FUJITSU

FUJITSU Software
Interstage Service Integrator V9.5.0



アプリケーション開発ガイド

Windows/Solaris/Linux

B1X1-0281-01Z0(00)
2013年6月

まえがき

本書の目的

本書は、Interstage Service Integrator(以降、ISIと略します)の共通APIを利用したアプリケーションの開発方法について説明しています。

本書の読者

本書は、ISIの共通APIを利用したアプリケーションの開発者を対象としています。
なお、本書を読むためには、以下の知識が必要です。

- インターネットに関する知識
- 使用するOSに関する知識
- XMLに関する知識
- Interstage Application Serverに関する知識
- Javaプログラミング言語に関する知識
- .NET Frameworkに関する知識

本書の構成

本書は、以下の構成になっています。

第1章 アプリケーション開発の概要

ISIのアプリケーション開発の概要について説明しています。

第2章 サービス利用側のアプリケーションの開発

ISIのサービス利用側アプリケーションの開発方法について説明しています。

第3章 メディエータファンクションの開発

メディエータファンクションとして呼び出すユーザプログラムの開発方法について説明しています。

第4章 アダプタの開発

ISIで使用できるアダプタの開発について説明しています。

第5章 変換定義の開発

ISIで利用する変換定義の開発について説明しています。

第6章 アダプタ利用時の定義の開発

ISIが提供しているアダプタを利用する場合に必要な定義の作成内容、および利用方法によって必要となるアプリケーションの開発内容について説明しています。

付録A 定義の概要

ISIの機能利用時に必要な定義の概要について説明しています。

付録B 定義の詳細

ISIの機能を利用するための定義の詳細について説明しています。

付録C 定義の例

定義ファイルの内容と定義する項目の例について説明しています。

付録D 互換機能

互換機能について説明しています。

付録E BPEL機能のアプリケーション開発

BPEL機能のアプリケーション開発について説明しています。

お願い

- このマニュアルは、予告なしに変更されることがあります。
- このマニュアルを無断で他に転用しないようお願いします。
- このマニュアルに記載されたデータの使用に起因する第三者の特許権、およびその他の権利の侵害については、当社はその責を負いません。

2013年6月 初版

Copyright 2013 FUJITSU LIMITED

目　次

第1章 アプリケーション開発の概要 1
1.1 サービス提供側のアプリケーションの設計 1
1.1.1 Webサービスアプリケーション 2
1.1.2 データベース 2
1.1.3 FTPサーバ 2
1.1.4 CORBAアプリケーション 2
1.1.5 WebSphere MQ 2
1.1.6 III連携 2
1.2 サービス利用側のアプリケーションの設計 2
1.2.1 Webサービスクライアント 3
1.2.2 CORBAクライアントアプリケーション 3
1.2.3 WebSphere MQ 3
1.2.4 III連携 3
1.3 メディエータファンクションの開発 3
1.4 カスタムアダプタの設計 4
1.4.1 インバウンドサービス 4
1.4.2 アウトバウンドサービス 4

第2章 サービス利用側のアプリケーションの開発 5
2.1 SOAPを利用してISI呼び出しを行うアプリケーションの開発 5
2.2 CORBAを利用してISI呼び出しを行うアプリケーションの開発 6

第3章 メディエータファンクションの開発 14
3.1 呼出し可能なユーザプログラムの種類 14
3.1.1 Javaユーザプログラム 14
3.2 ISIとユーザプログラム間のデータの受渡し 16
3.3 ユーザプログラム呼出しの概要 16
3.4 ユーザプログラムの作成 17
3.4.1 クラスの作成 18
3.4.2 定義の設定 18
3.4.3 アプリケーションのコンパイル 23
3.4.4 jarファイルの配置 24
3.5 ジャンプ用メディエータファンクションの開発 24
3.5.1 ユーザプログラムによる設定 24
3.5.2 メディエータファンクションの復帰値による設定 24
3.5.3 ジャンプ用メディエータファンクション実装例 25
3.6 ファイルベースシーケンス実行用メディエータファンクションの開発 26

第4章 アダプタの開発 27
4.1 ISI呼び出しをするカスタムアダプタの開発 27
4.1.1 サービスエンドポイント定義の作成 27
4.1.2 アプリケーションの作成 27
4.1.3 実行時の設定 30
4.2 サービス提供側アプリケーションを呼び出すカスタムアダプタの開発 34
4.2.1 サービスエンドポイント定義の作成 34
4.2.2 リソースアダプタの開発 34
4.2.3 リソースアダプタの配備 37

第5章 変換定義の開発 39
5.1 Formatmanagerクライアントによる変換定義登録 40
5.1.1 変換定義登録の手順 40
5.1.2 変換定義の関連について 42
5.1.3 Formatmanagerクライアントの環境設定 45
5.1.4 フォーマット登録について 45
5.1.5 フォーマット変換登録について 45

5.1.6 運用情報登録について....46
5.1.7 シミュレーションについて....46

第6章 アダプタ利用時の定義の開発....47
6.1 SOAPアダプタ利用時の定義....47
6.1.1 Oneway方式によるSOAPメッセージ受信....49
6.1.2 RequestResponse方式によるSOAPメッセージ受信....49
6.1.3 Oneway方式によるSOAPメッセージ送信....50
6.1.4 RequestResponse方式によるSOAPメッセージ送信....51
6.1.5 添付ファイルを操作する....52
6.2 CORBAアダプタ利用時の定義....55
6.2.1 RequestResponse方式による呼び出し....56
6.2.2 Oneway方式による呼び出し....57
6.3 データベースアダプタ利用時の定義....58
6.3.1 データベースの更新....59
6.3.2 データベースの検索....60
6.3.3 ストアドプロシージャの実行....61
6.3.4 バイナリデータの使用....63
6.3.5 データベースアダプタでアクセス可能なデータ型....64
6.3.6 代替文字への変換ついて....67
6.4 MQアダプタ利用時の定義....67
6.4.1 WebSphere MQのキューからメッセージを受信する....67
6.4.2 WebSphere MQのキューへメッセージを送信する....68
6.4.3 WebSphere MQのキューとRequestResponse方式で通信する....68
6.4.4 MQ連携時のメッセージマッピング....70

付録A 定義の概要....76
A.1 定義の種類....76
A.1.1 サービスエンドポイント定義....77
A.1.2 シーケンス定義....78
A.1.3 メディエータファンクション定義....79
A.1.4 ルーティング定義....80
A.1.5 キュー間転送定義....82
A.2 定義の作成と登録....83
A.2.1 定義の登録....84
A.2.2 定義の検証....84
A.2.3 定義の配置....84
A.2.4 定義の管理....84

付録B 定義の詳細....86
B.1 サービスエンドポイント定義の詳細....86
B.1.1 EndpointList....86
B.1.1.1 Endpoint....86
B.2 サービス情報の詳細....91
B.2.1 Service....91
B.3 シーケンス定義の詳細....94
B.3.1 Sequence....94
B.4 プロパティ情報の詳細....96
B.4.1 PropertyList....96
B.4.1.1 Property....96
B.5 シーケンスブロックの詳細....97
B.5.1 Block....97
B.6 実行ステップ情報の詳細....99
B.6.1 StepInformation....100
B.7 条件分岐ブロック情報の詳細....105
B.7.1 ConditionBranchInformation....105
B.8 メディエータファンクション定義の詳細....108
B.8.1 MediatorFunctionList....109

B.8.1.1 MediatorFunction....109
B.9 ルーティング定義の詳細....115
B.9.1 MessageDistribution....115
B.10 グループ定義の詳細....121
B.10.1 GroupList....121
B.10.1.1 Group....121
B.11 ヘッダ伝送定義の詳細....124

付録C 定義の例....126
C.1 サービスエンドポイント定義....126
C.1.1 サービスエンドポイント定義の構成....126
C.1.2 SOAPのメッセージング機能の定義....127
C.1.2.1 インバウンド非同期....127
C.1.2.2 インバウンド同期....128
C.1.2.3 アウトバウンド非同期....129
C.1.2.4 アウトバウンド同期....134
C.1.3 ファイル転送の定義....138
C.1.3.1 受信....139
C.1.3.2 送信....143
C.1.4 JMS連携の定義....145
C.1.5 ListCreator連携の定義....146
C.1.6 CORBA連携の定義....148
C.1.6.1 インバウンド....148
C.1.6.2 アウトバウンド....149
C.1.7 MQ連携の定義....152
C.1.7.1 インバウンド....152
C.1.7.2 アウトバウンド(Oneway方式)....153
C.1.7.3 アウトバウンド(RequestResponse方式)....154
C.1.8 MQD連携の定義 157
C.1.8.1 受信....157
C.1.8.2 送信....158
C.1.9 キュー間転送の定義....159
C.1.9.1 受信....159
C.1.9.2 送信....162
C.1.9.3 ISI同士のキュー間転送....164
C.1.10 データベース連携の定義....166
C.1.11 III連携の定義....170
C.1.11.1 インバウンド....170
C.1.11.2 アウトバウンド....171
C.1.12 互換機能の定義....172
C.1.12.1 メッセージ送受信APIの定義....172
C.1.12.2 SOAPのメッセージング機能の定義....177
C.1.12.3 SOAP(.NET)の定義....184
C.1.12.4 JMS-R連携の定義....185
C.1.12.5 JCA連携の定義....188
C.2 シーケンス定義....190
C.2.1 シーケンス定義の構成....190
C.2.1.1 プロパティ情報....193
C.2.1.2 シーケンスブロック....193
C.2.2 サービス呼出しの定義....193
C.2.2.1 同期サービス呼出し....193
C.2.2.2 ルーティングして同期サービス呼出し(ルーティング定義を利用しない)....194
C.2.2.3 ルーティングして同期サービス呼出し(ルーティング定義を利用)....197
C.2.2.4 非同期サービス呼出し....198
C.2.2.5 ルーティングして非同期サービス呼出し(ルーティング定義を利用しない)....199
C.2.2.6 ルーティングして非同期サービス呼出し(ルーティング定義を利用)....201
C.2.2.7 複数の非同期サービス呼出し....202

C.2.3 XML Validationの定義....204
C.2.4 XML Operationの定義....205
C.2.4.1 追加(固定値)....206
C.2.4.2 追加(XMLデータから情報取得)....207
C.2.4.3 挿入....209
C.2.4.4 更新(固定値)....210
C.2.4.5 更新(XMLデータから情報取得)....211
C.2.4.6 削除....213
C.2.5 XML Emergerの定義....213
C.2.6 FLAT Emergerの定義....215
C.2.7 SOAP Operationの定義....216
C.2.7.1 SOAPメッセージの作成....216
C.2.7.2 SOAPメッセージの作成(SOAPボディだけ)....217
C.2.7.3 SOAPヘッダ要素を取得....218
C.2.7.4 SOAPボディ要素を取得....219
C.2.7.5 XML文書にSOAPタグを追加....220
C.2.7.6 SOAPメッセージからSOAPタグを削除....221
C.2.8 XML Converterの定義....222
C.2.9 Formatmanagerの定義....223
C.2.9.1 通常のフォーマット変換....223
C.2.9.2 複数フォーマット入力を使用したフォーマット変換....224
C.2.10 BPM Analytics連携の定義....225
C.2.10.1 XMLデータのモニタリング定義....225
C.2.10.2 FLATデータのモニタリング定義....227
C.2.11 List Creator連携の定義....229
C.2.12 Command Launcherの定義....231
C.2.13 File Operatorの定義....232
C.2.13.1 ファイル読込み....232
C.2.13.2 ファイル出力....234
C.2.14 ユーザプログラム呼出しの定義....235
C.2.15 条件分岐ブロック情報の定義....236
C.2.16 互換機能の定義....238
C.2.16.1 メッセージ送信の定義(互換定義)....238
C.2.16.2 SOAP Gatewayの定義(互換定義)....245
C.2.16.3 JCA Adapterの定義（互換定義）....246
C.3 メディエータファンクション定義....247
C.3.1 メディエータファンクション定義の構成....247
C.4 ルーティング定義....249
C.4.1 ルーティング定義の構成....249

付録D 互換機能....253
D.1 サービス提供側のアプリケーションの設計....253
D.1.1 非同期通信を行うアプリケーション(JMSアプリケーション)....253
D.1.2 MQDを使用するアプリケーション....253
D.2 サービス利用側のアプリケーションの設計（共通APIを使用する場合）....253
D.2.1 RMIの利用....253
D.2.2 JMSの利用....254
D.2.3 SOAPのRPC機能の利用....254
D.2.4 SOAP(.NET)の利用....255
D.2.5 JMS-Rの利用....256
D.2.6 ローカルDBとのトランザクション機能の利用....257
D.2.7 MQDの利用....257
D.3 共通APIを利用したサービス利用側アプリケーションの開発....257
D.3.1 RMIの利用....257
D.3.2 JMS,JMS-Rの利用....258
D.3.3 SOAPのRPC機能(JAX-RPC)の利用....260
D.3.4 .NETの利用....263

D.3.5 ローカルDBとのトランザクション機能の利用....264
D.4 スタブを利用したサービス利用側アプリケーションの開発....268
D.4.1 Interstage Application Server....268
D.4.2 AXIS....276
D.4.3 アプリケーションの移行....283
D.5 共通メッセージの作成・操作....285
D.5.1 ヘッダの作成・操作....285
D.5.2 ISI処理結果の確認....286
D.5.3 ペイロードの作成・操作....287
D.5.4 共通メッセージの作成・操作(.NET)....289
D.6 アプリケーションのコンパイルと実行....289
D.7 メディエータファンクションの開発....296
D.7.1 .NETユーザプログラム呼出しの概要....296
D.7.2 .NETユーザプログラムの作成....297
D.8 サンプルプログラム....302
D.8.1 ESIAPISendTest.java(送信処理)....303
D.8.2 ESIAPISendTest.cs(送信処理)....304
D.8.3 ESIAPIReceiveTest.java(非同期受信処理)....306

付録E BPEL機能のアプリケーション開発....308
E.1 アプリケーション開発の流れ....308
E.2 Webサービスの開発....308
E.2.1 Webサービスの作成....308
E.2.2 Webサービスの配備....309
E.2.3 WebサービスではないサービスのWebサービス化....309
E.3 BPEL定義の開発....309
E.3.1 WebサービスのWSDL取得....309
E.3.2 BPEL定義の作成....310
E.3.3 ビジネス・プロセス実行環境への配備....310
E.3.4 ビジネス・プロセス実行環境のチューニング....312
E.3.5 ビジネス・プロセス実行環境のログ設定....317
E.3.6 ビジネス・プロセス実行環境からの配備解除....322
E.4 BPEL起動アプリケーションの開発....323
E.4.1 BPEL定義のWSDL取得....324
E.4.2 BPEL起動アプリケーションの作成....324
E.5 BPEL定義のシミュレーション実行....324
E.6 BPELエディタによるBPEL定義の作成方法....327
E.6.1 BPEL定義作成の流れ....327
E.6.2 Interstage Studioの起動....328
E.6.3 プロジェクトの作成....328
E.6.4 BPELプロセスファイルの作成....330
E.6.5 WSDL定義の編集....335
E.6.6 BPEL定義の作成/編集....344
E.6.7 Webアプリケーション環境定義ファイル（deployment descriptor）の作成....387
E.6.8 BPEL起動アプリケーション開発者へのWSDL定義の提供....391

索引....392

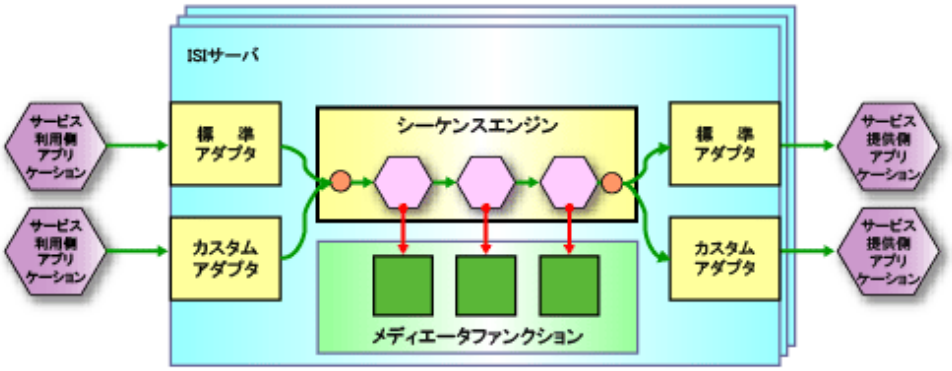
第1章 アプリケーション開発の概要

ここでは、ISIのアプリケーション開発の概要について説明します。
ISIで開発するアプリケーションには、以下の4種類があります。

- サービス提供側のアプリケーション
- サービス利用側のアプリケーション
- メディエータファンクション
  ISIが標準で提供するメディエータファンクションにはない機能が必要な場合に作成します。
- カスタムアダプタ
  ISIが標準で提供するアダプタでサポートしていないプロトコルを利用して、サービス利用側アプリケーションやサービス提供側アプリケーションと接続したい場合に作成します。

ISIと上記の各アプリケーション関係は、以下のようになります。
サービス提供側のアプリケーションとサービス利用側のアプリケーションは、標準アダプタ、またはカスタムアダプタを利用してISIサーバと接続します。サービス利用側のアプリケーションは、シーケンスエンジンを経由してサービス提供側のアプリケーションを呼び出します。シーケンスエンジンでは、シーケンス定義に定義された順番で各種のメディエータファンクションを利用してサービス提供側のアプリケーションとサービス利用側のアプリケーションのインタフェースを調整します。

図1.1 ISIで開発するアプリケーション



1.1 サービス提供側のアプリケーションの設計

サービス提供側のアプリケーションには、以下の種類があります。

表1.1 サービス提供側のアプリケーションの種類

接続方式	アプリケーション	概要
SOAP	Webサービスアプリケーション	Webサービスに接続して、メッセージ送信や取得を行います。
DB	データベース	データベースのテーブルに対してデータの取得・更新を行います。
FTP	FTPサーバ	FTPサーバに対してデータを送信します。
CORBA	CORBAアプリケーション EE	CORBAのサーバアプリケーションを呼び出します。
MQ	WebSphere MQ EE	WebSphere MQのキューにメッセージを送信します。
III連携	Interstage Information Integrator（以降、IIIと略します）の処理プロセス	IIIの処理プロセスのデータ収集ファンクションに対してデータを渡します。

ISIサーバ間連携を行う場合は、キュー間転送を利用することができます。

上記以外のアプリケーションと接続する場合は、カスタムアダプタを開発することでISIとの接続が可能です。カスタムアダプタの開発については、“第4章 アダプタの開発”を参照してください。

1.1.1 Webサービスアプリケーション

ISIのシーケンスからWebサービスアプリケーションに接続して、メッセージ送信や取得を行います。
Webサービスアプリケーションの設計は、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”の“Interstage Webサービスの機能”、または“Interstage Application Server/Interstage Web Server Express Java EE運用ガイド”の“Java EEアプリケーションの提供機能”を参照してください。

1.1.2 データベース

ISIのシーケンスから業務で利用するデータベースと連携し、データの抽出や登録を行います。
以下のデータベース製品に接続することができます。

- Symfoware Server
- Oracle
- SQL Server
- FUJITSU Integrated System HA Database Ready

データベースの設計は、各データベース製品のマニュアルを参照してください。

1.1.3 FTPサーバ

ISIのシーケンスからFTPサーバへファイルを転送します。
FTPサーバの設定については、各FTPサーバのマニュアルを確認してください。

1.1.4 CORBAアプリケーション EE

ISIのシーケンスからCORBAサーバのCORBAのサーバアプリケーションを呼び出します。
COARBAアプリケーションの設計は、“Interstage Application Server アプリケーション作成ガイド（CORBAサービス編）”の“CORBAアプリケーションの設計”を参照してください。

1.1.5 WebSphere MQ EE

ISIのシーケンスからWebSphere MQのキューにメッセージを送信します。
WebSphere MQの設定については、“ISI 導入ガイド”の“WebSphere MQのキューの設定”およびWebSphere MQの関連マニュアルを参照してください。

1.1.6 III連携

ISIのシーケンスから、IIIの処理プロセスにデータを渡します。IIIは、ISIから渡されたデータをデータ収集ファンクションで取得します。

ISIのアウトバウンドサービスエンドポイント定義に指定する「プロセスID」および「データボックス名」は、IIIのプロセス定義の「プロセスID」およびデータ収集定義の「データボックス」と一致させる必要があります。

IIIの定義方法については、IIIの「システム設計ガイド」やヘルプを参照してください。

1.2 サービス利用側のアプリケーションの設計

サービス利用側のアプリケーションには、以下の種類があります。

表1.2 サービス利用側のアプリケーションの種類

接続方法	アプリケーション	概要
SOAP	Webサービスクライアント	WebサービスクライアントからSOAPを使用してISIサーバに接続します。

接続方法	アプリケーション	概要
CORBA	CORBAクライアントアプリケーション EE	CORBAクライアントアプリケーションからISIサーバに接続します。
MQ	WebSphere MQ EE	WebSphere MQのキューからメッセージを受信します。
III連携	IIIの処理プロセス	IIIの処理プロセスのデータ配付ファンクションにからデータを受け取ります。

ISIサーバ間連携を行う場合は、キュー間転送を利用することができます。

上記以外のアプリケーションから接続する場合は、カスタムアダプタを開発することでISIと接続できます。カスタムアダプタの開発については、“第4章 アダプタの開発”を参照してください。

1.2.1 Webサービスクライアント

ISIのシーケンスをWebサービスとして公開し、Webサービスクライアントから ISIのシーケンスを呼び出します。
Webサービスクライアントから SOAPを利用して ISIの呼び出しを行う場合の開発については、“2.1 SOAPを利用してISI呼び出しを行うアプリケーションの開発”を参照してください。

1.2.2 CORBAクライアントアプリケーション EE

ISIのシーケンスをCORBAクライアントアプリケーションから呼び出します。
CORBAクライアントアプリケーションからISIの呼び出しを行う場合の開発については、“2.2 CORBAを利用してISI呼び出しを行うアプリケーションの開発”を参照してください。

1.2.3 WebSphere MQ EE

WebSphere MQのキューからメッセージを受信して、ISIのシーケンスを実行します。
WebSphere MQの設定については、“ISI 導入ガイド”の“WebSphere MQのキューの設定”およびWebSphere MQの関連マニュアルを参照してください。

1.2.4 III連携

IIIの処理プロセスのデータ配付ファンクションからデータを受け取り、ISIのシーケンスを実行します。

ISIのインバウンドサービスエンドポイント定義に指定する「プロセスID」は、IIIのプロセス定義の「プロセスID」と一致させる必要があります。また、IIIのデータ配付定義の「サービスエンドポイント」は、ISIのインバウンドサービスエンドポイント名と一致させる必要があります。

IIIの定義方法については、IIIの「システム設計ガイド」やヘルプを参照してください。

ポイント

III連携では、IIIからISIへのデータの受け渡し方法としてメモリベースかファイルベースかを選択できます。ファイルベースを選択した場合、ISIのシーケンスもファイルベースで動作します。

1.3 メディエータファンクションの開発

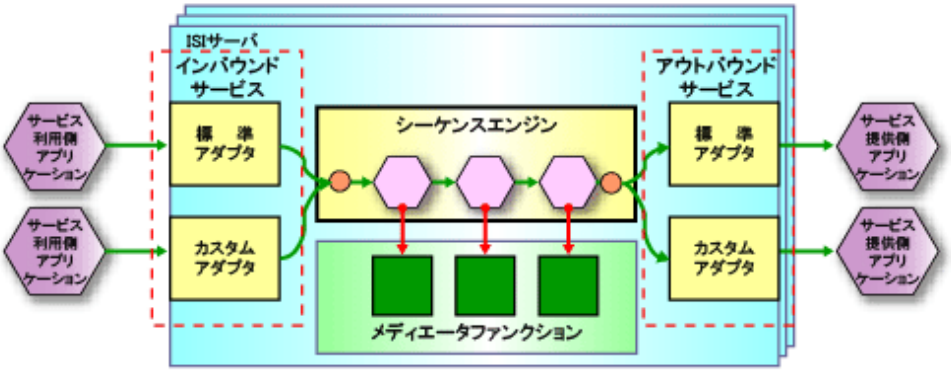
標準提供のメディエータファンクション以外に、Javaでプログラムを作成し、ユーザプログラム呼出しの機能を利用して、シーケンスエンジンから呼び出すことができます。
共通メッセージのヘッダパラメタやペイロードを、ユーザプログラムのパラメタにセットし、復帰値を共通メッセージに格納できます。
ユーザプログラムを作成すると、パラメタ、復帰値の型を自由に設定できます。

1.4 カスタムアダプタの設計

多様なアプリケーションと接続を行うために、カスタムアダプタとして独自にアダプタを提供することができます。カスタムアダプタは、Javaアプリケーションとして開発します。

カスタムアダプタは、サービス利用側アプリケーションおよびサービス提供側アプリケーションと接続するために、インバウンドサービスとアウトバウンドサービスの2種類があります。

図1.2 カスタムアダプタ



1.4.1 インバウンドサービス

サービス利用側アプリケーションとISIサーバを接続するためのアダプタを開発します。

インバウンドサービスのカスタムアダプタは、ISIが提供する、J2EE Connection Architectureのリソースアダプタを呼び出すことでISIサーバに接続します。

インバウンドサービスのカスタムアダプタの開発については、“4.1 ISI呼び出しをするカスタムアダプタの開発”を参照してください。

1.4.2 アウトバウンドサービス

サービス提供側アプリケーションとISIサーバを接続するためのアダプタを開発します。

アウトバウンドサービスのカスタムアダプタは、J2EE Connection Architectureのリソースアダプタとして提供する必要があります。

アウトバウンドサービスのカスタムアダプタの開発については、“4.2 サービス提供側アプリケーションを呼び出すカスタムアダプタの開発”を参照してください。

第2章 サービス利用側のアプリケーションの開発

ここでは、サービス利用側のアプリケーションの開発について説明します。

2.1 SOAPを利用してISI呼び出しを行うアプリケーションの開発

SOAPを利用してISI呼び出しを行うアプリケーションは、次の条件を満たすWebサービスクライアントとして作成します。

- WS-I Basic Profile 1.0に準拠していること
- 添付ファイルを扱う場合、WS-I Attachment Profile 1.0に準拠していること

  WSDLエクスポート機能を利用すれば、XMLスキーマファイルをもとにISIで作成したSOAPメッセージ受信用定義に対応したWSDLを自動で生成できるため、容易にISI呼び出しを行うアプリケーションを作成できます。WSDLエクスポート機能の利用方法は、“ISI Studio ヘルプ”の“サービスエンドポイントからWSDLを生成する”を参照してください。

Webサービスクライアントがすでに存在する場合

アプリケーションの開発は不要で、送信先URLをISIのSOAPアダプタにすることでISIを呼び出すことができます。送信先URLは以下の形式になります。

- サービスエンドポイント名指定によるISI呼出し

  サービス利用側からURLの後ろ(斜体部分)のクエリストリングでサービスエンドポイント名を付加した送信先に、SOAPメッセージを送信します。
  http://ホスト名:ポート番号/esisoap/ESISoapMsgServer*?サービスエンドポイント名*

  例)
  ホスト名 : MyHost
  ポート番号 : 省略(80)
  サービスエンドポイント名 : endpoint01
  http://MyHost/esisoap/ESISoapMsgServer?endpoint01

- インバウンドルーティングを行うISI呼出し

  サービス利用側からクエリストリングなしのURLにSOAPメッセージを送信します。
  http://ホスト名:ポート番号/esisoap/ESISoapMsgServer

  例)
  ホスト名 : MyHost
  ポート番号 : 省略(80)
  http://MyHost/esisoap/ESISoapMsgServer

Webサービスクライアントを開発する場合

ISIのWSDLエクスポート機能でWSDLファイルを生成し、WSDLファイルからWebサービスクライアントを生成する機能を用いて開発を行います。WSDLファイルからWebサービスクライアントを生成する機能はInterstage Application Serverの“iswsgen”コマンドやApache Axisなどがあります。

エラー発生時の動作

ISIサーバでエラーが発生した場合、以下の形式のSOAPFaultを返却します。

項目名	設定値
faultcode	soapenv:Server
faultstring	エラー原因

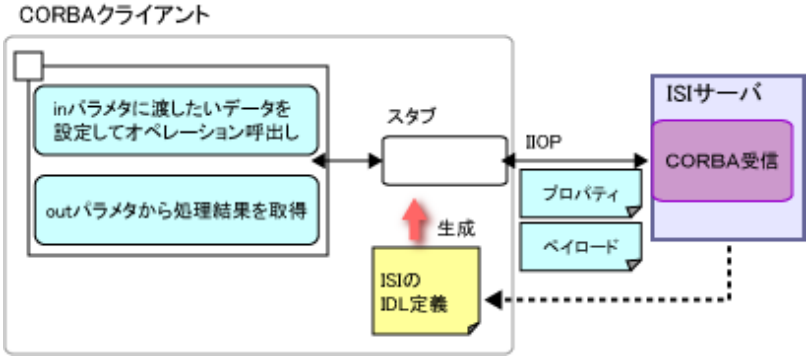
SOAPFaultの例を示します。

```
<soapenv:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
   <soapenv:Fault>
        <faultcode>soapenv:Server</faultcode>
        <faultstring>errorcode=SOAP-MSG-41002, message=サービスエンドポイント情報が取得できませんでした。[サービスエン
ドポイント名=xxx]</faultstring>
   </soapenv:Fault>
 </soapenv:Body>
</soapenv:Envelope>
```

2.2 CORBAを利用してISI呼び出しを行うアプリケーションの開発EE

アプリケーション開発時は、スタブに対して必要な項目を設定します。スタブは、ISIが準備したIDL定義から生成します。
inタイプのパラメタには、共通メッセージのペイロードに格納するデータおよびプロパティ情報を指定します。
オペレーション呼出し後の処理結果は、outタイプのパラメタに設定されます。

図2.1 CORBAクライアントアプリケーション利用時のイメージ

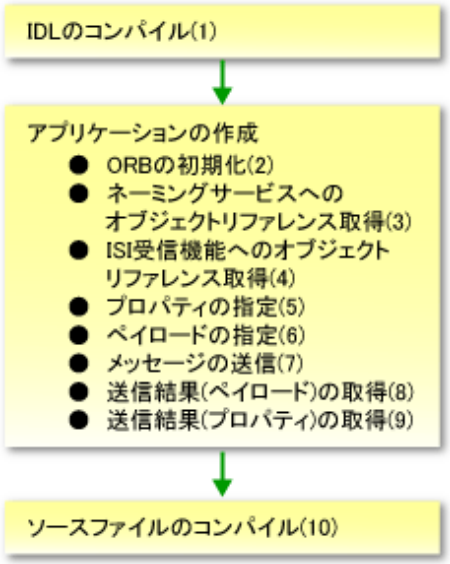


ISIサーバでエラーが発生した場合でも、CORBAクライアントアプリケーションへ正常復帰します。
ISIサーバでのエラーをユーザ例外としてエラー情報(エラーコード、エラーメッセージ、およびプロパティ情報)を通知する動作に変更することも可能です。
詳細は、“ISI 導入ガイド”の“環境設定ファイルの設定”を参照してください。

例をもとに、CORBA利用時にスタブを利用するアプリケーションの開発手順を説明します。

アプリケーションの開発は、以下の手順で行います。

図2.2 開発の手順



前提

作成するソースファイル名：ESICorbaClient.c
作成するアプリケーション名：ESICorbaClient
CORBA用のインバウンドルーティング定義は設定済み。

インバウンドルーティング定義の作成方法および設定方法については、“ISI Studio ヘルプ”、“ISI 導入ガイド”を参照してください。

手順

1. IDLをコンパイルします。
   ISIは、サービス利用側のアプリケーションに、以下のIDL定義を提供します。

   IDLファイル：“<INSDIR>¥etc¥idl¥corbaservice.idl”

   IDLファイル：“/opt/FJSVesi/etc/idl/corbaservice.idl”

```
module esi {
    interface corbaservice {

        //
        // プロパティ情報
        //
        struct ESIProperty {
            string      name;
            string      value;
        };

        //
        // 型宣言
        //
        typedef sequence<ESIProperty> ESIProperties;
        typedef sequence<octet> ESIPayload;

        //
        // 例外
        //
        exception ESIError {
            string      errorCode;
            string      errorMessage;
            ESIProperties outProperties;
        };

        //
        // メッセージ送信
        //
        void sendMessage(
            in   ESIPayload  inObject,
            in   ESIProperties inProperties,
            out  ESIPayload  outObject,
            out  ESIProperties outProperties)
            raises(ESIError);
    };
};
```

   IDLファイルを作業ディレクトリにコピー後、IDLコンパイラを使用して、クライアントスタブを自動生成します。

```
IDLc -lc corbaservice.idl
```

注意

ISIが提供するIDL定義は変更しないでください。

2. ORBを初期化します。
3. ネーミングサービスへのオブジェクトリファレンスを取得します。
4. インバウンドのCORBAアダプタのオブジェクトリファレンスを取得します。
   オブジェクト名は、“ESICORBARcv”です。
5. 必要に応じて共通メッセージに格納するプロパティ情報を指定します。
   この時、ヘッダのキー名“com.fujitsu.esi.endpoint.inbound”にCORBAのサービスエンドポイント名を指定することも可能です。
6. 共通メッセージに格納するペイロードを指定します。
7. メッセージを送信します。
8. 送信結果(ペイロード)を取得します。
9. 送信結果(プロパティ)を取得します。
   取得できる代表的なプロパティには、以下の情報があります。

表2.1 キー名と取得情報

キー名	情報
com.fujitsu.esi.status	ISIの処理結果です。 ・ SUCCESS 処理が正常終了した場合 ・ FAILURE 処理中にエラーが発生した場合
com.fujitsu.esi.error.code	ISIの処理中に最初に発生したエラーのエラーコードです。
com.fujitsu.esi.error.stepname	エラーが発生したステップです。 ISIの初期処理中にエラーが発生した場合は、nullになります。
com.fujitsu.esi.error.methodtype	エラーが発生したメソッド種別です。 ISIの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。
com.fujitsu.esi.error.handle.status	ISI処理のエラー発生時に呼び出したシーケンスの処理結果です。 ・ SUCCESS 処理が正常終了した場合 ・ FAILURE 処理中にエラーが発生した場合
com.fujitsu.esi.error.handle.code	エラー発生時に呼び出したシーケンスの中で、最初に発生したエラーのエラーコードです。
com.fujitsu.esi.error.handle.stepname	エラー発生時に呼び出したシーケンスの中で、エラーが発生したステップです。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。

キー名	情報
com.fujitsu.esi.error.handle.methodtype	エラー発生時に呼び出したシーケンスの中で、エラーが発生したメソッド種別です。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。

10. 作成したソースをコンパイル・リンクします。
    ESICorbaClient.c、corbaservice_stub.cのソースをコンパイル・リンクします。
    コンパイル・リンク時の注意事項は、“Interstage Application Server アプリケーション作成ガイド(CORBAサービス編)”を参照してください。

アプリケーションの例:サンプルプログラム“ESICorbaClient.c”を例にあげて説明します。

```
#include <stdio.h>
#include "orb.h"
#include "OM_imr.h"
#include "CosNamin.h"
#include "corbaservice.h"

#define CORBASV_OBJNAME "ESICORBARcv"              /* オブジェクトリファレンス名 */
#define PAYLOADDATA     "This is sample message."  /* 送信メッセージ             */


int main( int  argc, char**   argv )
{
  CORBA_ORB                Orb;
  CosNaming_NamingContext  CosNaming;
  CORBA_Object             ESICorbaRcvObj; /* CORBAアダプタのオブジェクトリファレンス */
  CORBA_Environment        Env;            /* CORBAエラー情報                         */
  CosNaming_Name           NSName;         /* NS用Name                                */
  CosNaming_NameComponent  NSComp;         /* NS用Nameコンポーネント                  */

  /* 送信用データ */
  esi_corbaservice_ESIProperties  InESIProperties;
  esi_corbaservice_ESIProperty    InESIPropertyArray[10];
  esi_corbaservice_ESIPayload     InESIPayload;

  /* 受信用データ */
  esi_corbaservice_ESIProperties* pOutESIProperties;
  esi_corbaservice_ESIPayload*    pOutESIPayload;
  esi_corbaservice_ESIProperty*   pOutESIProperty;

  int               i;
  char              wBuff[4096];

  /*
   * ORBの初期化(2)
   */
  Orb = CORBA_ORB_init( &argc,
              argv,
              FJ_OM_ORBid,
              &Env );
  if( Env._major != CORBA_NO_EXCEPTION ) {
    printf( "ERROR : CORBA_ORB_init()" );
    return -1;
```

```
}

/*
 * ネーミングサービスへのオブジェクトリファレンスを取得(3)
 */
CosNaming = CORBA_ORB_resolve_initial_references(
                Orb,
                CORBA_ORB_ObjectId_NameService,
                &Env );
if( Env._major != CORBA_NO_EXCEPTION ) {
  printf( "ERROR : CORBA_ORB_resolve_initial_references()" );
  return -1;
}

/*
 * インバウンドのCORBAアダプタへのオブジェクトリファレンスを取得(4)
 */
NSName._length  = 1;
NSName._maximum = 1;
NSName._buffer  = &NSComp;
NSComp.id   = CORBASV_OBJNAME;
NSComp.kind = "";
ESICorbaRcvObj = CosNaming_NamingContext_resolve( CosNaming,
                          &NSName,
                          &Env );
if( Env._major != CORBA_NO_EXCEPTION ) {
  printf( "ERROR : CosNaming_NamingContext_resolve()" );
  CORBA_Object_release( CosNaming , &Env );
  return -1;
}

/*
 * プロパティの指定(5)
 */
InESIProperties._length  = 1;
InESIProperties._maximum = 1;
InESIProperties._buffer  = InESIPropertyArray;
memset( &InESIPropertyArray, 0x00, sizeof( InESIPropertyArray ) );
InESIPropertyArray[0].name  = "category";
InESIPropertyArray[0].value = "A01";

/*
 * ペイロードの指定(6)
 */
InESIPayload._maximum = strlen( PAYLOADDATA );
InESIPayload._length  = strlen( PAYLOADDATA );
InESIPayload._buffer  = (unsigned char*)PAYLOADDATA;

/*
 * メッセージの送信(7)
 */
esi_corbaservice_sendMessage( ESICorbaRcvObj,
                &InESIPayload,
                &InESIProperties,
                &pOutESIPayload,
                &pOutESIProperties,
                &Env );
if( Env._major != CORBA_NO_EXCEPTION ) {
  printf( "ERROR : esi_corbaservice_sendMessage()" );
  CORBA_Object_release( CosNaming , &Env );
  CORBA_Object_release( ESICorbaRcvObj , &Env );
  CORBA_exception_free( &Env );
  return -1;
```

```
  }

  /*
   * 送信結果(ペイロード)の取得(8)
   */
  memset( wBuff, 0x00, sizeof( wBuff ) );
  memcpy( wBuff, pOutESIPayload->_buffer, pOutESIPayload->_length );
  printf( "payload : [%s]¥n", wBuff );

  /*
   * 送信結果(プロパティ)の取得(9)
   */
  printf( "property : ¥n" );
  for( i = 0; i < pOutESIProperties->_length; i ++ ) {
    pOutESIProperty = pOutESIProperties->_buffer;
    pOutESIProperty += i;
    printf( "  %s : [%s]¥n",
    pOutESIProperty->name, pOutESIProperty->value );
  }

  CORBA_free(pOutESIProperties);
  CORBA_free(pOutESIPayload);
  CORBA_Object_release( ESICorbaRcvObj , &Env );
  CORBA_Object_release( CosNaming , &Env );
  CORBA_exception_free( &Env );

  return 0;

}
```

S L

```
#include <stdio.h>
#include "orb.h"
#include "OM_impl_rep.h"
#include "CosNaming.h"
#include "corbaservice.h"

#define CORBASV_OBJNAME "ESICORBARcv"              /* オブジェクトリファレンス名 */
#define PAYLOADDATA     "This is sample message."  /* 送信メッセージ             */


int main( int  argc, char**   argv )
{
  CORBA_ORB                Orb;
  CosNaming_NamingContext  CosNaming;
  CORBA_Object             ESICorbaRcvObj; /* CORBAアダプタのオブジェクトリファレンス */
  CORBA_Environment        Env;            /* CORBAエラー情報                         */
  CosNaming_Name           NSName;         /* NS用Name                                */
  CosNaming_NameComponent  NSComp;         /* NS用Nameコンポーネント                  */

  /* 送信用データ */
  esi_corbaservice_ESIProperties  InESIProperties;
  esi_corbaservice_ESIProperty    InESIPropertyArray[10];
  esi_corbaservice_ESIPayload     InESIPayload;

  /* 受信用データ */
  esi_corbaservice_ESIProperties* pOutESIProperties;
  esi_corbaservice_ESIPayload*    pOutESIPayload;
  esi_corbaservice_ESIProperty*   pOutESIProperty;
```

```
int               i;
char              wBuff[4096];

/*
 * ORBの初期化(2)
 */
Orb = CORBA_ORB_init( &argc,
            argv,
            FJ_OM_ORBid,
            &Env );
if( Env._major != CORBA_NO_EXCEPTION ) {
  printf( "ERROR : CORBA_ORB_init()" );
  return -1;
}

/*
 * ネーミングサービスへのオブジェクトリファレンスを取得(3)
 */
CosNaming = CORBA_ORB_resolve_initial_references(
                Orb,
                CORBA_ORB_ObjectId_NameService,
                &Env );
if( Env._major != CORBA_NO_EXCEPTION ) {
  printf( "ERROR : CORBA_ORB_resolve_initial_references()" );
  return -1;
}

/*
 * インバウンドのCORBAアダプタのオブジェクトリファレンスを取得(4)
 */
NSName._length  = 1;
NSName._maximum = 1;
NSName._buffer  = &NSComp;
NSComp.id   = CORBASV_OBJNAME;
NSComp.kind = "";
ESICorbaRcvObj = CosNaming_NamingContext_resolve( CosNaming,
                          &NSName,
                          &Env );
if( Env._major != CORBA_NO_EXCEPTION ) {
  printf( "ERROR : CosNaming_NamingContext_resolve()" );
  CORBA_Object_release( CosNaming , &Env );
  return -1;
}

/*
 * プロパティの指定(5)
 */
InESIProperties._length  = 1;
InESIProperties._maximum = 1;
InESIProperties._buffer  = InESIPropertyArray;
memset( &InESIPropertyArray, 0x00, sizeof( InESIPropertyArray ) );
InESIPropertyArray[0].name  = "category";
InESIPropertyArray[0].value = "A01";

/*
 * ペイロードの指定(6)
 */
InESIPayload._maximum = strlen( PAYLOADDATA );
InESIPayload._length  = strlen( PAYLOADDATA );
InESIPayload._buffer  = (unsigned char*)PAYLOADDATA;

/*
```

```
   * メッセージの送信(7)
   */
  esi_corbaservice_sendMessage( ESICorbaRcvObj,
                  &InESIPayload,
                  &InESIProperties,
                  &pOutESIPayload,
                  &pOutESIProperties,
                  &Env );
  if( Env._major != CORBA_NO_EXCEPTION ) {
    printf( "ERROR : esi_corbaservice_sendMessage()" );
    CORBA_Object_release( CosNaming , &Env );
    CORBA_Object_release( ESICorbaRcvObj , &Env );
    CORBA_exception_free( &Env );
    return -1;
  }

  /*
   * 送信結果(ペイロード)の取得(8)
   */
  memset( wBuff, 0x00, sizeof( wBuff ) );
  memcpy( wBuff, pOutESIPayload->_buffer, pOutESIPayload->_length );
  printf( "payload : [%s]¥n", wBuff );

  /*
   * 送信結果(プロパティ)の取得(9)
   */
  printf( "property : ¥n" );
  for( i = 0; i < pOutESIProperties->_length; i ++ ) {
    pOutESIProperty = pOutESIProperties->_buffer;
    pOutESIProperty += i;
    printf( "  %s : [%s]¥n",
    pOutESIProperty->name, pOutESIProperty->value );
  }

  CORBA_free(pOutESIProperties);
  CORBA_free(pOutESIPayload);
  CORBA_Object_release( ESICorbaRcvObj , &Env );
  CORBA_Object_release( CosNaming , &Env );
  CORBA_exception_free( &Env );

  return 0;

}
```

第3章 メディエータファンクションの開発

シーケンスエンジンからユーザが作成したプログラムをメディエータファンクションとして呼び出すことができます。ここでは、メディエータファンクションとして呼び出すユーザプログラムの開発方法について説明します。

3.1 呼出し可能なユーザプログラムの種類

ISIではJavaおよび.NETのユーザプログラムをメディエータファンクションとして呼び出すことができます。呼び出すことができるユーザプログラムの種類について説明します。



.NETのユーザプログラムは、Windowsの場合だけ呼び出すことができます。

3.1.1 Javaユーザプログラム

以下の条件を満たすJavaユーザプログラムをメディエータファンクションとして呼び出すことができます。

クラス

- アクセス修飾子がpublic
- アクセス修飾子がpublicの引数なしコンストラクタがある

メソッド

- アクセス修飾子がpublic

パラメタ

以下のデータ型のパラメタだけを使用している必要があります。

基本データ型

- boolean
- short
- int
- float
- long
- double

基本データ型のラッパクラス

- java.lang.Boolean
- java.lang.Byte
- java.lang.Character
- java.lang.Short
- java.lang.Integer
- java.lang.Float
- java.lang.Long
- java.lang.Double

String型

- java.lang.String

String配列

- java.lang.String[]

共通メッセージ

- com.fujitsu.esi.message.ESIMessage

byte配列型

- byte[]

復帰値

復帰値の型は、以下のどれかです。

基本データ型

- boolean
- short
- int
- float
- long
- double

基本データ型のラッパクラス

- java.lang.Boolean
- java.lang.Byte
- java.lang.Character
- java.lang.Short
- java.lang.Integer
- java.lang.Float
- java.lang.Long
- java.lang.Double

String型

- java.lang.String

共通メッセージ

- com.fujitsu.esi.message.ESIMessage

注意

ISI 8.0以前で作成したプログラムのために残しています。
共通メッセージにはシーケンス動作に必要な情報が含まれています。したがって、復帰値で共通メッセージを返却した場合、シーケンス動作に影響を及ぼすことがあります。新規でメディエータファンクション用ユーザプログラムを作成する際には使用しないでください。

byte配列型

- byte[]

復帰値なし

- void

3.2 ISIとユーザプログラム間のデータの受渡し

ISIからユーザプログラムへ受け渡すパラメタ値は、以下のとおりです。

- 基本データ型、基本データ型のラッパクラス、String型
  定義時に指定した値または共通メッセージのヘッダの値を受け渡します。
- byte配列型
  共通メッセージのペイロードの値を受け渡します。
- 共通メッセージ
  共通メッセージ全体を受け渡します。

ユーザプログラムからISIへ受け渡す復帰値は、以下のとおりです。

- 基本データ型、基本データ型のラッパクラス、String型
  共通メッセージのヘッダに復帰値をセットします。
- byte配列型
  共通メッセージのペイロードに復帰値をセットします。

また、共通メッセージをパラメタとして取得するユーザプログラムにおいて、共通メッセージを書き換えた場合、以降のシーケンスの処理では書き換えた共通メッセージが使用されます。

ポイント

ユーザプログラムでシーケンスをエラーで終了させたい場合は、ユーザプログラムのメソッド内で例外をスローしてください。

3.3 ユーザプログラム呼出しの概要

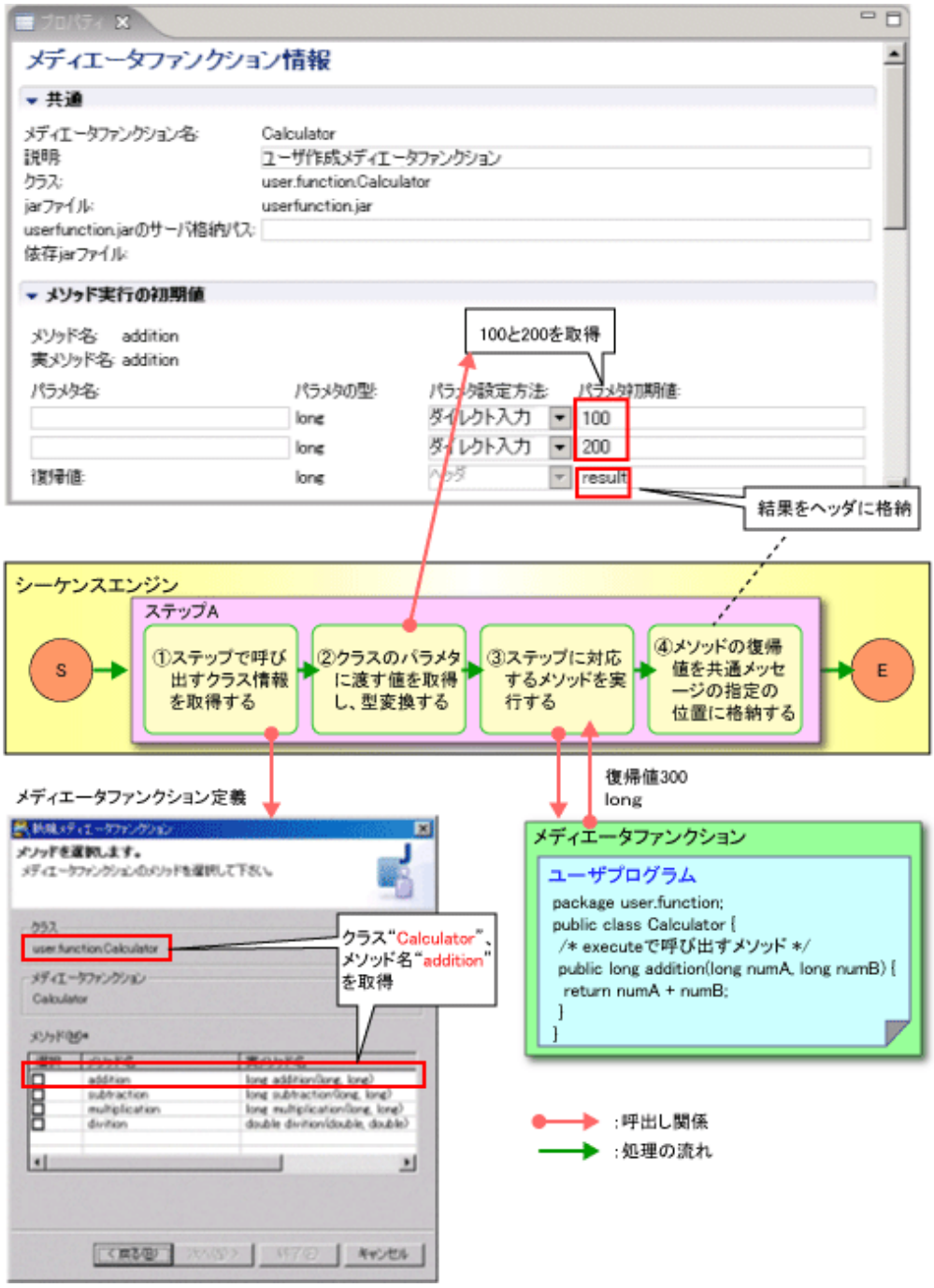
例をもとに、ユーザプログラム呼出しの概要について説明します。
この例では、以下の条件を満たすプログラムを作成します。

