aviso prévio.

Referências nestas informações a Web sites não-IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a esses Web sites. Os materiais contidos nesses Web sites não fazem parte deste produto IBM e sua utilização desses Web sites é de inteira responsabilidade do Cliente.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com o Cliente.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

*Gerência de Relações Comerciais e Industriais da  
Rational  
Brasil  
Av. Pasteur, 138-146  
Botafogo  
Rio de Janeiro, RJ  
CEP 22290-240*

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriadas, incluindo em alguns casos o pagamento de uma taxa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM, do Contrato de Licença do Programa Internacional IBM ou de qualquer outro contrato equivalente.

Todos os dados sobre desempenho aqui descritos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido tomadas em sistemas de nível de desenvolvimento e não há garantia de que estas medidas serão iguais em sistemas geralmente disponíveis. Além disso, algumas medidas podem ter sido estimadas por extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar os dados aplicáveis para seu ambiente específico.

As informações relativas a produtos não-IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não-IBM. Dúvidas sobre os recursos de produtos não-IBM devem ser encaminhadas diretamente a seus fornecedores.

Todas as declarações relacionadas aos objetivos e intenções futuras da IBM estão sujeitas a alterações ou cancelamento sem aviso prévio e representam apenas metas e objetivos.

Estas informações contêm exemplos de dados e relatórios utilizados nas operações diárias de negócios. Para ilustrá-los da forma mais completa possível, os exemplos contêm nomes de indivíduos, empresas, marcas e produtos. Todos esses nomes são fictícios e qualquer semelhança com nomes e endereços utilizados por uma empresa real é mera coincidência.

#### LICENÇA DE DIREITOS AUTORAIS:

Estas informações contêm programas de aplicativos de exemplo na linguagem fonte, ilustrando as técnicas de programação em diversas plataformas operacionais. O Cliente pode copiar, modificar e distribuir estes programas de exemplo sem a necessidade de pagar à IBM, com objetivos de desenvolvimento, utilização, marketing ou distribuição de programas aplicativos em conformidade com a interface de programação de aplicativo para a plataforma operacional para a qual os programas de exemplo são criados. Esses exemplos não foram testados completamente em todas as condições. Portanto, a IBM não pode garantir ou implicar a confiabilidade, manutenção ou função destes programas. O Cliente pode copiar, modificar e distribuir estes programas de exemplo de qualquer maneira sem



pagamento à IBM, com objetivos de desenvolvimento, utilização, marketing ou distribuição de programas aplicativos em conformidade com interfaces de programação de aplicativos da IBM.

Cada cópia ou parte destes programas de exemplo ou qualquer trabalho derivado deve incluir um aviso de direitos autorais com os dizeres:

(C) (nome da empresa) (ano). Partes deste código são derivadas dos Programas de Exemplo da IBM Corp. (C) Direitos Autorais IBM Corp. 2000, 2005. Todos os direitos reservados.

Se estas informações estiverem sendo exibidas em cópia eletrônica, as fotografias e ilustrações coloridas podem não aparecer.

## **Informações sobre a Interface de Programação**

As informações sobre interface de programação destinam-se a facilitar a criação de software aplicativo utilizando este programa.

As interfaces de programação de uso geral permitem que o Cliente desenvolva o software aplicativo que obtém os serviços das ferramentas deste programa.

Entretanto, também podem estar contidas informações de diagnóstico, modificação e ajuste. As informações sobre diagnósticos, modificações e ajustes são fornecidas para ajudá-lo a depurar seu software aplicativo.

**Aviso:** Não utilize estas informações sobre diagnósticos, modificações e ajustes como uma interface de programação, pois elas estão sujeitas a alterações.

## **Marcas Registradas e Marcas de Serviço**

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







Impresso em Brazil

S517-8345-01