- メソッドの種別は、実行処理メソッド
- クラス名は、“Calculator”
- メソッド名は、“addition”
- メディエータファンクション名は、“Calculator”
- パラメタは、numAとnumB
- numAに“100”、numBに“200”を指定
- 2つのパラメタの合計値“300”を返却
- 結果は、共通メッセージのヘッダに格納

シーケンスエンジン内の処理の流れ

1. メディエータファンクション定義から呼び出すプログラムのクラス情報を取得
2. シーケンス定義からパラメタの値を取得し、型変換
3. メソッド“addition”を実行(numA＋numB)
4. 復帰値“300”を共通メッセージのヘッダに格納

図3.1 ユーザプログラムの呼出しイメージ



3.4 ユーザプログラムの作成

メディエータファンクションのユーザプログラムは、以下の手順で開発します。

1. メディエータファンクションクラスを作成します。

2. 作成したユーザプログラムの情報を、メディエータファンクション定義ファイルおよびシーケンス定義ファイルに定義します。
3. アプリケーションのコンパイルを行います。
4. 作成したクラスが含まれるjarファイルを固定のディレクトリに配備します。

“3.3 ユーザプログラム呼出しの概要”の例に従って、作成手順を以下に説明します。

3.4.1 クラスの作成

シーケンスエンジンから実行されるクラスを作成します。

表3.1 クラスの機能一覧

メソッドの種別	クラス名	メソッド名	パラメタ	復帰値
-	user.function.Calculator	-	-	-
execute	-	addition	long numA long numB	long

1. クラスとメソッドを作成します。以下のサンプルコードを作成します。
   指定例

```
package user.function;
public class Calculator {

        /* executeで呼び出すメソッド */
        public long addition(long numA, long numB) {
                return numA + numB;
       }
}
```

2. 作成したクラスをコンパイルします。classファイルが生成されます。
   コンパイル方法は、ご使用の開発環境、またはjavacコマンドの使用方法を確認してください。
3. 生成したclassファイルを含むjarファイルを作成します。jarファイルの生成には、JDKに同梱されているjarコマンドを使用します。

ポイント

- Interstage Studioを利用してメディエータファンクションのユーザプログラムを作成できます。
  使用するプロジェクトは、“Java プロジェクト”です。
- ユーザプログラムでシーケンスをエラーで終了させたい場合は、ユーザプログラムのメソッド内で例外をスローしてください。

3.4.2 定義の設定

作成したユーザプログラムの情報を、ISI Studioを利用しメディエータファンクションとして定義します。
ここでは、“3.4.1 クラスの作成”で作成したユーザプログラム“C:¥Temp¥userfunction.jar”を例に、メディエータファンクション作成とシーケンスからの呼出し定義までを説明します。

メディエータファンクションの作成

ユーザプログラムをメディエータファンクションとしてシーケンスから呼び出すため、メディエータファンクションの定義を作成します。

1. ISIナビゲータービューで以下のどちらかを操作します。

   ー メディエータファンクションのツリーをダブルクリック、手順2へ進みます。

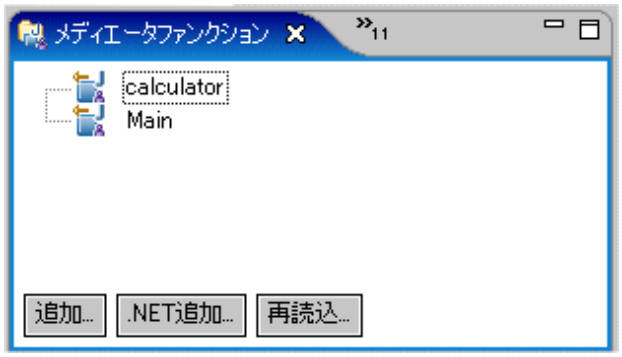
ー ISIプロジェクト配下のツリーを選択し、ポップアップメニューの[新規メディエータファンクション]をクリックし、手順3へ進みます。

図3.2 ISIナビゲータービューの画面例



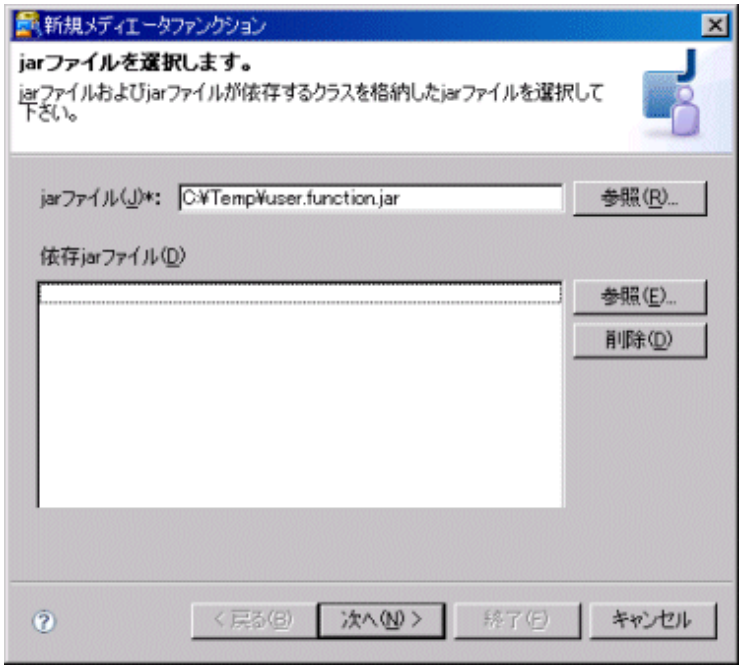
2. メディエータファンクションエディタで、[追加]ボタンをクリックし、手順3へ進みます。

図3.3 メディエータファンクションエディタの画面例



3. 作成したjarファイル“C:¥Temp¥userfunction.jar”を指定し、[次へ]ボタンをクリックします。

図3.4 [新規メディエータファンクション作成(1ページ目)]ダイアログボックスの画面例

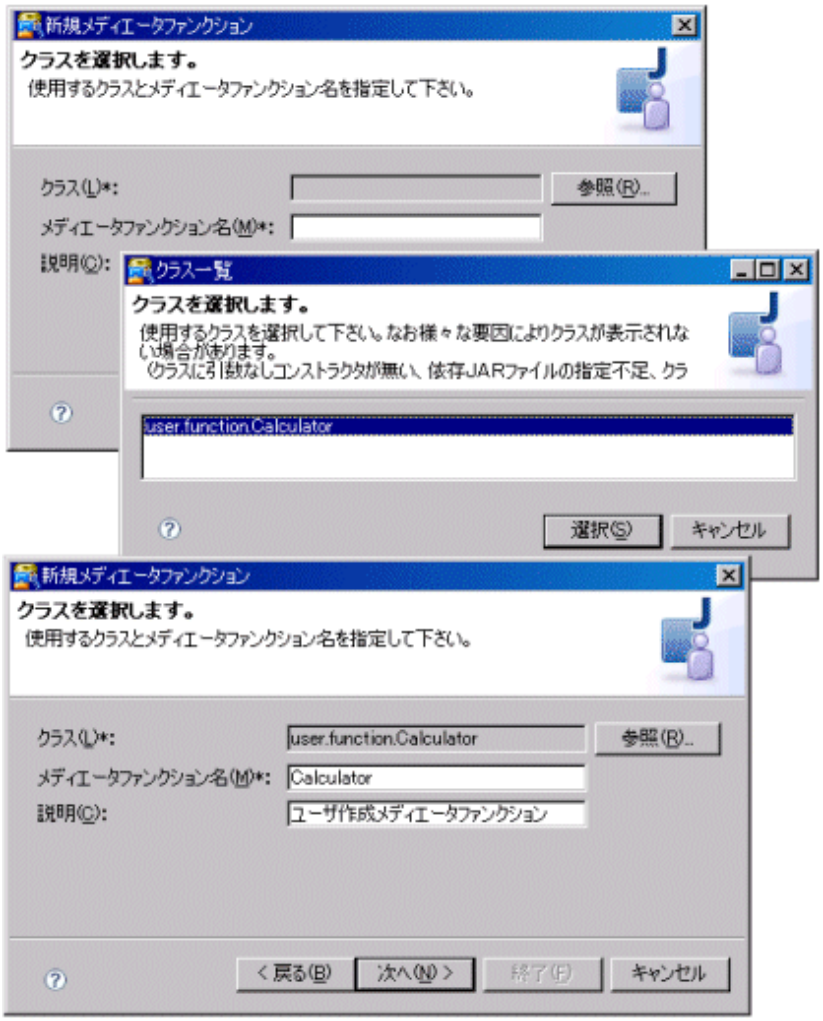


注意

使用するクラスに関連するjarファイルがある場合、関連するすべてのjarファイルを依存jarファイルに指定してください。指定がない場合、クラスの解析ができないため次の画面のクラス指定候補に使用するクラスが表示されません。

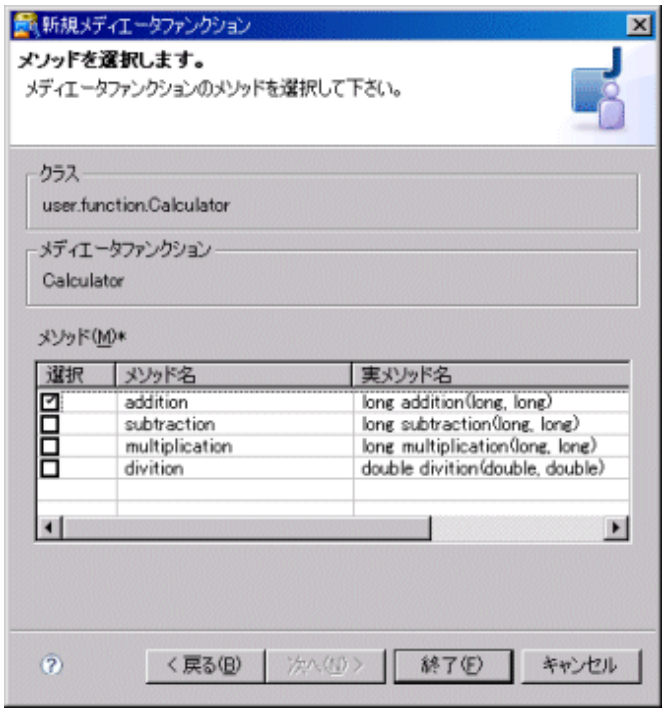
4. 作成したクラス“user.function.Calculator”を選択し、[次へ]ボタンをクリックします。
[メディエータファンクション説明]は省略可能です。

図3.5 [新規メディエータファンクション作成(2ページ目)]ダイアログボックスの画面例



5. 作成したメソッド“addition”を指定し、[終了]ボタンをクリックします。

図3.6 [新規メディエータファンクション作成(3ページ目)]ダイアログボックスの画面例



6. 作成されたメディエータファンクションのプロパティを確認します。
ユーザプログラムへの初期値を指定する場合は、プロパティを変更してください。
以下の例では、プロパティの変更は不要です。

図3.7 メディエータファンクションの定義例(メディエータファンクションエディタ)

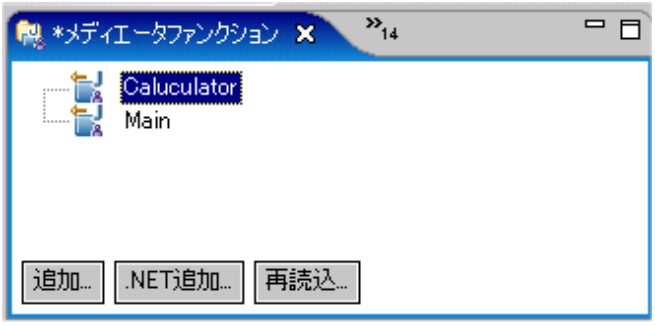
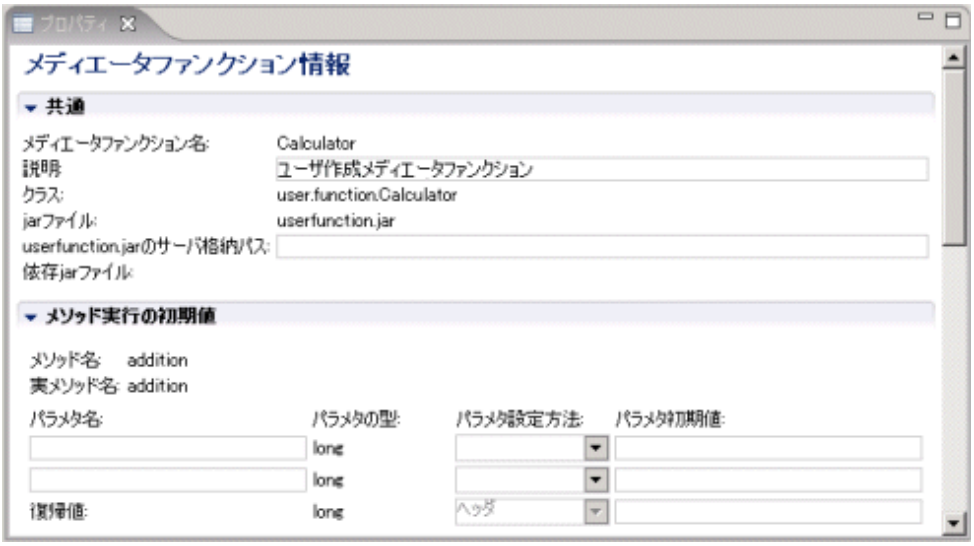


図3.8 メディエータファンクションの定義例(プロパティビュー)



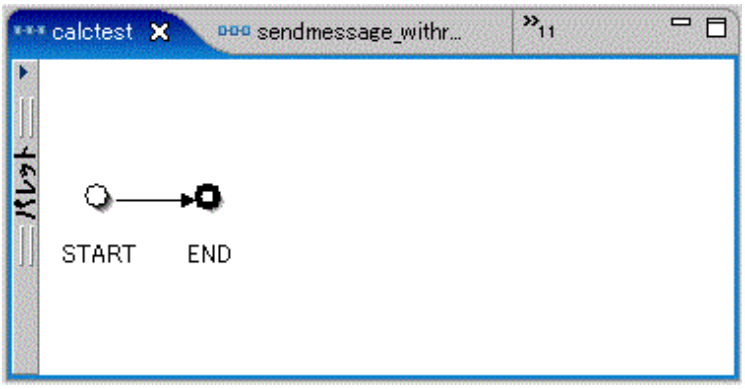
7. メディエータファンクションエディタを保存します。
   ショートカットキーの[Ctrl+S]、またはファイルメニューの[ファイル]－[保存]を選択し、メディエータファンクションエディタを保存します。

シーケンスの作成

ユーザプログラムを呼び出すシーケンスを作成します。

1. ユーザ作成のメディエータファンクションを挿入するシーケンスを表示します。
   以下の例では、シーケンス名“calctest”を使用しています。

   図3.9 シーケンスエディタの画面例



2. パレットから挿入するメディエータファンクションを選択し、シーケンスエディタの挿入ポイント(矢印上の矩形)へドラッグアンドドロップします。

   図3.10 ステップ挿入の画面例



3. シーケンスにユーザ作成のメディエータファンクションがステップとして挿入されます。
   ステップをクリックすると、プロパティが表示されます。
   この例では、実行するユーザプログラムのメソッドとして“addition”メソッドを指定し、メソッドに渡す第1パラメタとして“100”、第2パ

ラメタとして“200”を指定しています。
計算結果となる復帰値は、共通メッセージのヘッダの“result”に格納するように指定しています。

図3.11 シーケンスの定義例(シーケンスエディタ)

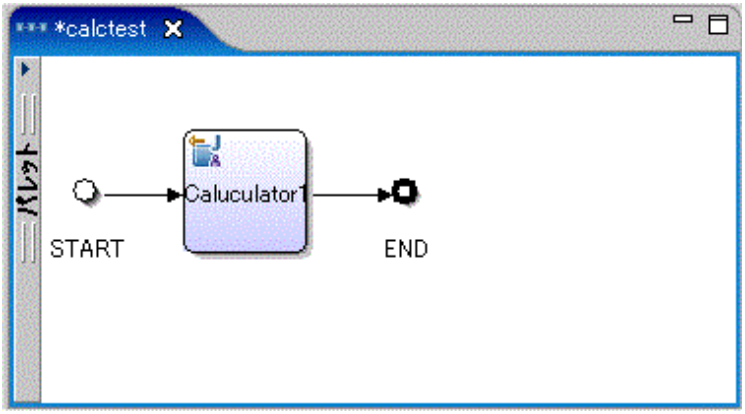


図3.12 シーケンスの定義例(プロパティビュー)



ISI Studioの操作や画面の詳細については、“ISI Studio ヘルプ”を参照してください。
定義の詳細については、“付録B 定義の詳細”を参照してください。

3.4.3 アプリケーションのコンパイル

アプリケーションをコンパイルするにはクラスパスに以下を含めてください。

- %ESI_HOME%¥lib¥esicore.jar(注)
- %CLASSPATH%

注:共通メッセージを使用しないアプリケーションでは不要です。

コンパイルの例

```
javac -classpath "%ESI_HOME%¥lib¥esicore.jar;%CLASSPATH%;." ESIJumpSample.java
```

3.4.4 jarファイルの配置

定義で指定したjarファイル格納パスにjarファイルを配置します。
定義のサーバ格納パスの指定を省略した場合は、以下に配置します。

W

```
<INSDIR>¥usr¥lib
```

S L

```
/opt/FJSVesi/usr/lib
```

3.5 ジャンプ用メディエータファンクションの開発

シーケンスエンジンでは、途中実行処理をジャンプする機能を提供しています。ジャンプする機能を利用するには、以下の方法があります。

- ユーザプログラムで設定
- メディエータファンクションの復帰値をジャンプ先として設定

注意

- 以下の場合、シーケンスを実行するとエラーになります。
  - ー 存在しないジャンプ先を指定した場合
  - ー ジャンプを指定するメディエータファンクションより前のステップ名を設定した場合
- ジャンプする機能と条件分岐機能を同時に利用することはできません。同時に利用した場合は、条件分岐機能が有効になります。

3.5.1 ユーザプログラムによる設定

メソッドのパラメタに共通メッセージをとるユーザプログラムを作成し、このメソッドの中で共通メッセージのヘッダにジャンプ先を設定します。共通メッセージのヘッダに値を設定するには、ヘッダパラメタキー名に“ESIHeaderConstants.ESI_HEADER_KEY_JUMP”を指定します。
また、ヘッダパラメタ値にはこのメディエータファンクションより後ろのステップ名またはシーケンスの終わりにジャンプする“ESIHeaderConstants.ESI_HEADER_VALUE_JUMP_EXIT”を指定します。

例:
共通メッセージmsgに対し、ジャンプ先ステップをSOAPGateway1に指定する場合

```
msg.getHeader().getParameter().put (
    ESIHeaderConstants.ESI_HEADER_KEY_JUMP, "SOAPGateway1");
```

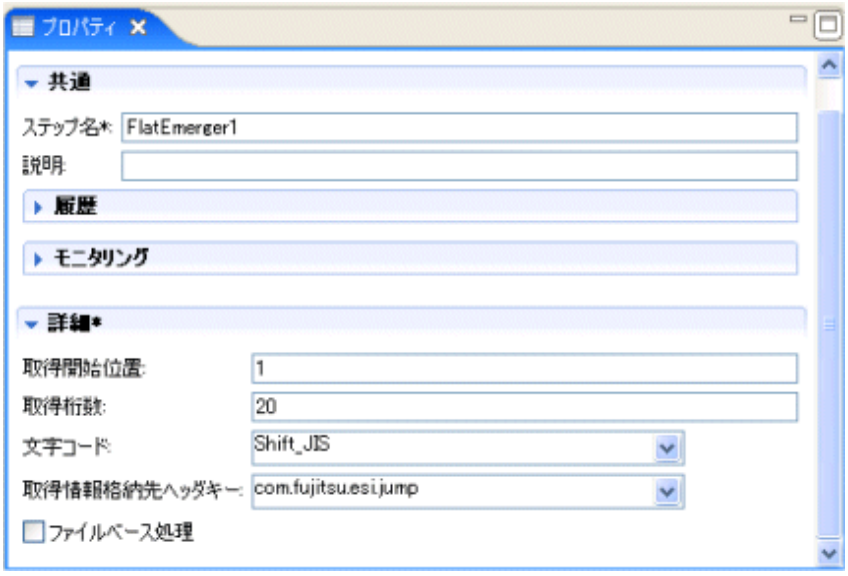
3.5.2 メディエータファンクションの復帰値による設定

メディエータファンクションの復帰値の格納先に“com.fujitsu.esi.jump”を指定します。
以下を復帰値として返すメディエータファンクションで利用できます。

- 基本型
- 基本データ型ラッパクラス
- String型

以下の例では、FLATEmergerの取得情報格納先ヘッダキーに“com.fujitsu.esi.jump”を指定しています。

図3.13 FLATEmergerの画面例(プロパティビュー)



3.5.3 ジャンプ用メディエータファンクション実装例

ジャンプ先を設定するESIJumpSample.java(ジャンプ用メディエータファンクション)のサンプルプログラムです。

```
        :
        :
public class ESIJumpSample {
    // (1) パラメタで共通メッセージを取得する。
    public void jump(ESIMessage msg, String result) {

        // 値によりジャンプ先を設定する。
        if ("SOAP".equals(result)) {
            // (2) ステップSOAPGateway1にジャンプする。
            msg.getHeader().getParameter().put(
               ESIHeaderConstants.ESI_HEADER_KEY_JUMP, "SOAPGateway1");
        } else if ("FAULT".equals(result)) {
            // (3) シーケンスの終わりにジャンプする。
            msg.getHeader().getParameter().put(
               ESIHeaderConstants.ESI_HEADER_KEY_JUMP,ESIHeaderConstants.ESI_HEADER_VALUE_JUMP_EXIT);
        }
    }
        :
        :
}
```

1. 引数の取得
   ジャンプ先を設定するために共通メッセージを引数として取得します。
2. ステップ名指定によるジャンプ先設定
   以下のキー名と値を指定し、共通メッセージのヘッダに格納します。
   - キー名:“ESIHeaderConstants.ESI_HEADER_KEY_JUMP”
   - 値:ステップ名(サンプルプログラムではSOAPGateway1)
3. シーケンスの終わりへのジャンプ設定
   以下のキー名と値を指定し、共通メッセージのヘッダに格納します。
   - キー名:“ESIHeaderConstants.ESI_HEADER_KEY_JUMP”
   - 値:“ESIHeaderConstants.ESI_HEADER_VALUE_JUMP_EXIT”

サンプルプログラムは、以下に格納されています。



“<INSDIR>¥sample¥App¥src”



“/opt/FJSVesi/sample/App/src”
クライアント“<INSDIR>¥sample¥App¥src”

Javaのサンプルプログラムは、JDK1.4互換ソースのため、JDK5.0以上でコンパイルした場合に警告が発生することがありますが、動作に問題はありません。
なお、上記に格納されている“Calculator.java”については、“3.3 ユーザプログラム呼出しの概要”を参照してください。

3.6 ファイルベースシーケンス実行用メディエータファンクションの開発

シーケンスエンジンは、データ(ペイロード)が大容量のときにファイルベースでシーケンス処理を行うことができます。
ファイルベースでシーケンス処理を行う場合、共通メッセージのヘッダに以下の情報が格納されています。

表3.2 ファイルベースシーケンスで使用するヘッダ情報

キー	値/説明
com.fujitsu.esi.filepayload.currentfile	現在のペイロードデータのファイル名です。
com.fujitsu.esi.filepayload.nextfile	メディエータファンクションでの処理結果を格納するファイル名です。 ファイルベースで結果を出力するメディエータファンクションは、このファイルに結果を出力してください。
com.fujitsu.esi.filepayload.workdir	ファイルベースでの処理ディレクトリです。 この情報を利用して、現在のペイロードデータ以外のファイルを参照できます。 “ディレクトリ名+java.io.File.separator+@start”が、このシーケンスの入力ファイル名です。 “ディレクトリ名+java.io.File.separator+ステップ名”が、指定したステップの実行結果のファイル名です。

上記の情報を利用して、ファイルベースシーケンス実行用のメディエータファンクションを開発します。



ファイル全体を、メモリ上に読み込まないでください。メモリ不足が発生し、ISIサーバが停止するおそれがあります。

第4章 アダプタの開発

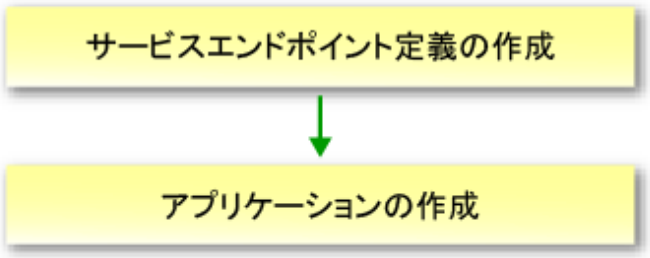
ここでは、アダプタの開発について説明します。

4.1 ISI呼び出しをするカスタムアダプタの開発

ISI呼び出しは標準規約であるJ2EE Connection Architectureを用いて実現できます。

開発の流れを以下に示します。

図4.1 開発の手順



4.1.1 サービスエンドポイント定義の作成

インバウンドサービスのカスタムアダプタのサービスエンドポイント定義を作成します。

サービスエンドポイント定義は、ISI Studioから作成します。サービスエンドポイント定義の作成方法については、“ISI Studio ヘルプ”の“インバウンドのサービスエンドポイント”の“カスタムアダプタの定義”を参照してください。

4.1.2 アプリケーションの作成

例をもとにISI呼び出しをするカスタムアダプタのアプリケーションの作成方法を説明します。

アプリケーションの例

```
import javax.resource.ResourceException;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.IndexedRecord;
import javax.resource.cci.Interaction;
import javax.resource.cci.InteractionSpec;
import javax.resource.cci.MappedRecord;
import javax.resource.cci.RecordFactory;
import com.fujitsu.esi.jca.inbound.ESIInteractionSpec;
import com.fujitsu.esi.jca.inbound.ESIManagedConnectionFactory;
import com.fujitsu.esi.message.ESIHeaderConstants;

public class ISICall {
    public byte[] call(String serviceEndpointName, byte[] payload) {
        byte[] returnPayload = null;
        Connection connection = null;
        Interaction interaction = null;
        try {
            // (1)ConnectionFactoryの生成
            ESIManagedConnectionFactory managedConnectionFactory = new ESIManagedConnectionFactory();
            ConnectionFactory connectionFactory =
                 (ConnectionFactory) managedConnectionFactory.createConnectionFactory();
            // (2)Connectionの取得
            connection = connectionFactory.getConnection();
```

```
            // (3)InteractonSpecの作成
            ESIInteractionSpec interactionSpec = new ESIInteractionSpec();
            interactionSpec.setInteractionVerb(InteractionSpec.SYNC_SEND_RECEIVE);
            interactionSpec.setServiceName(serviceEndpointName);
            // (4)入出力Recordの作成
            RecordFactory recordFactory = connectionFactory.getRecordFactory();
            // 入力Recordの作成
            IndexedRecord input = recordFactory.createIndexedRecord(null);
            // ペイロードの設定
            input.add(payload);
            // ヘッダの設定
            MappedRecord headerMap = recordFactory.createMappedRecord(null);
            headerMap.put("header1", "value1");
            input.add(headerMap);
            // 出力Recordの作成
            IndexedRecord output = recordFactory.createIndexedRecord(null);
            // (5)シーケンス処理の呼び出し
            interaction = connection.createInteraction();
            interaction.execute(interactionSpec, input, output);
            // (6)処理結果の取得
            returnPayload = (byte[]) output.get(0);
            MappedRecord returnHeader = (MappedRecord) output.get(1);
        } catch (ResourceException ex) {
            ex.printStackTrace();
        } finally {
            // (7)クローズ処理
            // Interactionのクローズ
            if (interaction != null) {
                try {
                    interaction.close();
                } catch (ResourceException ex) {
                    ex.printStackTrace();
                }
            }
            // Connectionのクローズ
            if (connection != null) {
                try {
                    connection.close();
                } catch (ResourceException ex) {
                    ex.printStackTrace();
                }
            }
        }
        return returnPayload;
    }
}
```

1. ConnectionFactoryの生成

   J2EE Connection Architectureの記述形式でConnectionFactoryを生成します。

2. Connectionの取得

   J2EE Connection Architectureの記述形式でConnectionを生成します。

3. InteractonSpecの作成

   ISI呼び出しの情報を設定するInteractionSpec(ESIInteractionSpec)を生成します。以下の情報を設定します。

   ー InteractionVerb

   ISIを同期で呼び出すか、非同期で呼び出すかを指定します。

   - 同期呼び出しの場合
     InteractionSpec.SYNC_SEND_RECEIVE

- 非同期呼び出しの場合
  InteractionSpec.SYNC_SEND

ー ServiceName

インバウンドのサービスエンドポイント名を指定します。
指定するサービスエンドポイントのメッセージング（同期/非同期）はInteractionVerbプロパティで指定した同期/非同期と必ず一致させてください。一致していない場合、実行エラーになります。

インバウンドルーティングを行う場合は、このプロパティは設定不要です。
このプロパティが設定されている場合、RoutingNameプロパティの設定は無視されます。

ー RoutingName

インバウンドルーティングを行う場合に、ルーティング定義名を指定します。
インバウンドルーティングを行わない場合、このプロパティは設定不要です。

4. 入出力Record の作成

J2EE Connection Architectureの記述形式でRecordを生成します。入出力Recordには以下の型を使用できます。

ー IndexedRecord

ー MappedRecord

また、入力Recordにリクエスト情報を設定します。入力Recordに設定するリクエスト情報は以下のとおりです

ー ペイロード

メッセージの本文を表します。ペイロードは必須です。byte[]型で入力Recordに設定します。

ー ヘッダパラメタ

メッセージのヘッダ情報を表します。ヘッダパラメタは省略できます。MappedRecord型で入力Recordに設定します。

入力Recordにリクエスト情報を設定する方法は、入力Recordに使用するRecordの型によって異なります。

ー IndexedRecordを使用する場合

- ペイロード

  インデックス0番目に設定します。

- ヘッダパラメタ

  メッセージのヘッダを設定します。

インデックス2番目以降に設定した情報は無視されます。

ー MappedRecordを使用する場合

- ペイロード

  キー名「com.fujitsu.esi.jca.payload」に設定します。

- ヘッダパラメタ

  キー名「com.fujitsu.esi.jca.header」に指定します。

上記以外のキー名で設定した情報は無視されます。

5. シーケンス処理の呼び出し

J2EE Connection Architectureの記述形式でInteractionを生成し、executeメソッドを呼び出します。

6. 処理結果の取得

出力Recordから結果を取り出します。

7. クローズ処理

InteractionとConnectionのクローズを行います。

コンパイルに必要なクラスパス

インバウンドのカスタムアダプタのコンパイル時には、以下のクラスパスを設定してください。

<INSDIR>¥lib¥esicore.jar

<Interstageインストールディレクトリ>¥J2EE¥lib¥isj2ee.jar

4.1.3 実行時の設定

カスタムアダプタをIJServer上で動作させる場合とスタンドアロンで動作させる場合について実行時に必要な設定を以下に示します。

注意

S L

カスタムアダプタは、管理者権限を所有するユーザ、またはInterstage運用者で動作させてください。

IJServer上で動作させる場合

IJServerに対して下記の環境設定を行ってください。

W

設定項目	設定するパス
JavaVMオプション	-Dcom.fujitsu.ObjectDirector.DeterredMessage=YES -Dcom.fujitsu.interstage.jms.queue_max_err=yes EE -DAPFW_DBDC_VERSION=9.1
クラスパス	<INSDIR>¥lib¥esicore.jar <INSDIR>¥lib¥esiejb_jar_client.jar <INSDIR>¥lib¥log4j-1.2.15.jar EE <Interstageインストールディレクトリ>¥APC¥lib¥uji.jar <Interstageインストールディレクトリ>¥APC¥lib¥ujief.jar <Interstageインストールディレクトリ>¥APC¥lib¥ujilog.jar <Interstageインストールディレクトリ>¥BAS¥lib¥apfwibscm.jar <Interstageインストールディレクトリ>¥BAS¥lib¥apfwasyncapi.jar <Interstageインストールディレクトリ>¥ots¥lib¥fjtsclient.jar <Interstageインストールディレクトリ>¥F3FMjs5¥common¥lib¥servlet-api.jar <Symfoware Serverクライアントインストールディレクトリ>¥JDBC¥fjjdbc¥lib¥fjsymjdbc2.jar

S L

設定項目	設定するパス
JavaVMオプション	-Dcom.fujitsu.ObjectDirector.DeterredMessage=YES -Dcom.fujitsu.interstage.jms.queue_max_err=yes -Dcom.fujitsu.esi.home=/opt/FJSVesi EE -DAPFW_DBDC_VERSION=9.1
クラスパス	/opt/FJSVesi/lib/esicore.jar /opt/FJSVesi/lib/esiejb_jar_client.jar /opt/FJSVesi/lib/log4j-1.2.15.jar EE /opt/FJSVwebc/lib/uji.jar /opt/FJSVapcef/lib/ujief.jar /opt/FJSVapclg/lib/ujilog.jar /opt/FJSVibs/lib/apfwibscm.jar /opt/FJSVibs/lib/apfwasyncapi.jar /opt/FJSVibs/lib/apfwsetsecurity.jar /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar
ライブラリパス EE	/opt/FJSVesi/lib /opt/FJSVibs/lib /opt/FJSVapcef/lib /opt/FJSVsymjd/fjjdbc/bin S /opt/FSUNrdb2b/lib L /opt/FJSVrdb2b/lib

スタンドアロンで動作させる場合

W

- VMオプション

  -Dcom.fujitsu.ObjectDirector.DeterredMessage=YES

  -Dcom.fujitsu.interstage.jms.queue_max_err=yes

  -Djava.naming.factory.initial=com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient

  -Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB

  -Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB

  -Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl

  -Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl

  -Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl

  EE

  -DAPFW_DBDC_VERSION=9.1

- クラスパス

  <INSDIR>¥lib¥esicore.jar

  <INSDIR>¥lib¥esiejb_jar_client.jar

  <INSDIR>¥lib¥log4j-1.2.15.jar

<Interstageインストールディレクトリ>¥ODWIN¥etc¥class¥ODjava4.jar

<Interstageインストールディレクトリ>¥eswin¥lib¥esnotifyjava4.jar

<Interstageインストールディレクトリ>¥J2EE¥lib¥isj2ee.jar

<Interstageインストールディレクトリ>¥jms¥lib¥fjmsprovider.jar

EE

<Interstageインストールディレクトリ>¥APC¥lib¥uji.jar

<Interstageインストールディレクトリ>¥APC¥lib¥ujief.jar

<Interstageインストールディレクトリ>¥APC¥lib¥ujilog.jar

<Interstageインストールディレクトリ>¥BAS¥lib¥apfwibscm.jar

<Interstageインストールディレクトリ>¥BAS¥lib¥apfwasyncapi.jar

<Interstageインストールディレクトリ>¥lib¥isjmxcommon.jar

<Interstageインストールディレクトリ>¥ots¥lib¥fjtsclient.jar

<Interstageインストールディレクトリ>¥F3FMjs5¥common¥lib¥servlet-api.jar

<Symfoware Serverクライアントインストールディレクトリ>¥JDBC¥fjjdbc¥lib¥fjsymjdbc2.jar

S L

- JavaVMオプション

  -Dcom.fujitsu.ObjectDirector.DeterredMessage=YES

  -Dcom.fujitsu.interstage.jms.queue_max_err=yes

  -Djava.naming.factory.initial=com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient

  -Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB

  -Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB

  -Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl

  -Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl

  -Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl

  -Dcom.fujitsu.esi.home=/opt/FJSVesi

  EE

  -DAPFW_DBDC_VERSION=9.1

- 環境変数の設定

  S

環境変数	設定するパス
JAVA_HOME	/opt/FJSVawjbk/jdk6
PATH	$JAVA_HOME/bin /opt/FJSVj2ee/bin /opt/FJSVesi/bin
CLASSPATH	/opt/FJSVesi/lib/esicore.jar /opt/FJSVesi/lib/esiejb_jar_client.jar /opt/FJSVesi/lib/log4j-1.2.15.jar /opt/FSUNod/etc/class/ODjava4.jar /opt/FJSVes/lib/esnotifyjava4.jar /opt/FJSVj2ee/lib/isj2ee.jar /opt/FJSVjms/lib/fjmsprovider.jar

環境変数	設定するパス
	EE /opt/FJSVwebc/lib/uji.jar /opt/FJSVapcef/lib/ujief.jar /opt/FJSVapclg/lib/ujilog.jar /opt/FJSVibs/lib/apfwibscm.jar /opt/FJSVibs/lib/apfwasyncapi.jar /opt/FJSVibs/lib/apfwsetsecurity.jar /opt/FJSVisjmx/lib/isjmxcommon.jar /opt/FJSVots/lib/fjtsclient.jar /opt/FJSVjs5/common/lib/servlet-api.jar /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar
LD_LIBRARY_PATH	/opt/FJSVesi/lib /opt/FSUNod/lib /opt/FJSVjms/lib EE /opt/FJSVibs/lib /opt/FJSVapcef/lib /opt/FSUNrdb2b/lib /opt/FJSVsymjd/fjjdbc/bin

L

環境変数	設定するパス
JAVA_HOME	/opt/FJSVawjbk/jdk6
PATH	$JAVA_HOME/bin /opt/FJSVj2ee/bin /opt/FJSVesi/bin
CLASSPATH	/opt/FJSVesi/lib/esicore.jar /opt/FJSVesi/lib/esiejb_jar_client.jar /opt/FJSVesi/lib/log4j-1.2.15.jar /opt/FJSVod/etc/class/ODjava4.jar /opt/FJSVes/lib/esnotifyjava4.jar /opt/FJSVj2ee/lib/isj2ee.jar /opt/FJSVjms/lib/fjmsprovider.jar EE /opt/FJSVwebc/lib/uji.jar /opt/FJSVapcef/lib/ujief.jar /opt/FJSVapclg/lib/ujilog.jar /opt/FJSVibs/lib/apfwibscm.jar /opt/FJSVibs/lib/apfwasyncapi.jar /opt/FJSVibs/lib/apfwsetsecurity.jar /opt/FJSVisjmx/lib/isjmxcommon.jar /opt/FJSVots/lib/fjtsclient.jar /opt/FJSVjs5/common/lib/servlet-api.jar /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar
LD_LIBRARY_PATH	/opt/FJSVesi/lib /opt/FJSVod/lib /opt/FJSVjms/lib EE /opt/FJSVibs/lib /opt/FJSVapcef/lib

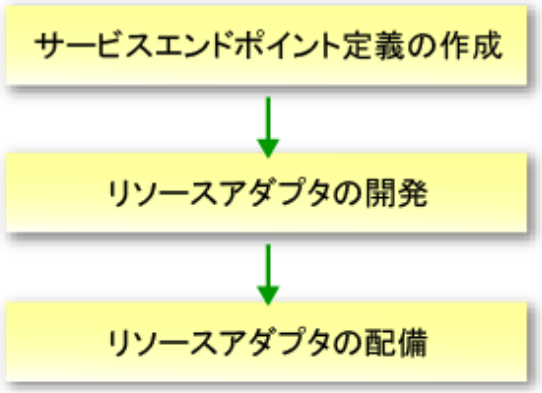
環境変数	設定するパス
	/opt/FJSVrdb2b/lib /opt/FJSVsymjd/fjjdbc/bin

4.2 サービス提供側アプリケーションを呼び出すカスタムアダプタの開発

アウトバウンドサービスのカスタムアダプタは、J2EE Connection Architecture 1.0のManaged対応のリソースアダプタとして開発します。

開発の流れを以下に示します。

図4.2 開発の手順



4.2.1 サービスエンドポイント定義の作成

アウトバウンドサービスのカスタムアダプタのサービスエンドポイント定義を作成します。

サービスエンドポイント定義は、ISI Studioから作成します。サービスエンドポイント定義の作成方法については、“ISI Studio ヘルプ”の“アウトバウンドのサービスエンドポイント”の“カスタムアダプタの定義”を参照してください。

4.2.2 リソースアダプタの開発

アウトバウンドサービスのカスタムアダプタは、J2EE Connection Architecture 1.0のManaged対応のリソースアダプタとして開発します。J2EE Connection Architecture 1.0の詳細は、Oracle Corporationから提供されているドキュメントを参照してください。

ここでは、処理の流れとデータの受け渡しについて説明します。実装が必要なJ2EE Connection Architecture標準インタフェースについては、“実装する必要のあるJ2EE Connection Architecture標準インタフェース”を参照してください。

処理の流れ

アウトバウンド側ISI接続FWがアウトバウンドアダプタを呼び出す手順は、J2EE Connection Architectureで定められた呼び出し手順に従います。

以下にISIサーバ起動とメッセージ処理の各タイミングで行う処理の流れを示します。

ISIサーバ起動時

1. コネクションファクトリの取得

   サービスエンドポイント定義に定義されているJNDI名を使用してコネクションファクトリを取得します。複数のサービスエンドポイント定義に同じJNDI名が定義されている場合、JNDI名ごとに1度だけコネクションファクトリの取得が行われます。

メッセージ処理時

1. コネクションの取得

   ISIサーバ起動時に取得したコネクションファクトリのgetConnectionメソッドを使用してコネクションを取得します。

2. サービス提供側アプリケーションの呼び出し

   Interactionオブジェクトのexecuteメソッドを実行します。
   executeメソッドの引数としてRecordが渡されます。Recordで渡される情報については、”入出力Record”を参照してください。

3. コネクションのクローズ

   1で取得したコネクションのcloseメソッドを使用してコネクションをクローズします。

データの受け渡し

ISIサーバからアウトバウンドサービスのカスタムアダプタに受け渡される情報は以下のとおりです。

・ 接続情報(ConnectionSpec)

アウトバウンド側ISI接続FWからアウトバウンドアダプタへのConnectionSpecの受け渡しには、アウトバウンドアダプタが提供する、javax.resource.cci.ConnectionSpecの実装クラスのオブジェクトを使用します。

ConnectionSpecの実装クラスに定義するプロパティは以下のとおりです。

ー 標準プロパティ

J2EE Connection Architectureで規定されるConnectionSpecの標準プロパティは以下のとおりです。

- UserName（String型）

  サービス提供側アプリケーションへの接続に使用するユーザ名を設定します。
  設定する値はサービスエンドポイント定義で指定することができます。サービスエンドポイント定義による指定がない場合、ConnectionSpec実装クラスのデフォルトの値がそのまま使用されます。

- Password（String型）

  サービス提供側アプリケーションへの接続に使用するパスワードを設定します。
  設定する値はサービスエンドポイント定義で指定することができます。サービスエンドポイント定義による指定がない場合、ConnectionSpec実装クラスのデフォルトの値がそのまま使用されます。

ー 拡張プロパティ

アウトバウンドアダプタは任意の名前で拡張プロパティを定義できます。
拡張プロパティで利用できる型は以下のとおりです。

- boolean
- short
- int
- long
- float
- double
- String

・ 相互作用情報(InteractionSpec)

アウトバウンド側ISI接続FWからアウトバウンドアダプタへのInteractionSpecの受け渡しには、アウトバウンドアダプタが提供する、javax.resource.cci.InteractionSpecの実装クラスのオブジェクトを使用します。InteractionSpecの実装クラスに定義するプロパティは以下のとおりです。

ー 標準プロパティ

J2EE Connection Architectureで規定されるInteractionSpecの標準プロパティは以下のとおりです。

- FunctionName

  EIS機能名を指定します。
  設定する値はサービスエンドポイント定義で指定することが可能です。サービスエンドポイント定義による指定がない場合、ConnectionSpec実装クラスのデフォルトの値がそのまま使用されます。

- InteractionVerb

  javax.resource.cci.InteractionSpecに定義されている相互作用モードを設定します。

  設定する値はサービスエンドポイント定義で指定できます。

値	意味
0（SYNC_SEND）	メッセージの送信だけ
1（SYNC_SEND_RECEIVE）	メッセージの送受信
2（SYNC_RECEIVE）	メッセージの受信

  定義による指定がない場合、提供側サービスの呼び出しに使用するメディエータファンクションに応じた値が設定されます。

呼び出し	設定される値
同期サービス呼び出し	0（SYNC_SEND）
非同期サービス呼び出し	1（SYNC_SEND_RECEIVE）

- ExecutionTimeout

  処理のタイムアウト時間をミリ秒で設定します。
  設定する値はサービスエンドポイント定義で指定することが可能です。サービスエンドポイント定義による指定がない場合、ConnectionSpec実装クラスのデフォルトの値がそのまま使用されます。

ー 拡張プロパティ

アウトバウンドアダプタは任意の名前で拡張プロパティを定義することができます。
拡張プロパティで利用可能な型は以下のとおりです。

- boolean
- short
- int
- long
- float
- double
- String

・ データ(Record)

アウトバウンド側ISI接続FWとアウトバウンドアダプタとの間でのRecordの受け渡しには、アウトバウンドアダプタが提供する以下のどちらかのインタフェースを実装したクラスのオブジェクトを使用します。

ー javax.resource.cci.IndexedRecord

ー javax.resource.cci.MappedRecord

受け渡されるRecordに格納される情報は、マッパーメディエータファンクションで定義します。
Recordで受け渡しされる情報の設定については、“ISI Studio ヘルプ”を参照してください。

実装する必要のあるJ2EE Connection Architecture標準インタフェース

アウトバウンドサービスのリソースアダプタで実装する必要のあるJ2EE Connection Architecture標準インタフェースは以下のとおりです。

ここでは、アウトバウンドサービスのリソースアダプタで利用するCCIインタフェースについてだけ記載しています。

表4.1 実装する必要のあるJ2EE Connection Architecture 標準インタフェース

J2EE Connection Architecture規定のインタフェース	アウトバウンドアダプタでの実装 ○:必須 △:任意 ×:不要※ ※ISIから使用しないインタフェースのため実装は不要です。	概要
Connection	○	EISインスタンスへの物理接続にアクセスするためのアプリケーションレベルのハンドルです。
ConnectionFactory	○	Connectionを取得するためのインタフェースを提供します。
ConnectionMetaData	×	Connectionを通じて接続されるEIS インスタンスに関するメタ情報を提供します。
ConnectionSpec	△ EISインスタンスへアクセスするために必要な情報がある場合は実装してください。	EISインスタンスへアクセスするためのConnectionSpecを保持します。
IndexedRecord	△ IndexedRecordまたはMappedRecordの少なくとも一方を実装してください。	java.util.Listインタフェースに基づいて順序付けられたレコード要素の集合を表します。
Interaction	○	EISインスタンスとの相互作用を実行するためのインタフェースです。
InteractionSpec	○	EISインスタンスとの相互作用を実行するためのプロパティを保持します。
LocalTransaction	×	アプリケーションレベルのローカルトランザクションの制御に使用します。
MappedRecord	△ IndexedRecordまたはMappedRecordの少なくとも一方を実装してください。	キーと値のマップをベースにした、レコード要素の表現に使用します。
MessageListener	×	要求/応答形式のMDBを実装するためのインタフェースです。
Record	○ Recordを継承するIndexedRecordまたはMappedRecordを実装してください。	Interaction内で定義される実行メソッドへの入力または出力表現の基本インタフェースです。
RecordFactory	○	MappedRecord および IndexedRecord インスタンスを作成するために使用します。
ResourceAdapterMetaData	×	リソースアダプタに関するメタ情報を表します。
ResultSet	×	EISインスタンスとの相互作用の実行により取得される表形式のRecordを表します。
ResultSetInfo	×	ResultSetの機能サポートに関する情報を提供します。
Streamable	×	Recordからバイトストリームとして値を取得したり、出力Recordにバイトストリームで値を設定したりするために使用します。

4.2.3 リソースアダプタの配備

リソースアダプタとして作成したカスタムアダプタの配備と配備解除について説明します。

リソースアダプタの配備

リソースアダプタの配備手順

1. resource adapterファイルを作成

   アウトバウンドサービスのアダプタのモジュールをresource adapterファイル(.rar形式)に圧縮します。

2. resource adapterファイルを運用環境に配備

   Interstage管理コンソールを使用して、[Interstage Application Server] > [システム] > [リソース] > [connector]にresource adapterファイルを配備します。

3. ISIサーバの環境設定

   Interstage管理コンソールを使用して、[Interstage Application Server] > [システム] > [ワークユニット]からISIサーバのワークユニットを選択します。[環境設定]のワークユニット設定にある[パス]、および[クラスパス]に以下の設定をしてください。

   ISIサーバの構成が非同期メッセージング構成の場合は、ISIサーバを構成する2つのIJServerで設定が必要です。

   複数のISIサーバが存在する場合、すべてのISIサーバをIJServerで設定する必要があります。

   W

   ー [パス]に設定する情報

   ```
   <J2EE共通ディレクトリ>¥deployed¥jca¥ra¥<リソース名>
   ```

   ー [クラスパス]に設定する情報

   ```
   <J2EE共通ディレクトリ>¥deployed¥jca¥ra¥<リソース名>
   ```

   S L

   ー [パス]に設定する情報

   ```
   <J2EE共通ディレクトリ>/deployed/jca/ra/<リソース名>
   ```

   ー [クラスパス]に設定する情報

   ```
   <J2EE共通ディレクトリ>/deployed/jca/ra/<リソース名>
   ```

リソースアダプタの配備解除

リソースアダプタの配備解除手順

1. resource adapterファイルを運用環境から配備解除

   Interstage管理コンソールを使用して、[Interstage Application Server] > [システム] > [リソース] > [connector]からresource adapterファイルを配備解除

2. ISIサーバにクラスパスの設定を戻す。

   Interstage管理コンソールを使用して、[Interstage Application Server] > [システム] > [ワークユニット]からISIサーバのIJServerを選択します。[環境設定]のワークユニット設定にある[パス]、および[クラスパス]からリソースアダプタの配備時に設定した情報を削除します。

   ISIサーバの構成が非同期メッセージング構成の場合は、ISIサーバを構成する2つのIJServerで設定が必要です。

   複数のISIサーバが存在する場合、すべてのISIサーバをIJServerで設定する必要があります。

第5章 変換定義の開発

ここでは、変換定義の開発について説明します。

トランスレーション仕様の決定

Formatmanagerクライアントにてフォーマット変換定義を入力する前に、フォーマット変換を実施するにあたって、事前に内容を確認しておくべき事項があります。
事前に決定しておく項目を、以下に示します。

1. **入力データと出力データの種類の確認**

   相手システムのデータと自システムのデータを対比しながら、入力データと出力データはそれぞれ何が必要で、どの入力データからどの出力データを作成するのかを確認します。

2. **入出力データの格納形式の確認**

   入力データがファイルに格納されてくるのか、メモリに格納されてくるのかを確認します。また、前処理との関連を確認します。同様に、出力データについても確認します。

3. **データのフォーマット形式の確認**

   それぞれのデータが、Formatmanagerで変換が可能なフォーマット形式になっているかを確認します。
   フォーマット形式が一致しない場合は、Formatmanagerの実行前後にフォーマットの整形用プログラムを別途作成するか、Formatmanagerのレイアウト調整機能を使用して、変換が可能なフォーマットにします。

   Formatmanagerで変換が可能なフォーマット形式については、“ISI 解説書”の“ユーザ固有フォーマットについて”、および“XML文書について”を参照してください。

4. **文字コードの確認**

   入力データと出力データの文字コードを確認します。Formatmanagerで入出力できる文字コードはシフトJIS、EUC、JEF、dbcs90、JIS、KEIS、JIPS(E)、UTF-8/UCS2です(入力データと出力データは同じ文字コードとなります。例えば、入力がシフトJISの場合は出力もシフトJIS、入力がEUCの場合は出力もEUCです)。
   入力データがその他の文字コード(EBCDICなど)の場合は、文字コード変換機能を使用して、フォーマット変換前後に文字コード変換を実施します。

   フォーマット変換機能において基準となる文字コードは、変換テーブルの文字コードで決定されます。変換テーブルがEUCの場合は、EUCの入力データであるとみなしてフォーマット変換が行われます。変換テーブルの文字コードは、Formatmanagerクライアントの[サーバ転送先設定]画面で指定します。詳細は、“ISI Formatmanagerクライアント(FEDIT/FL-TABLE)ヘルプ”を参照してください。

5. **出力データの各項目の設定方法の確認(フォーマット変換)**

   出力データの各項目は、入力データのどの項目から設定するかを確認します。以下の問題があった場合は、それぞれの対応を行ってください。

表5.1 問題点と対応策

問題点	対応策
入力データから取得できる値だけでは決定できない項目がある。	利用者プログラムの作成を検討してください。入力データ以外の値を使用して出力に設定するプログラムを作成することで、“フォーマット変換”と“XML変換”で呼び出すことができます。詳細は、“ISIリファレンス”の“ユーザ固有フォーマットの利用者プログラムAPI”または“ISI リファレンス”の“XML文書の利用者プログラムAPI”を参照してください。
入力項目と出力項目のデータ属性が一致していない。	フォーマット変換の“最適化転記”を使用してください。入力データを出力データの属性で整形して出力します。詳細は、“ISI 解説書”の“最適化転記”を参照してください。
値の読み替えが必要。 例) 入力データでは「男:1」、出力データでは「男:M」	フォーマット変換の“項目コード変換(CSV)”で、CSVファイルを用意するか、“項目コード変換(DB)”で、データベースを用意して読み替えを実施してください。
Formatmanagerの機能では設定できない。	利用者プログラムの作成を検討してください。“フォーマット変換”と“XML変換”で呼び出すことができます。詳細は、“ISIリファレンス”の“ユーザ固

問題点	対応策
	有フォーマットの利用者プログラムAPI”または“ISI リファレンス”の“XML 文書の利用者プログラムAPI”を参照してください。

5.1 Formatmanagerクライアントによる変換定義登録

ここでは、Formatmanagerクライアントを使用して、フォーマット変換定義を登録する際の概要を説明します。

5.1.1 変換定義登録の手順

変換定義登録の手順を以下に示します。

図5.1 変換定義登録手順

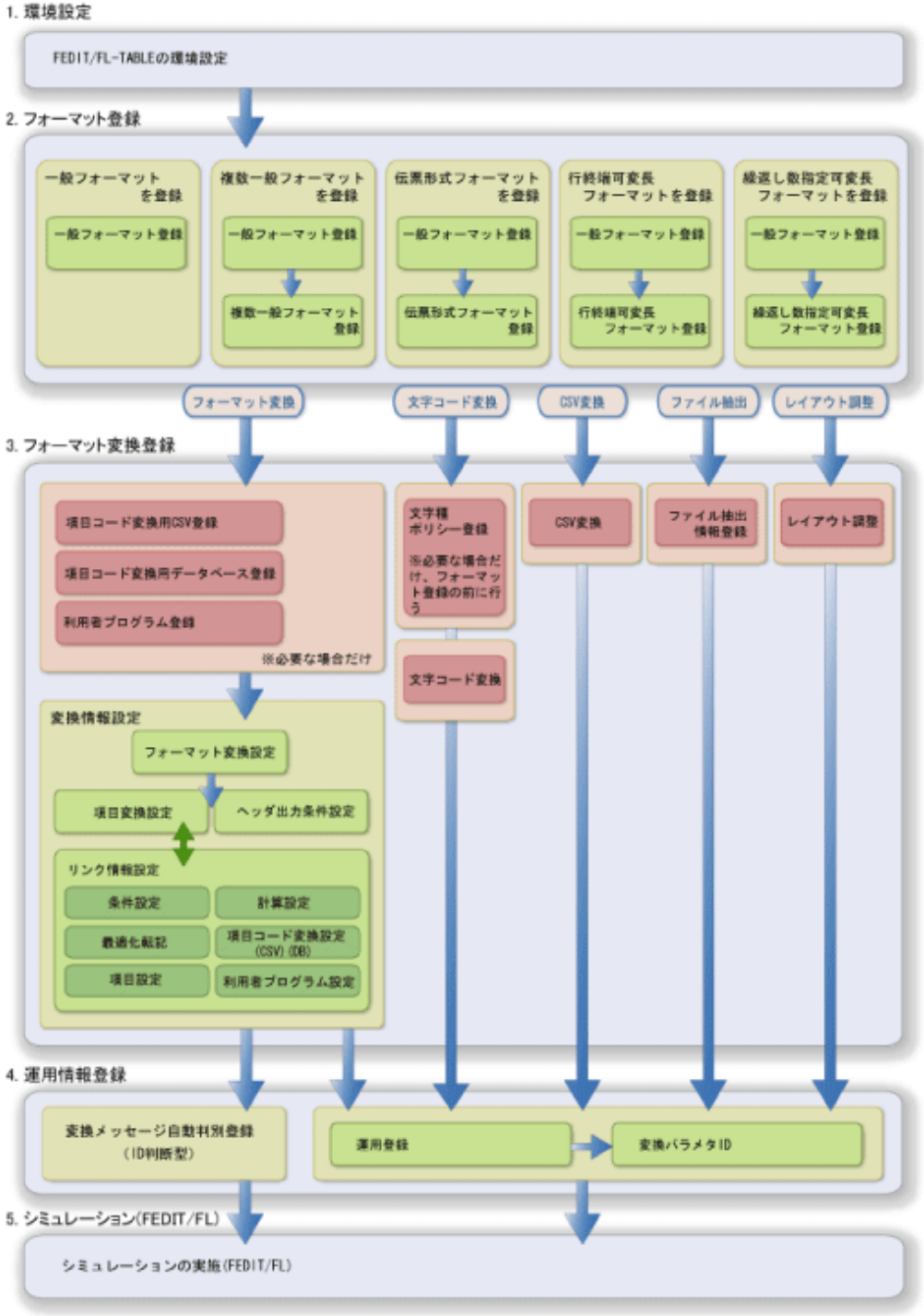


表5.2 作業と対応する章

順番	作業	対応する章
1	環境設定	“ISI 導入ガイド”の“Formatmanagerクライアントの環境設定”
2	フォーマット登録	“ISI 解説書”の“ユーザ固有フォーマットについて”
		“ISI 解説書”の“XML文書について”
		“ISI 解説書”の“項目の属性”
		“ISI 解説書”の“一般フォーマットの自動登録”
		“ISI 解説書”の“変換定義の一括登録”

順番	作業	対応する章
3	フォーマット変換登録	“ISI 解説書”の“ユーザ固有フォーマットのフォーマット変換”
		“ISI 解説書”の“フォーマット変換設定への自動登録”
		“ISI 解説書”の“ユーザ固有フォーマットのその他のフォーマット変換”
		“ISI 解説書”の“XML文書のフォーマット変換”
4	運用情報登録	“ISI 解説書”の“変換パラメタID登録”
5	シミュレーション	“ISI 解説書”の“シミュレーションの実施”

ウィザードによる変換定義のID自動作成

Formatmanagerクライアントでは、ウィザードにより変換定義に必要な一般フォーマットやレイアウト調整を新規作成することができます。その際、一般フォーマットのフラットIDやレイアウト調整のIDを自動的に生成・割当てし、また定義画面の表示までを自動的に行います。ウィザードで生成できる定義は、以下です。

- フォーマット変換定義
  - 一般フォーマット→一般フォーマット間のフォーマット変換
- 文字コード変換定義
  - 一般フォーマットの文字コード変換
- レイアウト調整定義
  - 集団項目－明細レコード調整
  - 可変長レコード－固定長レコード調整
  - パディングレコード調整
  - ヘッダの定量間隔挿入・削除

詳細は、“ISI Formatmanagerクライアント（FEDIT/FL-TABLE）ヘルプ”の“ひな型定義作成”を参照ください。

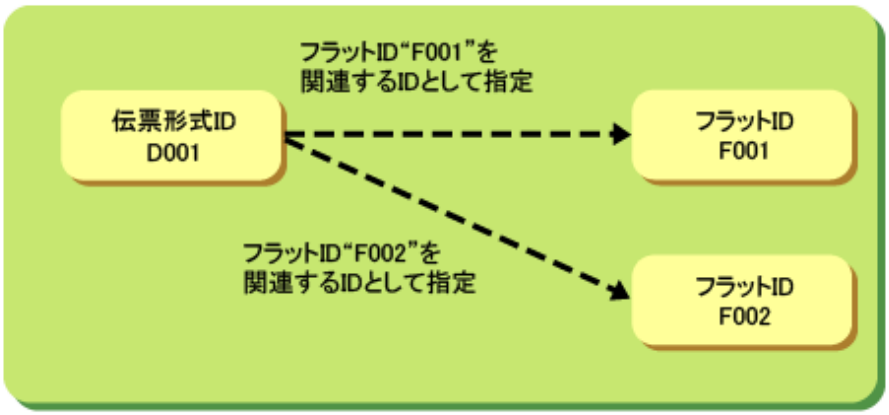
5.1.2 変換定義の関連について

変換定義を構成する各種IDの関連について説明します。

変換定義のIDの関連

変換定義は各種IDを組み合わせて作成します。IDを組み合わせるには、各IDの登録画面から対象のIDを指定します。このとき、指定したIDと指定されたIDをお互いに関連するIDと呼びます。例として、フラットID“F001”、“F002”を組み合わせて伝票形式ID“D001”を作成した場合、フラットID“F001”、“F002”、および伝票形式ID“D001”はお互いに関連するIDになります。

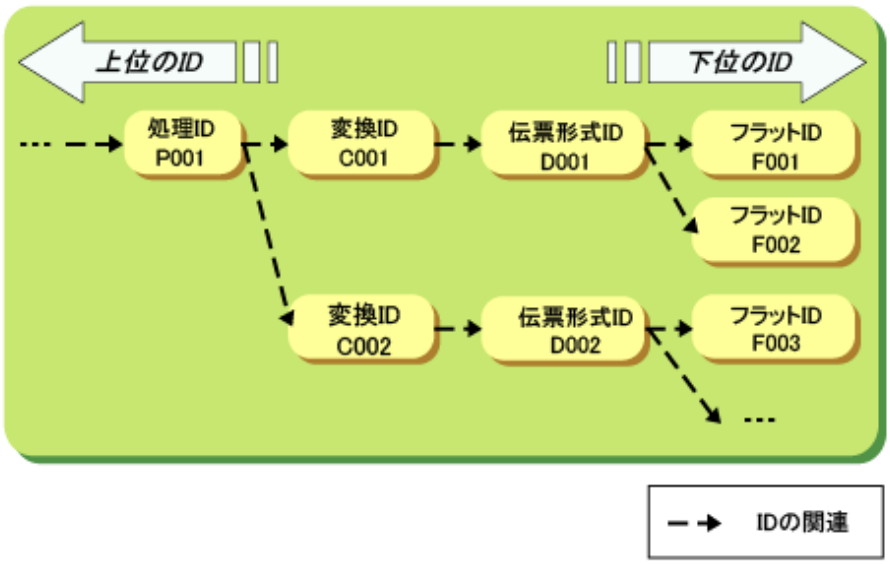
図5.2 図：関連するIDの例



変換定義の関連の上下関係

変換定義のIDの関連には上下関係があります。
関連するIDとして指定したIDが上位のID、指定されたIDが下位のIDとなります。
例として、伝票形式ID“D001”からフラットID“F001”と“F002”を指定した場合、伝票形式ID“D001”が上位のID、フラットID“F001”と“F002”が下位のIDとなります。

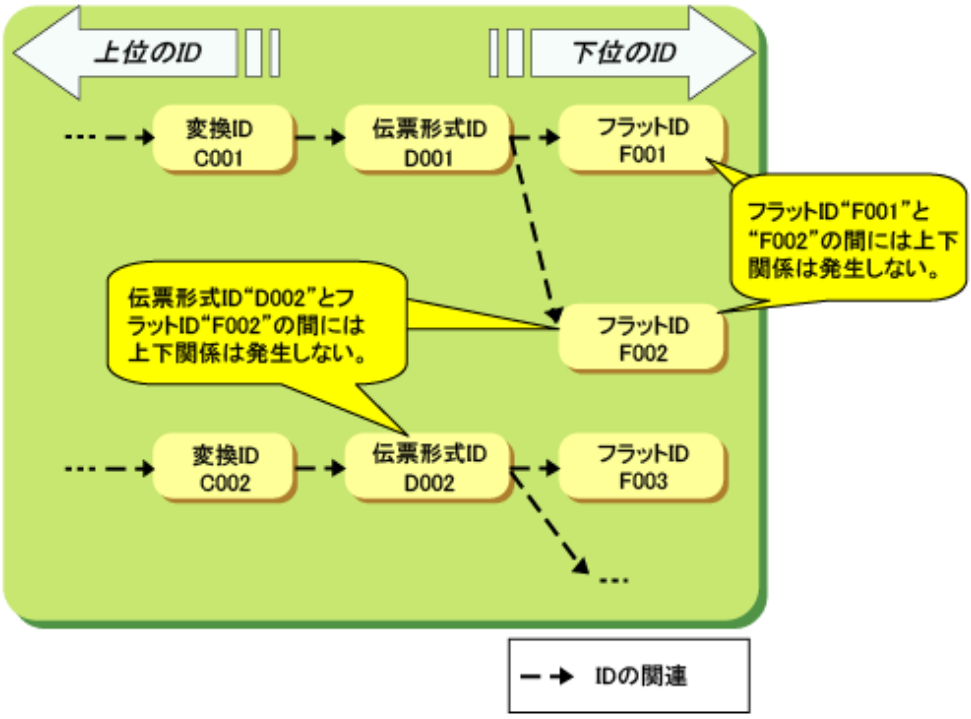
図5.3 図：関連するIDの上下関係



ただし、以下に示すIDの間には上下関係は発生しません。

- 上位のIDが同一である下位のIDの間
- 関連のないIDの間

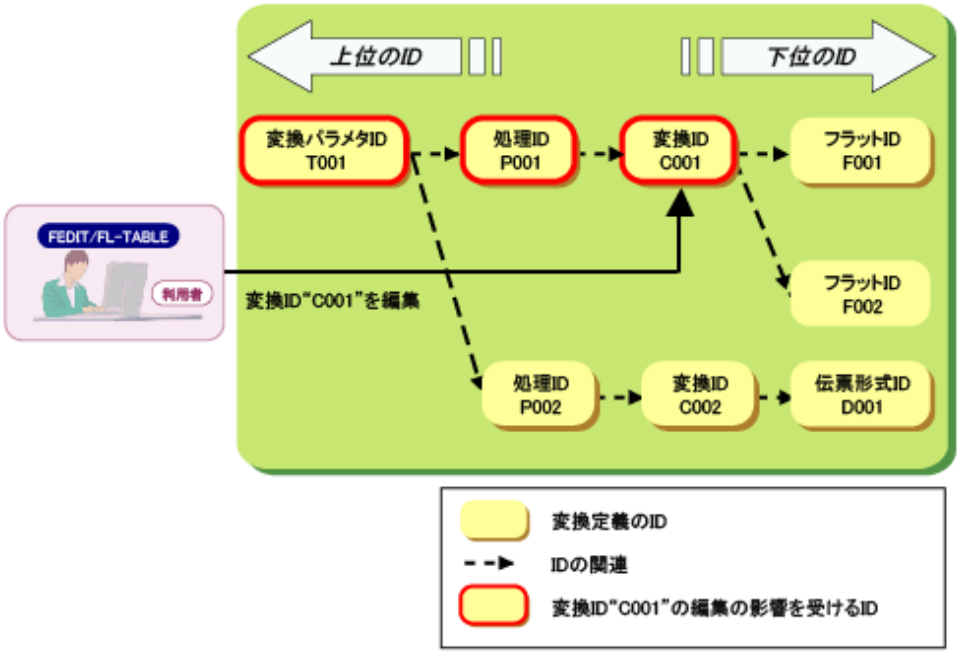
図5.4 図：関連の上下関係が発生しないIDの例



変換定義の編集による影響

変換定義を編集した場合、編集対象のIDと関連があるIDのうち、上位のIDが編集の影響を受けます。変換定義を編集する際は、編集の影響を考慮して実施してください。

図5.5 図：IDの編集による影響が発生する範囲



変換定義に編集による影響の詳細については、“ISI Formatmanagerクライアント（FEDIT/FL-TABLE）ヘルプ”を参照してください。

5.1.3 Formatmanagerクライアントの環境設定

Formatmanagerクライアントを動作させるためには、以下の環境設定が必要です。詳細は、“ISI 導入ガイド”の“Formatmanagerクライアントの環境設定”を参照してください。

- データベースの設定
- サーバ情報の設定

5.1.4 フォーマット登録について

フォーマット変換を行うデータを構成する項目のデータ長やデータ型を指定して、フォーマット情報を登録します。以下のフォーマットが登録できます。

表5.3 ユーザ固有フォーマットの種類

種類	ユーザ固有フォーマット同士の変換	ユーザ固有フォーマットとXML文書の変換	備考
一般フォーマット	○	○	自動登録が可
複数一般フォーマット	○	○	自動登録が可
伝票形式フォーマット	○	○	自動登録が可
行終端可変長フォーマット	○	○	自動登録が可
繰返し数指定可変長フォーマット	○	×	自動登録が可
短縮FLAT形式ファイル	×	○	一般フォーマットだけ可

繰返し数指定可変長フォーマットは、Formatmanagerクライアントで定義することは可能ですが、XML変換の対象にはなりません。

フォーマット登録では、各項目に対して属性を指定します。属性については、“ISI 解説書”の“項目の属性”を参照してください。

なお、一般フォーマットは、以下の方法で自動登録ができます。一般フォーマットの自動登録については、“ISI 解説書”の“一般フォーマットの自動登録”を参照してください。

- COBOL COPY句
- CSV
- DTD
- XML Schema
- RDB

5.1.5 フォーマット変換登録について

フォーマット変換登録では、フォーマット変換を行うための変換情報を設定します。フォーマット変換には以下の種類があり、その種類によって設定する内容が異なります。

- フォーマット変換
- 文字コード変換
- CSV変換
- ファイル抽出
- レイアウト調整
- XML変換

5.1.6 運用情報登録について

フォーマット変換を行う際に、どのような変換をどのような順番で呼び出すか「変換ID」などで処理を呼び出すことができます。この変換IDなどの登録を運用情報登録と呼びます。運用情報登録については、“ISI 解説書”の“変換パラメタID登録”を参照してください。

5.1.7 シミュレーションについて

Formatmanagerクライアントで運用情報を登録後、Formatmanagerクライアント上で実際に登録された変換内容が正しいかを確認します。この確認操作をシミュレーションと呼びます。シミュレーションについては、“ISI 解説書”の“シミュレーションの実施”を参照してください。

第6章 アダプタ利用時の定義の開発

ここでは、ISIが提供している標準アダプタを利用する場合に必要な定義の作成内容、および利用方法によって必要となるアプリケーションの開発内容について説明します。

6.1 SOAPアダプタ利用時の定義

ここでは、SOAPアダプタ利用時のシーケンス処理について、以下のパターンに分けて説明します。

- SOAPメッセージの受信
- SOAPメッセージの送信
- 添付ファイルの操作

SOAPメッセージの受信

SOAPアダプタを利用して、SOAPメッセージを受信する場合のシーケンス定義は以下のようになります。



1. SOAPボディの抽出
   受信したSOAPメッセージから、SOAPボディを抽出します。
   SOAPボディの抽出には、“SOAPメッセージの操作”メディエータファンクションの“SOAPボディ要素を取得”を利用します。
2. XMLの変換
   抽出したSOAPボディを出力メッセージの形式にあわせて変換します(この例では、SOAPボディを固定長メッセージに変換しています)。
   - 固定長メッセージへの変換には、“固定長メッセージの変換”メディエータファンクションを利用します。
   - XMLへの変換には、“XMLメッセージの変換”を利用します。
3. 連携システムの呼び出し
   変換したメッセージを使い、連携システムを呼び出します。連携システムの呼び出しには“サービス呼出し”のメディエータファンクションを利用します。
4. XMLへの変換
   入力メッセージをSOAPボディ用のXMLに変換します(この例では、固定長メッセージからSOAPボディ用のXMLへ変換しています)。メッセージ交換パターンがOneway方式の場合、このステップは省略できます。
   - 固定長メッセージからSOAPボディ用のXMLへ変換する場合は、“固定長メッセージの変換”を利用します。
   - XMLからSOAPボディ用のXMLへ変換する場合は、“XMLメッセージの変換”を利用します。
5. SOAPメッセージの作成
   Webサービス呼び出し用のSOAPメッセージを作成します。メッセージ交換パターンがOneway方式の場合、このステップは省略できます。
   SOAPメッセージの作成には、“SOAPメッセージの操作”メディエータファンクションの“SOAPメッセージの作成(SOAPボディだけ)”を利用します。
   SOAPヘッダのあるSOAPメッセージを作成する場合は、“SOAPメッセージの操作”メディエータファンクションの“SOAPメッセージの作成”を利用します。

また、ISIでSOAPメッセージを受信するには、SOAPのサービスエンドポイント定義を作成します。

ISIのSOAPアダプタでは、Oneway方式とRequestResponse方式の2つのメッセージ交換パターン（MEP：Message Exchange Patterns）を使うことができます。それぞれのメッセージ交換パターンでのサービスエンドポイント定義作成方法については以下を参照してください。

- 6.1.1 Oneway方式によるSOAPメッセージ受信
- 6.1.2 RequestResponse方式によるSOAPメッセージ受信

SOAPメッセージの送信

SOAPアダプタを利用して、SOAPメッセージを送信する場合のシーケンス定義は以下のようになります。



1. XMLへの変換
   入力メッセージをSOAPボディ用のXMLに変換します(この例では、固定長メッセージからSOAPボディ用のXMLへ変換しています)。
   - 固定長メッセージからSOAPボディ用のXMLへ変換する場合は、“固定長メッセージの変換”を利用します。
   - XMLからSOAPボディ用のXMLへ変換する場合は、“XMLメッセージの変換”を利用します。
2. SOAPメッセージの作成
   Webサービス呼び出し用のSOAPメッセージを作成します。SOAPメッセージの作成には、“SOAPメッセージの操作”メディエータファンクションの“SOAPメッセージの作成(SOAPボディだけ)”を利用します。
   SOAPヘッダのあるSOAPメッセージを作成する場合は、“SOAPメッセージの操作”メディエータファンクションの“SOAPメッセージの作成”を利用します。
3. Webサービスの呼び出し
   SOAPアダプタを呼び出してWebサービスを呼び出します。SOAPアダプタの場合は、“同期サービス呼び出し”、または“ルーティングして同期サービス呼び出し”のメディエータファンクションを選択します。
   この例では、“同期サービス呼び出し”でWebサービスを呼び出しています。
   また、SOAPアダプタでは、Oneway方式とRequestResponse方式の2つのメッセージ交換パターンを選択できます。
   それぞれのメッセージ交換パターンでのサービスエンドポイント定義作成方法については以下を参照してください。
   - 6.1.3 Oneway方式によるSOAPメッセージ送信
   - 6.1.4 RequestResponse方式によるSOAPメッセージ送信
4. SOAPボディの抽出
   受信したSOAPメッセージから、SOAPボディを抽出します。メッセージ交換パターンがOneway方式の場合、このステップは省略できます。
   SOAPボディの抽出には、“SOAPメッセージの操作”メディエータファンクションの“SOAPボディ要素を取得”を利用します。
5. XMLの変換
   抽出したSOAPボディを出力メッセージの形式にあわせて変換します(この例では、SOAPボディを固定長メッセージに変換しています)。メッセージ交換パターンがOneway方式の場合、このステップは省略できます。
   - 固定長メッセージへの変換には、“固定長メッセージの変換”メディエータファンクションを利用します。
   - XMLへの変換には、“XMLメッセージの変換”を利用します。

添付ファイルの操作

ここでは、SOAPアダプタを利用して受信した、添付ファイルを操作する場合のシーケンス定義について説明します。

SOAPアダプタでは以下のタイミングで添付ファイルを受信します。

- インバウンドのSOAPアダプタでSOAPの要求メッセージを受信した時
- アウトバウンドのSOAPアダプタでSOAPの応答メッセージを受信した時

添付ファイルを操作したい場合は、これらのタイミングで添付ファイル操作用のユーザプログラム呼出し用メディエータファンクションを呼び出します。

添付ファイル操作用メディエータファンクションを使い、添付ファイルを操作する場合のシーケンスは以下のようになります。

1. 添付ファイルの操作
   インバウンドのSOAPアダプタで受信した添付ファイルを操作するには、シーケンスの先頭で添付ファイル操作用のユーザプログラム呼出し用メディエータファンクションを呼び出します。
   添付ファイル操作用のユーザプログラム呼出し用メディエータファンクションの作成方法については、“6.1.5 添付ファイルを操作する”を参照してください。

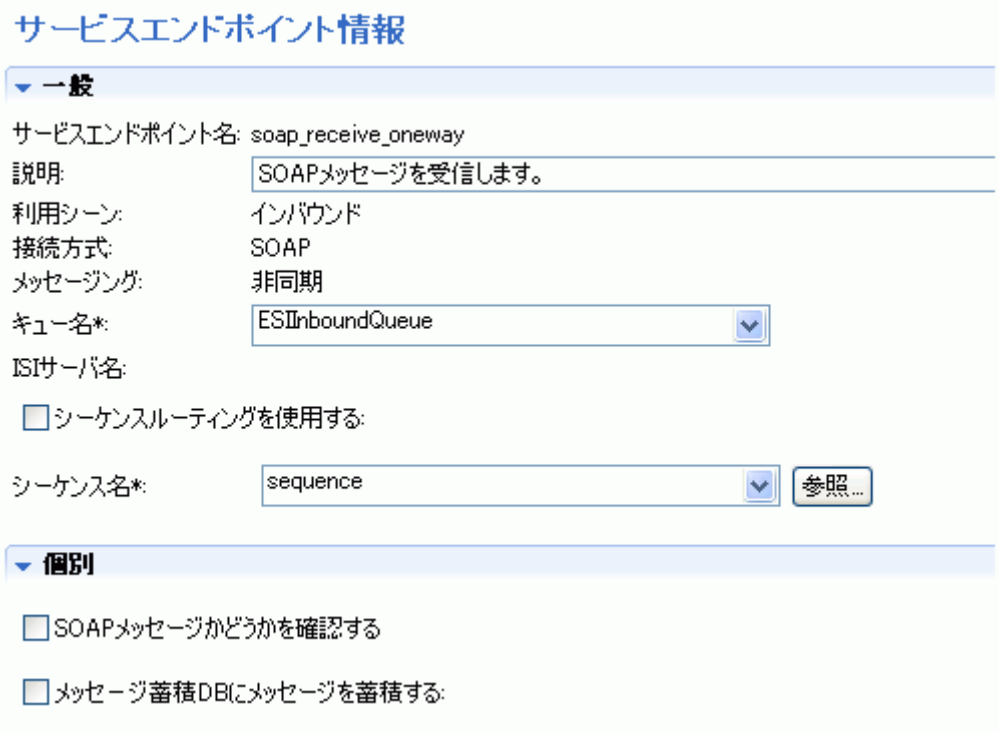
2. SOAP呼び出し
   SOAPアダプタを呼び出してWebサービスを呼び出します。詳しくは“SOAPメッセージの送信”を参照してください。

3. 添付ファイルの操作
   アウトバウンドのSOAPアダプタで受信した添付ファイルを操作するには、SOAP呼び出しの直後に添付ファイル操作用のユーザプログラム呼出し用メディエータファンクションを呼び出します。
   添付ファイル操作用のユーザプログラム呼出し用メディエータファンクションの作成方法については、“6.1.5 添付ファイルを操作する”を参照してください。

6.1.1 Oneway方式によるSOAPメッセージ受信

ここでは、Oneway方式でSOAPメッセージを受信する場合の定義について説明します。

1. サービスエンドポイントの作成
   ISIでSOAPメッセージを受信するには、SOAPのサービスエンドポイント定義を作成します。
   SOAPのOneway方式の場合、サービス利用側アプリケーションは呼び出し先の処理結果を知る必要がありません。このため、Oneway方式で処理を行う場合、メッセージングが“非同期”の定義を作成してください。
   Oneway方式の定義は、以下のようになります。



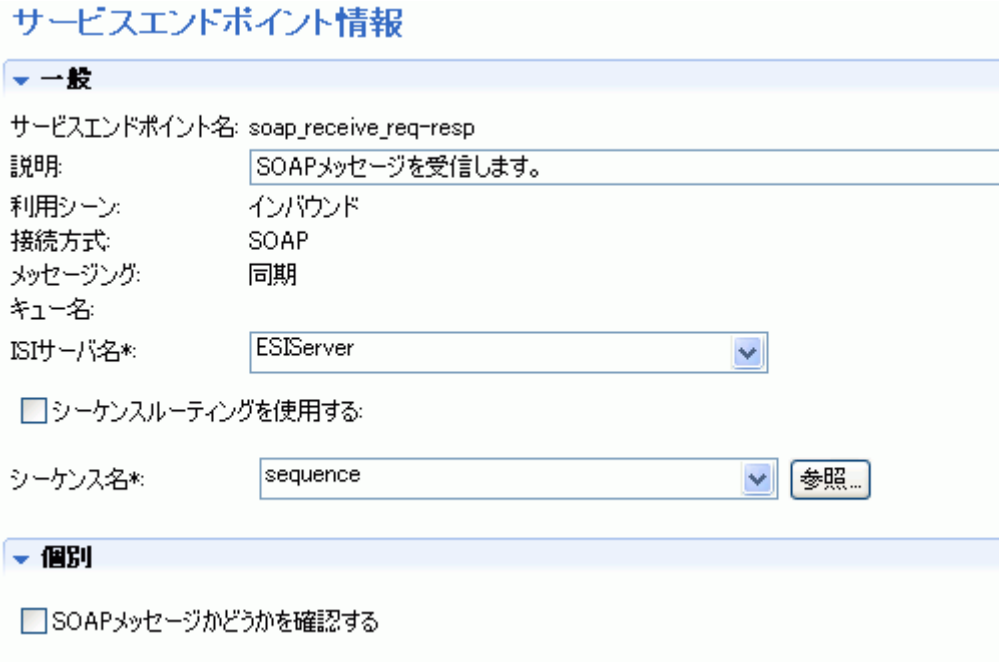
2. シーケンスの作成
   サービスと連携するためのシーケンスを作成します。

6.1.2 RequestResponse方式によるSOAPメッセージ受信

ここでは、RequestResponse方式でSOAPメッセージを受信する場合の定義について説明します。

1. サービスエンドポイントの作成
   ISIでSOAPメッセージを受信するには、SOAPのサービスエンドポイント定義を作成します。
   SOAPのRequestResponse方式の場合、サービス利用側アプリケーションは呼び出し先の処理結果を待ちあわせます。このため、RequestResponse方式で処理を行う場合、メッセージングが“同期”の定義を作成してください。

RequestResponse方式の定義は、以下のようになります。



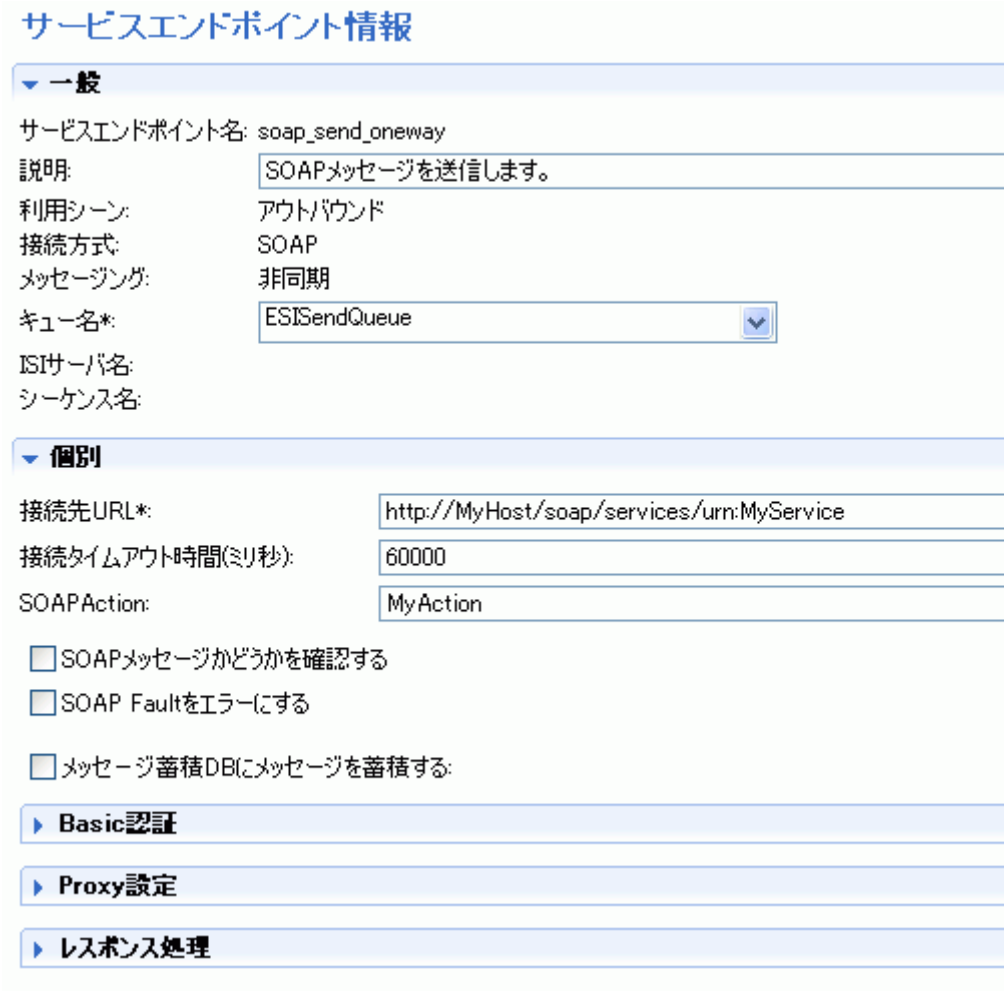
2. シーケンスの作成
   サービスと連携するためのシーケンスを作成します。

6.1.3 Oneway方式によるSOAPメッセージ送信

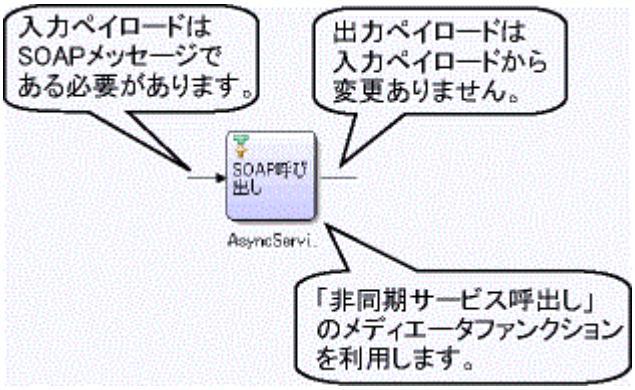
ここでは、Oneway方式でISIからSOAPメッセージを送信する場合の定義について説明します。

1. サービスエンドポイントの作成
   SOAPのOneway方式の場合、サービス提供側アプリケーションは、リクエストのSOAPメッセージを受け取るだけでレスポンスのSOAPメッセージを返却しません。このため、Oneway方式で処理を行う場合、“非同期”のサービスエンドポイント定義を作成してください。

Oneway方式のサービスエンドポイント定義は、以下のようになります。



2. シーケンスの作成
   1で作成したサービスエンドポイントを呼び出すシーケンス定義を作成します。



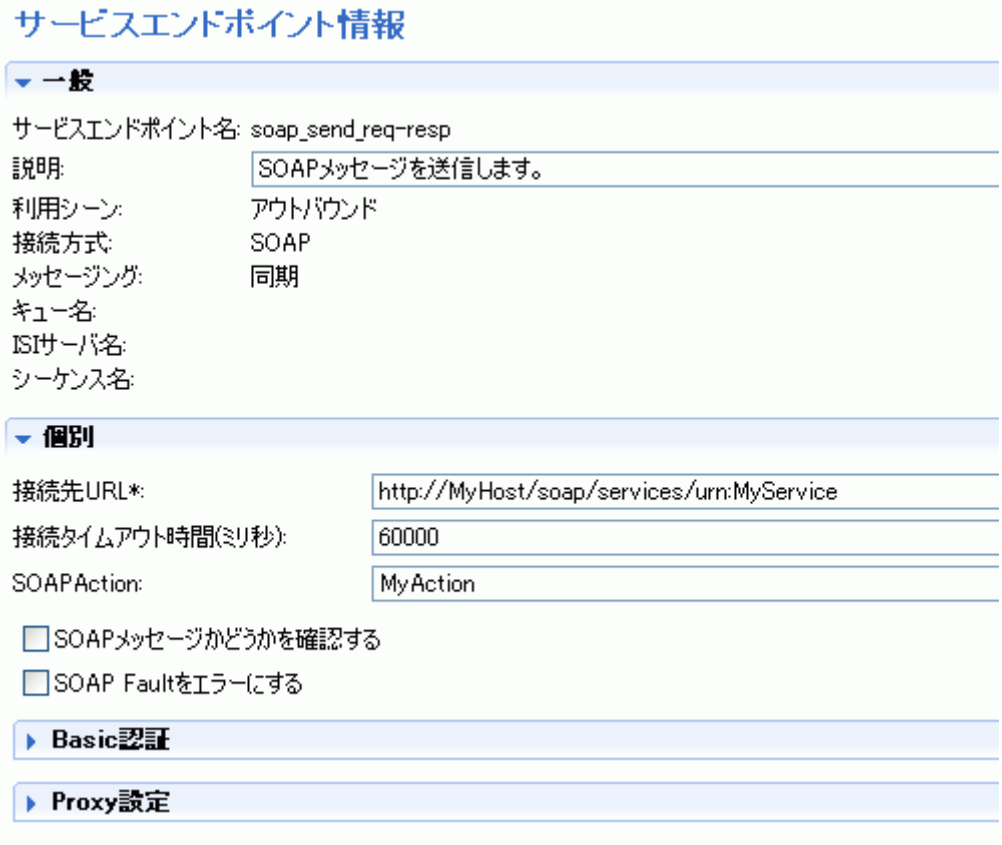
   サービスエンドポイント呼び出し用には、「非同期サービス呼び出し」のメディエータファンクションを利用します。

6.1.4 RequestResponse方式によるSOAPメッセージ送信

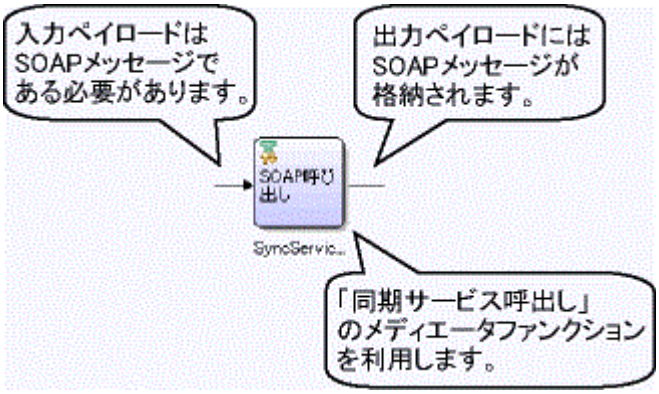
ここでは、RequestResponse方式でISIからSOAPメッセージを送信する場合の定義について説明します。

1. サービスエンドポイントの作成
   SOAPのRequestResponse方式の場合、サービス提供側アプリケーションは、リクエストのSOAPメッセージを処理した結果をレスポンスのSOAPメッセージとして返却します。このため、RequestResponse方式で処理を行う場合、“同期”のサービスエンドポイント定義を作成してください。

RequestResponse方式のサービスエンドポイント定義は、以下のようになります。



2. シーケンスの作成
1で作成したサービスエンドポイントを呼び出すシーケンス定義を作成します。



サービスエンドポイント呼び出し用には、「同期サービス呼び出し」のメディエータファンクションを利用します。
SOAP呼び出しを実行した結果、呼び出し先のWebサービスのレスポンスSOAPメッセージは、出力共通メッセージのペイロードに格納されます。

6.1.5 添付ファイルを操作する

ここでは、受信した添付ファイルを操作するユーザアプリケーション(メディエータファンクション)の作成方法について説明します。

SOAPアダプタで、どのように添付ファイルが保持されるかについては、“添付ファイルの保持形式”を参照してください。

受信した添付ファイルを操作するには、JavaMailのAPIを利用したユーザアプリケーションを開発します。ISIでは、添付ファイルをJavaMailのjavax.mail.internet.MimeMultipartオブジェクトとして操作できます。

以下に、ISIの共通メッセージからJavaMailのjavax.mail.internet.MimeMultipartオブジェクトを取得するユーザアプリケーションのコード例を示します。

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import javax.activation.FileDataSource;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMultipart;

import com.fujitsu.esi.message.ESIMessage;

public class SoapAttachmentSample {

    public void attachmentHandle(ESIMessage message) {

        // 変更結果出力用の FileOutputStream
        FileOutputStream fos = null;
        try {
            // (1) 添付ファイルデータが格納されているファイルパスの取得
            String currentAttachmentFilepath = message.getHeader().getParameter().get(
                    "com.fujitsu.esi.filepayload.currentfile");

            // (2) 変更結果書き出し先ファイルパスの取得
            String nextAttachmentFilepath = message.getHeader().getParameter().get(
                    "com.fujitsu.esi.filepayload.nextfile");

            // (3) ファイル情報から FileDataSource オブジェクトを作成
            FileDataSource source = new FileDataSource(currentAttachmentFilepath);

            // (4) MimeMultipart オブジェクトの作成
            MimeMultipart multipart = new MimeMultipart(source);

            // (5) MimeMultipart オブジェクトを利用し、添付データを操作
                :
            // (6) 変更結果をファイルに書き出す
            fos = new FileOutputStream(nextAttachmentFilepath);
            multipart.writeTo(fos);

        } catch (MessagingException e) {
            // 例外処理
        } catch (FileNotFoundException e) {
            // 例外処理
        } catch (IOException e) {
            // 例外処理
        } finally {
            try {
                fos.flush();
            } catch (IOException e) {
                // 例外処理
            } finally {
                try {
                    fos.close();
                } catch (IOException e) {
                    // 例外処理
                }
            }
        }
    }
}
```

1. 添付ファイルが格納されているファイルパスを取得します

   添付ファイルのデータが格納されているMIMEデータファイルのパスを、共通メッセージの以下のヘッダから取得します。

   ヘッダキー：com.fujitsu.esi.filepayload.currentfile

2. 添付ファイルの変更結果出力先ファイルパスを取得します

   添付ファイルを操作した結果を共通メッセージに反映させたい場合、MIMEデータファイルの出力先パスを、共通メッセージの以下のヘッダから取得しておきます。

   ヘッダキー：com.fujitsu.esi.filepayload.nextfile

   変更結果を共通メッセージに反映する必要がない（後続の処理で利用しない）のであれば、省略できます。

3. ファイルパスをもとに FileDataSource オブジェクトを作成します

   MIMEデータファイルから、MimeMultipart オブジェクトを作成用の FileDataSource オブジェクトを作成します。

4. FileDataSource オブジェクトをもとに MimeMultipart オブジェクトを作成します

   FileDataSource オブジェクトをもとに MimeMultipart オブジェクトを作成します。

5. MimeMultipart オブジェクトを利用し、添付データを操作します

   作成した MimeMultipart オブジェクトを利用し、添付データを操作します。MimeMultipart オブジェクトのルートパート（0番目の要素）は、SOAPメッセージが格納されています。添付ファイルを操作する場合、ルートパート以外のデータを操作します。

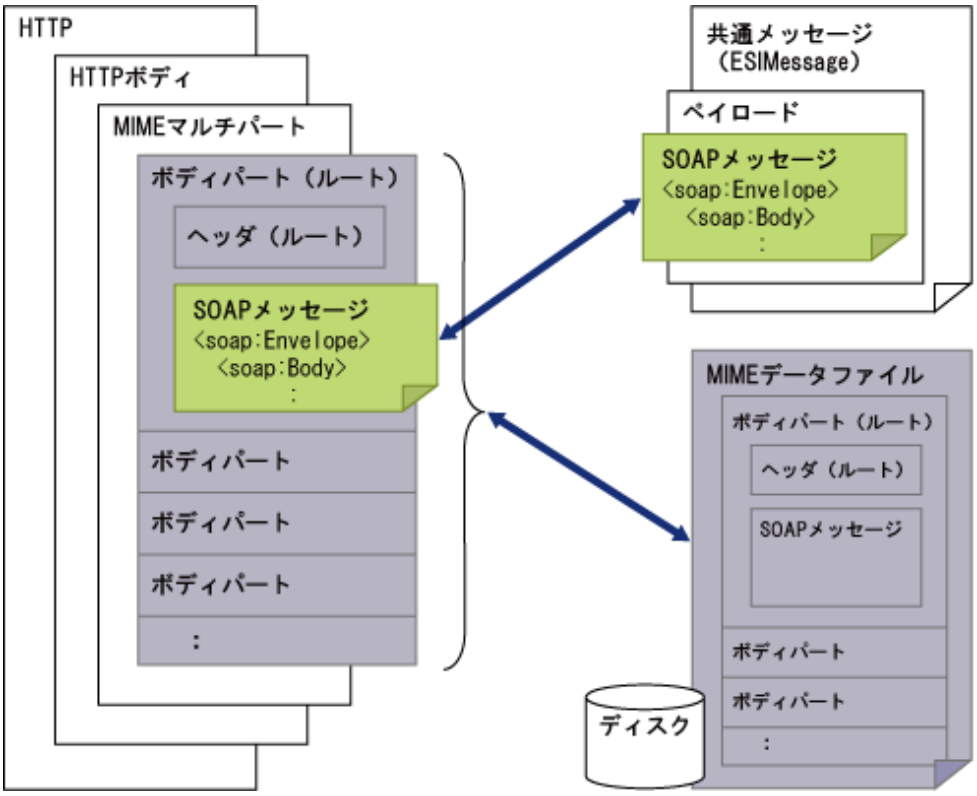
   SOAPメッセージを操作したい場合は、共通メッセージのペイロードに格納されたSOAPメッセージを操作してください。MIMEデータファイルのルートパートを操作しても、共通メッセージに反映されません。ただし、ルートパートのMIMEヘッダは共通メッセージに反映されるので、注意してください。

6. 添付データを操作した変更結果を、ファイルに書き出します

   操作した添付データを共通メッセージに反映させたい場合、操作結果をMIMEデータファイルに書き出します。書き出し先のファイルパスは、手順2で取得したファイルパスを利用してください。

添付ファイルの保持形式

ここでは、添付ファイルと共通メッセージの対応関係について説明します。



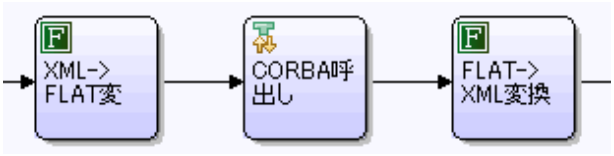
添付ファイル付きのSOAPメッセージを受信した場合も同様に、SOAPメッセージ（ルート要素 soap:Envelope）全体が共通メッセージのペイロードに格納されます。

添付ファイル部分は、添付ファイルが格納されているMIMEマルチパート全体（SOAPメッセージであるルートパート含む）をMIMEデータファイルとしてファイルに保存します。このMIMEデータファイルは、共通メッセージと関連付けられており、共通メッセージがメモリからなくなった時点でMIMEデータファイルも自動的に削除されます。

また、添付ファイル付きのSOAPメッセージを送信する場合は、共通メッセージのペイロード内容をSOAPメッセージ全体とし、保存された添付ファイルをSOAPメッセージに添付して送信を行います。

6.2 CORBAアダプタ利用時の定義 EE

CORBAアダプタ利用時のシーケンス定義は以下のようになります。



1. バイナリデータへの変換

   Formatmanagerを使用して受信したメッセージをFLAT形式のバイナリデータに変換します。受信したメッセージを加工せずにCORBAサーバに送信する場合、このステップは省略できます。

2. CORBAサーバ呼び出し

   CORBAアダプタを呼び出してCORBAサーバのサーバアプリケーションを呼び出します。CORBAアダプタの場合は、“同期サービス呼び出し”、または“ルーティングして同期サービス呼び出し”のメディエータファンクションを選択します。
   また、CORBAアダプタでは、CORBAのサーバアプリケーションの呼び出し方法に応じて、以下の2通りの方式を選択することができます。

   - RequestResponse方式
   - Oneway方式

3. バイナリデータからの変換

Formatmanagerを使用してCORBAサーバから受信したバイナリデータをサービス利用側アプリケーションで利用できるデータ形式に変換します。CORBAアダプタがOneway方式の場合、またはCORBAサーバからの受信データを加工しない場合、このステップは省略できます。

6.2.1 RequestResponse方式による呼び出し

RequestResponse方式では、CORBAのサーバアプリケーションからの処理結果を受け取ることができます。

1. CORBAのサーバアプリケーションを作成します。

RequestResponse方式で呼び出すことのできるCORBAのサーバアプリケーションは、sequence<octet>型のパラメタをinとoutで1つずつ持つ必要があります。

以下にRequestResponse方式の場合のIDL定義例を示します。モジュール名、インタフェース名、データ名、および、メソッド名は任意となります。

```
module samplemodule {
        interface sampleinterface {
                //
                // Type declaration
                //
                typedef sequence<octet> binary;

                //
                // Message invoke
                //
                void op1(
                        in binary inObject,
                        out binary outObject
                );
        };
};
```

上記の定義は、それぞれ下記の設定としました。

表6.1 IDL定義の設定値

項目	設定値
モジュール名	samplemodule
インタフェース名	sampleinterface
データ名	binary
メソッド名	op1

ネーミングサービスを設定します。

CORBAのサーバアプリケーションを呼び出すために以下の登録が必要です。

－ インプリメンテーションリポジトリへCORBAのサーバアプリケーションの登録

－ ネーミングサービスへCORBAのサーバアプリケーションのオブジェクトリファレンスの登録

以下に実行例を示します。

```
OD_impl_inst -ax def
OD_or_adm -c IDL:samplemodule/sampleinterface:1.0 -n samplemodule::sampleinterface
```

CORBAアプリケーション情報定義ファイル(def)の指定例

```
rep_id                  = IDL:samplemodule/sampleinterface:1.0
type                    = persistent
lang                    = JAVA
mode                    = SYNC_END
```

```
proc_conc_max           = 1
thr_conc_init           = 16
ior                     = 1.1
locale                  = UNICODE
```

2. CORBAアダプタのサービスエンドポイント定義を作成します。

   サービスエンドポイント定義の詳細にあるサーバアプリケーション情報のリクエスト方法でRequestResponse方式を選択します。



6.2.2 Oneway方式による呼び出し

Oneway方式では、CORBAのサーバアプリケーションからの処理を待たずにシーケンス処理を継続することができます。

1. CORBAアダプタのサービスエンドポイント定義を作成します。CORBAのサーバアプリケーションを作成します。

   Oneway方式で呼び出すことのできるCORBAのサーバアプリケーションは、sequence<octet>型のinパラメタを1つだけを持つ必要があります。

   以下にOneway方式の場合のIDL定義例を示します。モジュール名、インタフェース名、データ名、および、メソッド名は任意となります。

```
module samplemodule {
        interface sampleinterface {
                //
                // Type declaration
                //
                typedef sequence<octet> binary;

                //
                // Message invoke
                //
                oneway void op1(in binary seq1);
        };
};
```

   上記の定義は、それぞれ下記の設定としました。

表6.2 IDL定義の設定値

項目	設定値
モジュール名	samplemodule

項目	設定値
インタフェース名	sampleinterface
データ名	binary
メソッド名	op1

2. ネーミングサービスを設定します。

   CORBAのサーバアプリケーションを呼び出すために以下の登録が必要です。

   ー インプリメンテーションリポジトリへCORBAのサーバアプリケーションの登録

   ー ネーミングサービスへCORBAのサーバアプリケーションのオブジェクトリファレンスの登録

   以下に実行例を示します。

```
OD_impl_inst -ax def
OD_or_adm -c IDL:samplemodule/sampleinterface:1.0 -n samplemodule::sampleinterface
```

   CORBAアプリケーション情報定義ファイル(def)の指定例

```
rep_id                  = IDL:samplemodule/sampleinterface:1.0
type                    = persistent
lang                    = JAVA
mode                    = SYNC_END
proc_conc_max           = 1
thr_conc_init           = 16
ior                     = 1.1
locale                  = UNICODE
```

3. CORBAアダプタのサービスエンドポイント定義を作成します。

   サービスエンドポイント定義の詳細にあるサーバアプリケーション情報のリクエスト方法でOneway方式を選択します。



6.3 データベースアダプタ利用時の定義

データベースアダプタ(DBアダプタ)利用時のシーケンス定義は以下のようになります。



1. バイナリデータの設定
   DBアダプタを使用してバイナリデータをデータベースに格納する場合、ユーザプログラムを作成してバイナリデータを共通メッセージに格納します。バイナリデータを使用しない場合、このステップは省略できます。
2. CSV変換
   データベースを更新する場合、データベースアダプタに渡す共通メッセージのペイロードをCSV形式にします。Formatmanagerを使用して共通メッセージのペイロードをCSV形式に変換します。すでに共通メッセージのペイロードがCSV形式になっている場合、またはSQL文に値を動的に設定する必要がない場合、このステップは省略できます。
3. DBアダプタ呼出し
   DBアダプタを呼び出してデータベースにアクセスします。データベースの検索を行う場合、または復帰値があるストアドプロシージャを実行する場合は“同期サービス呼出しメディエータファンクション”を使用します。データベースの更新を行う場合、または復帰値がないストアドプロシージャを実行する場合は“同期サービス呼び出しメディエータファンクション”、“非同期サービス呼出しメディエータファンクション”のどちらかを使用します。
4. CSV変換
   データベースの呼出し結果はバイナリデータを除き共通メッセージのペイロードにCSV形式で格納されます。Formatmanagerを使用して共通メッセージのペイロードをCSV形式から別の形式に変換します。変換の必要がない場合、このステップは省略できます。
5. バイナリデータの取得
   データベースの呼出し結果にバイナリデータがある場合、ユーザプログラムを作成してバイナリデータを取り出します。バイナリデータを使用しない場合、このステップは省略できます。

ここでは、定義の例として次のデータベースへのアクセスを行うものとします。

■在庫表データベース

製品番号	製品名	在庫数量	倉庫番号
110	テレビ	85	2
111	テレビ	90	2
123	冷蔵庫	60	1
124	冷蔵庫	75	1
140	CDプレーヤー	120	2
212	テレビ	0	2
215	ビデオ	5	2
226	冷蔵庫	8	1
227	冷蔵庫	15	1
240	CDプレーヤー	25	2
243	CDプレーヤー	14	2
351	CD	2500	2

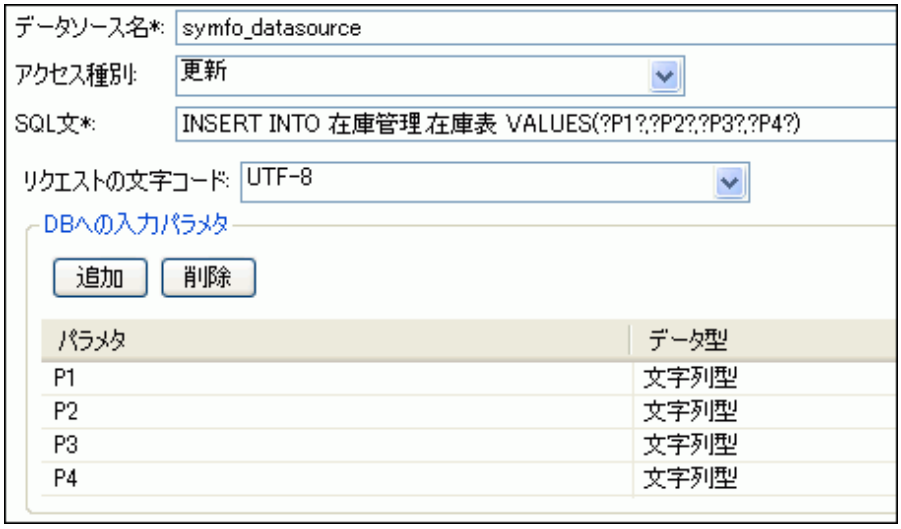
■在庫表データベースの列のデータ型

列名	列のデータ型
製品番号	SMALLINT
製品名	NCHAR(10)
在庫数量	INTEGER
倉庫番号	SMALLINT

6.3.1 データベースの更新

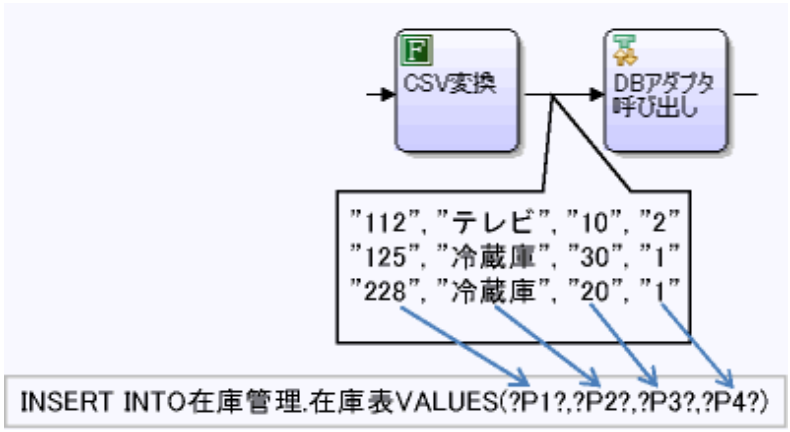
更新の例として在庫表テーブルにデータを格納する場合の定義について説明します。

1. DBアダプタのサービスエンドポイント定義を作成します。



列のデータ型がバイナリ以外の場合、サービスエンドポイントの入力パラメタのデータ型には文字列型を指定してください。

2. シーケンス定義を作成します。



DBアダプタに渡す共通メッセージのペイロードが上図のペイロードの場合、テーブルに3行レコードを追加します。

6.3.2 データベースの検索

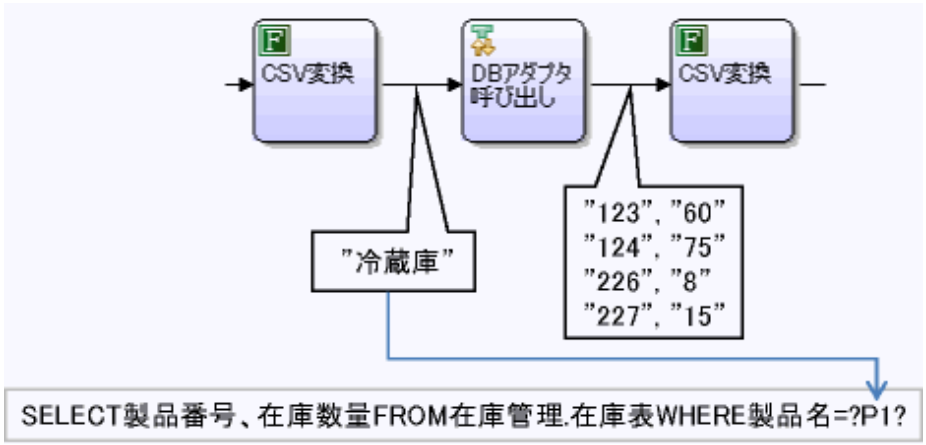
検索の例として在庫表テーブルからデータを取得する場合の定義について説明します。

1. DBアダプタのサービスエンドポイント定義を作成します。



2. 列のデータ型バイナリ以外の場合、サービスエンドポイントの入力パラメタおよび出力パラメタのデータ型には文字列型を指定してください。

3. シーケンス定義を作成します。



6.3.3 ストアドプロシージャの実行

ストアドプロシージャの例として指定された製品名の在庫合計を返すストアドプロシージャを実行する場合の定義について説明します。

この例では引数として製品名を渡すと在庫合計を返すストアドプロシージャを実行するものとします。

例えばこのストアドプロシージャに“冷蔵庫”を渡すと在庫合計の“158”を返します。

1. DBアダプタのサービスエンドポイント定義を作成します。



復帰値がある場合のストアドプロシージャ呼び出しCALL文の書き方は以下のようになります。

－ Oracle, SQL Serverの場合

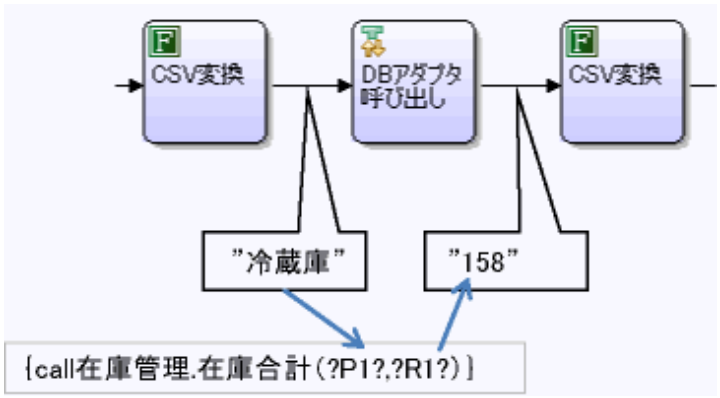
```
{?R1?=call プロシージャ名 (?P1?)}　(P1:入力パラメタ、R1:出力パラメタ)
```

－ Symfoware Serverの場合

```
{call プロシージャ名 (?P1?,?R1?)}　(P1:入力パラメタ、R1:出力パラメタ)
```

復帰値があるストアドプロシージャの場合、復帰値の型に合うJDBCデータ型を指定します。復帰値の型に合うJDBCデータ型についてはデータベースシステムのマニュアルを参照してください。

2. シーケンス定義を作成します。



注意

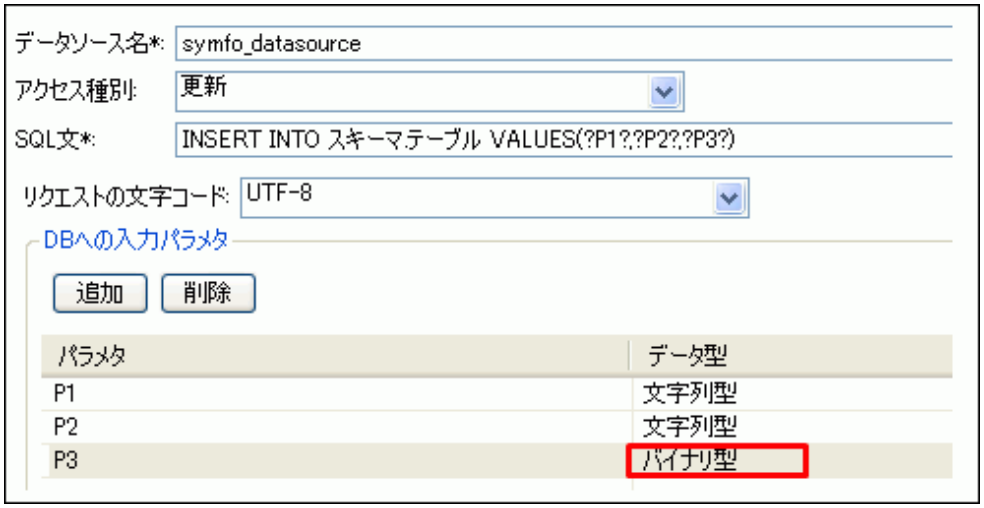
ISIのストアドプロシージャ呼出しでは、日付型、時間型、時刻型のパラメタの受け渡しはできません。文字列型で受け渡しを行ってください。

6.3.4 バイナリデータの使用

DBアダプタでバイナリデータを扱う場合、ユーザプログラムを作成します。

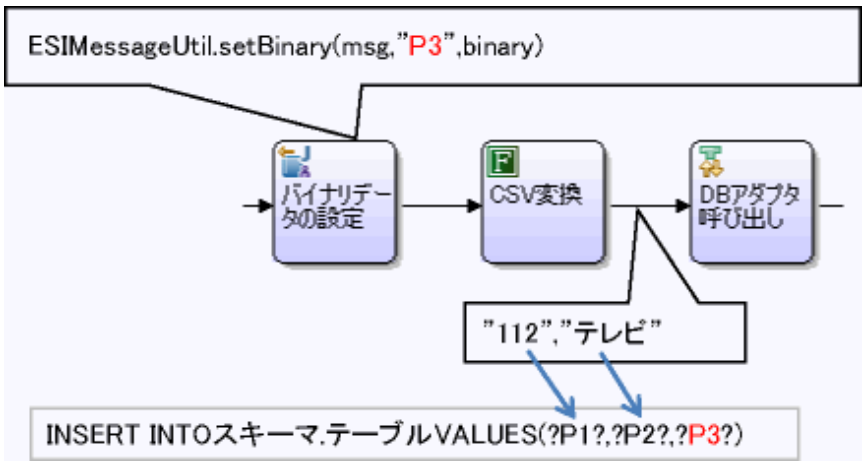
- 前提
  バイナリデータの設定/取得を行う場合、共通メッセージ(ESIMessage)が必要です。ユーザプログラムのメソッドの引数に必ず共通メッセージを加えてください。
- バイナリデータの設定
  バイナリデータを設定する場合、ISIの以下のメソッドを使用します。
  public void ESIMessageUtil.setBinary(ESIMessage msg, String key, byte[] binary)
- バイナリデータの取得
  バイナリデータを取得する場合、ISIの以下のメソッドを使用します。
  public byte[] ESIMessageUtil.removeBinary(ESIMessage msg, String key)

1. サービスエンドポイント定義を作成します。



この例ではP3(第3パラメタ)のデータ型をバイナリ型として指定しています。

2. シーケンス定義を作成します。



- バイナリ型はサービスエンドポイント定義で指定するパラメタの最後で指定してください。
- バイナリ型の更新を行う場合、DBアダプタに渡すCSVデータは1行にしてください。複数行のCSVデータをDBアダプタに渡した場合、同じバイナリが行数分挿入されます。

6.3.5 データベースアダプタでアクセス可能なデータ型

ここではデータベースアダプタでアクセス可能なデータ型について説明します。

Oracle

Oracleのデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型について記載します。

表6.3 Oracleのデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型

データベースのデータ型	データベースアダプタの定義で指定するデータ型
BINARY DOUBLE	文字列型
BINARY FLOAT	文字列型
BLOB	バイナリ型
CHAR	文字列型
CLOB	文字列型
DATE	文字列型
FLOAT	文字列型
LONG	文字列型
LONG RAW	バイナリ型
NCHAR	文字列型
NCLOB	文字列型
NUMBER	文字列型
NVARCHAR	文字列型
RAW	バイナリ型
ROWID	文字列型
TIMESTAMP	文字列型
TIMESTAMP WITH TIME ZONE	文字列型
VARCHAR	文字列型

SQL Server

SQL Serverのデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型について記載します。

表6.4 SQL Serverのデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型

データベースのデータ型	データベースアダプタの定義で指定するデータ型
BIGINT	文字列型
BINARY	バイナリ型
BIT	文字列型
CHAR	文字列型
DATETIME	文字列型
DECIMAL	文字列型
FLOAT	文字列型
IMAGE	バイナリ型
INT	文字列型

データベースのデータ型	データベースアダプタの定義で 指定するデータ型
MONEY	文字列型
NCHAR	文字列型
NTEXT	文字列型
NUMERIC	文字列型
NVARCHAR	文字列型
REAL	文字列型
SMALLDATETIME	文字列型
SMALLINT	文字列型
SMALLMONEY	文字列型
TEXT	文字列型
TIMESTAMP	文字列型
TINYINT	文字列型
UNIQUEIDENTIFIER	文字列型
VARBINARY	バイナリ型
VARCHAR	文字列型
XML	文字列型

Symfoware Server

Symfoware Serverのデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型について記載します。

表6.5 Symfoware Serverのデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型

データベースのデータ型	データベースアダプタの定義で 指定するデータ型
BLOB	バイナリ型
CHAR	文字列型
DATE	文字列型
DECIMAL	文字列型
DOUBLE	文字列型
FLOAT	文字列型
INTEGER	文字列型
NCHAR	文字列型
NCHAR VARYING	文字列型
NUMERIC	文字列型
REAL	文字列型
SMALLINT	文字列型
TIME	文字列型
TIMESTAMP	文字列型
VARCHAR	文字列型

HA Database Ready(Native SQLインタフェース)

HA Database Ready(Native SQLインタフェース)のデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型について記載します。

表6.6 HA Database Ready(Native SQLインタフェース)のデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型

データベースのデータ型	データベースアダプタの定義で指定するデータ型
BLOB	バイナリ型
CHAR	文字列型
DATE	文字列型
DECIMAL	文字列型
DOUBLE	文字列型
FLOAT	文字列型
INTEGER	文字列型
NCHAR	文字列型
NCHAR VARYING	文字列型
NUMERIC	文字列型
REAL	文字列型
SMALLINT	文字列型
TIME	文字列型
TIMESTAMP	文字列型
VARCHAR	文字列型

HA Database Ready(Open SQLインタフェース)

HA Database Ready(Open SQLインタフェース)のデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型について記載します。

表6.7 HA Database Ready(Open SQLインタフェース)のデータベースにアクセスするために指定するデータベースアダプタ定義のデータ型

データベースのデータ型	データベースアダプタの定義で指定するデータ型
BIGINT	文字列型
BYTEA	バイナリ型
CHARACTER	文字列型
CHARACTER VARYING	文字列型
DATE	文字列型
INTEGER	文字列型
MONEY	文字列型
NUMERIC	文字列型
REAL	文字列型
SMALLINT	文字列型
TEXT	文字列型
TIME	文字列型

データベースのデータ型	データベースアダプタの定義で 指定するデータ型
TIMESTAMP	文字列型

6.3.6 代替文字への変換ついて

UNICODE以外の文字コード系(例えば、Shift_JISやEUC-JP)の入出力を扱う場合、Javaのコード変換機能により代替文字への変換が行われることがあります。例えば、NEC特殊文字やIBM拡張文字(以下、特殊文字)について以下の例のような変換が行われます。

例

(株) ⇒ ?

髙 ⇒ ?

特殊文字を扱う場合は以下のようにしてください。

データ更新時

Formatmanagerを使用してUNICODEへの変換を行ってからDBアダプタを呼び出してください。

データ取得時

DBアダプタのサービスエンドポイント定義のレスポンスの文字コードをUTF-8で指定してデータを取得してください。このあとのステップでFormatmanagerを使用して文字コードの変換を行ってください。

6.4 MQアダプタ利用時の定義 EE

ここでは、WebSphere MQにメッセージを送受信する場合に必要なMQアダプタ利用時の定義について説明します。

- WebSphere MQのキューからメッセージを受信する
- WebSphere MQのキューへメッセージを送信する
- WebSphere MQのキューとRequestResponse方式で通信する

6.4.1 WebSphere MQのキューからメッセージを受信する

MQアダプタを使用してWebSphere MQのキューからメッセージを受信するための定義は以下のとおりです。

1. インバウンドルーティングの作成

   受信したWebSphere MQのメッセージをもとにインバウンドサービスエンドポイントを決定するため、インバウンドルーティングを作成します。

   詳細は“ISI Studio ヘルプ”の“ルーティングを操作する”を参照してください。

   作成したインバウンドルーティング名はインバウンドMQアダプタ設定ファイルに指定します。インバウンドMQアダプタ設定ファイルにインバウンドルーティング名を指定するには、以下の2つの方法があります。

   ー esisetup_mqrcvコマンドの引数に指定する

   ー インバウンドMQアダプタ設定ファイルを直接編集する

   esisetup_mqrcv コマンドについては、“ISI リファレンス”の“esisetup_mqrcv”コマンドを参照してください。

   インバウンドMQアダプタ設定ファイルについては、“ISI 導入ガイド”の“MQアダプタ利用時の設定”を参照してください。

2. インバウンドサービスエンドポイントの作成

   サービス利用側からISIを実行するため、インバウンドサービスエンドポイントを作成します。インバウンドサービスエンドポイントの接続方式は“MQ”を選択します。

詳細は“ISI Studio ヘルプ”の“サービスエンドポイントを操作する”を参照してください。

3. シーケンスの作成

   サービスと連携するためのシーケンスを作成します。

   詳細は“ISI Studio ヘルプ”の“シーケンスを操作する”を参照してください。

作成した定義はサーバへ登録・配置してください。登録・配置の詳細は“ISI Studio ヘルプ”の“ISIサーバに定義を登録・配置する”を参照してください。

WebSphere MQのキューからのメッセージ受信を確認するには、WebSphere MQのキューにテストメッセージを書き込み、シーケンスが正しく実行されることを確認してください。
WebSphere MQのキューにテストメッセージを書き込む方法は、WebSphere MQの関連マニュアルを参照してください。
シーケンスが正しく実行されたかどうかは、ISI運用管理コンソールから確認できます。詳細は、“ISI 運用ガイド”の“実行モニタ”を参照してください。

6.4.2 WebSphere MQのキューへメッセージを送信する

MQアダプタを使用してWebSphere MQのキューへメッセージを送信するための定義は以下のとおりです。

1. アウトバウンドサービスエンドポイントの作成

   WebSphere MQのキューへメッセージを送信するため、アウトバウンドサービスエンドポイントを作成します。アウトバウンドサービスエンドポイントの接続方式は“MQ”を選択します。

   詳細は“ISI Studio ヘルプ”の“サービスエンドポイントを操作する”を参照してください。

2. シーケンスの作成

   1で作成したサービスエンドポイントを呼び出すシーケンスを作成します。

   サービスエンドポイントを呼び出すメディエータファンクションには、“同期サービス呼び出し”または、“ルーティングして同期サービス呼び出し”のメディエータファンクションを利用します。

   詳細は“ISI Studio ヘルプ”の“シーケンスを操作する”を参照してください。

作成した定義はサーバへ登録・配置してください。登録・配置の詳細は“ISI Studio ヘルプ”の“ISIサーバに定義を登録・配置する”を参照してください。

WebSphere MQのキューへのメッセージ送信を確認するには、ISIのシーケンスを動作させ、WebSphere MQのキューにメッセージが送信されていることを確認してください。
ISIのシーケンスを動作させるには、ISI Studioのシミュレーション機能を利用できます。詳細は“ISI Studio ヘルプ”の“シーケンスの動作を確認する”を参照してください。
WebSphere MQのキューにメッセージが送信されていることを確認する方法は、WebSphere MQの関連マニュアルを参照してください。

ポイント

アウトバウンドのサービスエンドポイント呼び出し後のステップでエラーが発生した場合、メッセージが冗送する可能性があります。

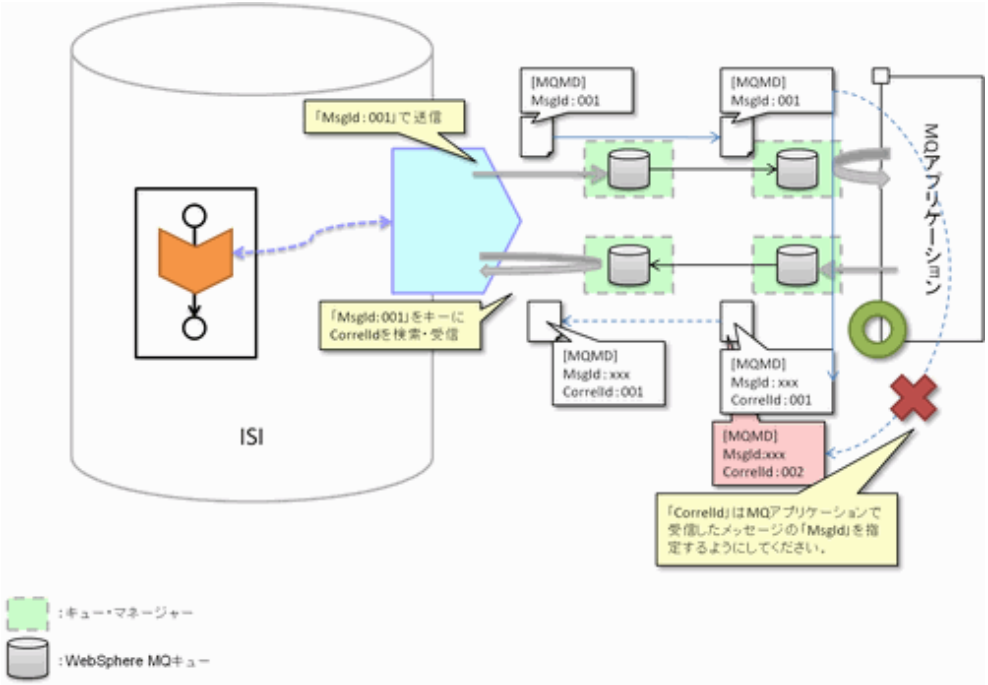
冗送を防止するためには、サービス提供側アプリケーションで冗送を検知・対処してください。

6.4.3 WebSphere MQのキューとRequestResponse方式で通信する

ここでは、MQアダプタにおけるRequestResponse方式の通信の実現方法および定義について説明します。

RequestResponse方式通信の実現方法

ISIのMQアダプタにおけるRequestResponse方式の通信は、MQMDのメッセージID(MsgId)および相関ID(CorrelId)を利用して実現されます。



RequestResponse方式では、ISIからMQキューへ要求メッセージを送信する場合、メッセージID(MsgId)を自動的に採番します。MQキューからの応答メッセージ受信時は、採番したメッセージID(MsgId)を利用し、応答メッセージの相関ID(CorrelId)に採番したメッセージID(MsgId)と同じものがあれば、MQキューからの応答メッセージを受信します。

ポイント

- サービス提供側のMQアプリケーションでは、受信したメッセージID(MsgId)を相関ID(CorrelId)にコピーするようにしてください。
- ユーザが採番したメッセージID(MsgId)をMQMDマッピング機能で連携させる場合、nullまたは空文字を指定しないでください。nullまたは空文字が指定された場合、ISIがMsgIdを新規に採番します。
- ISIの共通メッセージのヘッダにすでにMQMDのメッセージID(MsgId)が設定されている場合は、設定されているメッセージID(MsgId)が通信に利用されます。

応答メッセージの待ち時間

ISIのMQアダプタにおけるRequestResponse方式の通信では、要求メッセージの送信(Request)後即座に応答メッセージの受信(Response)を開始します。ただし、キューを用いた双方向の通信となるため、一定時間応答メッセージの到着を待ち合わせます。待ち時間は、サービスエンドポイント定義またはMQアダプタ設定ファイルで指定可能です。両方に定義された場合、サービスエンドポイント定義で指定した値が有効となります。

待ち時間以内に応答メッセージが返ってこなかった場合、通信が失敗したとみなされシーケンスがエラーとなります。エラー情報はOSのシステムログ、ISIのトレースログに出力されます。

ポイント

サービス提供側のMQアプリケーションの処理遅延のためにタイムアウトが発生した場合、応答メッセージが遅れてMQキューに到着する場合があります。この場合、遅れて到着した応答メッセージは受信されずにMQキューに残り続けます。

エラー情報に出力された情報を元に、送れて到着した応答メッセージをMQキューから削除してください。MQキューからメッセージを削除する方法は、WebSphere MQの関連マニュアルを参照してください。

プロセスダウン・サーバダウンが発生した場合

MQのRequestResponse方式通信では、REQUESTキューとRESPONSEキューの2つのキューを利用します。

REQUESTキューへのメッセージ送信が完了しなければ、サービス提供側アプリケーションはキューからメッセージを受信することが出来ません。そのため、MQアダプタはMQキューとの接続ごとに通信を完結させる必要があり、結果として、要求メッセージの送信（Request）と応答メッセージの受信（Response）のそれぞれでトランザクションが分かれます。

REQUESTキューへの要求メッセージ送信が完了すると、MQアダプタはREQUESTキューへの送信処理をコミットします。コミットが完了すると、MQアダプタはRESPONSEキューに接続し、応答メッセージの到着を待ち合わせますが、応答メッセージの受信が完了する前にMQアダプタに何か異常が発生した場合には、RESPONSEキュー内にメッセージが残り続けることがあります。

そのため運用時に異常が発生した場合には、連携しているMQキューに滞留している電文がないか調査を実施してください。調査方法は、WebSphere MQの関連マニュアルを参照してください。

ポイント

- 応答メッセージの受信トランザクションの処理中にプロセスダウンまたはサーバダウンが発生した場合、ISIのシーケンスではタイムアウトが発生し、シーケンスの実行に失敗します。
- サービス提供側のMQアプリケーションは2重送信を考慮して設計してください。

RequestResponse方式の定義

MQアダプタを使用してWebSphere MQのキューとRequestResponse方式で通信するための定義は以下のとおりです。

1. アウトバウンドサービスエンドポイントの作成

   WebSphere MQのキューへメッセージを送受信するため、アウトバウンドサービスエンドポイントを作成します。アウトバウンドサービスエンドポイントの接続方式は“MQ”を選択します。

   詳細は“ISI Studio ヘルプ”の“サービスエンドポイントを操作する”を参照してください。

2. シーケンスの作成

   1で作成したサービスエンドポイントを呼び出すシーケンスを作成します。

   サービスエンドポイントを呼び出すメディエータファンクションには、“同期サービス呼び出し”または、“ルーティングして同期サービス呼び出し”のメディエータファンクションを利用します。

   詳細は“ISI Studio ヘルプ”の“シーケンスを操作する”を参照してください。

作成した定義はサーバへ登録・配置してください。登録・配置の詳細は“ISI Studio ヘルプ”の“ISIサーバに定義を登録・配置する”を参照してください。

WebSphere MQのキューとのRequestResponse方式通信を確認するには、ISIのシーケンスを動作させ、シーケンスが正しく実行されることを確認してください。

ISIのシーケンスを動作させるには、ISI Studioのシミュレーション機能を利用できます。詳細は“ISI Studio ヘルプ”の“シーケンスの動作を確認する”を参照してください。

6.4.4 MQ連携時のメッセージマッピング

WebSphere MQのメッセージは、ユーザデータが格納されるアプリケーション・データ部分と、メッセージ・ヘッダー(MQMD)部分で構成されています。各データのISIでの扱いは以下の通りです。

WebSphere MQのメッセージの詳細は、WebSphere MQの関連マニュアルを参照してください。



アプリケーション・データ

ISIでWebSphere MQのキューからメッセージを受信する場合、WebSphere MQのメッセージのアプリケーション・データ部分がISIの共通メッセージのペイロードにマッピングされます。

ISIからWebSphere MQのキューへメッセージを送信する場合、ISIの共通メッセージのペイロードがWebSphere MQのメッセージのアプリケーション・データ部分にマッピングされます。

メッセージ・ヘッダー(MQMD)

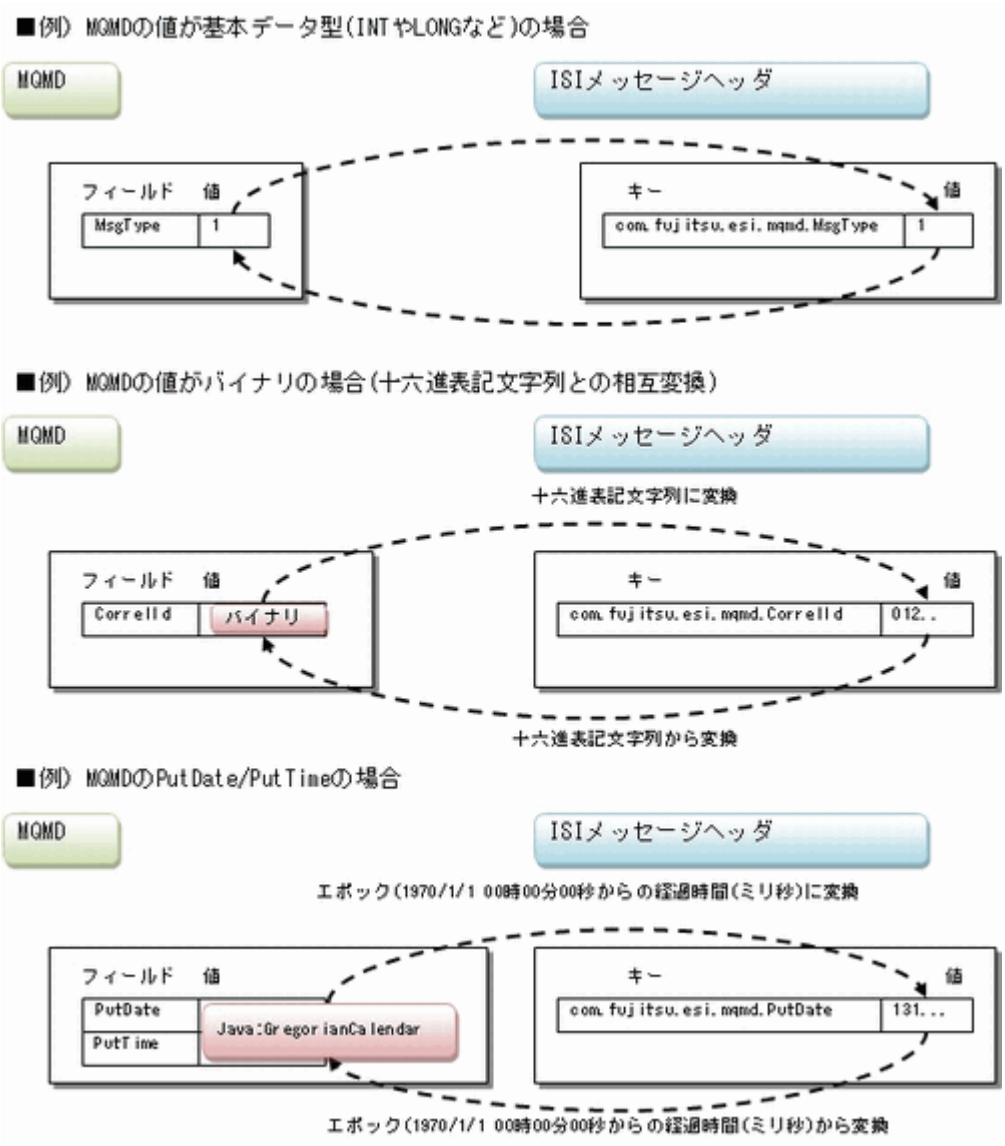
ISIからWebSphere MQのキューへメッセージを送信する場合、ISIの共通メッセージのヘッダに設定されたMQMD用のデータが、MQメッセージのフィールドにマッピングされます。ISIの共通メッセージのヘッダにMQMD用のデータが設定されていない場合、“表6.8 メッセージ・ヘッダー(MQMD)の設定値”に示す内容でMQMDの値が自動的に設定されます。

ISIがMQキューからメッセージを受信する場合、MQメッセージのMQMDのフィールドがISIの共通メッセージのヘッダに設定されます。

MQMDマッピングの動作

ここでは、MQMDマッピングの動作の詳細について説明します。

ISIの共通メッセージのヘッダとMQMDのフィールドを相互にマッピングする場合、MQMDのフィールドのデータ型に応じて以下のように動作します。ISIの共通メッセージのヘッダとMQMDのフィールドの対応関係については“MQMDのフィールドとISIの共通メッセージヘッダのキーの対応関係”を参照してください。



・ MQMDの値が基本データ型の場合

MQMDの値が基本データ型の場合、その値は対応するISIの共通メッセージのヘッダにそのままマッピングされます。

同様にISIの共通メッセージのヘッダの値も、対応するMQMDにそのままマッピングされます。

・ MQMDの値がバイナリ型(byte[])の場合

MQMDの値がバイナリ型（byte[]）の場合、その値は十六進表記文字列(String)に変換され、ISIの共通メッセージのヘッダにマッピングされます。

逆に、ISIの共通メッセージのヘッダの値は、十六進表記文字列(String)から使用してバイナリ型(byte[])に変換され、MQMDにマッピングされます。

－ マッピング例

MQMD値（バイナリ型）：0x0123456789ABCDEF

ISIでの形式（文字列型）：0123456789ABCDEF

例えば、MQMDの「CorrelId」ヘッダの値「0x0123456789ABCDEF」（バイナリ型）は、十六進表記文字列に変換されISIメッセージヘッダの「com.fujitsu.esi.mqmd.CorrelId」の値「0123456789ABCDEF」（文字列）にマッピングされます。逆に、ISIメッセージヘッダの「com.fujitsu.esi.mqmd.CorrelId」の値「0123456789ABCDEF」（文字列）は十六進表記文字列からバイナリ型（byte[]）の値「0x0123456789ABCDEF」に変換され、MQMDの「CorrelId」ヘッダにマッピングされます。

・ MQMDの値がPutDate/PutTimeの場合

MQMDの「PutDate」フィールドおよび「PutTime」フィールドの値は、WebSphere MQのJava用APIで「java.util.GregorianCalendar」の形式で保持しています。このため、MQMDの「PutDate」および「PutTime」の値は、エポック（1970/1/1 0時00分00秒）からの経過時間（ミリ秒）の文字列に変換され、ISIの共通メッセージのヘッダにマッピングされます。

逆に、ISメッセージのヘッダ「com.fujitsu.esi.mqmd.PutDate」の値は、エポックからの経過時間(ミリ秒)を表す文字列から「java.util.GregorianCalendar」の形式に変換され、MQMDにマッピングされます。

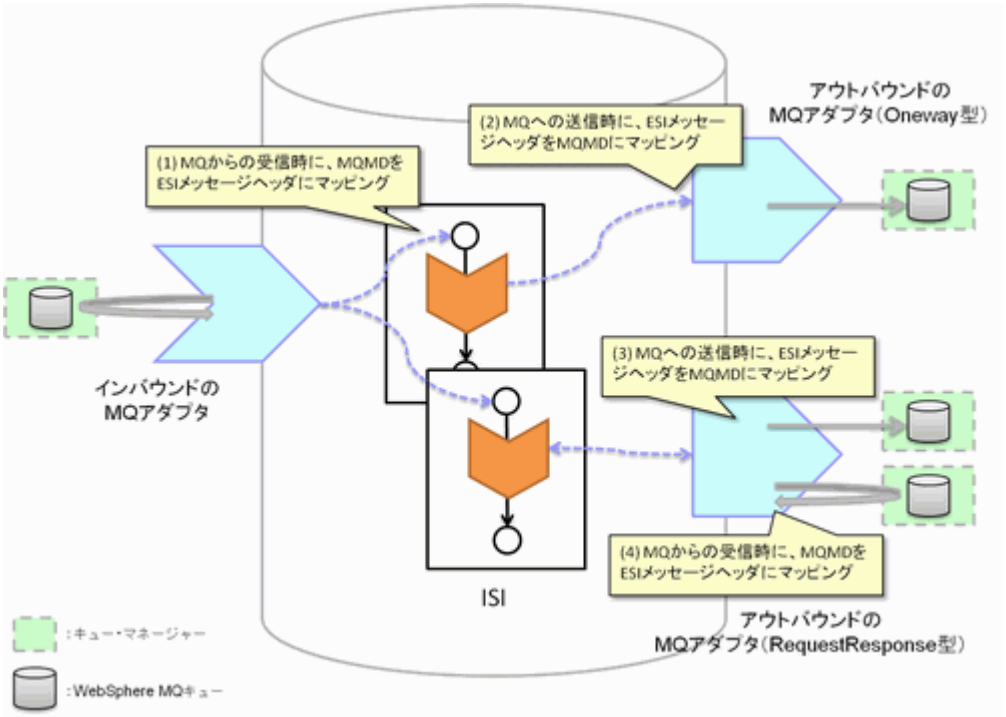
ポイント

－ MQMDの値がバイナリ型(byte[])のデータをユーザアプリケーションで直接編集する場合、十六進表記文字列を設定してください。

－ MQMDの「PutDate」および「PutTime」の値は、WebSphere MQ上1/100秒までしか保持されません。このため、ISIで上記経過時間「1310698800845」を指定した場合、一度MQキューを経由すると1/1000秒の単位が0で置き換えられ「1310698800840」となります。

MQMDマッピングが動作するタイミング

ここでは、MQMDマッピングが動作するタイミングについて説明します。



1. MQからの受信時

   WebSphere MQからメッセージを受信したタイミングで、MQMDのフィールドをISIの共通メッセージヘッダにマッピングします。

2. MQへの送信時(Oneway方式)

   ISIからWebSphere MQへOneway方式でメッセージを送信したタイミングで、ISIの共通メッセージヘッダにMQMD用のキーが設定されている場合、設定されている値をMQMDのフィールドにマッピングします。

3. MQへの送信時(RequestResponse方式の要求メッセージ送信)

   ISIからWebSphere MQへRequestResponse方式の要求メッセージを送信したタイミングで、ISIの共通メッセージヘッダにMQMD用のキーが設定されている場合、設定されている値をMQMDのフィールドにマッピングします。

4. MQへの応答時(RequestResponse方式の応答メッセージ受信)

   WebSphere MQからRequestResponse方式の応答メッセージを受信したタイミングで、MQMDのフィールドをISIの共通メッセージヘッダにマッピングします。

   MQMDマッピングを実施しないで運用したい場合、以下の2通りの方法があります。

・ 環境設定ファイル(ESISystemConfig.properties)に設定する方法

  ISIの環境設定ファイル(ESISystemConfig.properties)に設定する方法です。セットアップしたMQアダプタ全てに反映されます。

  キー名:com.fujitsu.esi.mqmd.mapping.compatibility

  指定値:true

  設定の詳細については、“ISI 導入ガイド”の“環境のカスタマイズ”を参照してください。

・ MQアダプタ設定ファイルに項目を設定する方法

  MQアダプタ設定ファイルに設定する方法です。セットアップしたMQアダプタ毎に反映されます。

  設定の詳細については“ISI 導入ガイド”の“MQアダプタ利用時の設定”を参照してください。

MQMDのフィールドとISIの共通メッセージヘッダのキーの対応関係

MQMDのフィールド名とISIの共通メッセージヘッダのキー名の対応関係は以下の通りです。

MQMDフィールド名	ISIの共通メッセージヘッダ キー名	備考
Report	com.fujitsu.esi.mqmd.Report	NULL指定不可(注1)

MQMDフィールド名	ISIの共通メッセージヘッダ キー名	備考
MsgType	com.fujitsu.esi.mqmd.MsgType	NULL指定不可(注1)
Expiry	com.fujitsu.esi.mqmd.Expiry	NULL指定不可(注1)
Feedback	com.fujitsu.esi.mqmd.Feedback	NULL指定不可(注1)
Encoding	com.fujitsu.esi.mqmd.Encoding	NULL指定不可(注1)
CodedCharSetId	com.fujitsu.esi.mqmd.CodedCharSetId	NULL指定不可(注1)
Format	com.fujitsu.esi.mqmd.Format	
Priority	com.fujitsu.esi.mqmd.Priority	NULL指定不可(注1)
Persistence	com.fujitsu.esi.mqmd.Persistence	NULL指定不可(注1)
MsgId	com.fujitsu.esi.mqmd.MsgId	バイナリ型(注2)
CorrelId	com.fujitsu.esi.mqmd.CorrelId	バイナリ型(注2)
BackoutCount	com.fujitsu.esi.mqmd.BackoutCount	ユーザ変更不可(注3)
ReplyToQ	com.fujitsu.esi.mqmd.ReplyToQ	
ReplyToQMgr	com.fujitsu.esi.mqmd.ReplyToQMgr	(注4)
UserIdendifier	com.fujitsu.esi.mqmd.UserIdentifier	
AccountingToken	com.fujitsu.esi.mqmd.AccountingToken	バイナリ型(注2)
ApplIdentityData	com.fujitsu.esi.mqmd.ApplIdentityData	
PutApplType	com.fujitsu.esi.mqmd.PutApplType	NULL指定不可(注1)
PutApplName	com.fujitsu.esi.mqmd.PutApplName	
PutDate	com.fujitsu.esi.mqmd.PutDate	エポックからの経過時間(ミリ秒)で保持
PutTime		
ApplOriginData	com.fujitsu.esi.mqmd.ApplOriginData	
以下は、サービスエンドポイント定義の“メッセージ・ヘッダー(MQMD)のバージョン”に“バージョン2”が指定された場合に付加されるフィールドです。		
GroupId	com.fujitsu.esi.mqmd.GroupId	バイナリ型(注2) ユーザ変更不可(注3)
MsgSeqNumber	com.fujitsu.esi.mqmd.MsgSeqNumber	ユーザ変更不可(注3)
Offset	com.fujitsu.esi.mqmd.Offset	ユーザ変更不可(注3)
MsgFlags	com.fujitsu.esi.mqmd.MsgFlags	NULL指定不可(注1)
OriginalLength	com.fujitsu.esi.mqmd.OriginalLength	ユーザ変更不可(注3)

注1:値としてNULLが指定された場合は、マッピングが実施されず“メッセージ・ヘッダー(MQMD)の設定値”に示す内容が自動的に設定されます。

注2:十六進表記文字列、または指定された文字コード(デフォルトは十六進表記文字列)を使用して、文字列型(String)とバイナリ型(byte[])を相互に変換します。

注3:WebSphere MQからのメッセージ受信時にはMQMDの値をISIの共通メッセージヘッダにマッピングしますが、WebSphere MQへのメッセージ送信時にはISIの共通メッセージヘッダの値をMQMDにマッピングできません。

注4: MQメッセージの送信時に「ReplyToQMgr」の値に空文字(“”)を指定した場合、WebSphere MQがデフォルトの値を自動的に付与します。

表6.8 メッセージ・ヘッダー(MQMD)の設定値

フィールド名	設定値
StrucId	MQMD_STRUC_ID

フィールド名	設定値
Version	サービスエンドポイント定義の“メッセージ・ヘッダー(MQMD)のバージョン”に指定された内容が設定されます。 詳細については、“ISI Studioヘルプ”を参照してください。
Report	MQRO_NONE
MsgType	MQMT_DATAGRAM
Expiry	MQEI_UNLIMITED
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	MQCCSI_Q_MGR
Format	MQFMT_NONE
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_PERSISTENCE_AS_Q_DEF
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	空白
ReplyToQMgr	空白
UserIdendifier	空白
AccountingToken	MQACT_NONE
ApplIdentityData	空白
PutApplType	MQAT_NO_CONTEXT
PutApplName	空白
PutDate	空白
PutTime	空白
ApplOriginData	空白
以下は、サービスエンドポイント定義の“メッセージ・ヘッダー(MQMD)のバージョン”に“バージョン2”が指定された場合に付加されるフィールドです。	
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

各フィールドの詳細は、WebSphere MQの関連マニュアルを参照してください。

付録A　定義の概要

ここでは、ISIの機能利用時に必要な定義について説明します。

A.1　定義の種類

ISIの機能を利用するために、以下の定義を作成し、ISIサーバへ登録します。定義の作成・登録は、ISI Studioを利用します。ISI Studioの操作方法については、“ISI Studio ヘルプ”を参照してください。
なお、コマンドを使用して定義を登録することもできます。コマンドを使用した定義の登録については、“A.2 定義の作成と登録”を参照してください。

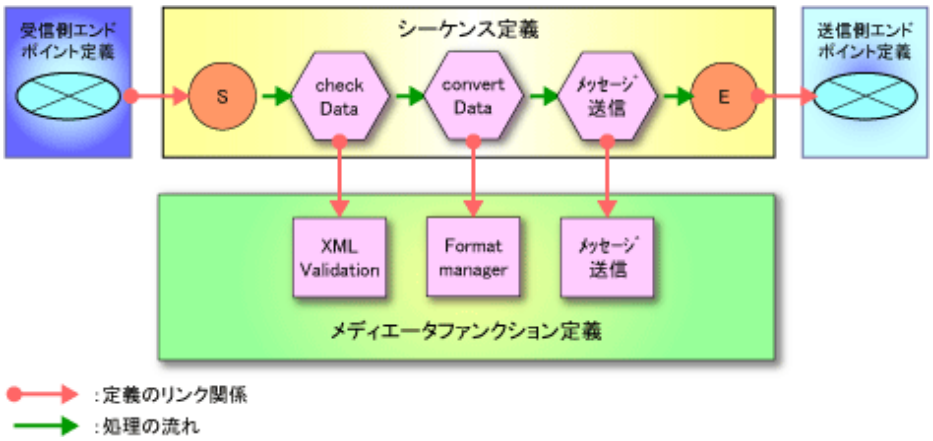
- サービスエンドポイント定義
  サービス提供側やサービス利用側と連携するための情報を定義します。
- シーケンス定義
  メディエータファンクションの実行順やメディエータファンクション実行時に必要な情報を定義します。
- メディエータファンクション定義
  メディエータファンクションのインタフェース情報を定義します。
  ユーザプログラムを呼び出す場合は、メディエータファンクション定義を追加します。
- ルーティング定義
  サービス利用側から受信したメッセージの内容によって、接続するインバウンドサービスエンドポイント、アウトバウンドサービスエンドポイント、シーケンスを特定するための情報を定義します。
- キュー間転送定義
  キュー間転送を行うために必要な情報を定義します。

注意

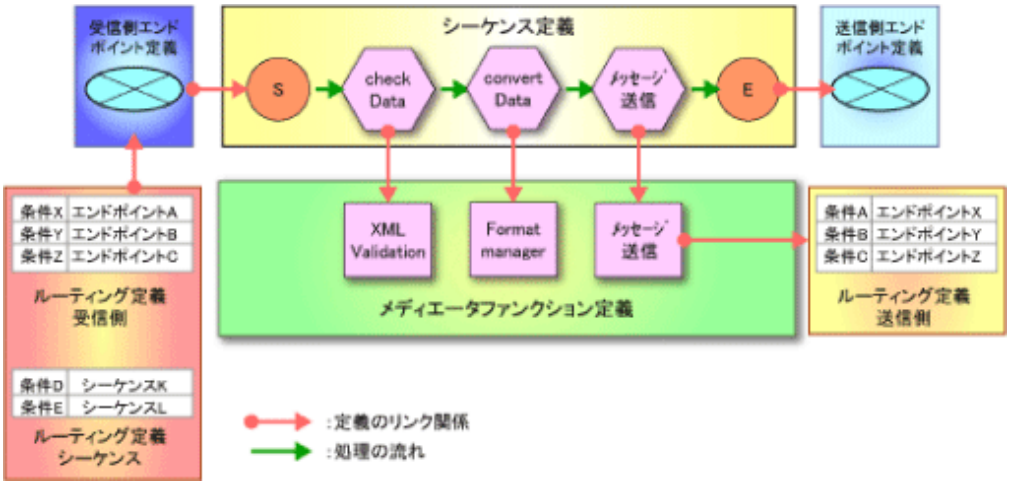
- Systemwalker SQC連携機能利用時に使用するグループ定義は、ISI Studioで作成・登録できません。
- ルーティング定義は、インバウンドルーティング、アウトバウンドルーティング、シーケンスルーティングで使用します。なお、インバウンドルーティングでは、SOAP、CORBA EE、MQ EE、キュー間転送 EE、MQD S EE、およびカスタムアダプタを利用してサービス利用側からメッセージを送信する場合に定義します。

ISIでの各定義間のリンク関係は、以下のとおりです。

図A.1 定義間のリンク関係

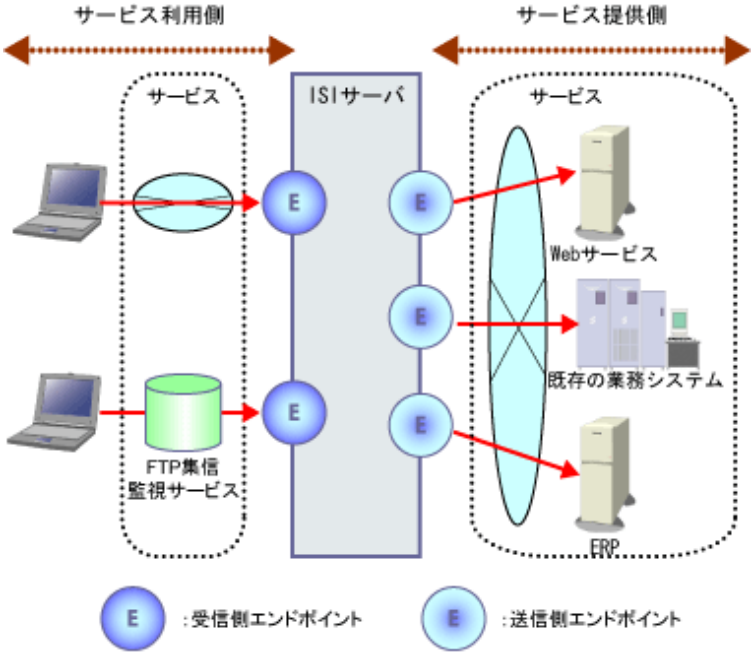


図A.2 定義間のリンク関係(メッセージによってサービスエンドポイント、シーケンスを振り分ける場合)



サービスエンドポイント定義には、サービス利用側とISIとの連携を定義するインバウンドサービスエンドポイント定義、ISIとサービス提供側との連携を定義するアウトバウンドサービスエンドポイント定義があります。

図A.3 サービスエンドポイント定義の位置付け

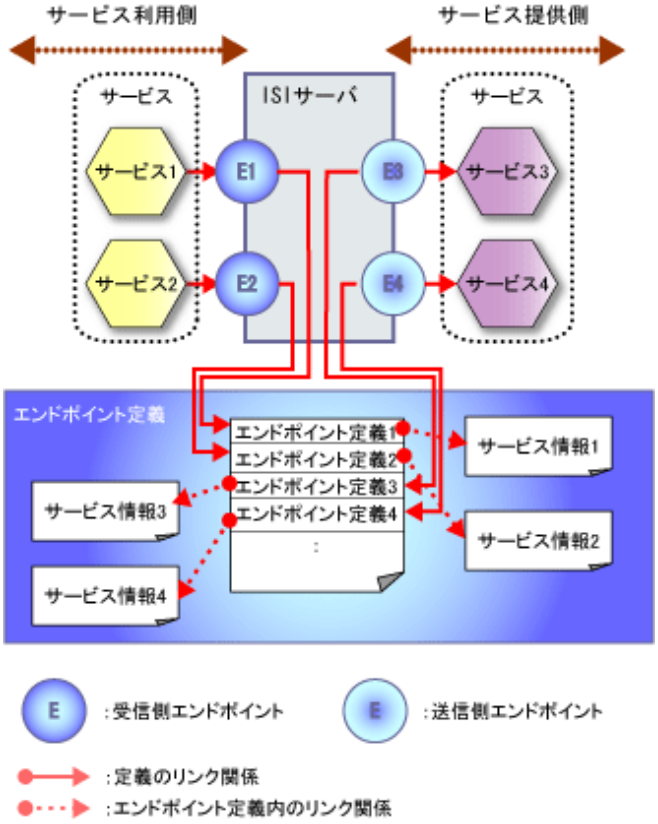


A.1.1 サービスエンドポイント定義

サービスエンドポイント定義は、サービスとISIを連携するための情報を定義します。
サービスエンドポイント定義は、2種類の情報で構成されます。

- サービスエンドポイント定義
  サービス利用側とISI、サービス提供側とISIを接続するための情報を定義します。
  ISIシステム内に1つ定義します。
- サービス情報
  サービスエンドポイントから呼び出すサービスの情報を定義します。
  サービスごとに定義します。1サービスを1ファイルで定義します。

図A.4 サービスエンドポイント定義のリンク関係



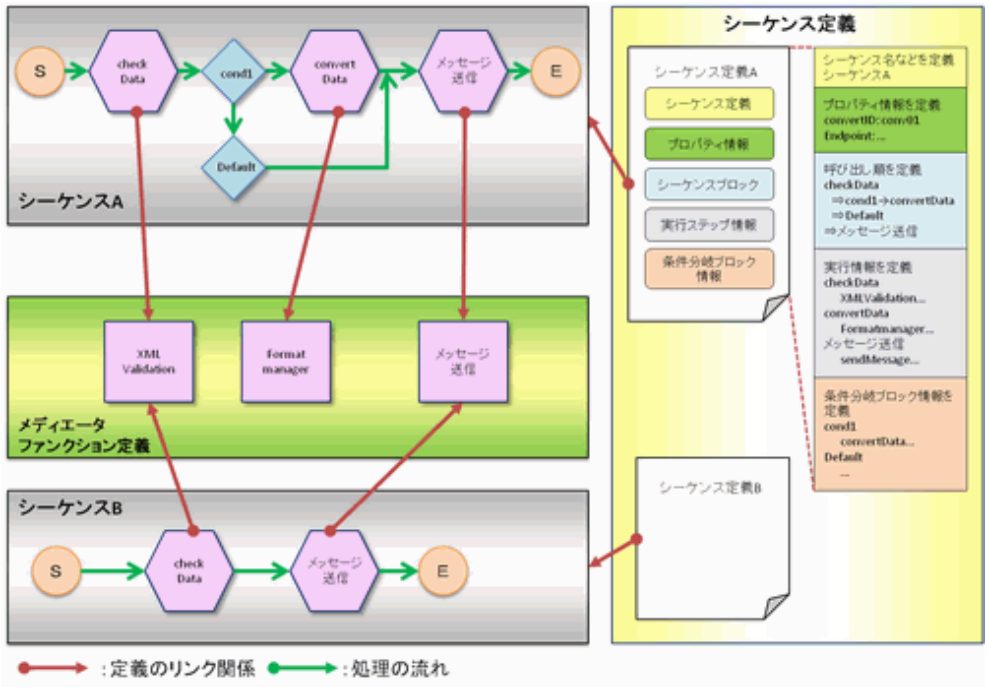
A.1.2 シーケンス定義

シーケンス定義は、メディエータファンクションの実行順やメディエータファンクション実行時に必要な情報を定義します。

シーケンス定義は、シーケンスごとに1シーケンスを1ファイルで定義します。
シーケンス定義は、5種類の情報で構成されます。

- シーケンス定義
  シーケンス名などを定義します。
- プロパティ情報
  シーケンスで利用するプロパティを定義します。指定したプロパティ情報は、シーケンス上を流れる共通メッセージのヘッダパラメタに設定されます。
- シーケンスブロック
  メディエータファンクションの呼出し順を定義します。エラー時のキャンセル処理の有無も定義します。
- 実行ステップ情報
  メディエータファンクションの呼出し情報、各実行ステップの動作内容を定義します。
- 条件分岐ブロック情報
  条件分岐ブロックで利用するデータのタイプ、ポジション情報、分岐経路の条件を定義します。

図A.5 シーケンス定義のリンク関係



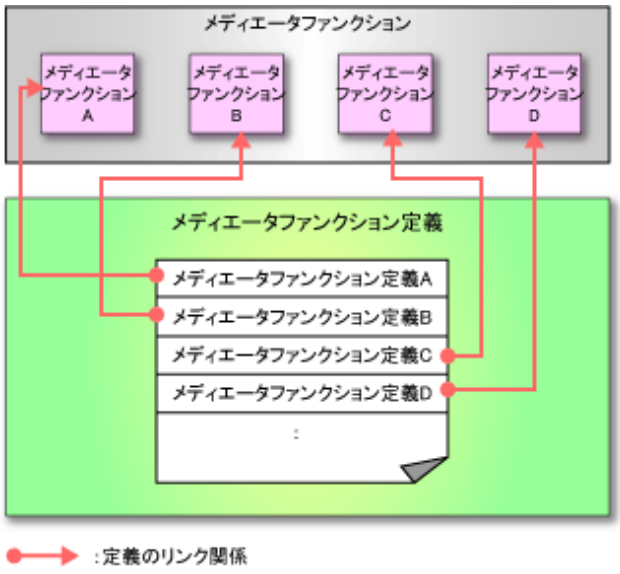
A.1.3 メディエータファンクション定義

メディエータファンクションのインタフェース情報を定義する定義体です。シーケンスからメディエータファンクションを呼び出す際に使用するメディエータファンクションのクラス名、メソッド名、パラメタ、復帰値などの情報が定義されています。

ポイント

標準提供のメディエータファンクションを利用する場合は、特に変更する必要はありません。
追加でユーザプログラム呼出しをする場合には、メディエータファンクション定義を設定してください。

図A.6 メディエータファンクション定義のリンク関係



A.1.4 ルーティング定義

ルーティング定義は、サービス利用側から受信したメッセージの内容によって、接続するインバウンドサービスエンドポイント、アウトバウンドサービスエンドポイント、シーケンスを特定するための情報(振分け条件とサービスエンドポイント、シーケンス)を定義します。1サービスを1ファイルで定義します。
ルーティング定義には、以下のルーティングタイプがあります。

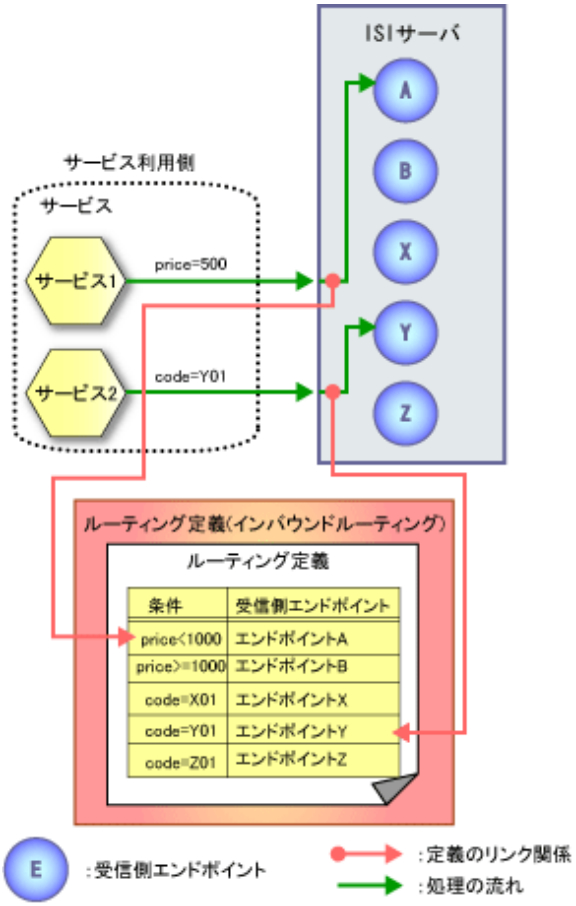
- インバウンドルーティング
- アウトバウンドルーティング
- シーケンスルーティング

振分け条件は、ルーティング定義内に定義された順番で判断されます。指定された位置に判断するデータが存在しない場合は、次の振分け条件に移ります。

インバウンドルーティング

接続するインバウンドサービスエンドポイントを特定するための情報(振分け条件と接続するインバウンドサービスエンドポイント)を定義します。
共通APIを使用することなく(サービスエンドポイント名を指定できない場合)、ISIサーバに接続する場合に利用します。

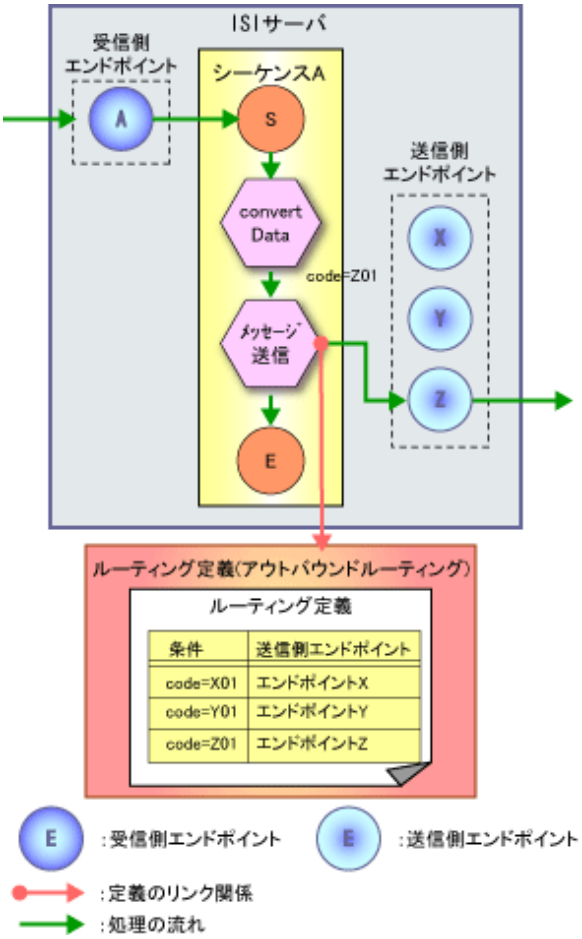
図A.7 ルーティング定義(インバウンドルーティング)のリンク関係



アウトバウンドルーティング

接続するアウトバウンドサービスエンドポイントを特定するための情報(振分け条件と接続するアウトバウンドサービスエンドポイント)を定義します。

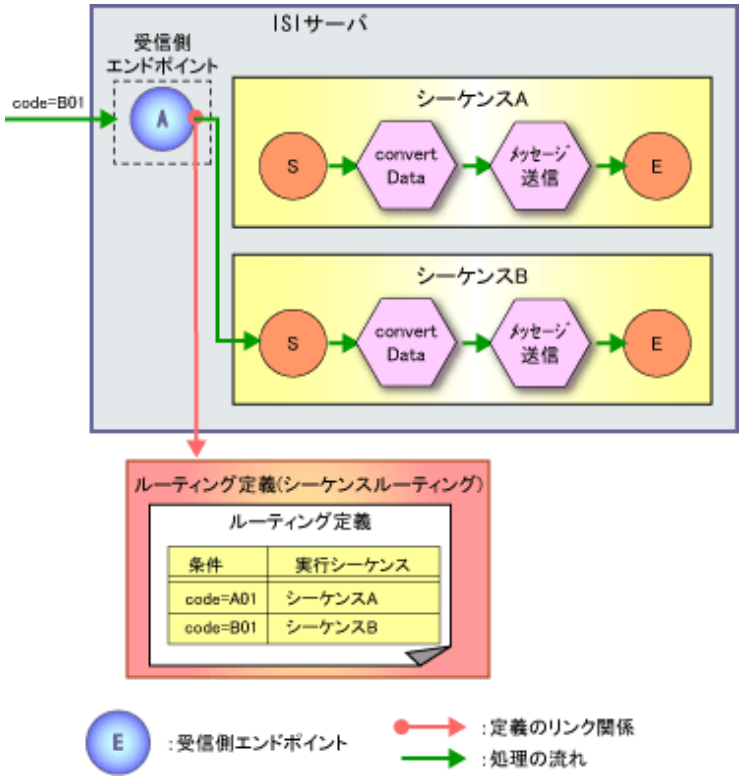
図A.8 ルーティング定義(アウトバウンドルーティング)のリンク関係



シーケンスルーティング

実行するシーケンスを特定するためのルーティング情報(振分け条件と実行するシーケンス)を定義します。

図A.9 ルーティング定義(シーケンスルーティング)のリンク関係

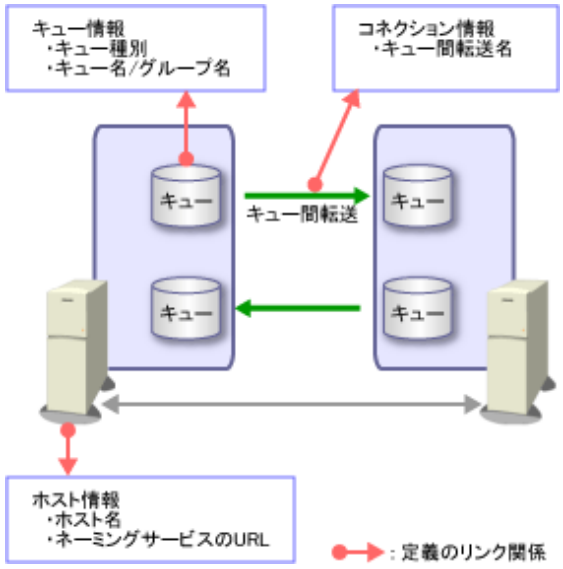


A.1.5 キュー間転送定義

キュー間転送定義は、キュー間転送を行うホストやキュー間の接続情報を定義します。
接続情報には、以下の種類があります。

- ホスト情報
  キュー間転送の接続先となるホストに関するプロパティのことです。
- キュー情報
  キュー間転送で接続するキューに関するプロパティのことです。
- コネクション情報
  キュー間転送の接続に関するプロパティのことです。

図A.10 キュー間転送の接続情報



A.2 定義の作成と登録

ここではコマンドを使用した定義の管理について説明します。
ISI Studioを利用した定義の作成、登録方法については、“ISI Studio ヘルプ”を参照してください。

定義の作成

構築するシステム要件に応じて、ISI Studioを操作し、必要な定義ファイルを作成します。

定義の登録

作成・編集した定義は、コマンドを実行してISIサーバに登録します。
定義を登録するためには、以下のコマンドを使用します。

- 定義の登録コマンド(esidefadd)
  定義をISIサーバに登録します。
- 定義の検証コマンド(esidefverify)
  定義の妥当性を検証します。
- 定義の配置コマンド(esidefdeploy)
  登録済み定義を運用環境に配置します。

定義の登録手順を以下に示します。

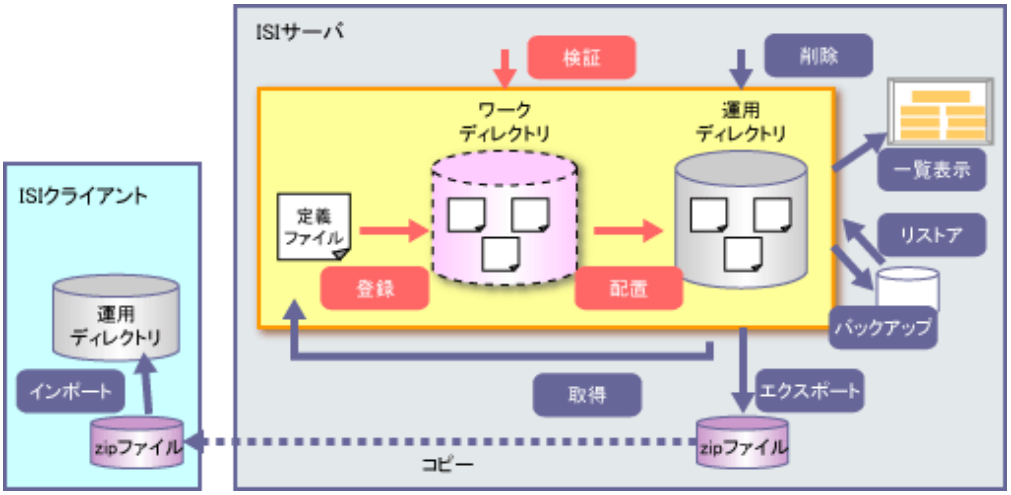
1. 定義の登録
   作成・編集した定義を定義の登録コマンドでISIサーバに登録します。
2. 定義の検証
   定義検証コマンドで、登録した定義の妥当性を検証します。検証で不具合が検出された場合は、定義を修正し、定義の登録から再度実施してください。
3. 定義の配置
   定義配置コマンドで、登録済みの定義を運用環境に配置します。

定義の管理

登録した定義を管理します。定義の管理には、以下の種類があります。

- 定義の一覧表示
- 定義の削除
- 定義の取得
- 定義のバックアップ
- 定義のリストア
- サービスエンドポイント定義のエクスポート
- サービスエンドポイント定義のインポート

図A.11 定義の作成・登録・管理



A.2.1 定義の登録

esidefaddコマンドを実行して、定義をISIサーバに登録します。esidefaddコマンドは、ISIサーバで実行します。
定義の登録は各定義単位で行います。登録時には、各定義ファイル内の範囲で定義の妥当性が検証されます。
esidefaddコマンドの詳細については、“ISI リファレンス”の“esidefadd”コマンドを参照してください。

A.2.2 定義の検証

esidefverifyコマンドを実行して、登録した定義の検証を行います。esidefverifyコマンドは、ISIサーバで実行します。
esidefverifyコマンドでは、登録済み定義全体の妥当性が検証されます。各定義間の参照関係などを検証し、問題点が指摘されます。
なお、esidefverifyコマンドで問題点が指摘された場合、運用環境に定義を配置することはできません。esidefverifyコマンドで指摘された問題点は、定義を配置する前に必ず解決してください。
esidefverifyコマンドの詳細については、“ISI リファレンス”の“esidefverify”コマンドを参照してください。

A.2.3 定義の配置

esidefdeployコマンドを実行して、登録済み定義を運用環境に配置します。esidefdeployコマンドは、ISIサーバで実行します。
定義を配置することで、登録した定義内容が運用環境に反映され、利用可能となります。なお、登録済み定義の妥当性に問題がある場合は、esidefdeployコマンドはエラーとなります。
esidefdeployコマンドの詳細については、“ISI リファレンス”の“esidefdeploy”コマンドを参照してください。

A.2.4 定義の管理

定義を管理するために、以下のコマンドが用意されています。コマンドの詳細については、“ISI リファレンス”を参照してください。

- 定義の一覧表示(esideflist)
  ISIサーバに登録されている定義の一覧を表示します。
- 定義の削除(esidefdel)
  定義を削除します。
- 定義の取得(esidefget)
  登録済みの定義を一括取得します。
- 定義のバックアップコマンド(esidefbackup)
  ISIサーバに登録されている定義を指定したファイル(zip形式)にバックアップします。
- 定義のリストアコマンド(esidefrestore)
  バックアップした定義を、リストアします。
- サービスエンドポイント定義のエクスポート(esidefexportep)
  ISIサーバに登録されているサービスエンドポイント定義を、zip形式でエクスポートします。

・ サービスエンドポイント定義のインポート(esidefimportep)
ISIクライアントでサービスエンドポイント定義を利用可能とするため、ISIサーバでエクスポートしたサービスエンドポイント定義を運用環境に反映します。(事前にエクスポートされたサービスエンドポイント定義(zip形式)を、ISIクライアントにコピーしておく必要があります。)

付録B　定義の詳細

ここでは、ISIの機能を利用するための定義の詳細について説明します。
定義は、ISI Studioで作成・編集してください。ISI Studioで作成・編集した定義の内容を、特に変更する必要はありません。ただし、Systemwalker SQC連携機能利用時に使用するグループ定義は、ISI Studioで作成・編集できません。
なお、定義の構成については、“付録C 定義の例”を参照してください。

注意

定義内の要素名・属性名は、英大文字、英小文字が区別されます。

B.1　サービスエンドポイント定義の詳細

サービスエンドポイント定義の詳細について説明します。

B.1.1　EndpointList

EndpointList(サービスエンドポイント定義のルート要素)について説明します。

定義項目

表B.1 EndpointListの定義項目

種類	名称	説明	備考
子要素	Endpoint	サービスエンドポイント情報を指定します。	複数指定可

指定例

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
      <Endpoint>…</Endpoint>
      <Endpoint>…</Endpoint>
</EndpointList>
```

B.1.1.1 Endpoint

Endpoint(サービスエンドポイント情報)について説明します。

定義項目

表B.2 Endpointの定義項目

種類	名称	説明	備考
属性	name	サービスエンドポイント名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・半角文字 　－ 英小文字 　－ 数字 　－ -(ハイフン) 　－ _(アンダーバー) ・日本語	必須 最大256文字

種類	名称	説明	備考
		ー 全角漢字(JIS第1水準漢字・第2水準漢字) ー 全角ひらがな ー 全角カタカナ ー 全角英字 ー 全角数字 ー 全角記号(機種依存文字を除く)	
属性	sync	サービスエンドポイントの同期タイプを指定します。以下のどちらかを指定します。 ・ true 同期通信を行います。 ・ false 非同期通信を行います。	必須
子要素	Description	サービスエンドポイント定義についての説明を指定します。	
子要素	ServiceName	サービスエンドポイントのサービス名を指定します。	サービス情報を記述する場合、必須
子要素	EsiName	接続するISIサーバ名を指定します。 なお、キュー名で接続先サーバを指定するタイプのサービスエンドポイントでかつ、キュー名が指定されている場合、キューの指定が優先されます。	デフォルトは、“ESIServer”です。
子要素	QueueName	サービスエンドポイントが使用するキュー名を指定します。	以下を利用したサービスエンドポイントの場合、必須 ・ JMS ・ FTP ・ SOAP-RPC(非同期) ・ SOAP-MSG(非同期) ・ JMS-R ・ QUECONNECTED
子要素	ApisideQueueName	キュー間転送機能利用時に、サービスエンドポイントが使用するキュー名を指定します。 サービス利用側またはサービス提供側のアプリケーションにあるキュー名を指定してください。	サービス情報の“type”属性が“QUECONNECTED”で、キュー間転送機能を利用して接続する2つのキュー名が異なる場合、必須
子要素	SequenceName	実行するシーケンス名を指定します。	インバウンドサービスエンドポイント定義で、シーケンスを直接指定する場合、必須
子要素	DistributionName	サービスエンドポイントで使用するルーティング名を指定します。	インバウンドサービスエンドポイント定義で、実行するシーケンスをルーティング定義から取得する場合、必須
子要素	SequenceList	実行するシーケンスのリストを指定します。 シーケンスを複数指定した場合、受信メッセージに対して、複数のシーケンスが実行されます。	

種類	名称	説明	備考
			インバウンドサービスエンドポイント定義で、サービス情報の“type”属性が“JMS-R”の場合、SequenceNameに代えて指定可能 S L インバウンドサービスエンドポイント定義で、サービス情報の“type”属性が“JMS-R”または“MQD” S EE の場合、SequenceNameに代えて指定可能

注意

インバウンドサービスエンドポイント定義の場合、以下のどれか1つを指定してください。

- SequenceName
- DistributionName
- SequenceList

指定例

```
<Endpoint name="endpoint01" sync="true">
      <Description>…</Description>
      <ServiceName>…</ServiceName>
      <EsiName>…</EsiName>
      <QueueName>…</QueueName>
      <SequenceName>…</SequenceName>
</Endpoint>
```

Description

Description(説明)について説明します。

定義項目

表B.3 Descriptionの定義項目

種類	名称	説明	備考
値	---	サービスエンドポイント定義についての説明を指定します。	

指定例

```
<Description>見積り業務用サービスエンドポイント</Description>
```

ServiceName

ServiceName(サービス名)について説明します。

定義項目

表B.4 ServiceNameの定義項目

種類	名称	説明	備考
値	---	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	必須

指定例

```
<ServiceName>service01</ServiceName>
```

EsiName

EsiName(ISIサーバ名)について説明します。

定義項目

表B.5 EsiNameの定義項目

種類	名称	説明	備考
値	---	接続するISIサーバ名を指定します。 システム内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英小文字、または数字を指定してください。 ・ 英小文字 ・ 数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大64バイト

指定例

```
<EsiName>ESIServer</EsiName>
```

QueueName

QueueName(キュー名)について説明します。

定義項目

表B.6 QueueNameの定義項目

種類	名称	説明	備考
値	---	キュー名を指定します。	必須

注意事項

- システム内のJMSサーバに登録されたキュー名だけを指定できます。外部のJMSサーバに登録されたキュー名は指定できません。
- キュー間転送機能を利用する場合は、ISIサーバ側のキュー名を指定してください。

指定例

```
<QueueName>InboundQueue01</QueueName>
```

ApisideQueueName EE

ApisideQueueName(外部キュー名)について説明します。

定義項目

表B.7 ApisideQueueNameの定義項目

種類	名称	説明	備考
値	---	キュー間転送機能で接続する2つのキュー名が異なる場合に指定します。 サービス利用側のアプリケーション、またはサービス提供側のアプリケーション内のキュー名を指定します。	必須

指定例

```
<ApisideQueueName>InboundQueue01</ApisideQueueName>
```

SequenceName

SequenceName(シーケンス名)について説明します。

定義項目

表B.8 SequenceNameの定義項目

種類	名称	説明	備考
値	---	サービスエンドポイントから実行するシーケンス名を指定します。 シーケンス定義に指定したシーケンス名を指定してください。	必須

指定例

```
<SequenceName>sequence01</SequenceName>
```

DistributionName

DistributionName(ルーティング名)について説明します。

定義項目

表B.9 DistributionNameの定義項目

種類	名称	説明	備考
値	---	シーケンスルーティングを使用する場合に、ルーティング名を指定します。	必須

指定例

```
<DistributionName>distribution01</DistributionName>
```

SequenceList EE

SequenceList(シーケンス名のリスト)について説明します。

定義項目

表B.10 SequenceListの定義項目

種類	名称	説明	備考
子要素	SequenceName	サービスエンドポイントから実行するシーケンス名を指定します。	複数指定可

指定例

```
<SequenceList>
        <SequenceName>…</SequenceName>
        <SequenceName>…</SequenceName>
</SequenceList>
```

SequenceName(SequenceListの子要素)

SequenceName(シーケンス名)について説明します。

定義項目

表B.11 SequenceNameの定義項目

種類	名称	説明	備考
値	---	シーケンスルーティングを使用する場合に、ルーティング名を指定します。	必須

指定例

```
<SequenceName>sequence01</SequenceName>
```

B.2 サービス情報の詳細

サービス情報の詳細について説明します。

B.2.1 Service

Service(サービス情報のルート要素)について説明します。

定義項目

表B.12 Serviceの定義項目

種類	名称	説明	備考
属性	name	サービス名を指定します。 システム内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英小文字、または数字を指定してください。 ・ 英小文字 ・ 数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大64バイト
属性	type	サービスのタイプ名を指定します。以下のどれかを指定します。 ・ filewatch ファイルの受信を監視し、シーケンスを開始します。	必須 最大64バイト

種類	名称	説明	備考
		• ftpsend シーケンスの実行結果(ペイロード)をファイル転送機能(FTP)で送信します。 • SOAP-MSG シーケンスの実行結果(ペイロード)をSOAP送信します。 • SOAP-RPC SOAPのRPC機能を利用して、ペイロードをISIサーバに送信し、シーケンスを開始します。 • JMS-EXTERNAL シーケンスの実行結果(ペイロード)をJMS送信します。 • LIST-CREATOR List Creator連携を行います。 • JCA JCAを利用した連携を行います。 • SOAP-NET .NET互換SOAP通信機能を利用して、ペイロードをISIサーバに送信し、シーケンスを開始します。 • CORBA EE CORBA接続により、ペイロードをISIサーバに送信し、シーケンスを開始します。 • MQ-MSG EE WebSphere MQと連携し、メッセージを送受信します。 • JMS-R EE JMS-Rのキューを使用して、ペイロードをISIサーバに送信し、シーケンスを開始します。または、実行結果(ペイロード)を送信します。 • MQD S EE MQDを使用して、ペイロードをISIサーバに送信し、シーケンスを開始します。または、実行結果(ペイロード)を送信します。 • QUECONNECTED EE メッセージ(ペイロードもしくは、共通メッセージ)をサービス利用側システムのキューに格納し、キュー間転送機能を利用してISIサーバに送信します。 または、シーケンスの実行結果(ペイロードまたは共通メッセージ)をISIサーバ内のキューに格納し、キュー間転送機能を利用してサービス提供側システムのキューに送信します。	
子要素	Description	サービスについての説明を指定します。	
子要素	PropertyList	プロパティ情報を指定します。	プロパティが必要なサービスの場合、必須

指定例

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="FTPService" type="filewatch" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
```

```
    <Description>…</Description>
    <PropertyList>…</PropertyList>
</Service>
```

Description

Description(説明)について説明します。

定義項目

表B.13 Descriptionの定義項目

種類	名称	説明	備考
値	---	サービスについての説明を指定します。	

指定例

```
<Description>見積り業務用のファイル受信を監視し、シーケンスを開始します</Description>
```

PropertyList

PropertyList(プロパティ情報のリスト)について説明します。

定義項目

表B.14 PropertyListの定義項目

種類	名称	説明	備考
子要素	Property	プロパティ情報を指定します。	複数指定可

指定例

```
<PropertyList>
        <Property>…</Property>
        <Property>…</Property>
</PropertyList>
```

Property

Property(プロパティ情報)について説明します。

定義項目

表B.15 Propertyの定義項目

種類	名称	説明	備考
属性	name	プロパティ名を指定します。 サービス内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー) ・ .(ピリオド)	必須 最大255バイト
属性	secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。以下のどちらかを指定します。	デフォルトは、“false”です。

種類	名称	説明	備考
		・ true 暗号化します。 ・ false 暗号化しません。 “true”を指定した場合は、esidefaddコマンド実行時に、定義の情報が暗号化され登録されます。	
値	---	プロパティの情報を指定します。	

指定例

```
<Property name="hostname">hostA</Property>
<Property name="username">anonymous</Property>
```

B.3 シーケンス定義の詳細

シーケンス定義の詳細について説明します。

B.3.1 Sequence

Sequence(シーケンス定義のルート要素)について説明します。

定義項目

表B.16 Sequenceの定義項目

種類	名称	説明	備考
属性	name	シーケンス名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 — 英小文字 — 数字 — -(ハイフン) — _(アンダーバー) ・ 日本語 — 全角漢字(JIS第1水準漢字・第2水準漢字) — 全角ひらがな — 全角カタカナ — 全角英字 — 全角数字 — 全角記号(機種依存文字を除く)	必須 最大256文字
属性	logging	シーケンスログを採取するかどうかを指定します。 以下のどちらかを指定します。 ・ true ログを採取します。	デフォルトは、“false”です。

種類	名称	説明	備考
		・ false ログを採取しません。	
属性	history	シーケンスの履歴を取得するかどうかを指定します。以下のどちらかを指定します。 ・ true 履歴を取得します。 ・ false 履歴を取得しません。	デフォルトは、“true”です。
子要素	Retry	シーケンスリトライの情報を指定します。	利用側サービスの通信方式が同期の場合だけ有効です。 通信方式が非同期の場合のリトライ回数の設定方法については、“ISI運用ガイド”の“機能のチューニング”を参照してください。
子要素	Description	シーケンスについての説明を指定します。	
子要素	PropertyList	プロパティ情報を指定します。	
子要素	Block	実行ステップを指定します。	必須
子要素	StepInformation	実行ステップの詳細情報を指定します。	必須 複数指定可
子要素	ConditionBranchInformation	条件分岐ブロックの詳細情報を指定します	複数指定可
子要素	ErrorHandleSequence	エラー発生時に呼び出すシーケンス名を指定します。	

指定例

```
<?xml version="1.0" encoding="UTF-8" ?>
<Sequence name="sequence01" logging="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-sequence.xsd">
      <Description>…</Description>
      <PropertyList>…</PropertyList>
      <Block>…</Block>
      <StepInformation>…</StepInformation>
</Sequence>
```

Retry

Retry(リトライ)について説明します。

定義項目

表B.17 Retryの定義項目

種類	名称	説明	備考
属性	interval	リトライの間隔をミリ秒で指定します。	
値	---	リトライの回数を指定します。	

指定例

```
<Retry interval="1000">10</Retry>
```

Description

Description(説明)について説明します。

定義項目

表B.18 Descriptionの定義項目

種類	名称	説明	備考
値	---	シーケンスについての説明を指定します。	

指定例

```
<Description>見積り業務用シーケンス</Description>
```

B.4 プロパティ情報の詳細

プロパティ情報の詳細について説明します。

B.4.1 PropertyList

PropertyList(プロパティ情報のリスト)について説明します。

定義項目

表B.19 PropertyListの定義項目

種類	名称	説明	備考
子要素	Property	プロパティ情報を指定します。	複数指定可

指定例

```
<PropertyList>
      <Property>…</Property>
      <Property>…</Property>
</PropertyList>
```

B.4.1.1 Property

Property(プロパティ情報)について説明します。

定義項目

表B.20 Propertyの定義項目

種類	名称	説明	備考
属性	name	プロパティ名を指定します。 シーケンス定義内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大255バイト

種類	名称	説明	備考
		・ .(ピリオド)	
値	---	プロパティのデフォルト値を指定します。	

指定例

```
<Property name="convert.result">0</Property>
<Property name="ftp.username">anonymous</Property>
```

B.5 シーケンスブロックの詳細

シーケンスブロックの詳細について説明します。

B.5.1 Block

Block(シーケンスブロック)について説明します。

定義項目

表B.21 Blockの定義項目

種類	名称	説明	備考
子要素	Step	実行ステップの情報を指定します。 ステップの実行順に指定します。	必須 複数指定可
子要素	ConditionBranch	条件分岐ブロックの情報を指定します。	複数指定可

指定例

```
<Block>
      <Step>…</Step>
      <ConditionBranch>…</ConditionBranch>
      <Step>…</Step>
</Block>
```

Step

Step(実行ステップ)について説明します。

定義項目

表B.22 Stepの定義項目

種類	名称	説明	備考
属性	name	ステップ名を指定します。 シーケンス定義内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 　ー 英小文字 　ー 数字 　ー -(ハイフン) 　ー _(アンダーバー) ・ 日本語	必須 最大256文字

種類	名称	説明	備考
		ー 全角漢字(JIS第1水準漢字・第2水準漢字) ー 全角ひらがな ー 全角カタカナ ー 全角英字 ー 全角数字 ー 全角記号(機種依存文字を除く)	
属性	cancel	エラー時のキャンセル処理の有無を指定します。以下のどちらかを指定します。 ・ true キャンセル処理があります。 “true”を指定した場合は、“StepInformation”に、キャンセル処理用のメソッドを定義してください。 ・ false キャンセル処理がありません。	デフォルトは、“false”です。
属性	history	ステップの履歴を取得するかどうかを指定します。以下のどちらかを指定します。 ・ true 履歴を取得します。 ・ false 履歴を取得しません。	デフォルトは、“true”です。

指定例

```
<Step name="checkXML" cancel="true" history="false" />
<Step name="convertData" />
```

ConditionBranch

ConditionBranch(条件分岐ブロック)について説明します。

定義項目

表B.23 ConditionBranchの定義項目

種類	名称	説明	備考
属性	name	条件分岐ブロック名を指定します。 シーケンス定義内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大64バイト
子要素	Condition	分岐経路の情報を指定します。	必須 複数指定可
子要素	DefaultCondition	条件不成立時の経路情報を指定します。	必須

指定例

```
<ConditionBranch name="branch1">
      <Condition>…</Condition>
      <DefaultCondition>…</DefaultCondition>
</ConditionBranch>
```

Condition

Condition(分岐経路情報)について説明します。

定義項目

表B.24 Conditionの定義項目

種類	名称	説明	備考
属性	name	分岐経路名を指定します。 条件分岐ブロック内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大64バイト
子要素	Step	条件を満たした場合の実行ステップの情報を指定します。 ステップの実行順に指定します。	複数指定可

指定例

```
<Condition name="1">
      <Step>…</Step>
      <Step>…</Step>
</Condition>
```

DefaultCondition

DefaultCondition(条件不成立時の経路情報)について説明します。

定義項目

表B.25 DefaultConditionの定義項目

種類	名称	説明	備考
子要素	Step	条件不成立時の実行ステップの情報を指定します。 ステップの実行順に指定します。	複数指定可

指定例

```
<DefaultCondition name="1">
      <Step>…</Step>
      <Step>…</Step>
</DefaultCondition>
```

B.6 実行ステップ情報の詳細

実行ステップ情報の詳細について説明します。

B.6.1 StepInformation

StepInformation(実行ステップ詳細情報)について説明します。

定義項目

表B.26 StepInformationの定義項目

種類	名称	説明	備考
属性	stepname	ステップ名を指定します。 シーケンス定義内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 ー 英小文字 ー 数字 ー -(ハイフン) ー _(アンダーバー) ・ 日本語 ー 全角漢字(JIS第1水準漢字・第2水準漢字) ー 全角ひらがな ー 全角カタカナ ー 全角英字 ー 全角数字 ー 全角記号(機種依存文字を除く)	必須 最大256文字
属性	monitoring	モニタリングを行うかどうかを指定します。以下のどちらかを指定します。 ・ true モニタリングします。 ・ false モニタリングしません。	デフォルトは、“false”です。
子要素	MediatorFunction	メディエータファンクションの呼出し情報を指定します。	必須
子要素	Description	実行ステップについての説明を指定します。 モニタリングのサービス名として利用されます。	

指定例

```
<StepInformation stepname="checkXML">
      <MediatorFunction>…</MediatorFunction>
</StepInformation>
```

MediatorFunction

MediatorFunction(メディエータファンクションの呼出し情報)について説明します。

定義項目

表B.27 MediatorFunctionの定義項目

種類	名称	説明	備考
属性	name	メディエータファンクション名を指定します。標準提供のメディエータファンクションを利用する場合は、以下のどれかを指定します。 ・ XMLValidation XML Validationを利用します。 ・ XMLOperation XML Operationを利用します。 ・ XMLEmergerExt XML Emergerを利用します。 ・ FlatEmerger FLAT Emergerを利用します。 ・ SOAPOperation SOAP Operationを利用します。 ・ XMLConverter XML Converterを利用します。 ・ Formatmanager Formatmanagerを利用します。 ・ SendMessage メッセージ送信を利用します。 ・ ESISoapGw SOAP Gatewayを利用します。 ・ BPMMonitoring Interstage BPM Analyticsを利用します。 ・ ListCreator List Creatorを利用します。 ・ JCAAdapter JCA Adapterを利用します。 ・ CommandLauncher Command Launcherを利用します。 ・ FileOperator File Operatorを利用します。	必須
子要素	Method	メディエータファンクションの呼出し情報を指定します。	必須 複数(4つまで)指定可

指定例

```
<MediatorFunction name="XMLValidation">
      <Method>…</Method>
      <Method>…</Method>
</MediatorFunction>
```

Method

Method(ファンクションのメソッド情報)について説明します。

定義項目

表B.28 Methodの定義項目

種類	名称	説明	備考
属性	type	メソッドを呼び出す契機を指定します。以下のどれかを指定します。 ・ init インスタンス初期化時 ・ execute ステップ実行時 ・ cancel キャンセル実行時(ステップ実行後の例外発生時) ・ destroy インスタンス削除時	必須
属性	name	メディエータファンクション定義に指定したメソッド名を指定します。	必須
子要素	ParameterList	メソッドのパラメタ情報を指定します。	
子要素	Return	メソッドの復帰値情報を指定します。	

注意事項

- 属性“type”がexecute指定のMethod定義は必須です。
- 属性“type”がinit、execute、cancel、destroyのMethod定義は、それぞれ1回ずつだけ記述できます。

指定例

```
<Method type="execute" name="convert">
      <ParameterList>…</ParameterList>
      <Return>…</Return>
</Method>
```

Description

Description(説明)について説明します。

定義項目

表B.29 Descriptionの定義項目

種類	名称	説明	備考
値	---	実行ステップについての説明を指定します。	

指定例

```
<Description>商品照会サービスです</Description>
```

ParameterList

ParameterList(パラメタ情報のリスト)について説明します。

定義項目

表B.30 ParameterListの定義項目

種類	名称	説明	備考
子要素	Parameter	メソッドのパラメタ情報を指定します。	複数指定可
子要素	ParameterArray	配列型のパラメタ情報を指定します。	複数指定可

指定例

```
<ParameterList>
      <Parameter>…</Parameter>
      <ParameterArray>…</ParameterArray>
</ParameterList>
```

Parameter

Parameter(パラメタ情報)について説明します。

定義項目

表B.31 Parameterの定義項目

種類	名称	説明	備考
属性	name	メディエータファンクション定義に指定したパラメタ名を指定してください。	必須
属性	valuetype	パラメタに指定する値のタイプを指定します。以下のどれかを指定します。 ・ esi.header 共通メッセージ内のヘッダパラメタ ・ esi.payload 共通メッセージ内のペイロード ・ esi.esimessage 共通メッセージ本体 ・ direct ここに記述した値を直接使用します。値を省略した場合、パラメタ型が基本型・基本型のラッパクラスの場合は、javaの初期値を指定したものとして扱われます。またString型の場合は空文字を指定したものとして扱われます。	
値	---	パラメタ値を指定します。指定した値は、“valuetype”の設定値に応じて、以下のようになります。 ・ esi.header 共通メッセージのヘッダパラメタキー名として扱われます。 ・ esi.payload どの時点のペイロードにパラメタを渡すかを指定します。以下のどれかを指定します。デフォルトは、“@current”です。 － @start シーケンス起動時のペイロード － @current 現在のペイロード － ステップ名を指定 指定したステップで出力したペイロード ・ esi.esimessage 指定は不要です。指定した値は、無効となります。	valuetypeに“esi.payload”を指定した場合のデフォルトは、“@current”です。 valuetypeに“esi.header”を指定した場合、値は必須です。

種類	名称	説明	備考
		・ direct ここに記述した値を直接使用します。値を省略した場合、パラメタ型が基本型・基本型のラッパクラスの場合は、javaの初期値が、String型の場合は空文字となります。	

指定例

```
<Parameter name="esi.Formatmanager.inputData" valuetype="esi.payload">@current</Parameter>
<Parameter name="esi.XmlOperation.in" valuetype="esi.payload">@current</Parameter>
```

ParameterArray

ParameterArray(配列型のパラメタ情報)について説明します。

定義項目

表B.32 ParameterArrayの定義項目

種類	名称	説明	備考
属性	name	パラメタ名を指定します。メディエータファンクション定義に指定したパラメタ名を指定してください。	必須
子要素	Parameter	パラメタ情報を指定します。 パラメタの配列順に指定します。	必須 複数指定可

指定例

```
<ParameterArray name="esi.SendMessage.endpoint">
      <Parameter>…</Parameter>
      <Parameter>…</Parameter>
</ ParameterArray>
```

Parameter

Parameter(配列内のパラメタ情報)について説明します。

定義項目

表B.33 Parameterの定義項目

種類	名称	説明	備考
属性	valuetype	パラメタに指定する値のタイプを指定します。以下のどちらかを指定します。 ・ esi.header 共通メッセージ内のヘッダパラメタ ・ direct 固定値	デフォルトは、“direct”です。
値	---	パラメタ値を指定します。指定した値は、“valuetype”の設定値に応じて、以下のようになります。 ・ esi.header 共通メッセージのヘッダパラメタキー名として扱われます。 ・ direct ここに記述した値を直接使用します。省略した場合は、空文字となります。	必須

指定例

```
<Parameter valuetype="direct">endpointA</Parameter>
<Parameter valuetype="esi.header">com.fujitsu.esi.endpoint.outbound</Parameter>
```

Return

Return(復帰値情報)について説明します。

定義項目

表B.34 Returnの定義項目

種類	名称	説明	備考
属性	valuetype	復帰値の格納先を指定します。以下のどれかを指定します。 ・ esi.header 共通メッセージ内のヘッダパラメタ ・ esi.payload 共通メッセージ内のペイロード ・ esi.esimessage 共通メッセージ本体	必須
値	---	復帰値の設定先を指定します。指定した値は、“valuetype”の設定値に応じて、以下のようになります。 ・ esi.header 共通メッセージのヘッダパラメタキー名として扱われます。 ・ esi.payload 指定は不要です。 ・ esi.esimessage 指定は不要です。	valuetypeにesi.headerを指定した場合、値は必須です。

指定例

```
<Return valuetype="esi.payload" />
```

B.7 条件分岐ブロック情報の詳細

条件分岐ブロック情報の詳細について説明します。

B.7.1 ConditionBranchInformation

ConditionBranchInformation(条件分岐ブロック詳細情報)について説明します。

定義項目

表B.35 ConditionBranchInformationの定義項目

種類	名称	説明	備考
属性	name	条件分岐ブロック名を指定します。 シーケンス定義内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字	必須 最大64バイト

種類	名称	説明	備考
		・ -(ハイフン) ・ _(アンダーバー)	
子要素	DataType	判断するデータのタイプを指定します。	必須
子要素	PositionList	ポジションを指定します。	必須
子要素	ConditionInformation	分岐経路の条件を指定します。	必須 複数指定可

指定例

```
<ConditionBranchInformation name="branch1">
      <DataType>…</DataType>
      <PositionList>…</PositionList>
      <Conditioninformation>…</ConditionInformation>
      <Conditioninformation>…</ConditionInformation>
</ConditionBranch>
```

DataType

DataType(データタイプ)について説明します。

定義項目

表B.36 DataTypeの定義項目

種類	名称	説明	備考
値	---	判断するデータタイプを指定します。以下のどれかを指定します。 ・ xml XML形式のペイロード ・ flat FLAT形式のペイロード ・ csv CSV形式のペイロード ・ header 共通メッセージのヘッダパラメタ	

指定例

```
<DataType>xml</DataType>
```

PositionList

PositionList(ポジションのリスト)について説明します。

定義項目

表B.37 PositionListの定義項目

種類	名称	説明	備考
子要素	Position	ポジションを指定します。	複数指定可

指定例

```
<PositionList>
      <Position>…</Position>
      <Position>…</Position>
</PositionList>
```

Position

Position(ポジション)について説明します。

定義項目

表B.38 Positionの定義項目

種類	名称	説明	備考
属性	name	ポジション名を指定します。 条件分岐ブロック内で一意な名前を設定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー) 予約語“AND”、“OR”は指定できません。	必須 最大64バイト
属性	charset	使用しているJavaVMがサポートする文字コードを指定します。 文字コードにUTF-16を使用する場合、リトルエンディアンは“UTF-16LE”、ビックエンディアンは“UTF-16BE”を指定してください。	
値	---	判断するデータの位置を指定します。 DataTypeに指定した値によって指定方法が異なります。 ・ DataTypeに“xml”を指定した場合 XPathで指定してください。 ・ DataTypeに“flat”を指定した場合 データの開始位置と桁数をバイト単位で、カンマで区切って指定してください。FLATデータの先頭の開始位置は“0”です。 ・ DataTypeに“csv”を指定した場合 行番号とカラム番号を、カンマで区切って指定してください。CSVデータの先頭の行番号とカラム番号は“1”です。 ・ DataTypeに“header”を指定した場合 共通メッセージのヘッダキーで指定してください。	

指定例

```
<Position name="title">/booklist/book/title</Position>
<Position name="price">/booklist/book/price</Position>
```

ConditionInformation

ConditionInformation(分岐経路情報)について説明します。

定義項目

表B.39 ConditionInformationの定義項目

種類	名称	説明	備考
子要素	name	分岐経路名を指定します。 条件分岐ブロック内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大64バイト
子要素	Description	分岐経路の説明を指定します。	
子要素	Condition	分岐経路の条件を指定します。	必須

指定例

```
<Condition name="1">
      <Step>…</Step>
      <Step>…</Step>
</Condition>
```

Description

Description(分岐経路の説明)について説明します。

定義項目

表B.40 Descriptionの定義項目

種類	名称	説明	備考
値	---	分岐経路の説明を指定します。	

指定例

```
<Description>2000円以下の場合</Description>
```

Condition

Condition(分岐経路の条件)について説明します。

定義項目

表B.41 Conditionの定義項目

種類	名称	説明	備考
値	---	分岐経路の条件を指定します。	必須

指定例

```
<Condition>?price? &lt;= 2000</Condition>
```

B.8 メディエータファンクション定義の詳細

メディエータファンクション定義の詳細について説明します。

B.8.1 MediatorFunctionList

MediatorFunctionList(メディエータファンクション定義のルート要素)について説明します。

定義項目

表B.42 MediatorFunctionListの定義項目

種類	名称	説明	備考
子要素	MediatorFunction	メディエータファンクション情報を指定します。	必須 複数指定可

指定例

```
<?xml version="1.0" encoding="UTF-8" ?>
<MediatorFunctionList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-mediatorfunction.xsd">
      <MediatorFunction>…</MediatorFunction>
      <MediatorFunction>…</MediatorFunction>
</MediatorFunctionList>
```

B.8.1.1 MediatorFunction

MediatorFunction(メディエータファンクション情報)について説明します。

定義項目

表B.43 MediatorFunctionの定義項目

種類	名称	説明	備考
属性	name	メディエータファンクション名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 ー 英小文字 ー 数字 ー -(ハイフン) ー _(アンダーバー) ・ 日本語 ー 全角漢字(JIS第1水準漢字・第2水準漢字) ー 全角ひらがな ー 全角カタカナ ー 全角英字 ー 全角数字 ー 全角記号(機種依存文字を除く)	必須 最大256文字
属性	jar	メディエータファンクションのjarファイル名を指定します。 標準提供のメディエータファンクションを利用する場合は、jarファイル名の指定は、不要です。	
属性	class	メディエータファンクションのクラス名を指定します。	必須
子要素	Description	メディエータファンクションについての説明を指定します。	
子要素	DependJar	依存するjarファイルのリストを指定します。	

種類	名称	説明	備考
子要素	Method	メソッド情報を指定します。	必須 複数指定可

指定例

```
<MediatorFunction name="XMLvalidation" class="com.fujitsu.esi.function.validation.xml.ESIXMLValidation">
        <Description>…</Description>
        <Method>…</Method>
</MediatorFunction>
<MediatorFunction name="UserFunction" jar="gyoumu01.jar" class="Gyoumu01">
        <Description>…</Description>
        <DependJar>…</DependJar>
        <Method>…</Method>
</MediatorFunction>
```

Description

Description(説明)について説明します。

定義項目

表B.44 Descriptionの定義項目

種類	名称	説明	備考
値	---	メディエータファンクションについての説明を指定します。	

指定例

```
<Description>XML Validation</Description>
```

DependJar

DependJar(依存するjarファイルのリスト)について説明します。

定義項目

表B.45 DependJarの定義項目

種類	名称	説明	備考
子要素	Jar	依存するjarファイルの情報を指定します。	

指定例

```
<DependJar>
        <Jar>…</Jar>
        <Jar>…</Jar>
</DependJar>
```

Jar

Jar(依存するjarファイル)について説明します。

定義項目

表B.46 Jarの定義項目

種類	名称	説明	備考
値	---	依存するjarファイル名を指定します。	必須

指定例

```
<Jar>GyoumuCommon.jar</Jar>
```

Method

Method(メソッド情報)について説明します。

定義項目

表B.47 Methodの定義項目

種類	名称	説明	備考
属性	name	メディエータファンクションのメソッド名を指定します。 メディエータファンクション内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー)	必須 最大64バイト
属性	methodname	メディエータファンクションの実メソッド名を指定します。	必須
子要素	ParameterList	メソッドのパラメタ情報を指定します。	メソッドにパラメタがある場合、必須
子要素	Return	メソッドの復帰値情報を指定します。	復帰値がvoid型でない場合、必須

指定例

```
<Method name="checkXML" methodname="checkXML">
        <ParameterList>…</ParameterList>
        <Return>…</Return>
</Method>
<Method name="convertbyTransformationSequenceID" methodname="convertbyTransformationSequenceID">
        <ParameterList>…</ParameterList>
        <Return>…</Return>
</Method>
```

ParameterList

ParameterList(パラメタ情報のリスト)について説明します。

定義項目

表B.48 ParameterListの定義項目

種類	名称	説明	備考
子要素	Parameter	メソッドのパラメタ情報を指定します。	複数指定可
子要素	ParameterArray	配列型のパラメタ情報を指定します。	複数指定可

指定例

```
<ParameterList>
      <Parameter>…</Parameter>
      <ParameterArray>…</ParameterArray>
</ParameterList>
```

Parameter

Parameter(パラメタ情報)について説明します。

定義項目

表B.49 Parameterの定義項目

種類	名称	説明	備考
属性	name	パラメタ名を指定します。 メソッド内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー) ・ . (ピリオド)	必須 最大255バイト
属性	type	パラメタのデータ型を指定します。以下のどれかを指定します。 ・ 基本データ型 　－ boolean 　－ short 　－ int 　－ float 　－ long 　－ double ・ 基本データ型のラッパクラス 　－ java.lang.Boolean 　－ java.lang.Byte 　－ java.lang.Character 　－ java.lang.Short 　－ java.lang.Integer 　－ java.lang.Float 　－ java.lang.Long 　－ java.lang.Double ・ String型 　－ java.lang.String ・ 共通メッセージ 　－ com.fujitsu.esi.message.ESIMessage	必須

種類	名称	説明	備考
		・ byte配列型 　－ byte[] “java.lang.String”の配列については、ParameterArrayで指定してください。	
属性	valuetype	パラメタに指定する値のタイプを指定します。以下のどれかを指定します。 ・ esi.header 共通メッセージ内のヘッダパラメタ ・ esi.payload 共通メッセージ内のペイロード ・ esi.esimessage 共通メッセージ本体 ・ direct 固定値	省略した場合、デフォルトは設定されません。
属性	value	パラメタ値を指定します。指定した値は、“valuetype”の設定値に応じて、以下のようになります。 ・ esi.header 共通メッセージのヘッダパラメタキー名として扱われます。 ・ esi.payload どの時点のペイロードにパラメタを渡すかを指定します。以下のどちらかを指定します。デフォルトは、“@current”です。 　－ @start 　シーケンス起動時のペイロード 　－ @current 　現在のペイロード ・ esi.esimessage 指定は不要です。指定した値は、無効となります。	

注意事項

・ 属性“valuetype”を省略した場合、属性“value”は指定できません。指定した値は、無効となります。

指定例

```
<Parameter name="esi.Formatmanager.transSeqID"
type="java.lang.String"
valuetype="esi.header"
value="com.fujitsu.esi.formatmanager.transSeqID" />
```

ParameterArray

ParameterArray(配列型のパラメタ情報)について説明します。

定義項目

表B.50 ParameterArrayの定義項目

種類	名称	説明	備考
属性	name	パラメタ名を指定します。 メソッド内で一意な名前を指定してください。	必須 最大255バイト

種類	名称	説明	備考
		以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー) ・ . (ピリオド)	
属性	type	パラメタのデータ型を指定します。“java.lang.String”を指定してください。	必須

指定例

```
<ParameterArray name="esi.SendMessage.endCondition" type="java.lang.String" />
```

Return

Return(復帰値情報)について説明します。

定義項目

表B.51 Returnの定義項目

種類	名称	説明	備考
属性	type	復帰値のデータ型を指定します。以下のどれかを指定します。 ・ 基本データ型 　ー boolean 　ー short 　ー int 　ー float 　ー long 　ー double ・ 基本データ型のラッパクラス 　ー java.lang.Boolean 　ー java.lang.Byte 　ー java.lang.Character 　ー java.lang.Short 　ー java.lang.Integer 　ー java.lang.Float 　ー java.lang.Long 　ー java.lang.Double ・ String型 　ー java.lang.String ・ 共通メッセージ 　ー com.fujitsu.esi.message.ESIMessage	必須

種類	名称	説明	備考
		・ byte配列型 　－ byte[] ・ 復帰値なし 　－ void	
属性	valuetype	復帰値の格納先を指定します。以下のどれかを指定します。 ・ esi.header 共通メッセージ内のヘッダパラメタ ・ esi.payload 共通メッセージ内のペイロード ・ esi.esimessage 共通メッセージ本体	省略した場合、デフォルトは設定されません。
属性	value	復帰値の設定先を指定します。指定した値は、“valuetype”の設定値に応じて、以下のようになります。 ・ esi.header 共通メッセージのヘッダパラメタキー名として扱われます。 ・ esi.payload 指定は不要です。指定した値は、無効となります。 ・ esi.esimessage 指定は不要です。指定した値は、無効となります。	

注意事項

- 属性“valuetype”を省略した場合、属性“value”は指定できません。指定した値は、無効となります。
- 属性“type”に“void”を指定した場合、属性“valuetype”および“value”で指定した値は、無効となります。

指定例

```
<Return type="byte[]" valuetype="esi.payload" />
```

B.9 ルーティング定義の詳細

ルーティング定義の詳細について説明します。

B.9.1 MessageDistribution

MessageDistribution(ルーティング定義のルート要素)について説明します。

定義項目

表B.52 MessageDistributionの定義項目

種類	名称	説明	備考
属性	name	ルーティング定義名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 　－ 英小文字 　－ 数字	必須 最大256文字

種類	名称	説明	備考
		ー -(ハイフン) ー _(アンダーバー) ・ 日本語 ー 全角漢字(JIS第1水準漢字・第2水準漢字) ー 全角ひらがな ー 全角カタカナ ー 全角英字 ー 全角数字 ー 全角記号(機種依存文字を除く)	
属性	routing	ルーティング種別を指定します。以下のどれかを指定します。 ・ INBOUND 受信メッセージを各インバウンドサービスエンドポイントへルーティングするための定義です。“inboundtype”属性で指定するサービスで使用します。 ・ OUTBOUND 送信メッセージを各アウトバウンドサービスエンドポイントへルーティングするための定義です。サービス呼出しメディエータファンクションで使用します。 ・ SEQUENCE インバウンドサービスエンドポイントから各シーケンスへルーティングするための定義です。	デフォルトは“INBOUND”です。
属性	inboundtype	ルーティング種別が“INBOUND”の場合に、接続のタイプ名を指定します。以下のどれかを指定します。 ・ SOAP-MSG SOAP接続でのルーティングを行います。 ・ CORBA EE CORBA接続でのルーティングを行います。 ・ MQ-MSG EE MQ連携でのルーティングを行います。 ・ QUECONNECTED EE キュー間転送でのルーティングを行います。 ・ MQD S EE MQD接続でのルーティングを行います。 ・ COMMON カスタムアダプタ接続でのルーティングを行います。	ルーティング種別が“INBOUND”の場合、必須
属性	outboundtype	ルーティング種別が“OUTBOUND”の場合に、同期か非同期かを指定します。指定しない場合は互換定義になります。 ・ SYNC 同期サービスエンドポイントのルーティングを行います。 ・ ASYNC 非同期サービスエンドポイントのルーティングを行います。	

種類	名称	説明	備考
子要素	Description	ルーティング定義についての説明を指定します。	
子要素	PositionList	判断するデータの位置を指定します。	必須
子要素	DistributionList	データを振り分ける情報を指定します。	必須

指定例

```
<MessageDistribution name="soap_distribution" inboundtype="SOAP-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="esi-endpointdistribution.xsd">
      <Description>…</Description>
      <PositionList>…</PositionList>
      <DistributionList>…</DistributionList>
</MessageDistribution>
```

Description

Description(説明)について説明します。

定義項目

表B.53 Descriptionの定義項目

種類	名称	説明	備考
値	---	ルーティング定義についての説明を指定します。	

指定例

```
<Description>SOAP用振分け定義</Description>
```

PositionList

PositionList(データ判断位置のリスト)について説明します。

定義項目

表B.54 PositionListの定義項目

種類	名称	説明	備考
子要素	Position	判断するデータの位置を指定します。	複数指定可

指定例

```
<PositionList>
      <Position>…</Position>
      <Position>…</Position>
</PositionList>
```

Position

Position(データ判断位置情報)について説明します。

定義項目

表B.55 Positionの定義項目

種類	名称	説明	備考
属性	name	データ判断位置名を指定します。 ルーティング定義内で一意な名前を指定してください。 以下の半角文字列を指定できます。ただし、先頭は、英数字を指定してください。 ・ 英数字 ・ -(ハイフン) ・ _(アンダーバー) ・ . (ピリオド)	必須 最大255バイト
属性	type	データのタイプ名を指定します。以下のどれかを指定します。 ・ xml XML形式 ・ flat FLAT形式 ・ csv CSV形式	必須
属性	charset	使用しているJavaVMがサポートする文字コードを指定します。 文字コードにUTF-16を使用する場合、リトルエンディアンは“UTF-16LE”、ビックエンディアンは“UTF-16BE”を指定してください。	
値	---	判断するデータの位置を指定します。 “type”に“xml”を指定した場合は、XPathで指定してください。 “type”に“flat”を指定した場合は、データの開始位置とけた数をバイト単位で、カンマで区切って指定してください。FLATデータの先頭の開始位置は“0”です。 “type”に“csv”を指定した場合は、行番号とカラム番号を、カンマで区切って指定してください。CSVデータの先頭の行番号とカラム番号は“1”です。	

指定例

```
<Position name="price" type="xml">/booklist/book/price</Position>
<Position name="division" type="flat" charset="EUC-JP">0,3</Position>
<Position name="spreadsheet" type="csv" charset="Shift-jis">1,2</Position>
```

DistributionList

DistributionList(振分け情報のリスト)について説明します。

定義項目

表B.56 DistributionListの定義項目

種類	名称	説明	備考
子要素	Distribution	データを振り分ける情報を指定します。	複数指定可

指定例

```
<DistributionList>
      <Distribution>…</Distribution>
      <Distribution>…</Distribution>
</DistributionList>
```

Distribution

Distribution(振分け情報)について説明します。

定義項目

表B.57 Distributionの定義項目

種類	名称	説明	備考
子要素	Condition	振分け条件を指定します。	必須
子要素	Endpoint	振分け先サービスエンドポイント名を指定します。	アウトバウンドルーティングを使用する場合、必須
子要素	Sequence	振分け先シーケンス名を指定します。	シーケンスルーティングを使用する場合、必須

Endpoint、Sequenceはどちらか一方を指定します。

指定例

```
<Distribution>
      <Condition>…</Condition>
      <Endpoint>…</Endpoint>
</Distribution>
```

Condition

Condition(振分け条件)について説明します。

定義項目

表B.58 Conditionの定義項目

種類	名称	説明	備考
値	---	データを振り分ける条件を指定します。 条件式については、“条件式”を参照してください。	必須

指定例

```
<Condition>$price == 1000</Condition>
```

Endpoint

Endpoint(振分けサービスエンドポイント)について説明します。

定義項目

表B.59 Endpointの定義項目

種類	名称	説明	備考
値	---	振分け先のサービスエンドポイント名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。	必須 最大256文字

種類	名称	説明	備考
		・ 半角文字 　ー 英小文字 　ー 数字 　ー -(ハイフン) 　ー _(アンダーバー) ・ 日本語 　ー 全角漢字(JIS第1水準漢字・第2水準漢字) 　ー 全角ひらがな 　ー 全角カタカナ 　ー 全角英字 　ー 全角数字 　ー 全角記号(機種依存文字を除く)	

指定例

```
<Endpoint>endpoint01</Endpoint>
```

Sequence

Sequence(振分けシーケンス)について説明します。

定義項目

表B.60 Sequenceの定義項目

種類	名称	説明	備考
値	---	振分け先のシーケンス名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 　ー 英小文字 　ー 数字 　ー -(ハイフン) 　ー _(アンダーバー) ・ 日本語 　ー 全角漢字(JIS第1水準漢字・第2水準漢字) 　ー 全角ひらがな 　ー 全角カタカナ 　ー 全角英字 　ー 全角数字 　ー 全角記号(機種依存文字を除く)	必須 最大256文字

指定例

```
<Sequence>sequence01</Sequence>
```

B.10 グループ定義の詳細

グループ定義の詳細について説明します。

B.10.1 GroupList

GroupList(SQC連携グループ定義のルート要素)について説明します。

定義項目

表B.61 GroupListの定義項目

種類	名称	説明	備考
子要素	Group	グループ情報を指定します。	複数指定可

指定例

```
<GroupList>
      <Group>…</Group>
      <Group>…</Group>
</GroupList>
```

B.10.1.1 Group

Group(グループ情報)について説明します。

定義項目

表B.62 Groupの定義項目

種類	名称	説明	備考
属性	name	グループ名を指定します。 システム内で一意な名前を指定してください。	必須 最大64バイト グループ名に":"(コロン)は、使用できません。
子要素	EndpointList	グループに属するサービスエンドポイントのリストを指定します。	
子要素	QueueList	グループに属するキューのリストを指定します。	

指定例

```
<Group name="group01">
      <EndpointList>…</EndpointList>
      <QueueList>…</QueueList>
</Group>
```

EndpointList

EndpointList(サービスエンドポイントのリスト)について説明します。

定義項目

表B.63 EndpointListの定義項目

種類	名称	説明	備考
要素	Endpoint	グループに属するサービスエンドポイントのリストを指定します。	複数指定可

指定例

```
<EndpointList>
      <Endpoint>…</Endpoint>
      <Endpoint>…</Endpoint>
</EndpointList>
```

Endpoint

Endpoint(サービスエンドポイント情報)について説明します。

定義項目

表B.64 Endpointの定義項目

種類	名称	説明	備考
属性	name	サービスエンドポイント名を指定します。 システム内で一意な名前を指定してください。 以下の文字を組み合わせて指定できます。 ・ 半角文字 ー 英小文字 ー 数字 ー -(ハイフン) ー _(アンダーバー) ・ 日本語 ー 全角漢字(JIS第1水準漢字・第2水準漢字) ー 全角ひらがな ー 全角カタカナ ー 全角英字 ー 全角数字 ー 全角記号(機種依存文字を除く)	必須 最大256文字

指定例

```
<Endpoint name="endpoint01"/>
```

QueueList

QueueList(キューのリスト)について説明します。

定義項目

表B.65 QueueListの定義項目

種類	名称	説明	備考
要素	Queue	グループに属するキューのリストを指定します。	複数指定可

指定例

```
<QueueList>
      <Queue>…</Queue>
      <Queue>…</Queue>
</QueueList>
```

Queue

Queue(キュー情報)について説明します。

定義項目

表B.66 Queueの定義項目

種類	名称	説明	備考
属性	name	キュー名を指定します。キューの種類がMQDの場合は、MQDキュー名指定してください。 S EE キューの種類がMQD以外の場合は、Destination名を指定してください。	必須
属性	type	キューのタイプを指定します。以下のどれかを指定します。 ・ JMS JMSキューの場合 ・ MQD S EE MQDキューの場合 ・ JMS-R JMS-Rキューの場合	必須
子要素	Description	キューについての説明を指定します。	
子要素	MQDSystem	MQDシステム名を指定します。type属性がMQDの場合に指定してください。省略した場合、MQDシステム名は“MQD001”となります。 S EE	

指定例

```
<Queue name="queue01" type="JMS">
      <Description>…</Description>
</Queue>
<Queue name="QUEUE1" type="MQD"/>
      <Description>…</Description>
      <MQDSystem>…</MQDSystem>
</Queue>
```

Description

Description(説明)について説明します。

定義項目

表B.67 Descriptionの定義項目

種類	名称	説明	備考
値	---	キューについての説明を指定します。	グループ名に“:”(コロン)は、

種類	名称	説明	備考
			使用できません。

指定例

```
<Description>見積り業務システム用受信キュー</Description>
```

MQDSystem S EE

MQDSystem(MQDシステム情報)について説明します。

定義項目

表B.68 MQDSystemの定義項目

種類	名称	説明	備考
値	---	MQDシステム名を指定します。省略した場合、MQDシステム名は“MQD001”となります。	

指定例

```
<MQDSystem>MQD001</MQDSystem>
```

B.11 ヘッダ伝送定義の詳細

ヘッダ伝送定義の詳細について説明します。

用語の定義:

ここでは、RFC2616に従いHTTPヘッダのfield-nameとfield-valueという単語を使います。

例）

Content-Type : text/xml

field-nameは“Content-Type”になります。

field-valueは“text/xml”になります。

ヘッダ伝送定義とは、ISIのSOAPアダプタにてカスタムHTTPヘッダの送受信を行うための定義です。

ヘッダ伝送の定義を行うことで、SOAP通信のサービスエンドポイント単位で、カスタムHTTPヘッダの送受信が可能になります。インバウンド、アウトバウンドどちらのサービスエンドポイントでも利用できます。

具体的には、ヘッダ伝送定義を作成することで以下の処理が可能になります。

- SOAPメッセージ受信時に、受信したHTTPヘッダの値(field-value)を、共通メッセージのヘッダパラメタ値に格納
- SOAPメッセージ送信時に、共通メッセージのヘッダパラメタ値を、HTTPヘッダの値に指定して送信

ヘッダ伝送定義ファイル

ヘッダ伝送の定義は、以下のファイルを編集します。

本定義ファイルは、インストール時に作成されません。必要に応じてユーザが作成してください。

W

<INSDIR>¥etc¥def¥soap¥headercoop.properties

S L

/opt/FJSVesi/etc/def/soap/headercoop.properties

ヘッダ伝送定義フォーマット

ヘッダ伝送定義ファイルに記述する、ヘッダ伝送定義のフォーマットについて説明します。ヘッダ伝送の定義は、以下のフォーマットに従った内容を、ヘッダ伝送定義ファイルに一行ずつ記述することで定義されます。

ヘッダ伝送定義の定義フォーマットは以下のとおりです。

- 送信ヘッダの定義フォーマット
  “サービスエンドポイント名”.**send**.“field-name”=“共通メッセージのヘッダキー値”

項目名	説明	指定例
サービスエンドポイント名	HTTPヘッダ伝送を行うSOAPアダプタのサービスエンドポイント名を指定します。	service1
field-name	送信したいHTTPヘッダ名(field-name)を指定します。	X-SAMPLE
共通メッセージのヘッダキー値	field-valueとして送信したい値が格納された、共通メッセージのヘッダキー値を指定します。	user.send.header1

```
指定例）
service1.send.X-SAMPLE=user.send.header1
```

この例では、“service1”というサービスエンドポイント定義について、SOAPメッセージ送信時に、“X-SAMPLE”というfield-nameのHTTPヘッダが送信されます。

このHTTPヘッダのfield-valueには、共通メッセージのヘッダキー値“user.send.header1”に対応する値が指定されます。

- 受信ヘッダの定義フォーマット
  “サービスエンドポイント名”.**receive**.“field-name”=“共通メッセージのヘッダキー値”

項目名	説明	指定例
サービスエンドポイント名	HTTPヘッダ伝送を行うSOAPアダプタのサービスエンドポイント名を指定します。	service1
field-name	受信したいHTTPヘッダ名(field-name)を指定します。	X-SAMPLE
共通メッセージのヘッダキー値	受信したfield-valueを格納する、共通メッセージのヘッダキー値を指定します。	user.receive.header1

```
指定例）
service1.receive.X-SAMPLE=user.receive.header1
```

この例では、“service1”というサービスエンドポイント定義について、SOAPメッセージ受信時に、“X-SAMPLE”というfield-nameのHTTPヘッダのfield-valueが、共通メッセージのヘッダキー値“user.receive.header1”に格納されます。

付録C　定義の例

ここでは、定義ファイルの内容と定義する項目の例について説明します。

注意

定義内の要素名・属性名は、英大文字、英小文字が区別されます。

C.1　サービスエンドポイント定義

サービスエンドポイント定義の詳細について説明します。

C.1.1　サービスエンドポイント定義の構成

サービスエンドポイント定義の構成と指定例は、以下のとおりです。
なお、定義する項目についての詳細は、定義する情報に応じて以下を参照してください。

- サービスエンドポイント定義:“サービスエンドポイント定義の詳細”
- サービス情報:“サービス情報の詳細”

構成

サービスエンドポイント定義

```
EndpointList
        Endpoint
                Description
                ServiceName
                EsiName
                QueueName
                SequenceName
                DistributionName
                SequenceList
                        SequenceName
```

サービス情報

```
Service
      Description
      PropertyList
              Property
```

指定例

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpointa" sync="false">
        <Description>業務Ａ用サービスエンドポイント</Description>
        <SequenceName>gyoumu01</SequenceName>
        <QueueName>ESIInboundQueue</QueueName>
    </Endpoint>
    <Endpoint name="endpointb" sync="true">
        <Description>業務Ｂ用集信サービスエンドポイント</Description>
        <ServiceName>service01</ServiceName> ……(a)
```

```
    </Endpoint>
</EndpointList>
```

a. サービス情報“service01.xml”で指定したサービス名を指定します。

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="filewatch" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="filename">/FTPhome/recvdata.dat</Property>
        <Property name="watch">true</Property>
        <Property name="split">CR</Property>
        <Property name="split_CR">0x0D0A</Property>
    </PropertyList>
</Service>
```

C.1.2 SOAPのメッセージング機能の定義

SOAP を利用した転送を行う場合、サービスエンドポイント定義に、SOAP通信で必要な情報を定義します。
SOAPの定義には、以下の種類があります。

- インバウンド非同期
- インバウンド同期
- アウトバウンド非同期
- アウトバウンド同期

C.1.2.1 インバウンド非同期

インバウンド非同期のサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <QueueName>ESIInboundQueue</QueueName>
        <SequenceName>sequence01</SequenceName>
        <ServiceName>msg_ws</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.1 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。 サービス利用側からの送信先となるURLの後ろのクエリストリングで指定されます。 また、インバウンドルーティングを利用する場合は、ルーティング定義の振分け先サービスエンドポイント名で指定されます。	endpoint01

キー名	説明	指定例
	インバウンドルーティング定義名の指定については、“ISI 導入ガイド”を参照してください。	
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
QueueName	JMSキュー名を指定します。	ESIInboundQueue
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	msg_ws

サービス情報

サービス情報:msg_ws.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="msg_ws" type="SOAP-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.2 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	msg_ws
type	サービスのタイプ“SOAP-MSG”を指定します。	SOAP-MSG
Property		
jmsr.messageAccumulate	JMS-Rキューを使用する場合、メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

C.1.2.2 インバウンド同期

インバウンド同期のサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>msg_ws</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.3 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。 サービスエンドポイント定義で使用するサービスエンドポイント名は、サービス利用側からの送信先となるURLの後ろのクエリストリングで指定されます。 また、インバウンドルーティングを利用する場合は、ルーティング定義の振分け先サービスエンドポイント名で指定されます。 インバウンドルーティング定義名の指定については、“ISI 導入ガイド”を参照してください。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	msg_ws

サービス情報

サービス情報:msg_ws.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="msg_ws" type="SOAP-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
</Service>
```

表C.4 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	msg_ws
type	サービスのタイプ“SOAP-MSG”を指定します。	SOAP-MSG

C.1.2.3 アウトバウンド非同期

アウトバウンド非同期のサービスエンドポイント定義の指定例は、以下のとおりです。

注意

- メッセージ送信呼出時の共通メッセージのペイロードに、SOAPメッセージ(XML文書)が格納されている必要があります。
- サービス情報でレスポンス処理を指定し、かつ、通信先がOneway方式のWebサービスでレスポンスにSOAPメッセージを含まない場合は、レスポンス処理で格納される共通メッセージのペイロードは空になります。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <QueueName>ESISendQueue</QueueName>
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.5 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
QueueName	JMSキュー名を指定します。	ESISendQueue
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="SOAP" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <Description>0</Description>
    <PropertyList>
        <Property name="JCA.Version" secret="false">1</Property>
        <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/esisoapadapter</Property>
        <Property name="JCA.Interaction.ImplClass.type" secret="false">bool_return</Property>
        <Property name="JCA.InteractionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.soap.outbound.cci.ESISoapAdapterInteractionSpec</Property>
        <Property name="JCA.InteractionSpec.Param1.name" secret="false">setResponseSet</Property>
        <Property name="JCA.InteractionSpec.Param1.type" secret="false">boolean</Property>
        <Property name="JCA.InteractionSpec.Param1.value" secret="false">true</Property>
        <Property name="JCA.InteractionSpec.Param2.name" secret="false">setReqrespParameter</Property>
        <Property name="JCA.InteractionSpec.Param2.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param2.value" secret="false">usr.param1,usr.param2</Property>
        <Property name="JCA.InteractionSpec.Param3.name" secret="false">setResponseEndpoint</Property>
        <Property name="JCA.InteractionSpec.Param3.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param3.value" secret="false">endpointRsp</Property>
        <Property name="JCA.InteractionSpec.Param4.name" secret="false">setTimeout</Property>
        <Property name="JCA.InteractionSpec.Param4.type" secret="false">int</Property>
        <Property name="JCA.InteractionSpec.Param4.value" secret="false">60000</Property>
        <Property name="JCA.InteractionSpec.Param5.name" secret="false">setSoapAction</Property>
        <Property name="JCA.InteractionSpec.Param5.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param5.value" secret="false">callBytes</Property>
        <Property name="JCA.InteractionSpec.Param6.name" secret="false">setParse</Property>
        <Property name="JCA.InteractionSpec.Param6.type" secret="false">boolean</Property>
        <Property name="JCA.InteractionSpec.Param6.value" secret="false">true</Property>
        <Property name="JCA.InteractionSpec.Param7.name" secret="false">setFaultHandle</Property>
```

```
        <Property name="JCA.InteractionSpec.Param7.type" secret="false">boolean</Property>
        <Property name="JCA.InteractionSpec.Param7.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.soap.outbound.cci.ESISoapAdapterConnectionSpec</Property>
        <Property name="JCA.ConnectionSpec.Param0.name" secret="false">setServiceName</Property>
        <Property name="JCA.ConnectionSpec.Param0.type" secret="false">@INPUT</Property>
        <Property name="JCA.ConnectionSpec.Param0.value" secret="false">SERVICE_NAME</Property>
        <Property name="JCA.ConnectionSpec.Param1.name" secret="false">setUrlEndpoint</Property>
        <Property name="JCA.ConnectionSpec.Param1.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param1.value" secret="false">http://MyHost/esisoap/services/
ESISoapRpcProvider</Property>
        <Property name="JCA.ConnectionSpec.Param2.name" secret="false">setBasicAuthSet</Property>
        <Property name="JCA.ConnectionSpec.Param2.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param2.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.Param3.name" secret="false">setBasicAuthUsr</Property>
        <Property name="JCA.ConnectionSpec.Param3.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param3.value" secret="false">userName</Property>
        <Property name="JCA.ConnectionSpec.Param4.name" secret="false">setBasicAuthPwd</Property>
        <Property name="JCA.ConnectionSpec.Param4.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param4.value" secret="true">userPassword</Property>
        <Property name="JCA.ConnectionSpec.Param5.name" secret="false">setProxySet</Property>
        <Property name="JCA.ConnectionSpec.Param5.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param5.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.Param6.name" secret="false">setNonProxyHosts</Property>
        <Property name="JCA.ConnectionSpec.Param6.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param6.value" secret="false">localhost|*.myhost.com</Property>
        <Property name="JCA.ConnectionSpec.Param7.name" secret="false">setProxyHost</Property>
        <Property name="JCA.ConnectionSpec.Param7.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param7.value" secret="false">proxy.proxyhost.com</Property>
        <Property name="JCA.ConnectionSpec.Param8.name" secret="false">setProxyPort</Property>
        <Property name="JCA.ConnectionSpec.Param8.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param8.value" secret="false">8080</Property>
        <Property name="JCA.ConnectionSpec.Param9.name" secret="false">setProxyAuthSet</Property>
        <Property name="JCA.ConnectionSpec.Param9.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param9.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.Param10.name" secret="false">setProxyAuthUsr</Property>
        <Property name="JCA.ConnectionSpec.Param10.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param10.value" secret="false">proxyUserName</Property>
        <Property name="JCA.ConnectionSpec.Param11.name" secret="false">setProxyAuthPwd</Property>
        <Property name="JCA.ConnectionSpec.Param11.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param11.value" secret="true">proxyPassword</Property>
        <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.InputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
        <Property name="JCA.InputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.InputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
        <Property name="JCA.InputRecord.Map2.type" secret="false">headerSet</Property>
        <Property name="JCA.InputRecord.Map2.value" secret="false">com.fujitsu.esi</Property>
        <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.OutputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
        <Property name="JCA.OutputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.OutputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
        <Property name="JCA.OutputRecord.Map2.type" secret="false">headerSet</Property>
        <Property name="JCA.OutputRecord.Map2.value" secret="false">com.fujitsu.esi</Property>
    </PropertyList>
</Service>
```

ユーザが指定するPropertyの項目について以下の定義項目の表に記載します。

この表で記載した以外の項目は、上記の定義をそのままサービス情報に記載してください。

表C.6 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“SOAP”を指定します。	SOAP
Property		
JCA.InteractionSpec.Param1.value	レスポンス処理を実行するかどうかを指定します。 ・ true：実行する ・ false：実行しない	true
JCA.InteractionSpec.Param2.value	レスポンス処理を実行する場合に、メッセージ送信された共通メッセージのヘッダ情報からコピーするパラメタ名を指定します。 複数のパラメタ名を指定する場合は、,(カンマ)で区切って指定してください。	usr.param1,usr.param2
JCA.InteractionSpec.Param3.value	レスポンスを格納するサービスエンドポイント定義名を指定します。	endpointRsp
JCA.InteractionSpec.Param4.value	接続のタイムアウト時間をミリ秒で指定します。 以下の文字列を指定できます。 ・ 正の整数値 デフォルトは、利用するSOAPエンジンに依存します。 Interstage Application Server：300000(5分)	60000
JCA.InteractionSpec.Param5.value	接続先のWebサービスが要求するSOAP Actionを指定します。 SOAP Actionが必要かどうかは接続先のWebサービスに確認してください。	callBytes
JCA.InteractionSpec.Param6.value	SOAPメッセージの妥当性検証を行うかどうかを指定します。 SOAP1.1およびSOAP1.2のスキーマ定義でリクエストとレスポンスのSOAPメッセージの妥当性検証を行います。 ・ true：確認する ・ false：確認しない(デフォルト)	true
JCA.InteractionSpec.Param7.value	Webサービス呼び出し結果がSOAP Faultだった場合にシーケンスをエラーにするかどうかを指定します。	true

キー名	説明	指定例
	・ true：エラーにする ・ false：エラーにしない(デフォルト)	
JCA.ConnectionSpec.Param1.value	接続先のURLを指定します。	http://MyHost/esisoap/services/ESISoapRpcProvider
JCA.ConnectionSpec.Param2.value	Basic認証を行うかどうかを指定します。 ・ true：行う “true”を指定した場合は、“JCA.ConnectionSpec.Param3.value”でユーザ名を、“JCA.ConnectionSpec.Param4.value”でパスワードを指定してください。 ・ false：行わない(デフォルト)	true
JCA.ConnectionSpec.Param3.value	“JCA.ConnectionSpec.Param2.value”にtrueを指定した場合、Basic認証のログインユーザ名を指定します。	userName
JCA.ConnectionSpec.Param4.value	“JCA.ConnectionSpec.Param2.value”にtrueを指定した場合、Basic認証のログインパスワードを指定します。	userPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true：暗号化する ・ false：暗号化しない	true
JCA.ConnectionSpec.Param5.value	Proxyサーバを利用するかどうかを指定します。 ・ true：利用する “true”を指定した場合は、“JCA.ConnectionSpec.Param7.value”でホスト名を、“JCA.ConnectionSpec.Param8.value”でポート番号を指定してください。 ・ false：利用しない(デフォルト)	true
JCA.ConnectionSpec.Param6.value	プロキシを経由せずに接続するホスト名を指定します。 複数のホスト名を指定する場合は、“|”(パイプ文字)で区切って指定してください。	localhost|*.myhost.com
JCA.ConnectionSpec.Param7.value	“JCA.ConnectionSpec.Param5.value”に“true”を指定した場合、Proxyサーバのホスト名を指定します。	proxy.proxyhost.com

キー名	説明	指定例
JCA.ConnectionSpec.Param8.value	“JCA.ConnectionSpec.Param5.value”に“true”を指定した場合、Proxyサーバのポート番号を指定します。	8080
JCA.ConnectionSpec.Param9.value	Proxyサーバで利用者認証を行うかどうかを指定します。以下のどちらかを指定します。 ・ true:行う “true”を指定した場合は、“JCA.ConnectionSpec.Param10.value”でユーザ名を、“JCA.ConnectionSpec.Param11.value”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true
JCA.ConnectionSpec.Param10.value	“JCA.ConnectionSpec.Param9.value”に“true”を指定した場合、Proxyサーバのログインユーザ名を指定します。	proxyUserName
JCA.ConnectionSpec.Param11.value	“JCA.ConnectionSpec.Param9.value”に“true”を指定した場合、Proxyサーバのログインパスワードを指定します。	proxyPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true

C.1.2.4 アウトバウンド同期

アウトバウンド同期のサービスエンドポイント定義の指定例は、以下のとおりです。

注意

サービス呼出時の共通メッセージのペイロードに、SOAPメッセージが格納されている必要があります。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
    <Endpoint name="outbound02" sync="true">
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.7 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	outbound02
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="SOAP" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <Description>outbound02</Description>
    <PropertyList>
        <Property name="JCA.Version" secret="false">1</Property>
        <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/esisoapadapter</Property>
        <Property name="JCA.Interaction.ImplClass.type" secret="false">bool_return</Property>
        <Property name="JCA.InteractionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.soap.outbound.cci.ESISoapAdapterInteractionSpec</Property>
        <Property name="JCA.InteractionSpec.Param4.name" secret="false">setTimeout</Property>
        <Property name="JCA.InteractionSpec.Param4.type" secret="false">int</Property>
        <Property name="JCA.InteractionSpec.Param4.value" secret="false">60000</Property>
        <Property name="JCA.InteractionSpec.Param5.name" secret="false">setSoapAction</Property>
        <Property name="JCA.InteractionSpec.Param5.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param5.value" secret="false">callBytes</Property>
        <Property name="JCA.InteractionSpec.Param6.name" secret="false">setParse</Property>
        <Property name="JCA.InteractionSpec.Param6.type" secret="false">boolean</Property>
        <Property name="JCA.InteractionSpec.Param6.value" secret="false">true</Property>
        <Property name="JCA.InteractionSpec.Param7.name" secret="false">setFaultHandle</Property>
        <Property name="JCA.InteractionSpec.Param7.type" secret="false">boolean</Property>
        <Property name="JCA.InteractionSpec.Param7.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.soap.outbound.cci.ESISoapAdapterConnectionSpec</Property>
        <Property name="JCA.ConnectionSpec.Param0.name" secret="false">setServiceName</Property>
        <Property name="JCA.ConnectionSpec.Param0.type" secret="false">@INPUT</Property>
        <Property name="JCA.ConnectionSpec.Param0.value" secret="false">SERVICE_NAME</Property>
        <Property name="JCA.ConnectionSpec.Param1.name" secret="false">setUrlEndpoint</Property>
        <Property name="JCA.ConnectionSpec.Param1.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param1.value" secret="false">http://MyHost/esisoap/services/
ESISoapRpcProvider</Property>
        <Property name="JCA.ConnectionSpec.Param2.name" secret="false">setBasicAuthSet</Property>
        <Property name="JCA.ConnectionSpec.Param2.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param2.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.Param3.name" secret="false">setBasicAuthUsr</Property>
        <Property name="JCA.ConnectionSpec.Param3.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param3.value" secret="false">userName</Property>
        <Property name="JCA.ConnectionSpec.Param4.name" secret="false">setBasicAuthPwd</Property>
        <Property name="JCA.ConnectionSpec.Param4.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param4.value" secret="true">userPassword</Property>
        <Property name="JCA.ConnectionSpec.Param5.name" secret="false">setProxySet</Property>
        <Property name="JCA.ConnectionSpec.Param5.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param5.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.Param6.name" secret="false">setNonProxyHosts</Property>
        <Property name="JCA.ConnectionSpec.Param6.type" secret="false">String</Property>
```

```
        <Property name="JCA.ConnectionSpec.Param6.value" secret="false">localhost|*.myhost.com</Property>
        <Property name="JCA.ConnectionSpec.Param7.name" secret="false">setProxyHost</Property>
        <Property name="JCA.ConnectionSpec.Param7.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param7.value" secret="false">proxy.proxyhost.com</Property>
        <Property name="JCA.ConnectionSpec.Param8.name" secret="false">setProxyPort</Property>
        <Property name="JCA.ConnectionSpec.Param8.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param8.value" secret="false">8080</Property>
        <Property name="JCA.ConnectionSpec.Param9.name" secret="false">setProxyAuthSet</Property>
        <Property name="JCA.ConnectionSpec.Param9.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param9.value" secret="false">true</Property>
        <Property name="JCA.ConnectionSpec.Param10.name" secret="false">setProxyAuthUsr</Property>
        <Property name="JCA.ConnectionSpec.Param10.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param10.value" secret="false">proxyUserName</Property>
        <Property name="JCA.ConnectionSpec.Param11.name" secret="false">setProxyAuthPwd</Property>
        <Property name="JCA.ConnectionSpec.Param11.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param11.value" secret="true">proxyPassword</Property>
        <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.InputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
        <Property name="JCA.InputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.InputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
        <Property name="JCA.InputRecord.Map2.type" secret="false">headerSet</Property>
        <Property name="JCA.InputRecord.Map2.value" secret="false">*</Property>
        <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.OutputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
        <Property name="JCA.OutputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.OutputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
        <Property name="JCA.OutputRecord.Map2.type" secret="false">headerSet</Property>
        <Property name="JCA.OutputRecord.Map2.value" secret="false">*</Property>
    </PropertyList>
</Service>
```

ユーザが指定するPropertyの項目について以下の定義項目の表に記載します。

この表で記載した以外の項目は、上記の定義をそのままサービス情報に記載してください。

表C.8 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“SOAP”を指定します。	SOAP
Property		
JCA.InteractionSpec.Param4.value	接続のタイムアウト時間をミリ秒で指定します。 以下の文字列を指定できます。 ・ 正の整数値 デフォルトは、利用するSOAPエンジンに依存します。 Interstage Application Server：300000(5分)	60000
JCA.InteractionSpec.Param5.value	接続先のWebサービスが要求するSOAP Actionを指定します。 SOAP Actionが必要かどうかは接続先のWebサービスに確認してください。	callBytes

キー名	説明	指定例
JCA.InteractionSpec.Param 6.value	SOAPメッセージの妥当性検証を行うかどうかを指定します。SOAP1.1およびSOAP1.2のスキーマ定義でリクエストとレスポンスのSOAPメッセージの妥当性検証を行います。 ・ true:確認する ・ false:確認しない(デフォルト)	true
JCA.InteractionSpec.Param 7.value	Webサービス呼び出し結果がSOAP Faultだった場合にシーケンスをエラーにするかどうかを指定します。 ・ true:エラーにする ・ false:エラーにしない(デフォルト)	true
JCA.ConnectionSpec.Param 1.value	接続先のURLを指定します。	http://MyHost/esisoap/services/ESISoapRpcProvider
JCA.ConnectionSpec.Param 2.value	Basic認証を行うかどうかを指定します。 ・ true:行う “true”を指定した場合は、“JCA.ConnectionSpec.Param3.value”でユーザ名を、“JCA.ConnectionSpec.Param4.value”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true
JCA.ConnectionSpec.Param 3.value	“JCA.ConnectionSpec.Param 2.value”にtrueを指定した場合、Basic認証のログインユーザ名を指定します。	userName
JCA.ConnectionSpec.Param 4.value	“JCA.ConnectionSpec.Param 2.value”にtrueを指定した場合、Basic認証のログインパスワードを指定します。	userPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true
JCA.ConnectionSpec.Param 5.value	Proxyサーバを利用するかどうかを指定します。 ・ true:利用する “true”を指定した場合は、“JCA.ConnectionSpec.Param7.value”でホスト名を、	true

キー名	説明	指定例
	“JCA.ConnectionSpec.Param8.value”でポート番号を指定してください。 ・ false：利用しない(デフォルト)	
JCA.ConnectionSpec.Param6.value	プロキシを経由せずに接続するホスト名を指定します。 複数のホスト名を指定する場合は、“|”(パイプ文字)で区切って指定してください。	localhost|*.myhost.com
JCA.ConnectionSpec.Param7.value	“JCA.ConnectionSpec.Param5.value”に“true”を指定した場合、Proxyサーバのホスト名を指定します。	proxy.proxyhost.com
JCA.ConnectionSpec.Param8.value	“JCA.ConnectionSpec.Param5.value”に“true”を指定した場合、Proxyサーバのポート番号を指定します。	8080
JCA.ConnectionSpec.Param9.value	Proxyサーバで利用者認証を行うかどうかを指定します。以下のどちらかを指定します。 ・ true：行う “true”を指定した場合は、“JCA.ConnectionSpec.Param10.value”でユーザ名を、“JCA.ConnectionSpec.Param11.value”でパスワードを指定してください。 ・ false：行わない(デフォルト)	true
JCA.ConnectionSpec.Param10.value	“JCA.ConnectionSpec.Param9.value”に“true”を指定した場合、Proxyサーバのログインユーザ名を指定します。	proxyUserName
JCA.ConnectionSpec.Param11.value	“JCA.ConnectionSpec.Param9.value”に“true”を指定した場合、Proxyサーバのログインパスワードを指定します。	proxyPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true：暗号化する ・ false：暗号化しない	true

C.1.3 ファイル転送の定義

FTPを利用したファイル転送を行う場合、サービスエンドポイント定義に、ファイル転送で必要な情報を定義します。
ファイル転送の定義には、以下の種類があります。

・ 受信

・ 送信

C.1.3.1 受信

インバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>gyoumu01</SequenceName>
        <ServiceName>service001</ServiceName>
        <QueueName>ESIInboundQueue</QueueName>
    </Endpoint>
    <Endpoint name="endpoint02" sync="false">
        <SequenceName>gyoumu02</SequenceName>
        <ServiceName>service002</ServiceName>
        <QueueName>ESIInboundQueue</QueueName>
    </Endpoint>
    <Endpoint name="endpoint03" sync="false">
        <SequenceName>gyoumu03</SequenceName>
        <ServiceName>service003</ServiceName>
        <QueueName>ESIInboundQueue</QueueName>
    </Endpoint>
</EndpointList>
```

表C.9 定義項目

キー名		説明	指定例
1	Endpoint		
	name	サービスエンドポイント名を指定します。	endpoint01
	sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
	SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	gyoumu01
	ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service001
	QueueName	JMSキュー名を指定します。	ESIInboundQueue
2	Endpoint		
	name	サービスエンドポイント名を指定します。	endpoint02
	sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
	SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	gyoumu02
	ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service002
	QueueName	JMSキュー名を指定します。	ESIInboundQueue
3	Endpoint		

キー名		説明	指定例
	name	サービスエンドポイント名を指定します。	endpoint03
	sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
	SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	gyoumu03
	ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service003
	QueueName	JMSキュー名を指定します。	ESIInboundQueue

サービス情報

サービス情報:service001.xml(改行コードを指定して分割する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service001" type="filewatch" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="filename">/FTPhome/recvdata.dat</Property>
        <Property name="watch">true</Property>
        <Property name="split">CR</Property>
        <Property name="split_CR">0x0D0A</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.10 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service001
type	“filewatch”(ファイルの受信を監視し、シーケンスを開始)を指定します。	filewatch
Property		
filename	監視対象のファイル名を指定します。ファイル名は、絶対パスで指定してください。	/FTPhome/recvdata.dat
watch	ファイルを監視するかどうかを指定します。 ・ true:ファイルを監視する(デフォルト) ・ false:ファイルを監視しない	true
split	ファイル分割の種類を指定します。 ・ no:ファイルを分割しない(デフォルト) ・ Size:ファイルサイズで分割 “Size”を指定した場合は、“split_size”で、分割するファイルサイズを指定してください。 ・ CR:監視ファイルのデータを改行コードで分割 “CR”を指定した場合は、“split_CR”で、改行コードを指定してください。	CR
split_CR	“split”に“CR”を指定した場合、監視ファイルのデータの改行コードを指定します。	0x0D0A

キー名	説明	指定例
	・ 0x0D0A:監視ファイルのデータの改行コードがCRLFの場合 ・ 0x0A:監視ファイルのデータの改行コードがCRの場合 ・ 0x0D:監視ファイルのデータの改行コードがLFの場合	
jmsr.messageAccumulate	JMS-Rキューを使用する場合、メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

サービス情報:service002.xml(サイズを指定して分割する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service002" type="filewatch" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="filename">/FTPhome/recvdata2.dat</Property>
        <Property name="watch">true</Property>
        <Property name="split">Size</Property>
        <Property name="split_size">1024</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.11 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service002
type	“filewatch”(ファイルの受信を監視し、シーケンスを開始)を指定します。	filewatch
Property		
filename	監視対象のファイル名を指定します。ファイル名は、絶対パスで指定してください。	/FTPhome/recvdata2.dat
watch	ファイルを監視するかどうかを指定します。 ・ true:ファイルを監視する(デフォルト) ・ false:ファイルを監視しない	true
split	ファイル分割の種類を指定します。 ・ no:ファイルを分割しない(デフォルト) ・ Size:ファイルサイズで分割 “Size”を指定した場合は、“split_size”で、分割するファイルサイズを指定してください。 ・ CR:監視ファイルのデータを改行コードで分割 “CR”を指定した場合は、“split_CR”で、改行コードを指定してください。	Size
split_size	“split”に“Size”を指定した場合、分割するファイルサイズを指定します。	1024

キー名	説明	指定例
jmsr.messageAccumulate	JMS-Rキューを使用する場合、メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

サービス情報:service003.xml(ファイルを監視する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service003" type="filewatch" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="watchtype">ackfile</Property>
        <Property name="filename">/FTPhome/*.dat</Property>
        <Property name="ackfilename">/FTPhome/$[FILENAME].ack</Property>
    </PropertyList>
</Service>
```

表C.12 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service003
type	“filewatch”(ファイルの受信を監視し、シーケンスを開始)を指定します。	filewatch
Property		
watchtype	ファイルの監視方法を指定します。以下のどちらかを指定します。 ・ time 時間で監視します。 ・ ackfile 指定したファイルが到着するまで監視します。	ackfile
filename	監視対象のファイル名を指定します。ファイル名は、絶対パスで指定してください。 ファイル名には、ワイルドカードを指定できます。	/FTPhome/*.dat
ackfilename	監視する完了ファイルのファイルパス(絶対パス)を指定します。ファイルパス内の監視完了ファイル名には、埋込み文字列“$[FILENAME]”を必ず指定してください。埋込み文字列“$[FILENAME]”は、[filename]で指定したファイル名に置き換えられます。	/FTPhome/$[FILENAME].ack “/FTPhome/*.dat.ack”になります。

ポイント

- シーケンス定義やメディエータファンクション定義を変更する場合、ほかの監視部分は動作させたまま、一部分だけを監視対象からはずしたいときは、Property項目の“watch”に“false”を指定してください。“false”を指定した定義は、監視対象外となり動作しません。
- 監視対象のファイルを指定する“filename”は、複数のサービスエンドポイントで同じファイルを指定しないでください。複数のサービスエンドポイントから同じ監視対象のファイルを共有した場合は、どのサービスエンドポイントから実行されるか保証されません。

C.1.3.2 送信

アウトバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint02" sync="false">
        <ServiceName>service002</ServiceName>
        <QueueName>ESIFTPQueue</QueueName>
    </Endpoint>
</EndpointList>
```

表C.13 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint02
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service002
QueueName	JMSキュー名を指定します。	ESIFTPQueue

サービス情報

サービス情報:service002.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service002" type="ftpsend" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="dirname">/FTPhome/</Property>
        <Property name="filename">senddata_$[DATE].dat</Property>
        <Property name="sendackfile">true</Property>
        <Property name="ackdirname">/FTPhome/</Property>
        <Property name="ackfilename">$[FILENAME].ack</Property>
        <Property name="hostname">127.0.0.1</Property>
        <Property name="port">21</Property>
        <Property name="username">user</Property>
        <Property name="password" secret="true">password</Property>
        <Property name="merge">0</Property>
        <Property name="overwrite">true</Property>
        <Property name="statuscheck">true</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.14 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します	service002

キー名	説明	指定例
type	“ftpsend”(シーケンスの実行結果をFTP送信)を指定します。	ftpsend
Property		
dirname	送信先のディレクトリ名を指定します。 デフォルトは、“./”(ログイン時のカレント・ディレクトリ)です。 ディレクトリ名には、アスキーコードを指定してください。	/FTPhome/
filename	送信先のファイル名を指定します。 ファイル名にはアスキーコードを指定してください。 埋込み文字列“$[DATE]”、“$[SEQNAME]”を指定できます。	senddata_$[DATE].dat
sendackfile	送信完了ファイルを転送するかどうかを指定します。以下のどちらかを指定します。 ・ true 転送します。 ・ false 転送しません。	true
ackdirname	送信する完了ファイルのディレクトリを指定します。 デフォルトは、“./”(ログイン時のカレント・ディレクトリ)です。ディレクトリ名には、アスキーコードを指定してください。	/FTPhome/
ackfilename	送信する完了ファイルのファイル名を指定します。ファイル名にはアスキーコードを指定してください。 埋込み文字列“$[FILENAME]”を指定すると、[filename]で指定したファイル名に置き換えられます。	$[FILENAME].ack “senddata_$[DATE].dat.ack”になります。
hostname	接続先のサーバ名、またはIPアドレスを指定します。	127.0.0.1
port	接続先のポート番号を指定します。標準は、“21”です。	21
username	接続サーバのログインユーザ名を指定します。	user
password	接続サーバのログインパスワードを指定します。	password
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true
merge	ファイルを集約するかどうかを指定します。 ・ 0:ファイルをすべて集約して送信(デフォルト) ・ 1:ファイルを集約せず、1ファイルずつ送信	0

キー名	説明	指定例
overwrite	送信先に同一のファイル名が存在した場合、上書きするかどうかを指定します。 ・ true:上書き保存する(デフォルト) ・ false:上書き保存しない	true
statuscheck	ファイル転送結果を確認するかどうかを指定します。 ・ true:転送が成功したかどうかを確認する(デフォルト) ・ false:転送が成功したかどうかを確認しない 通常は“true”を指定してください。 転送先のファイル一覧を取得し、転送結果を確認します。転送先のFTPサーバがlsコマンドをサポートしていない場合は、“false”を指定してください。	true
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

注意

- FTPのファイル転送モードには、アスキー転送モードとバイナリ転送モードの2つがあります。ISIでは、バイナリ転送だけをサポートしています。
- statuscheckプロパティが“false”の場合は、転送結果の確認は行われません。書き込み権限がないなどの理由でファイル転送に失敗している場合でも、esiftpsendコマンドは正常終了します。
- Interstage CollaborationRing 業務連携機能(内蔵FTP集信監視機能)を利用してEAIと連携する場合には、statuscheckプロパティを“false”にし、かつoverwriteプロパティを“true”にしてください。

C.1.4 JMS連携の定義

シーケンス実行した結果のペイロードを、JMSサーバのキューに送信するためのサービスエンドポイント定義を作成します。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="jmssend-ext" sync="false">
        <ServiceName>jmsservice02</ServiceName>
        <QueueName>ESISendQueue</QueueName>
    </Endpoint>
</EndpointList>
```

表C.15 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	jmssend-ext
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	jmsservice02
QueueName	JMSキュー名を指定します。	ESISendQueue

サービス情報

サービス情報:jmsservice02.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="jmsservice02" type="JMS-EXTERNAL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="requestQueueName">PrivateQueue</Property>
    </PropertyList>
</Service>
```

表C.16 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	jmsservice02
type	サービスのタイプ“JMS-EXTERNAL”を指定します。	JMS-EXTERNAL
Property		
requestQueueName	JMS送信先のキュー名を指定します。	PrivateQueue

C.1.5 ListCreator連携の定義

List Creator連携を行うためのサービスエンドポイント定義を作成します。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="listcreator" sync="true">
        <ServiceName>listcreator-service</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.17 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	listcreator

キー名	説明	指定例
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	listcreator-service

サービス情報

サービス情報:listcreator-service.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="listcreator-service" type="LIST-CREATOR" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="soap.urlEndpoint">http://MyHost/LCWebService/services/LCWebSvc</Property>
        <Property name="soap.basicAuthSet">true</Property>
        <Property name="soap.basicAuthUsr">userName</Property>
        <Property name="soap.basicAuthPwd" secret="true">userPassword</Property>
        <Property name="soap.proxySet">true</Property>
        <Property name="soap.nonProxyHosts">localhost|*.myhost.com</Property>
        <Property name="soap.proxyHost">proxy.proxyhost.com</Property>
        <Property name="soap.proxyPort">8080</Property>
        <Property name="soap.proxyAuthSet">true</Property>
        <Property name="soap.proxyAuthUsr">proxyUserName</Property>
        <Property name="soap.proxyAuthPwd" secret="true">proxyPassword</Property>
        <Property name="soap.timeout">60000</Property>
    </PropertyList>
</Service>
```

表C.18 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	listcreator-service
type	サービスのタイプ“LIST-CREATOR”を指定します。	LIST-CREATOR
Property		
soap.urlEndpoint	List Creator Webサービスに接続するためのURLを指定します。	http://MyHost/LCWebService/services/LCWebSvc
soap.basicAuthSet	Basic認証の設定を行うかどうかを指定します。 ・ true:行う “true”を指定した場合は、“soap.basicAuthUsr”でユーザ名を、“soap.basicAuthPwd”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true
soap.basicAuthUsr	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインユーザ名を指定します。	userName
soap.basicAuthPwd	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインパスワードを指定します。	userPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する	true

キー名	説明	指定例
	・ false：暗号化しない	
soap.proxySet	プロキシサーバを利用するかどうかを指定します。 ・ true：利用する “true”を指定した場合は、“soap.proxyHost”でホスト名を、“soap.proxyPort”でポート番号を指定してください。 ・ false：利用しない(デフォルト)	true
soap.nonProxyHosts	プロキシを経由せずに接続するホスト名を指定します。 複数のホスト名を指定する場合は、“|”(パイプ文字)で区切って指定してください。	localhost|*.myhost.com
soap.proxyHost	“soap.proxySet”に“true”を指定した場合、プロキシサーバのホスト名を指定します。	proxy.proxyhost.com
soap.proxyPort	“soap.proxySet”に“true”を指定した場合、プロキシサーバのポート番号を指定します。	8080
soap.proxyAuthSet	プロキシサーバの利用者認証の設定を行うかどうかを指定します。以下のどちらかを指定します。 ・ true：行う “true”を指定した場合は、“soap.proxyAuthUsr”でユーザ名を、“soap.proxyAuthPwd”でパスワードを指定してください。 ・ false：行わない(デフォルト)	true
soap.proxyAuthUsr	“soap.proxyAuthSet”に“true”を指定した場合、プロキシサーバのログインユーザ名を指定します。	proxyUserName
soap.proxyAuthPwd	“soap.proxyAuthSet”に“true”を指定した場合、プロキシサーバのログインパスワードを指定します。	proxyPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true：暗号化する ・ false：暗号化しない	true
soap.timeout	接続先から応答がない場合、接続を切断するタイムアウト時間をミリ秒で指定します。 以下の文字列を指定できます。 ・ 正の整数値 デフォルトは、利用するSOAPエンジンに依存します。 ・ Interstage Application Server：300000(5分)	60000

C.1.6 CORBA連携の定義EE

サービス利用側のアプリケーションからのメッセージを、CORBAを利用して受信するためのサービスエンドポイント定義を作成します。

C.1.6.1 インバウンド

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.19 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="CORBA" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
</Service>
```

表C.20 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“CORBA”を指定します。	CORBA

C.1.6.2 アウトバウンド

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <ServiceName>service01</ServiceName>
```

```
    </Endpoint>
</EndpointList>
```

表C.21 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Service name="" type="CORBA">
    <Description/>
    <PropertyList>
        <Property name="JCA.Version" secret="false">1</Property>
        <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/esicorbaadapter</Property>
        <Property name="JCA.Interaction.ImplClass.type" secret="false">bool_return</Property>
        <Property name="JCA.ConnectionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.corba.outbound.ESICorbaAdapterConnectionSpec</Property>
        <Property name="JCA.ConnectionSpec.Param1.name" secret="false">setServiceName</Property>
        <Property name="JCA.ConnectionSpec.Param1.type" secret="false">@INPUT</Property>
        <Property name="JCA.ConnectionSpec.Param1.value" secret="false">SERVICE_NAME</Property>
        <Property name="JCA.ConnectionSpec.Param2.name" secret="false">setNsConnectFlag</Property>
        <Property name="JCA.ConnectionSpec.Param2.type" secret="false">boolean</Property>
        <Property name="JCA.ConnectionSpec.Param2.value" secret="false">false</Property>
        <Property name="JCA.ConnectionSpec.Param3.name" secret="false">setNsHost</Property>
        <Property name="JCA.ConnectionSpec.Param3.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param3.value" secret="false">localhost</Property>
        <Property name="JCA.ConnectionSpec.Param4.name" secret="false">setNsPort</Property>
        <Property name="JCA.ConnectionSpec.Param4.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param4.value" secret="false">8002</Property>
        <Property name="JCA.ConnectionSpec.Param5.name" secret="false">setObjectName</Property>
        <Property name="JCA.ConnectionSpec.Param5.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param5.value" secret="false">samplemodule::sampleintarface</Property>
        <Property name="JCA.InteractionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.corba.outbound.ESICorbaAdapterInteractionSpec</Property>
        <Property name="JCA.InteractionSpec.Param1.name" secret="false">setOperationName</Property>
        <Property name="JCA.InteractionSpec.Param1.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param1.value" secret="false">op1</Property>
        <Property name="JCA.InteractionSpec.Param2.name" secret="false">setClientTimeOutFlag</Property>
        <Property name="JCA.InteractionSpec.Param2.type" secret="false">boolean</Property>
        <Property name="JCA.InteractionSpec.Param2.value" secret="false">false</Property>
        <Property name="JCA.InteractionSpec.Param3.name" secret="false">setClientTimeOut</Property>
        <Property name="JCA.InteractionSpec.Param3.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param3.value" secret="false">0</Property>
        <Property name="JCA.InteractionSpec.Param4.name" secret="false">setRequestType</Property>
        <Property name="JCA.InteractionSpec.Param4.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param4.value" secret="false">requestresponse</Property>
        <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.InputRecord.Map1.name" secret="false">com.fujitsu.esi.esimessage</Property>
        <Property name="JCA.InputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.OutputRecord.Map1.name" secret="false">com.fujitsu.esi.esimessage</Property>
```

```
        <Property name="JCA.OutputRecord.Map1.type" secret="false">payload</Property>
    </PropertyList>
</Service>
```

表C.22 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“CORBA”を指定します。	CORBA
Property		
JCA.ConnectionSpec.Param2.value	ネーミングサービスの参照先をサービスエンドポイントごとに個別に指定するか否かを指定します。 “true”または“false”を指定します。	false
JCA.ConnectionSpec.Param3.value	ネーミングサービスの参照先を指定します。 ネーミングサービスの参照先を個別に指定する場合だけ使用されます。	localhost
JCA.ConnectionSpec.Param4.value	ネーミングサービスの参照先ポート番号を指定します。 ネーミングサービスの参照先を個別に指定する場合だけ使用されます。 1～65535の範囲で半角数字だけ指定できます。範囲外の値を指定した場合は、誤動作する場合があります。	8002
JCA.ConnectionSpec.Param5.value	サーバアプリケーションのオブジェクト名を指定します。	samplemodule::sampleintarface
JCA.InteractionSpec.Param1.value	サーバアプリケーションのメソッド名を指定します。	op1
JCA.InteractionSpec.Param2.value	クライアントタイムアウト時間をサービスエンドポイントごとに個別に指定するか否かを指定します。 “true”または“false”を指定します。 サービスエンドポイントごとに細かくタイムアウトを決定する必要があるような特別な場合を除いては、“false”を選択してください。	false
JCA.InteractionSpec.Param3.value	クライアントタイムアウト時間(秒)を指定します。 0～100000000(秒)を指定することができます。0を指定する場合は、CORBAサーバアプリ	0

キー名	説明	指定例
	ケーションが復帰するまでの待機時間を監視しません。	
JCA.InteractionSpec.Param4.value	リクエストの方法を指定します。以下のどちらかを指定します。 ・ requestresponse ・ oneway 各リクエストの方法の詳細は“CORBAアダプタ利用時の定義”を参照してください。	requestresponse

Propertyについてはユーザが指定する項目について表に記載します。

この表で記載した以外の項目は必須ですのでそのままサービス情報に記載してください。

C.1.7 MQ連携の定義EE

WebSphere MQと連携したメッセージの送受信を行うためのサービスエンドポイント定義を作成します。

C.1.7.1 インバウンド

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.23 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="MQ-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="esi-service.xsd">
</Service>
```

表C.24 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“MQ-MSG”を指定します。	MQ-MSG

C.1.7.2 アウトバウンド(Oneway方式)

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.25 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="MQ-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
   <PropertyList>
      <Property name="JCA.Version" secret="false">1</Property>
      <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/esimqadapter</Property>
      <Property name="JCA.ConnectionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.mq.outbound.cci.ESIMQAdapterConnectionSpec</Property>
      <Property name="JCA.ConnectionSpec.Param0.name" secret="false">setMQAdapterName</Property>
      <Property name="JCA.ConnectionSpec.Param0.type" secret="false">String</Property>
      <Property name="JCA.ConnectionSpec.Param0.value" secret="false">ISIMQSnd_00001</Property>
      <Property name="JCA.Interaction.ImplClass.type" secret="false">record_return</Property>
      <Property name="JCA.InteractionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.mq.outbound.cci.ESIMQAdapterInteractionSpec</Property>
      <Property name="JCA.InteractionSpec.Param0.name" secret="false">setMQMDVersion</Property>
      <Property name="JCA.InteractionSpec.Param0.type" secret="false">int</Property>
      <Property name="JCA.InteractionSpec.Param0.value" secret="false">2</Property>
```

```
      <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
      <Property name="JCA.InputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
      <Property name="JCA.InputRecord.Map1.type" secret="false">payload</Property>
      <Property name="JCA.InputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
      <Property name="JCA.InputRecord.Map2.type" secret="false">headerSet</Property>
      <Property name="JCA.InputRecord.Map2.value" secret="false">*</Property>
      <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
      <Property name="JCA.OutputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
      <Property name="JCA.OutputRecord.Map1.type" secret="false">payload</Property>
      <Property name="JCA.OutputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
      <Property name="JCA.OutputRecord.Map2.type" secret="false">headerSet</Property>
      <Property name="JCA.OutputRecord.Map2.value" secret="false">*</Property>
   </PropertyList>
</Service>
```

表C.26 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“MQ-MSG”を指定します。	MQ-MSG
Property		
JCA.ConnectionSpec.Param0.value	ISIから呼び出すアウトバウンドのMQアダプタのワークユニット名を指定してください。	ISIMQSnd_00001
JCA.InteractionSpec.Param0.value	送信するWebSphere MQのメッセージの構造体のバージョン番号として、以下のどちらかを指定してください。 ・1:バージョン1で設定されます。 ・2:バージョン2で設定されます。	2

C.1.7.3 アウトバウンド(RequestResponse方式)

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <ServiceName>mqreqresp01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.27 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01

キー名	説明	指定例
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください	mqreqresp01

サービス情報

サービス情報:mqreqresp01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="mqreqresp01" type="MQ-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
   <Description>MQサービスエンドポイントのサンプルです。</Description>
   <PropertyList>
      <Property name="JCA.Version" secret="false">1</Property>
      <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/esimqadapter</Property>
      <Property name="JCA.Interaction.ImplClass.type" secret="false">record_return</Property>
      <Property name="JCA.InteractionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.mq.outbound.cci.ESIMQAdapterInteractionSpec</Property>
      <Property name="JCA.InteractionSpec.Param0.name" secret="false">setMQMDVersion</Property>
      <Property name="JCA.InteractionSpec.Param0.type" secret="false">int</Property>
      <Property name="JCA.InteractionSpec.Param0.value" secret="false">2</Property>
      <Property name="JCA.InteractionSpec.Param1.name" secret="false">setReqresp</Property>
      <Property name="JCA.InteractionSpec.Param1.type" secret="false">boolean</Property>
      <Property name="JCA.InteractionSpec.Param1.value" secret="false">true</Property>
      <Property name="JCA.InteractionSpec.Param2.name" secret="false">setUseInterval</Property>
      <Property name="JCA.InteractionSpec.Param2.type" secret="false">boolean</Property>
      <Property name="JCA.InteractionSpec.Param2.value" secret="false">true</Property>
      <Property name="JCA.InteractionSpec.Param3.name" secret="false">setInterval</Property>
      <Property name="JCA.InteractionSpec.Param3.type" secret="false">int</Property>
      <Property name="JCA.InteractionSpec.Param3.value" secret="false">1000</Property>
      <Property name="JCA.InteractionSpec.Param4.name" secret="false">setUseRecvkeyMsgId</Property>
      <Property name="JCA.InteractionSpec.Param4.type" secret="false">boolean</Property>
      <Property name="JCA.InteractionSpec.Param4.value" secret="false">true</Property>
      <Property name="JCA.ConnectionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.mq.outbound.cci.ESIMQAdapterConnectionSpec</Property>
      <Property name="JCA.ConnectionSpec.Param0.name" secret="false">setMQAdapterName</Property>
      <Property name="JCA.ConnectionSpec.Param0.type" secret="false">String</Property>
      <Property name="JCA.ConnectionSpec.Param0.value" secret="false">ISIMQReq_00001</Property>
      <Property name="JCA.ConnectionSpec.Param1.name" secret="false">setMQResponseAdapterName</Property>
      <Property name="JCA.ConnectionSpec.Param1.type" secret="false">String</Property>
      <Property name="JCA.ConnectionSpec.Param1.value" secret="false">ISIMQRes_00001</Property>
      <Property name="JCA.ConnectionSpec.Param2.name" secret="false">setServiceName</Property>
      <Property name="JCA.ConnectionSpec.Param2.type" secret="false">@INPUT</Property>
      <Property name="JCA.ConnectionSpec.Param2.value" secret="false">SERVICE_NAME</Property>
      <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
      <Property name="JCA.InputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
      <Property name="JCA.InputRecord.Map1.type" secret="false">payload</Property>
      <Property name="JCA.InputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
      <Property name="JCA.InputRecord.Map2.type" secret="false">headerSet</Property>
      <Property name="JCA.InputRecord.Map2.value" secret="false">*</Property>
      <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
      <Property name="JCA.OutputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
      <Property name="JCA.OutputRecord.Map1.type" secret="false">payload</Property>
      <Property name="JCA.OutputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
      <Property name="JCA.OutputRecord.Map2.type" secret="false">headerSet</Property>
      <Property name="JCA.OutputRecord.Map2.value" secret="false">*</Property>
   </PropertyList>
</Service>
```

表C.28 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	mqreqresp01
type	サービスのタイプ“MQ-MSG”を指定します。	MQ-MSG
Property		
JCA.ConnectionSpec.Param0.value	要求メッセージを送信するMQアダプタのワークユニット名を指定します。	ISIMQReq_00001
JCA.ConnectionSpec.Param1.value	応答メッセージを受信するMQアダプタのワークユニット名を指定します。	ISIMQRes_00001
JCA.InteractionSpec.Param0.value	送信するMQメッセージの構造体のバージョン番号を、以下のいずれかで指定します。 ・1：MQMD_VERSION_1で設定されます。 ・2：MQMD_VERSION_2で設定されます（デフォルト）。	2
JCA.InteractionSpec.Param1.value	RequestResponse方式を利用するための設定です。 必ず“true”を設定してください。	true
JCA.InteractionSpec.Param2.value	サービスエンドポイントで個別に応答メッセージの待ち時間を指定する場合“true”を指定します。 “false”を指定した場合、アウトバウンドのMQアダプタ(Response)設定ファイルの“recvTimeout”で指定された値が応答メッセージの待ち時間に使用されます。	true
JCA.InteractionSpec.Param3.value	応答メッセージの受信待ち時間をミリ秒単位で指定します。 指定可能な値は0～60000（1分）の整数です。0を指定した場合、応答メッセージの待ち合わせを実施しません。 省略した場合、1000（ミリ秒）が設定されます。	1000
JCA.InteractionSpec.Param4.value	MQメッセージの受信時に、検索キーとして追加でMsgIdを利用するかどうかを指定します。 デフォルトおよび省略時は“false”となります。 ・true：MsgIdを検索キーとして利用する。	true

キー名	説明	指定例
	・false：MsgIdを検索キーとして利用しない。	

上記の表に記載されていないパラメタについては、定義例の記載通りに設定してください。

C.1.8 MQD連携の定義 S EE

MQDを利用した非同期メッセージの送信/受信を行う場合、サービスエンドポイント定義に、MQD連携で必要な情報を定義します。

MQD連携の定義には、以下の種類があります。

- 受信
- 送信

C.1.8.1 受信

インバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

ポイント

MQD連携でメッセージ蓄積機能を利用するかどうか場合は、MQD受信環境定義ファイルに指定します。MQD受信環境定義ファイルについては、“ISI 導入ガイド”を参照してください。

指定例

サービスエンドポイント定義

サービスエンドポイント定義：endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>mqdrecvservice01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.29 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	mqdrecvservice01

サービス情報

サービス情報：mqdrecvservice01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="mqdrecvservice01" type="MQD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="esi-service.xsd">
</Service>
```

表C.30 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	mqdrecvservice01
type	サービスのタイプ“MQD”を指定します。	MQD

C.1.8.2 送信

アウトバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

注意

MQD連携でメッセージ蓄積機能を利用するかどうか場合は、MQD送信環境定義ファイルに指定します。MQD送信環境定義ファイルについては、“ISI 導入ガイド”を参照してください。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <ServiceName>mqdsendservice01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.31 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	mqdsendservice01

サービス情報

サービス情報:mqdsendservice01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="mqdsendservice01" type="MQD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="mqdsend.SystemName">MQDSYS01</Property>
        <Property name="mqdsend.QueueName">MQDQUE01</Property>
        <Property name="mqdsend.DivisionSize">32000</Property>
```

```
    </PropertyList>
</Service>
```

表C.32 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	mqdsendservice01
type	サービスのタイプ“MQD”を指定します。	MQD
Property		
mqdsend.SystemName	MQD送信先のMQDシステム名を指定します。	MQDSYS01
mqdsend.QueueName	MQD送信先のメッセージキュー名を指定します。	MQDQUE01
mqdsend.DivisionSize	データの分割サイズをバイト単位で指定します。“0”または省略した場合、データは分割されません。	32000

C.1.9 キュー間転送の定義EE

キュー間転送機能を利用した非同期メッセージの送信/受信を行う場合、サービスエンドポイント定義に、キュー間転送で必要な情報を定義します。

キュー間転送の定義には、以下の種類があります。

- 受信
- 送信
- ISI同士のキュー間転送

C.1.9.1 受信

インバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml(ペイロードを受信する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <ServiceName>quecntsndservice01</ServiceName>
        <QueueName>OutboundQueue01</QueueName>
    </Endpoint>
</EndpointList>
```

表C.33 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01

キー名	説明	指定例
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	quecntsndservice01
QueueName	キュー名を指定します。ISIシステム内のキュー名を指定してください。	OutboundQueue01

サービスエンドポイント定義:endpoint.xml(共通メッセージを受信する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <ServiceName>quecntsndservice02</ServiceName>
        <QueueName>OutboundQueue01</QueueName>
        <ApisideQueueName>InboundQueue02</ApisideQueueName>
    </Endpoint>
</EndpointList>
```

表C.34 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	quecntsndservice02
QueueName	キュー名を指定します。ISIシステム内のキュー名を指定してください。	OutboundQueue02
ApisideQueueName	キュー名を指定します。サービス提供側のアプリケーションにあるキュー名を指定してください。	InboundQueue02

注意

キュー間転送機能を利用して接続する2つのキュー名を統一してください。
キュー名を統一した場合は、統一したキュー名を、“QueueName”に指定してください。キュー名を統一できない場合は、ISIサーバ側にあるキュー名を“QueueName”に、サービス提供側のアプリケーションにあるキュー名を“ApisideQueueName”に指定してください。

サービス情報

サービス情報:quecntsndservice01.xml(ペイロードのサービス情報)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="quecntsndservice01" type="QUECONNECTED" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="quecnt.messageType">payload</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.35 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	quecntsndservice01
type	サービスのタイプ “QUECONNECTED”を指定します。	QUECONNECTED
Property		
quecnt.messageType	メッセージの種別“payload”を指定します。	payload
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

サービス情報:quecntsndservice02.xml(共通メッセージのサービス情報)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="quecntrcvservice02" type="QUECONNECTED" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="quecnt.messageType">common</Property>
        <Property name="quecnt.queueType">JMS-R</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
        <Property name="jmsr.wait.time">1000</Property>
    </PropertyList>
</Service>
```

表C.36 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	quecntsndservice02
type	サービスのタイプ “QUECONNECTED”を指定します。	QUECONNECTED
Property		
quecnt.messageType	メッセージの種別“common”を指定します。	common
quecnt.queueType	サービス利用側のアプリケーション、またはサービス提供側のアプリケーションにあるキューの種別を指定します。 以下のどちらかを指定してください。 ・ JMS-R(デフォルト) ・ JMS	JMS-R
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト)	true

キー名	説明	指定例
	・ false:蓄積しない	
jmsr.wait.time	キューにメッセージが格納されるまで待機する時間をミリ秒で指定します。 0～65535の範囲の整数を指定してください。 デフォルトは、“0”です。	1000

C.1.9.2 送信

アウトバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml(ペイロードを送信する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>quecntrcvservice01</ServiceName>
        <QueueName>InboundQueue01</QueueName>
    </Endpoint>
</EndpointList>
```

表C.37 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	quecntrcvservice01
QueueName	キュー名を指定します。ISIシステム内のキュー名を指定してください。	InboundQueue01

サービスエンドポイント定義:endpoint.xml(共通メッセージを送信する場合)

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>quecntrcvservice02</ServiceName>
        <QueueName>InboundQueue02</QueueName>
        <ApisideQueueName>OutboundQueue02</ApisideQueueName>
    </Endpoint>
</EndpointList>
```

表C.38 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	quecntrcvservice02
QueueName	キュー名を指定します。ISIシステム内のキュー名を指定してください。	InboundQueue02
ApisideQueueName	キュー名を指定します。サービス利用側のアプリケーションにあるキュー名を指定してください。	OutboundQueue02

注意

キュー間転送機能を利用して接続する2つのキュー名を統一してください。
キュー名を統一した場合は、統一したキュー名を、“QueueName”に指定してください。キュー名を統一できない場合は、ISIサーバ側にあるキュー名を“QueueName”に、サービス利用側のアプリケーションにあるキュー名を“ApisideQueueName”に指定してください。

サービス情報

サービス情報:quecntrcvservice01.xml(ペイロードのサービス情報)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="quecntrcvservice01" type="QUECONNECTED" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="quecnt.messageType">payload</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.39 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	quecntrcvservice01
type	サービスのタイプ“QUECONNECTED”を指定します。	QUECONNECTED
Property		
quecnt.messageType	メッセージの種別“payload”を指定します。	payload
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

サービス情報:quecntrcvservice02.xml(共通メッセージのサービス情報)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="quecntrcvservice02" type="QUECONNECTED" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="quecnt.messageType">common</Property>
        <Property name="quecnt.queueType">JMS-R</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.40 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	quecntrcvservice02
type	サービスのタイプ “QUECONNECTED”を指定します。	QUECONNECTED
Property		
quecnt.messageType	メッセージの種別“common”を指定します。	common
quecnt.queueType	サービス利用側のアプリケーション、またはサービス提供側のアプリケーションにあるキューの種別を指定します。 以下のどちらかを指定してください。 ・ JMS-R(デフォルト) ・ JMS	JMS-R
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

C.1.9.3 ISI同士のキュー間転送

ISIサーバ間のサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>quecntisiservice01</ServiceName>
        <QueueName>OutboundQueue01</QueueName>
        <ApisideQueueName>InboundQueue01</ApisideQueueName>
```

```
    </Endpoint>
</EndpointList>
```

表C.41 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
SequenceName	メッセージ送信先のISIサーバで動作させるシーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	quecntisiservice01
QueueName	キュー名を指定します。メッセージ送信元ISIサーバ内のキュー名を指定してください。	OutboundQueue01
ApisideQueueName	キュー名を指定します。メッセージ送信先ISIサーバ内のキュー名を指定してください。	InboundQueue01

注意

キュー間転送機能を利用して接続する2つのキュー名を統一してください。
キュー名を統一した場合は、統一したキュー名を、“QueueName”に指定してください。キュー名を統一できない場合は、メッセージ送信元ISIサーバ内のキュー名を“QueueName”に、メッセージ送信先ISIサーバ内のキュー名を“ApisideQueueName”に指定してください。

サービス情報

サービス情報:quecntisiservice01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="quecntisiservice01" type="QUECONNECTED" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="quecnt.isicnt">true</Property>
        <Property name="quecnt.msgAccumSndsystem">true</Property>
        <Property name="quecnt.msgAccumRcvsystem">true</Property>
    </PropertyList>
</Service>
```

表C.42 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	quecntisiservice01
type	サービスのタイプ“QUECONNECTED”を指定します。	QUECONNECTED
Property		
quecnt.isicnt	ISIサーバを連携させる“true”を指定します。	true

キー名	説明	指定例
quecnt.msgAccumSndsystem	メッセージ送信元のISIサーバに、メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true
quecnt.msgAccumRcvsystem	メッセージ送信先のISIサーバに、メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

C.1.10 データベース連携の定義

データベース連携を行うための定義を作成します。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.43 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプを指定します。 ・ true:同期送信 ・ false:非同期送信	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="DB" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <Description>zaiko_insert</Description>
    <PropertyList>
        <Property name="JCA.Version" secret="false">1</Property>
        <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/esidbadapter</Property>
        <Property name="JCA.ConnectionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.db.outbound.cci.ESIDBAdapterConnectionSpec</Property>
```

```
        <Property name="JCA.ConnectionSpec.Param1.name" secret="false">setDataSource</Property>
        <Property name="JCA.ConnectionSpec.Param1.type" secret="false">String</Property>
        <Property name="JCA.ConnectionSpec.Param1.value" secret="false">symfo_datasource</Property>
        <Property name="JCA.ConnectionSpec.Param2.name" secret="false">setServiceName</Property>
        <Property name="JCA.ConnectionSpec.Param2.type" secret="false">@INPUT</Property>
        <Property name="JCA.ConnectionSpec.Param2.value" secret="false">SERVICE_NAME</Property>
        <Property name="JCA.ConnectionSpec.Param3.name" secret="false">setLocalConnection</Property>
        <Property name="JCA.ConnectionSpec.Param3.type" secret="false">@INPUT</Property>
        <Property name="JCA.ConnectionSpec.Param3.value" secret="false">JMSR_DB_CONNECTION</Property>
        <Property name="JCA.InteractionSpec.ImplClass.name"
secret="false">com.fujitsu.esi.adapter.db.outbound.cci.ESIDBAdapterInteractionSpec</Property>
        <Property name="JCA.Interaction.ImplClass.type" secret="false">record_return</Property>
        <Property name="JCA.InteractionSpec.Param1.name" secret="false">setSql</Property>
        <Property name="JCA.InteractionSpec.Param1.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param1.value" secret="false">INSERT INTO 在庫管理.在庫表 VALUES(?P1?,?
P2?,?P3?,?P4?)</Property>
        <Property name="JCA.InteractionSpec.Param2.name" secret="false">setAccessType</Property>
        <Property name="JCA.InteractionSpec.Param2.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param2.value" secret="false">UPDATE</Property>
        <Property name="JCA.InputRecord.code" secret="false">UTF-8</Property>
        <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.OutputRecord.code" secret="false">UTF-8</Property>
        <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.OutputRecord.Map0.name" secret="false">com.fujitsu.esi.payload</Property>
        <Property name="JCA.OutputRecord.Map0.type" secret="false">payload</Property>
        <Property name="JCA.OutputRecord.Map0.value" secret="false"></Property>
        <Property name="JCA.InputRecord.Map0.name" secret="false">com.fujitsu.esi.payload</Property>
        <Property name="JCA.InputRecord.Map0.type" secret="false">payload</Property>
        <Property name="JCA.InputRecord.Map0.value" secret="false"></Property>
        <Property name="JCA.InputRecord.Map1.name" secret="false">P1</Property>
        <Property name="JCA.InputRecord.Map1.type" secret="false">db</Property>
        <Property name="JCA.InputRecord.Map1.value" secret="false">String</Property>
        <Property name="JCA.InputRecord.Map2.name" secret="false">P2</Property>
        <Property name="JCA.InputRecord.Map2.type" secret="false">db</Property>
        <Property name="JCA.InputRecord.Map2.value" secret="false">String</Property>
        <Property name="JCA.InputRecord.Map3.name" secret="false">P3</Property>
        <Property name="JCA.InputRecord.Map3.type" secret="false">db</Property>
        <Property name="JCA.InputRecord.Map3.value" secret="false">String</Property>
        <Property name="JCA.InputRecord.Map4.name" secret="false">P4</Property>
        <Property name="JCA.InputRecord.Map4.type" secret="false">db</Property>
        <Property name="JCA.InputRecord.Map4.value" secret="false">String</Property>
    </PropertyList>
</Service>
```

ユーザが指定するPropertyの項目について以下の定義項目の表に記載します。

この表で記載した以外の項目は、上記の定義をそのままサービス情報に記載してください。

表C.44 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“DB”を指定します。	DB
Property		
JCA.ConnectionSpec.Param1.value	データベース連携で使用するデータソース名を指定します。	symfo_datasource

キー名	説明	指定例
	JMS-RキューのDBと同一トランザクションで連携する場合は指定不要です。	
JCA.ConnectionSpec.Param3.name	JMS-RキューのDBと同一トランザクションで連携する場合に指定します。値は“setLocalConnection”で固定です。	setLocalConnection
JCA.ConnectionSpec.Param3.type	JMS-RキューのDBと同一トランザクションで連携する場合に指定します。値は“@INPUT”で固定です。	@INPUT
JCA.ConnectionSpec.Param3.value	JMS-RキューのDBと同一トランザクションで連携する場合に指定します。値は“JMSR_DB_CONNECTION”で固定です。	JMSR_DB_CONNECTION
JCA.InteractionSpec.Param1.value	実行するSQL文を指定します。	INSERT INTO 在庫管理.在庫表 VALUES(?P1?,?P2?,?P3?,?P4?)
JCA.InteractionSpec.Param2.value	実行するSQL種別を指定します。 ・ UPDATE：DBの更新時 ・ SELECT：DBの検索時 ・ STORED：復帰値がないストアドプロシージャ実行時 ・ STORED_SELECT：結果セットを返すストアドプロシージャ実行時 ・ STORED_FUNC：パラメタ復帰のストアドプロシージャ実行時	UPDATE
JCA.InputRecord.code	入力データ(DBアダプタに渡されたデータ)の文字コードを指定します。	UTF-8
JCA.InputRecord.MapN.name	SQL文に記述したSQL文への設定パラメタを指定します。パラメタの数だけ指定し、プロパティ名のMapNのNには自然数を指定します。	P1
JCA.InputRecord.MapN.type	SQL文に記述したSQL文への設定パラメタに設定する値により以下のように指定します。 ・ db:文字列を設定する場合 ・ binary:バイナリデータを設定する場合	db
JCA.InputRecord.MapN.value	SQL文に記述したSQL文への設定パラメタに設定する値により以下のように指定します。	String

キー名	説明	指定例
	・String:文字列を設定する場合 ・JCA.InputRecord.MapN.nameで設定したパラメタ名:バイナリデータを設定する場合	
以降はDBの検索時または復帰値があるストアドプロシージャ実行時に指定します。		
JCA.OutputRecord.code	出力データ(DBからの返却値)の文字コードを指定します。	UTF-8
JCA.OutputRecord.MapN.name	復帰値のパラメタ名を指定します。復帰値の数だけ指定します。	
JCA.OutputRecord.MapN.type	復帰値の値により以下のように指定します。 ・db:文字列の場合 ・binary:バイナリデータの場合	
JCA.OutputRecord.MapN.value	復帰値の値により以下のように指定します。 ・String:文字列の場合 ・JCA.OutputRecord.MapN.nameで設定したパラメタ名:バイナリデータを設定する場合 ・また、パラメタ復帰のストアドプロシージャ実行時は※を参照してください。	

※復帰値があるストアドプロシージャ実行時のJCA.OutputRecord.MapN.valueの値

VARCHAR	12
BIGINT	-5
BIT	-7
INTEGER	4
SMALLINT	5
TINYINT	-6
DOUBLE	8
FLOAT	6
REAL	7
NUMERIC	2
DECIMAL	3
DATE	91
TIME	92
TIMESTAMP	93
CHAR	1

LONGVARCHAR	-1
BINARY	-2
VARBINARY	-3
LONGVARBINARY	-4

C.1.11 III連携の定義

Interstage Information Integrator(以降、IIIと略します)と連携する場合、サービスエンドポイント定義に、IIIとの連携に必要な情報を定義します。

C.1.11.1 インバウンド

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.45 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="III" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="iii.processID" secret="false">iiiprocess1</Property>
    </PropertyList>
</Service>
```

表C.46 定義項目

キー名	説明	指定例
Service		

キー名	説明	指定例
name	サービス名を指定します。	service01
type	サービスのタイプ“III”を指定します。	III
Property		
iii.processID	IIIのプロセス定義のIDを指定します。	iiiprocess1

C.1.11.2 アウトバウンド

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <ServiceName>service01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.47 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	service01

サービス情報

サービス情報:service01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="service01" type="III" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
   <PropertyList>
        <Property name="JCA.Version" secret="false">1</Property>
        <Property name="JCA.JNDI.name" secret="false">java:comp/env/eis/isiiiiadapter</Property>
        <Property name="JCA.Interaction.ImplClass.type" secret="false">bool_return</Property>
        <Property name="JCA.InteractionSpec.ImplClass.name" secret="false">
com.fujitsu.ifl.adp.isi.outbound.cci.IFLInteractionSpec </Property>
        <Property name="JCA.InteractionSpec.Param1.name" secret="false">setProcessId</Property>
        <Property name="JCA.InteractionSpec.Param1.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param1.value" secret="false">iiiprocess1</Property>
        <Property name="JCA.InteractionSpec.Param2.name" secret="false">setDataBox</Property>
        <Property name="JCA.InteractionSpec.Param2.type" secret="false">String</Property>
        <Property name="JCA.InteractionSpec.Param2.value" secret="false">databox1</Property>
        <Property name="JCA.InputRecord.type" secret="false">Mapped</Property>
        <Property name="JCA.InputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
        <Property name="JCA.InputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.InputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
        <Property name="JCA.InputRecord.Map2.type" secret="false">headerSet</Property>
        <Property name="JCA.InputRecord.Map2.value" secret="false">com.fujitsu.esi</Property>
        <Property name="JCA.OutputRecord.type" secret="false">Mapped</Property>
```

```
        <Property name="JCA.OutputRecord.Map1.name" secret="false">com.fujitsu.esi.jca.payload</Property>
        <Property name="JCA.OutputRecord.Map1.type" secret="false">payload</Property>
        <Property name="JCA.OutputRecord.Map2.name" secret="false">com.fujitsu.esi.jca.header</Property>
        <Property name="JCA.OutputRecord.Map2.type" secret="false">headerSet</Property>
        <Property name="JCA.OutputRecord.Map2.value" secret="false">com.fujitsu.esi</Property>
    </PropertyList>
</Service>
```

この表で記載した以外の項目は、上記の定義をそのままサービス情報に記載してください。

表C.48 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	service01
type	サービスのタイプ“III”を指定します。	III
Property		
JCA.InteractionSpec.Param1.value	IIIのプロセス定義のIDを指定します。	iiiprocess1
JCA.InteractionSpec.Param2.value	IIIのデータボックスの名前を指定します。	databox1

C.1.12 互換機能の定義

C.1.12.1 メッセージ送受信APIの定義

メッセージ送受信APIを利用した転送を行う場合、サービスエンドポイント定義に、メッセージ送受信APIで必要な情報を定義します。
メッセージ送受信APIの定義には、以下の種類があります。

- 同期処理
- 非同期処理
  - 受信
  - 送信
- SOAPのRPC機能(JAX-RPC)の場合

同期処理

同期処理のサービスエンドポイント定義の指定例は、以下のとおりです(同期のサービスエンドポイントに対して、メッセージ送受信APIの送信APIを使用する時の定義です)。

指定例

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
    </Endpoint>
</EndpointList>
```

表C.49 定義項目

キー名	説明	指定例
Endpoint		

キー名	説明	指定例
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01

非同期処理

非同期処理のサービスエンドポイント定義には、以下の種類があります。

- 送信
- 受信

注意

JMSキュー名には、受信と送信で異なるものを指定してください。

送信

インバウンドサービスエンドポイント定義の指定例は、以下のとおりです(非同期のサービスエンドポイントに対して、メッセージ送受信APIの送信APIを使用する時の定義です)。

指定例

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <QueueName>ESIInboundQueue</QueueName>
    </Endpoint>
</EndpointList>
```

表C.50 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
QueueName	JMSキュー名を指定します。	ESIInboundQueue

受信

アウトバウンドサービスエンドポイント定義の指定例は、以下のとおりです(非同期のサービスエンドポイントに対して、メッセージ送受信APIの受信APIを使用する時の定義です)。

指定例

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint02" sync="false">
        <QueueName>ESIOutboundQueue</QueueName>
    </Endpoint>
</EndpointList>
```

表C.51 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint02
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
QueueName	JMSキュー名を指定します。	ESIOutboundQueue

SOAPのRPC機能(JAX-RPC)の場合

JAX-RPCによるSOAPのRPC機能を利用した転送を行う場合、サービスエンドポイント定義に、SOAP送信処理で必要な情報を定義します。
サービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

同期送信の場合のサービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>rpc01</ServiceName>
    </Endpoint>
</EndpointList>
```

非同期送信の場合のサービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <QueueName>ESIInboundQueue</QueueName>
        <ServiceName>rpc01</ServiceName>
    </Endpoint>
</EndpointList>
```

非同期受信の場合のサービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <QueueName>ESIOutboundQueue</QueueName>
        <ServiceName>rpc01</ServiceName>
```

```
    </Endpoint>
</EndpointList>
```

注意

非同期処理の場合、JMSキュー名には、受信と送信で異なるものを指定してください。

表C.52 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプを指定します。 ・ true:同期通信 ・ false:非同期通信	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
QueueName	syncで“false”を指定した場合に、JMSキュー名を指定します。	ESIInboundQueue
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	rpc01

各機能によるサービスエンドポイント定義の違いを以下に示します。

表C.53 定義項目の違い

	同期送信の場合	非同期送信の場合	非同期受信の場合
sync	true	false	false
<SequenceName>	○	○	－
<QueueName>	－	△	○
<ServiceName>	○	○	○

○:設定必須
△:設定可(省略時は、デフォルト値が有効となります。)
－:設定不要(設定は無効となります。)

サービス情報

サービス情報:rpc01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="rpc01" type="SOAP-RPC" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="soap.urlEndpoint">http://MyHost/esisoap/services/ESISoapRpcProvider</Property>
        <Property name="soap.basicAuthSet">true</Property>
        <Property name="soap.basicAuthUsr">userName</Property>
        <Property name="soap.basicAuthPwd" secret="true">userPassword</Property>
        <Property name="soap.proxySet">true</Property>
        <Property name="soap.nonProxyHosts">localhost|*.myhost.com</Property>
        <Property name="soap.proxyHost">proxy.proxyhost.com</Property>
        <Property name="soap.proxyPort">8080</Property>
        <Property name="soap.proxyAuthSet">true</Property>
```

```
        <Property name="soap.proxyAuthUsr">proxyUserName</Property>
        <Property name="soap.proxyAuthPwd" secret="true">proxyPassword</Property>
        <Property name="soap.timeout">60000</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.54 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	rpc01
type	“SOAP-RPC”を指定します。	SOAP-RPC
Property		
soap.urlEndpoint	ISIのJAX-RPC用のURLを指定します。	http://MyHost/esisoap/services/ESISoapRpcProvider
soap.basicAuthSet	Basic認証の設定を行うかどうかを指定します。 ・ true:行う “true”を指定した場合は、“soap.basicAuthUsr”でユーザ名を、“soap.basicAuthPwd”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true
soap.basicAuthUsr	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインユーザ名を指定します。	userName
soap.basicAuthPwd	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインパスワードを指定します。	userPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true
soap.proxySet	プロキシサーバを利用するかどうかを指定します。 ・ true:利用する “true”を指定した場合は、“soap.proxyHost”でホスト名を、“soap.proxyPort”でポート番号を指定してください。 ・ false:利用しない(デフォルト)	true
soap.nonProxyHosts	プロキシを経由せずに接続するホスト名を指定します。 複数のホスト名を指定する場合は、“|”(パイプ文字)で区切って指定してください。	localhost|*.myhost.com
soap.proxyHost	“soap.proxySet”に“true”を指定した場合、プロキシサーバのホスト名を指定します。	proxy.proxyhost.com
soap.proxyPort	“soap.proxySet”に“true”を指定した場合、プロキシサーバのポート番号を指定します。	8080
soap.proxyAuthSet	プロキシサーバの利用者認証の設定を行うかどうかを指定します。以下のどちらかを指定します。 ・ true:行う “true”を指定した場合は、“soap.proxyAuthUsr”でユーザ名を、	true

キー名	説明	指定例
	“soap.proxyAuthPwd”でパスワードを指定してください。 ・ false：行わない(デフォルト)	
soap.proxyAuthUsr	“soap.proxyAuthSet”に“true”を指定した場合、プロキシサーバのログインユーザ名を指定します。	proxyUserName
soap.proxyAuthPwd	“soap.proxyAuthSet”に“true”を指定した場合、プロキシサーバのログインパスワードを指定します。	proxyPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true：暗号化する ・ false：暗号化しない	true
soap.timeout	接続先から応答がない場合、接続を切断するタイムアウト時間をミリ秒で指定します。(※1) 以下の文字列を指定できます。 ・ 正の整数値 デフォルトは、利用するSOAPエンジンに依存します。 ・ Interstage Application Server：300000(5分) ・ AXIS：60000(1分)	60000
jmsr.messageAccumulate	JMS-Rキューを使用する場合、メッセージを蓄積するかどうかを指定します。 ・ true：蓄積する(デフォルト) ・ false：蓄積しない	true

※1：タイムアウト時間がISIサーバの処理時間より短い場合、接続タイムアウトが発生します。接続タイムアウトが発生すると、クライアント側でエラーになったり、データが消失したりする可能性があるため、タイムアウト時間は、ISIサーバの処理時間より長めに設定してください。

C.1.12.2 SOAPのメッセージング機能の定義

JAXMによるSOAPのメッセージング機能を利用した転送を行う場合、サービスエンドポイント定義に、SOAP送信処理で必要な情報を定義します。

非同期処理

アウトバウンドJAXM非同期のサービスエンドポイント定義の指定例は、以下のとおりです。

注意

- メッセージ送信呼出時の共通メッセージのペイロードに、SOAPメッセージ(XML文書)が格納されている必要があります。
- サービス情報でレスポンス処理を指定し、かつ、通信先がOneway方式のWebサービスでレスポンスにSOAPメッセージを含まない場合は、レスポンス処理で格納される共通メッセージのペイロードは空になります。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <QueueName>ESISendQueue</QueueName>
        <ServiceName>msg_mdb</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.55 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
QueueName	JMSキュー名を指定します。	ESISendQueue
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	msg_mdb

サービス情報

サービス情報:msg_mdb.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="msg_mdb" type="SOAP-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="reqrespParameter">usr.param1,usr.param2</Property>
        <Property name="responseSet">true</Property>
        <Property name="responseEndpoint">endpointRsp</Property>
        <Property name="soap.urlEndpoint">http://MyHost/esisoap/services/ESISoapRpcProvider</Property>
        <Property name="soap.basicAuthSet">true</Property>
        <Property name="soap.basicAuthUsr">userName</Property>
        <Property name="soap.basicAuthPwd" secret="true">userPassword</Property>
        <Property name="soap.proxySet">true</Property>
        <Property name="soap.nonProxyHosts">localhost|*.myhost.com</Property>
        <Property name="soap.proxyHost">proxy.proxyhost.com</Property>
        <Property name="soap.proxyPort">8080</Property>
        <Property name="soap.proxyAuthSet">true</Property>
        <Property name="soap.proxyAuthUsr">proxyUserName</Property>
        <Property name="soap.proxyAuthPwd" secret="true">proxyPassword</Property>
        <Property name="soap.timeout">60000</Property>
        <Property name="soap.action">callBytes</Property>
        <Property name="soap.parse">false</Property>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.56 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	msg_mdb
type	サービスのタイプ“SOAP-MSG”を指定します。	SOAP-MSG
Property		
reqrespParameter	レスポンス処理を実行する場合に、メッセージ送信された共通メッセージのヘッダ情報からコピーするパラメタ名を指定します。 複数のパラメタ名を指定する場合は、,(カンマ)で区切って指定してください。	usr.param1,usr.param2
responseSet	レスポンス処理を実行するかどうかを指定します。 ・ true:実行する ・ false:実行しない	true
responseEndpoint	レスポンスを格納するサービスエンドポイント定義名を指定します。	endpointRsp
soap.urlEndpoint	接続先のURLを指定します。	http://MyHost/esisoap/services/ESISoapRpcProvider
soap.basicAuthSet	Basic認証を行うかどうかを指定します。 ・ true:行う “true”を指定した場合は、“ soap.basicAuthUsr ”でユーザ名を、“soap.basicAuthPwd”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true
soap.basicAuthUsr	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインユーザ名を指定します。	username
soap.basicAuthPwd	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインパスワードを指定します。	userPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true
soap.proxySet	Proxyサーバを利用するかどうかを指定します。 ・ true:利用する “true”を指定した場合は、	true

キー名	説明	指定例
	“soap.proxyHost”でホスト名を、“soap.proxyPort”でポート番号を指定してください。 ・ false：利用しない(デフォルト)	
soap.nonProxyHosts	プロキシを経由せずに接続するホスト名を指定します。 複数のホスト名を指定する場合は、“|”(パイプ文字)で区切って指定してください。	localhost|*.myhost.com
soap.proxyHost	“soap.proxySet”に“true”を指定した場合、Proxyサーバのホスト名を指定します。	proxy.proxyhost.com
soap.proxyPort	“soap.proxySet”に“true”を指定した場合、Proxyサーバのポート番号を指定します。	8080
soap.proxyAuthSet	Proxyサーバで利用者認証を行うかどうかを指定します。以下のどちらかを指定します。 ・ true：行う “true”を指定した場合は、“soap.proxyAuthUsr”でユーザ名を、“soap.proxyAuthPwd”でパスワードを指定してください。 ・ false：行わない(デフォルト)	true
soap.proxyAuthUsr	“soap.proxyAuthSet”に“true”を指定した場合、Proxyサーバのログインユーザ名を指定します。	proxyUserName
soap.proxyAuthPwd	“soap.proxyAuthSet”に“true”を指定した場合、Proxyサーバのログインパスワードを指定します。	proxyPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true：暗号化する ・ false：暗号化しない	true
soap.timeout	接続のタイムアウト時間をミリ秒で指定します。 以下の文字列を指定できます。 ・ 正の整数値 デフォルトは、利用するSOAPエンジンに依存します。	60000

キー名	説明	指定例
	・ Interstage Application Server:300000(5分)	
soap.action	接続先のWebサービスが要求するSOAP Actionを指定します。 SOAP Actionが必要かどうかは接続先のWebサービスに確認してください。	callBytes
soap.parse	SOAPメッセージの妥当性検証を行うかどうかを指定します。 SOAP1.1およびSOAP1.2のスキーマ定義でリクエストとレスポンスのSOAPメッセージの妥当性検証を行います。 ・ true:確認する ・ false:確認しない(デフォルト)	false
jmsr.messageAccumulate	JMS-Rキューを使用する場合、メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true

同期処理

同期処理のサービスエンドポイント定義の指定例は、以下のとおりです。

注意

- SOAP Gateway呼出時の共通メッセージのペイロードに、SOAPメッセージ(XML文書)が格納されている必要があります。
- 通信先からのレスポンスのSOAPメッセージがSOAPFaultの場合も、SOAPメッセージを共通メッセージのペイロードに格納し、正常に終了します。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
    <Endpoint name="outbound02" sync="true">
        <ServiceName>msg_gw</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.57 定義項目

キー名	説明	指定例
Endpoint		

キー名	説明	指定例
name	サービスエンドポイント名を指定します。	outbound02
sync	サービスエンドポイントの同期タイプ“true”(同期通信)を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	msg_gw

サービス情報

サービス情報:msg_gw.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="msg_gw" type="SOAP-MSG" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="soap.urlEndpoint">http://MyHost/esisoap/services/ESISoapRpcProvider</Property>
        <Property name="soap.basicAuthSet">true</Property>
        <Property name="soap.basicAuthUsr">userName</Property>
        <Property name="soap.basicAuthPwd" secret="true">userPassword</Property>
        <Property name="soap.proxySet">true</Property>
        <Property name="soap.nonProxyHosts">localhost|*.myhost.com</Property>
        <Property name="soap.proxyHost">proxy.proxyhost.com</Property>
        <Property name="soap.proxyPort">8080</Property>
        <Property name="soap.proxyAuthSet">true</Property>
        <Property name="soap.proxyAuthUsr">proxyUserName</Property>
        <Property name="soap.proxyAuthPwd" secret="true">proxyPassword</Property>
        <Property name="soap.timeout">60000</Property>
        <Property name="soap.parse">false</Property>
    </PropertyList>
</Service>
```

表C.58 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	msg_gw
type	サービスのタイプ“SOAP-MSG”を指定します。	SOAP-MSG
Property		
soap.urlEndpoint	接続先のURLを指定します。	http://MyHost/esisoap/services/ESISoapRpcProvider
soap.basicAuthSet	Basic認証を行うかどうかを指定します。 ・ true:行う “true”を指定した場合は、“ soap.basicAuthUsr ”でユーザ名を、“soap.basicAuthPwd”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true

キー名	説明	指定例
soap.basicAuthUsr	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインユーザ名を指定します。	username
soap.basicAuthPwd	“soap.basicAuthSet”に“true”を指定した場合、Basic認証のログインパスワードを指定します。	userPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true
soap.proxySet	Proxyサーバを利用するかどうかを指定します。 ・ true:利用する “true”を指定した場合は、“soap.proxyHost”でホスト名を、“soap.proxyPort”でポート番号を指定してください。 ・ false:利用しない(デフォルト)	true
soap.nonProxyHosts	プロキシを経由せずに接続するホスト名を指定します。 複数のホスト名を指定する場合は、|(パイプ文字)で区切って指定してください。	localhost|*.myhost.com
soap.proxyHost	“soap.proxySet”に“true”を指定した場合、Proxyサーバのホスト名を指定します。	proxy.proxyhost.com
soap.proxyPort	“soap.proxySet”に“true”を指定した場合、Proxyサーバのポート番号を指定します。	8080
soap.proxyAuthSet	Proxyサーバで利用者認証を行うかどうかを指定します。以下のどちらかを指定します。 ・ true:行う “true”を指定した場合は、“ soap.proxyAuthUsr ”でユーザ名を、“ soap.proxyAuthPwd ”でパスワードを指定してください。 ・ false:行わない(デフォルト)	true
soap.proxyAuthUsr	“soap.proxyAuthSet”に“true”を指定した場合、Proxyサーバ	proxyUserName

キー名	説明	指定例
	のログインユーザ名を指定します。	
soap.proxyAuthPwd	“soap.proxyAuthSet”に“true”を指定した場合、Proxyサーバのログインパスワードを指定します。	proxyPassword
secret	プロパティの情報(パスワードなど)を暗号化するかどうかを指定します。 ・ true:暗号化する ・ false:暗号化しない	true
soap.timeout	接続のタイムアウト時間をミリ秒で指定します。 以下の文字列を指定できます。 ・ 正の整数値 デフォルトは、利用するSOAPエンジンに依存します。 ・ Interstage Application Server:300000(5分)	60000
soap.parse	SOAPメッセージの妥当性検証を行うかどうかを指定します。 SOAP1.1およびSOAP1.2のスキーマ定義でリクエストとレスポンスのSOAPメッセージの妥当性検証を行います。 ・ true:確認する ・ false:確認しない(デフォルト)	false

C.1.12.3 SOAP(.NET)の定義

SOAPのRPC機能を利用し、.NETから呼び出しを行うためのサービスエンドポイント定義を作成します。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpoint.xsd">
    <Endpoint name="endpoint01" sync="true">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>soap01</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.59 定義項目

キー名	説明	指定例
Endpoint		

キー名	説明	指定例
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプを指定します。 ・ true:同期通信 ・ false:非同期通信	true
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	soap01

サービス情報

サービス情報:soap01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="soap01" type="SOAP-NET" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="soap.urlEndpoint">http://MyHost/esilightservice/services/ESIServerLightProvider</Property>
    </PropertyList>
</Service>
```

表C.60 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	soap01
type	“SOAP-NET”を指定します。	SOAP-NET
Property		
soap.urlEndpoint	ISIのSOAP-NET用のURLを指定します。	http://MyHost/esilightservice/ services/ESIServerLightProvider

上記の他に接続タイムアウト時間、Basic認証、Proxy設定に関するプロパティが設定可能です。
接続タイムアウト時間、Basic認証、Proxy設定に関するプロパティについては、“SOAPのRPC機能(JAX-RPC)の場合”を参照してください。

C.1.12.4 JMS-R連携の定義EE

JMS-Rを利用した非同期メッセージの送信/受信行う場合、サービスエンドポイント定義に、JMS-R連携で必要な情報を定義します。

JMS-R連携の定義には、以下の種類があります。

- 受信
- 送信

受信

インバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <ServiceName>jmsrsendservice01</ServiceName>
        <QueueName>jmsrqueue01</QueueName>
    </Endpoint>
</EndpointList>
```

表C.61 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	jmsrsendservice01
QueueName	JMS-Rのキュー名を指定します。	jmsrqueue01

サービス情報

サービス情報:jmsrsendservice01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="jmsrsendservice01" type="JMS-R" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="jmsr.messageAccumulate">true</Property>
        <Property name="jmsr.wait.time">1000</Property>
    </PropertyList>
</Service>
```

表C.62 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	jmsrsendservice01
type	サービスのタイプ“JMS-R”を指定します。	JMS-R
Property		
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト) ・ false:蓄積しない	true
jmsr.wait.time	キューにメッセージが格納されるまで待機する時間をミリ秒で指定します。 0～65535の範囲の整数を指定してください。 デフォルトは、“0”です。	1000

送信

アウトバウンドサービスエンドポイント定義の指定例は、以下のとおりです。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="endpoint01" sync="false">
        <SequenceName>sequence01</SequenceName>
        <ServiceName>jmsrrecvservice01</ServiceName>
        <QueueName>jmsrqueue01</QueueName>
    </Endpoint>
</EndpointList>
```

表C.63 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	endpoint01
sync	サービスエンドポイントの同期タイプ“false”(非同期通信)を指定します。	false
SequenceName	シーケンス名を指定します。シーケンス定義に指定したシーケンス名を指定してください。	sequence01
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	jmsrrecvservice01
QueueName	JMS-Rのキュー名を指定します。	jmsrqueue01

サービス情報

サービス情報:jmsrrecvservice01.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="jmsrrecvservice01" type="JMS-R" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="jmsr.messageAccumulate">true</Property>
    </PropertyList>
</Service>
```

表C.64 定義項目

キー名	説明	指定例
Service		
name	サービス名を指定します。	jmsrrecvservice01
type	サービスのタイプ“JMS-R”を指定します。	JMS-R
Property		
jmsr.messageAccumulate	メッセージを蓄積するかどうかを指定します。 ・ true:蓄積する(デフォルト)	true

キー名	説明	指定例
	・ false:蓄積しない	

C.1.12.5 JCA連携の定義

JCAを利用した連携を行うためのサービスエンドポイント定義を作成します。

指定例

サービスエンドポイント定義

サービスエンドポイント定義:endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="jca_endpoint" sync="true">
        <ServiceName>jca001</ServiceName>
    </Endpoint>
</EndpointList>
```

表C.65 定義項目

キー名	説明	指定例
Endpoint		
name	サービスエンドポイント名を指定します。	jca_endpoint
sync	サービスエンドポイントの同期タイプ“true”を指定します。	true
ServiceName	サービス名を指定します。サービス情報に指定したサービス名を指定してください。	jca001

サービス情報

サービス情報:jca001.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<Service name="jca001" type="JCA" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-service.xsd">
    <PropertyList>
        <Property name="JCA.JNDI.name">java:comp/env/eis/EISVendorConnectionFactory</Property>
        <Property name="JCA.ConnectionSpec.ImplClass.name">com.vendor.eis.cci.EISVendorConnectionSpec</Property>
        <Property name="JCA.ConnectionSpec.Param0.name">setAdapterName</Property>
        <Property name="JCA.ConnectionSpec.Param0.type">java.lang.String</Property>
        <Property name="JCA.ConnectionSpec.Param0.value">RDBMS</Property>
        <Property name="JCA.ConnectionSpec.Param1.name">setConfig</Property>
        <Property name="JCA.ConnectionSpec.Param1.type">java.lang.String</Property>
        <Property name="JCA.ConnectionSpec.Param1.value">jca_config</Property>
        <Property name="JCA.ConnectionSpec.Param2.name">setUserName</Property>
        <Property name="JCA.ConnectionSpec.Param2.type">java.lang.String</Property>
        <Property name="JCA.ConnectionSpec.Param2.value">scott</Property>
        <Property name="JCA.ConnectionSpec.Param3.name">setPassword</Property>
        <Property name="JCA.ConnectionSpec.Param3.type">java.lang.String</Property>
        <Property name="JCA.ConnectionSpec.Param3.value">tiger</Property>
        <Property name="JCA.InteractionSpec.ImplClass.name">com.vendor.eis.cci.EISVendorInteractionSpec</Property>
        <Property name="JCA.InteractionSpec.Param0.name">setFunctionName</Property>
        <Property name="JCA.InteractionSpec.Param0.type">java.lang.String</Property>
        <Property name="JCA.InteractionSpec.Param0.value">PROCESS</Property>
        <Property name="JCA.Interaction.ImplClass.type">record_return</Property>
        <Property name="JCA.InputRecord.type">Indexed</Property>
        <Property name="JCA.InputRecord.List.name">input</Property>
```

```
        <Property name="JCA.InputRecord.List0.type">payload</Property>
        <Property name="JCA.OutputRecord.type">Indexed</Property>
        <Property name="JCA.OutputRecord.List0.type">payload</Property>
    </PropertyList>
</Service>
```

表C.66 定義項目

キー名		説明	指定例
Service			
name		サービス名を指定します。	jca001
type		“JCA”を指定します。	JCA
Property			
JCA.JNDI.name		Interstage Application Serverに配備したリソースアダプタのJNDI名を指定します。	java:comp/env/eis/EISVendorConnectionFactory
JCA.ConnectionSpec.ImplClass.name		リソースアダプタが提供しているConnectionSpecの実装クラス名を指定します。	com.vendor.eis.cci.EISVendorConnectionSpec
1	JCA.ConnectionSpec.Param0.name	ConnectionSpecが提供しているメソッド名を指定します。	setAdapterName
	JCA.ConnectionSpec.Param0.type	メソッドに渡す値の型を指定します。	java.lang.String
	JCA.ConnectionSpec.Param0.value	メソッドに渡す値を指定します。	RDBMS
2	JCA.ConnectionSpec.Param1.name	ConnectionSpecが提供しているメソッド名を指定します。	setConfig
	JCA.ConnectionSpec.Param1.type	メソッドに渡す値の型を指定します。	java.lang.String
	JCA.ConnectionSpec.Param1.value	メソッドに渡す値を指定します。	jca_config
3	JCA.ConnectionSpec.Param2.name	ConnectionSpecが提供しているメソッド名を指定します。	setUserName
	JCA.ConnectionSpec.Param2.type	メソッドに渡す値の型を指定します。	java.lang.String
	JCA.ConnectionSpec.Param2.value	メソッドに渡す値を指定します。	scott
4	JCA.ConnectionSpec.Param3.name	ConnectionSpecが提供しているメソッド名を指定します。	setPassword
	JCA.ConnectionSpec.Param3.type	メソッドに渡す値の型を指定します。	java.lang.String
	JCA.ConnectionSpec.Param3.value	メソッドに渡す値を指定します。	tiger
JCA.InteractionSpec.ImplClass.name		リソースアダプタが提供しているInteractionSpecの実装クラス名を指定します。	com.vendor.eis.cci.EISVendorInteractionSpec
1	JCA.InteractionSpec.Param0.name	InteractionSpecが提供しているメソッド名を指定します。	setFunctionName
	JCA.InteractionSpec.Param0.type	メソッドに渡す値の型を指定します。	java.lang.String
	JCA.InteractionSpec.Param0.value	メソッドに渡す値を指定します。	PROCESS

キー名	説明	指定例
JCA.Interaction.ImplClass.type	Interactionクラスのexecuteメソッドのタイプを指定します。 以下のどちらかを指定してください。 ・ Recordが復帰する場合 record_return ・ booleanが復帰する場合 bool_return	record_return
JCA.InputRecord.type	入力レコードの型“Indexed”(固定)を指定します。	Indexed
JCA.InputRecord.List.name	IndexedRecordのリストの名前“input”(固定)を指定します。	input
JCA.InputRecord.List0.type	リストの最初のインデックスに渡すデータ“payload”(固定)を指定します。	payload
JCA.OutputRecord.type	入力レコードの型“Indexed”(固定)を指定します。	Indexed
JCA.OutputRecord.List0.type	リストの最初のインデックスから取得するデータ“payload”(固定)を指定します。	payload

C.2 シーケンス定義

シーケンス定義の構成と、実行ステップ情報に定義するペイロード操作について説明します。

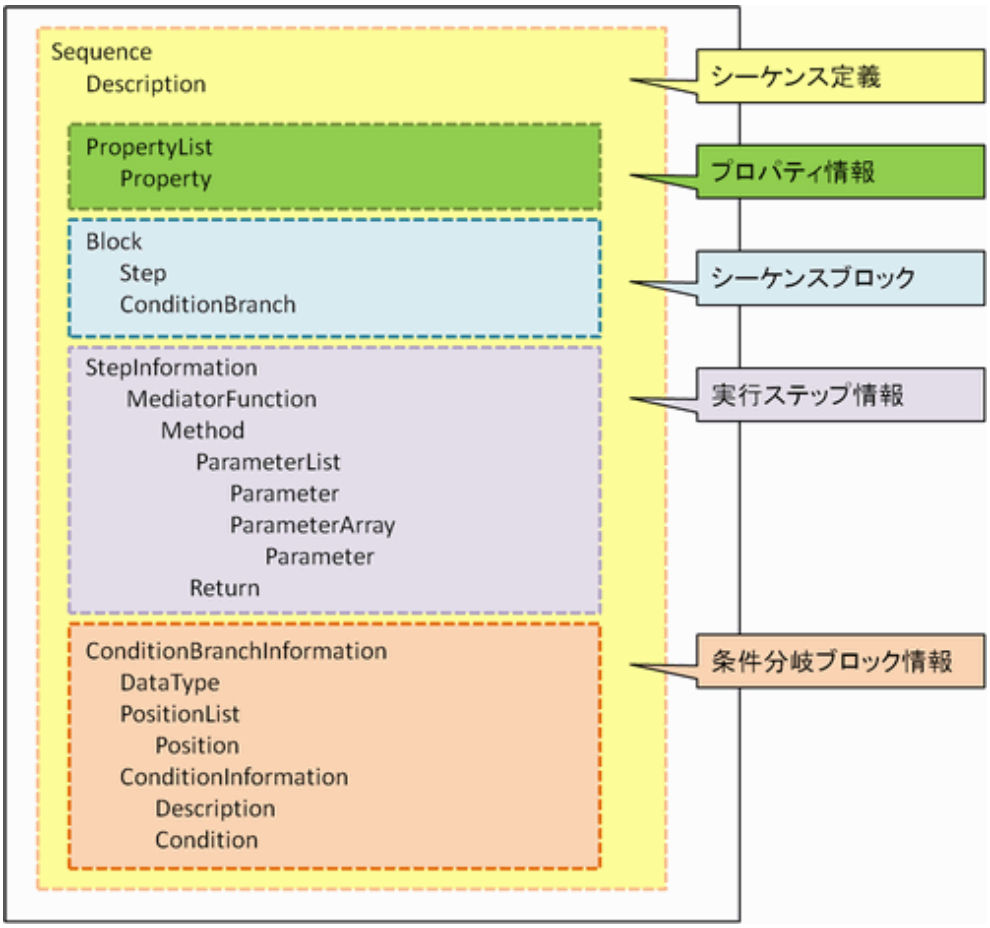
C.2.1 シーケンス定義の構成

シーケンス定義の構成と指定例は、以下のとおりです。
なお、定義する項目についての詳細は、定義する情報に応じて以下を参照してください。

- シーケンス定義:“シーケンス定義の詳細”
- プロパティ情報:“プロパティ情報の詳細”
- シーケンスブロック:“シーケンスブロックの詳細”
- 実行ステップ情報:“実行ステップ情報の詳細”
- 条件分岐ブロック情報:“条件分岐ブロック情報の詳細”

構成

シーケンス定義の構成を以下に示します。



指定例

```
<?xml version="1.0" encoding="UTF-8" ?>
<Sequence name="sequence01" logging="false" history="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="esi-sequence.xsd">
    <PropertyList>
        <Property name="com.fujitsu.esi.formatmanager.transSeqID">GYOUMU01</Property>
        <Property name="com.fujitsu.esi.endpoint.outbound2">endpointB</Property>
    </PropertyList>
    <Block>
        <Step name="checkXML" cancel="false" history="false"></Step>
        <ConditionBranch name="ConditionBranchBlock1">
            <Condition name="ConditionBranch1">
                <Step name="formatConversion1" cancel="false" history="false"></Step>
            </Condition>
            <Condition name="ConditionBranch2">
                <Step name="formatConversion2" cancel="false" history="false"></Step>
            </Condition>
            <DefaultCondition>
            </DefaultCondition>
        </ConditionBranch>
        <Step name="SyncServiceCall1" cancel="false" history="false"></Step>
    </Block>
    <StepInformation stepname="checkXML" monitoring="false">
        <MediatorFunction name="XMLValidation">
            <Method type="execute" name="checkXML">
                <ParameterList>
                  <Parameter name="esi.XmlValidation.ESIMessageID" valuetype="esi.header">com.fujitsu.esi.message.id</
Parameter>
                   <Parameter name="esi.XmlValidation.in" valuetype="esi.payload">@current</Parameter>
```

```
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
    <StepInformation stepname="formatConversion1" monitoring="false">
        <MediatorFunction name="Formatmanager">
            <Method type="execute" name="convertbyTransformationSequenceID">
                <ParameterList>
                  <Parameter name="esi.FormatManager.esiMessageID" valuetype="esi.header">com.fujitsu.esi.message.id</
Parameter>
                    <Parameter name="esi.FormatManager.transSeqID" valuetype="direct">GYOUMU01_CNV01</Parameter>
                    <Parameter name="esi.FormatManager.inputData" valuetype="esi.payload">@current</Parameter>
                </ParameterList>
                <Return valuetype="esi.payload"></Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
    <StepInformation stepname="formatConversion2" monitoring="false">
        <MediatorFunction name="Formatmanager">
            <Method type="execute" name="convertbyTransformationSequenceID">
                <ParameterList>
                  <Parameter name="esi.FormatManager.esiMessageID" valuetype="esi.header">com.fujitsu.esi.message.id</
Parameter>
                    <Parameter name="esi.FormatManager.transSeqID" valuetype="direct">GYOUMU01_CONV2</Parameter>
                    <Parameter name="esi.FormatManager.inputData" valuetype="esi.payload">@current</Parameter>
                </ParameterList>
                <Return valuetype="esi.payload"></Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
    <StepInformation stepname="SyncServiceCall1" monitoring="false">
        <MediatorFunction name="SyncServiceCall">
            <Method type="execute" name="send">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <Parameter name="endpoint" valuetype="direct">mq_snd_out</Parameter>
                    <Parameter name="use_fault_message">false</Parameter>
                </ParameterList>
                <Return valuetype="esi.header">com.fujitsu.esi.endpoint.outbound</Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
    <ConditionBranchInformation name="ConditionBranchBlock1">
        <DataType>xml</DataType>
        <PositionList>
            <Position name="point">/info/user/@point</Position>
        </PositionList>
        <ConditionInformation name="ConditionBranch1">
            <Description>ゴールド会員の場合</Description>
            <Condition>?point? &gt;= 10000</Condition>
        </ConditionInformation>
        <ConditionInformation name="ConditionBranch2">
            <Description>シルバー会員の場合</Description>
            <Condition>?point? &lt; 10000 AND ?point? &gt;= 5000</Condition>
        </ConditionInformation>
    </ConditionBranchInformation>
</Sequence>
```

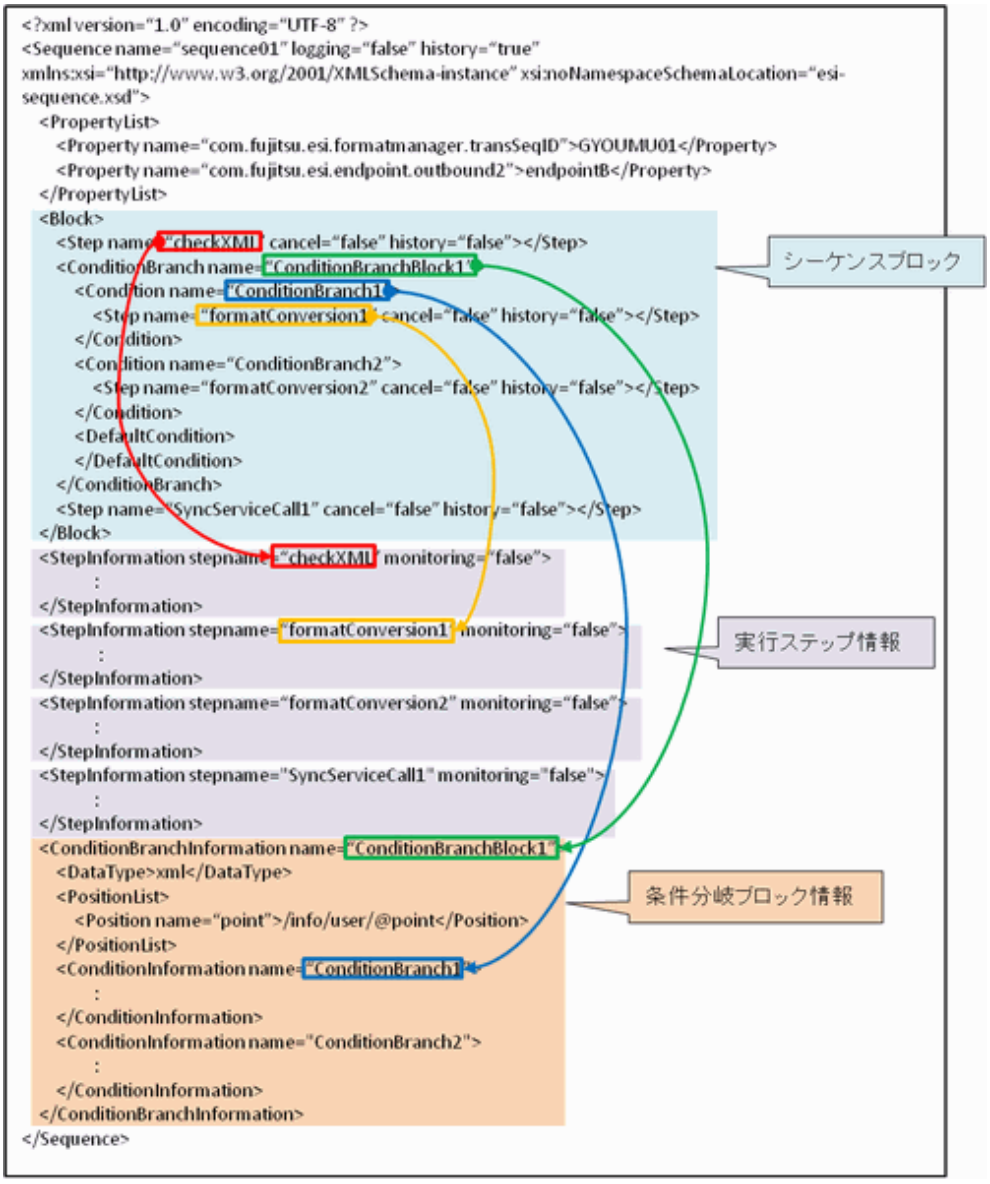
C.2.1.1 プロパティ情報

シーケンスで利用するプロパティを定義します。共通メッセージのヘッダパラメタに独自のプロパティを追加したいときに、プロパティ名(キー名)を指定します。また定数(ダイレクト値)も指定できます。指定したプロパティ情報は、シーケンス上を流れる共通メッセージのヘッダパラメタに設定されます。

C.2.1.2 シーケンスブロック

シーケンスブロックには、メディエータファンクションおよび条件分岐の呼出し順を定義します。エラー時のキャンセル処理の有無も定義します。

呼出し順は、実行ステップ情報のステップ名、条件分岐ブロック名および分岐経路名で指定します。シーケンスブロックと実行ステップ情報、条件分岐ブロックおよび分岐経路の対応関係を以下に示します。



C.2.2 サービス呼出しの定義

サービス呼出しの定義について説明します。

C.2.2.1 同期サービス呼出し

同期でサービスを呼び出す場合、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SyncServiceCall)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SyncServiceCallの呼出し情報(メソッド(send)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、サービスエンドポイント"endpoint01"で定義されているサービス呼出しを行います。

```
:
    <StepInformation stepname="SyncServiceCall1" monitoring="false">
        <MediatorFunction name="SyncServiceCall">
            <Method type="execute" name="send">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <Parameter name="endpoint" valuetype="direct">endpoint01</Parameter>
                    <Parameter name="use_fault_message">false</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.67 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SyncServiceCall1
MediatorFunction			
name		メディエータファンクション名“SyncServiceCall”を指定します。	SyncServiceCall
Method			
type		実行種別“execute”を指定します。	execute
name		SyncServiceCallのメソッド名“send”を指定します。	send
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
2	サービスエンドポイント名を指定します。		
	name	“endpoint”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	サービスエンドポイント名を指定します。	endpoint01
3	呼出し先サービスからのエラーメッセージをエラーシーケンスに通知するかどうかを指定します。		
	name	“use_fault_message”を指定します。	use_fault_message
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	trueまたはfalseを指定します。	false

C.2.2.2 ルーティングして同期サービス呼出し(ルーティング定義を利用しない)

ルーティング機能を使用して、条件に合った同期サービス呼出しを行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SyncServiceCallWithRouting)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SyncServiceCallWithRoutingの呼出し情報(メソッド(sendWithConditionRouting)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、“$price == 1000”の条件に合えばサービスエンドポイント“endpointA”に転送し、“$company @EQUAL@ "Fujitsu"”の条件に合えばサービスエンドポイント“endpointB”に転送し、その他の場合には、“endpointC”に転送しています。

```
:
    <StepInformation stepname="SyncServiceCallWithRouting1">
        <MediatorFunction name="SyncServiceCallWithRouting">
            <Method type="execute" name="sendWithConditionRouting">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <ParameterArray name="condition_list">
                        <Parameter valuetype="direct">$price == 1000</Parameter>
                        <Parameter valuetype="direct">$company @EQUAL@ "Fujitsu"</Parameter>
                        <Parameter valuetype="direct">*</Parameter>
                    </ParameterArray>
                    <ParameterArray name="endpoint_list">
                        <Parameter valuetype="direct">endpointA</Parameter>
                        <Parameter valuetype="direct">endpointB</Parameter>
                        <Parameter valuetype="direct">endpointC</Parameter>
                    </ParameterArray>
                    <Parameter name="use_fault_message">false</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.68 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SyncServiceCallWithRouting1
MediatorFunction			
name		メディエータファンクション名 “SyncServiceCallWithRouting”を指定します。	SyncServiceCallWithRouting
Method			
type		実行種別“execute”を指定します。	execute
name		SyncServiceCallWithRoutingのメソッド名 “sendWithConditionRouting”を指定します。	sendWithConditionRouting
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
ParameterArray			
name		“condition_list”を指定します。条件リストの配列です。	condition_list
1	Parameter		

キー名		説明	指定例
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	$price == 1000
2	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	$company @EQUAL@ "Fujitsu"
3	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	*
ParameterArray			
name		“endpoint_list”を指定します。サービスエンドポイントリストの配列です。	endpoint_list
1	Parameter		
	条件式1のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointA
2	Parameter		
	条件式2のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointB
3	Parameter		
	条件式3のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointC
Parameter			
1	呼出し先サービスからのエラーメッセージをエラーシーケンスに通知するかどうかを指定します。		
	name	“use_fault_message”を指定します。	use_fault_message
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	trueまたはfalseを指定します。	false

条件式

ここでは、条件式で使用できる演算子と文法について説明します。

演算子

条件式で使用できる演算子を以下に示します。

表C.69 演算子の一覧

演算子	説明
>	数値比較を行います。左辺の値が右辺の値よりも大きい場合に真となります。
<	数値比較を行います。左辺の値が右辺の値よりも小さい場合に真となります。
>=	数値比較を行います。左辺の値が右辺の値以上の場合に真となります。
<=	数値比較を行います。左辺の値が右辺の値以下の場合に真となります。
==	数値比較を行います。左辺の値が右辺の値と等しい場合に真となります。
!=	数値比較を行います。左辺の値が右辺の値と異なる場合に真となります。
@EQUAL@	文字列比較を行います。左辺の文字列が右辺の文字列と等しい場合に真となります。
@NOTEQUAL@	文字列比較を行います。左辺の文字列が右辺の文字列と異なる場合に真となります。

条件式の文法

条件式には、以下の文法があります。

- 文字列を表す場合は、“ " ”(ダブルクォーテーション)でくくります。
  “ " ”がない場合は、数値とみなされます。
- 条件式に括弧は使用できません。
- 論理演算子は、ANDだけ使用でき、“&&”と記述します。
- 共通メッセージのヘッダパラメタを指定する場合は、ヘッダパラメタキー名の先頭に“$”を付けます。
- 比較対象値、比較演算子、および論理演算子の間には、デリミタとして必ず半角スペースを入れます。
- 条件式に“*”(半角のアスタリスク)だけを指定すると、無条件に判定結果が真となります。

注意

XML文書中には直接“ " ”、“&”、“<”および“>”を記述することができません。条件式でこれらの文字を使用する場合は、CDATAセクションまたは実体参照を使用して記述してください。

$price >= 1000を定義する場合

- CDATAセクションを使用する場合の定義例
  <![CDATA[$price >= 1000]]>
- 実体参照を使用する場合の定義例
  $price >= 1000

C.2.2.3 ルーティングして同期サービス呼出し(ルーティング定義を利用)

ルーティング機能を使用して、条件に合った同期サービス呼出しを行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SyncServiceCallWithRouting)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SyncServiceCallWithRoutingの呼出し情報(メソッド(sendWithDefinitionRouting)やパラメタ)を定義します。
- パラメタにOUTBOUND用のルーティング定義名を指定します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、OUTBOUND用のルーティング定義“routingA”にメッセージを送信しています。

```
:
    <StepInformation stepname="SyncServiceCallWithRouting1">
        <MediatorFunction name="SyncServiceCallWithRouting">
```

```
            <Method type="execute" name="sendWithDefinitionRouting">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <Parameter name="routing">routingA</Parameter>
                    <Parameter name="use_fault_message">false</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.70 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SyncServiceCallWithRouting1
MediatorFunction			
name		メディエータファンクション名“SyncServiceCallWithRouting”を指定します。	SyncServiceCallWithRouting
Method			
type		実行種別“execute”を指定します。	execute
name		SyncServiceCallWithRoutingのメソッド名“sendWithDefinitionRouting”を指定します。	sendWithDefinitionRouting
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
2	ルーティング定義名を指定します。		
	name	“routing”を指定します。	routing
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	ルーティング定義名を指定します。	routingA
3	呼出し先サービスからのエラーメッセージをエラーシーケンスに通知するかどうかを指定します。		
	name	“use_fault_message”を指定します。	use_fault_message
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	trueまたはfalseを指定します。	false



ルーティング定義名で指定するルーティング定義は、送信メッセージルーティング(OUTBOUND)を指定してください。

C.2.2.4 非同期サービス呼出し

非同期でサービスを呼び出す場合、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(AsyncServiceCall)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、AsyncServiceCallの呼出し情報(メソッド(send)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、サービスエンドポイント"endpoint01"で定義されているサービス呼出しを行います。

```
:
    <StepInformation stepname="AsyncServiceCall1" monitoring="false">
        <MediatorFunction name="AsyncServiceCall">
            <Method type="execute" name="send">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <Parameter name="endpoint" valuetype="direct">endpoint01</Parameter>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.71 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	AsyncServiceCall1
MediatorFunction			
name		メディエータファンクション名“AsyncServiceCall”を指定します。	AsyncServiceCall
Method			
type		実行種別“execute”を指定します。	execute
name		AsyncServiceCallのメソッド名“send”を指定します。	send
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
2	サービスエンドポイント名を指定します。		
	name	“endpoint”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	サービスエンドポイント名を指定します。	endpoint01
Method			
type		実行種別“cancel”を指定します。	cancel
name		AsyncServiceCallのメソッド名“cancel”を指定します。	cancel

C.2.2.5 ルーティングして非同期サービス呼出し(ルーティング定義を利用しない)

ルーティング機能を使用して、条件に合った同期サービス呼出しを行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(AsyncServiceCallWithRouting)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、AsyncServiceCallWithRoutingの呼出し情報(メソッド(sendWithConditionRouting)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、“$price == 1000”の条件に合えばサービスエンドポイント“endpointA”に転送し、“$company @EQUAL@ "Fujitsu"”の条件に合えばサービスエンドポイント“endpointB”に転送し、その他の場合には、“endpointC”に転送しています。

```
:
    <StepInformation stepname="AsyncServiceCallWithRouting1">
        <MediatorFunction name="AsyncServiceCallWithRouting">
            <Method type="execute" name="sendWithConditionRouting">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <ParameterArray name="condition_list">
                        <Parameter valuetype="direct">$price == 1000</Parameter>
                        <Parameter valuetype="direct">$company @EQUAL@ "Fujitsu"</Parameter>
                        <Parameter valuetype="direct">*</Parameter>
                    </ParameterArray>
                    <ParameterArray name="endpoint_list">
                        <Parameter valuetype="direct">endpointA</Parameter>
                        <Parameter valuetype="direct">endpointB</Parameter>
                        <Parameter valuetype="direct">endpointC</Parameter>
                    </ParameterArray>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.72 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	AsyncServiceCallWithRouting1
MediatorFunction			
name		メディエータファンクション名“AsyncServiceCallWithRouting”を指定します。	AsyncServiceCallWithRouting
Method			
type		実行種別“execute”を指定します。	execute
name		AsyncServiceCallWithRoutingのメソッド名“sendWithConditionRouting”を指定します。	sendWithConditionRouting
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
ParameterArray			
name		“condition_list”を指定します。条件リストの配列です。	condition_list
1	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)

キー名		説明	指定例
	値	条件式を指定します。詳細は、“条件式”を参照してください。	$price == 1000
2	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	$company @EQUAL@ "Fujitsu"
3	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	*
ParameterArray			
name		“endpoint_list”を指定します。サービスエンドポイントリストの配列です。	endpoint_list
1	Parameter		
	条件式1のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointA
2	Parameter		
	条件式2のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointB
3	Parameter		
	条件式3のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointC
Method			
type		実行種別“cancel”を指定します。	cancel
name		AsyncServiceCallWithRoutingのメソッド名“cancel”を指定します。	cancel

C.2.2.6 ルーティングして非同期サービス呼出し(ルーティング定義を利用)

ルーティング機能を使用して、条件に合った非同期サービス呼出しを行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(AsyncServiceCallWithRouting)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、AsyncServiceCallWithRoutingの呼出し情報(メソッド(sendWithDefinitionRouting)やパラメタ)を定義します。
- パラメタにOUTBOUND用のルーティング定義名を指定します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、OUTBOUND用のルーティング定義“routingA”にメッセージを送信しています。

```
:
    <StepInformation stepname="AsyncServiceCallWithRouting1">
        <MediatorFunction name="AsyncServiceCallWithRouting">
            <Method type="execute" name="sendWithDefinitionRouting">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <Parameter name="routing">routingA</Parameter>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.73 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	AsyncServiceCallWithRouting1
MediatorFunction			
name		メディエータファンクション名 “AsyncServiceCallWithRouting”を指定します。	AsyncServiceCallWithRouting
Method			
type		実行種別“execute”を指定します。	execute
name		AsyncServiceCallWithRoutingのメソッド名 “sendWithDefinitionRouting”を指定します。	sendWithDefinitionRouting
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
2	ルーティング定義名を指定します。		
	name	“routing”を指定します。	routing
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	ルーティング定義名を指定します。	routingA
Method			
type		実行種別“cancel”を指定します。	cancel
name		AsyncServiceCallWithRoutingのメソッド名“cancel”を指定します。	cancel

注意

ルーティング定義名で指定するルーティング定義は、送信メッセージルーティング(OUTBOUND)を指定してください。

C.2.2.7 複数の非同期サービス呼出し

複数の非同期サービス呼出しを行うには以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(AsyncServiceMulticast)の呼出しを定義します。

- シーケンス定義の実行ステップ情報に、AsyncServiceMulticastの呼出し情報(メソッド(multicast)やパラメタ)を定義します。
- パラメタに送信先のサービスエンドポイント名を複数指定します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、サービスエンドポイント“endpointA”、“endpointB”、“endpointC”に同報でメッセージを送信しています。

```
:
    <StepInformation stepname="AsyncServiceMulticast1">
        <MediatorFunction name="AsyncServiceMulticast">
            <Method type="execute" name="multicast">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage"></Parameter>
                    <ParameterArray name="endpoint_list">
                          <Parameter valuetype="direct">endpointA</Parameter>
                          <Parameter valuetype="direct">endpointB</Parameter>
                          <Parameter valuetype="direct">endpointC</Parameter>
                    </ParameterArray>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.74 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	AsyncServiceMulticast1
MediatorFunction			
name		メディエータファンクション名“AsyncServiceMulticast”を指定します。	AsyncServiceMulticast
Method			
type		実行種別“execute”を指定します。	execute
name		SendMessageのメソッド名“multicast”を指定します。	multicast
Parameter			
1	パラメタとして渡すメッセージを指定します。		
	name	“message”を指定します。	name
	valuetype	パラメタに指定する値のタイプを指定します。	esi.esimessage(固定値)
ParameterArray			
name		“endpoint_list”を指定します。サービスエンドポイントリストの配列です。	endpoint_list
1	Parameter		
	送信先1のサービスエンドポイント名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointA
2	Parameter		
	送信先2のサービスエンドポイント名を指定します。		

キー名		説明	指定例
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointB
3	Parameter		
	送信先3のサービスエンドポイント名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointC
Method			
type		実行種別“cancel”を指定します。	cancel
name		SendMessageのメソッド名“cancel”を指定します。	cancel

C.2.3 XML Validationの定義

XML Validationを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XML Validation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLValidationの呼出し情報(メソッド(checkXML)やパラメタ)を定義します。



- XML Validationは、XML SchemaまたはDTDで定義されたXMLデータの妥当性をチェックします。なお、DTDファイル、XML Schemaファイルは、以下に格納してください。

  W

  <INSDIR>¥usr¥schema

  S L

  /opt/FJSVesi/usr/schema

- XML Validationでエラーが発生すると、以下に作業用のファイルが作成されます。出力されるエラーメッセージを参照し、エラー対処後に作業用のファイルを削除してください。

  W

  <INSDIR>¥var¥temp

  S L

  /opt/FJSVesi/var/temp

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在の共通メッセージ内のペイロードをチェックしています。

```
:
    <StepInformation stepname="checkXML">
        <MediatorFunction name="XMLValidation">
            <Method type="execute" name="checkXML" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.75 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	checkXML
MediatorFunction		
name	メディエータファンクション名“XMLValidation”を指定します。	XMLValidation
Method		
type	実行種別“execute”を指定します。	execute
name	XMLValidationのメソッド名“checkXML”を指定します。	checkXML

C.2.4 XML Operationの定義

XML Operationでは、以下の操作ができます。

- 要素/属性の追加(固定値)
- 要素/属性の追加(XMLデータから情報取得)
- 要素の挿入
- 要素/属性の更新(固定値)
- 要素/属性の更新(XMLデータから情報取得)
- 要素/属性の削除

該当する要素のあり/なしによる動作を、操作別に以下に示します。

種別	操作	該当要素あり	該当要素なし
要素	追加	要素を追加	要素を追加
	更新	要素内容を更新	エラー(例外をスロー)
	削除	要素を削除	エラー(例外をスロー)
	挿入	要素を挿入	要素を挿入
属性	追加	属性値を更新	属性を追加
	更新	属性値を更新	エラー(例外をスロー)
	削除	属性を削除	エラー(例外をスロー)
	挿入	－(操作不可)	－(操作不可)

XPathについて

XML Operationでは、要素/属性をXPathで指定できます。



- XPathは、絶対パスで指定してください。相対パスは使用できません。
- 複数の同一要素に対して、一括に操作することはできません。複数の同一要素に対して操作をする場合は、XML Operationを複数回呼び出してください。
- 配列のインデックスを指定しないと、最初に出現した要素が指定されたとみなされます。

・ 以下は、ISI 9.0以前のXPath指定を行う場合に注意してください。

　― ワイルドカードは指定できません。

　― 軸指定は“/”と“@”だけ使用できます。

XPathの指定例については、“ISI Studio ヘルプ”を参照してください。
なお、ISI 9.0以前のXPath指定を行うかどうかの設定については、“ISI 導入ガイド”を参照してください。

C.2.4.1 追加(固定値)

ペイロードに要素/属性を追加するには、以下のようにします。

・ シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLOperation)の呼出しを定義します。

・ シーケンス定義の実行ステップ情報に、XMLOperationの呼出し情報(メソッド(add)やパラメタ)を定義します。

なお、追加する要素は、共通メッセージのヘッダから取得することも可能です。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath2”で示される位置の最後に、要素名が“company”で、値が“Fujitsu”の要素を追加しています。

```
:
    <StepInformation stepname="addXML">
        <MediatorFunction name="XMLOperation">
            <Method type="execute" name="add">
                <ParameterList>
                    <Parameter name="esi.XmlOperation.XPath" valuetype="esi.header">com.fujitsu.esi.xpath2</Parameter>
                    <Parameter name="esi.XmlOperation.addType" valuetype="direct">element</Parameter>
                    <Parameter name="esi.XmlOperation.position" valuetype="direct">last</Parameter>
                    <Parameter name="esi.XmlOperation.name" valuetype="direct">company</Parameter>
                    <Parameter name="esi.XmlOperation.value" valuetype="direct">Fujitsu</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.76 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	addXML
MediatorFunction			
name		メディエータファンクション名“XMLOperation”を指定します。	XMLOperation
Method			
type		実行種別“execute”を指定します。	execute
name		XMLOperationのメソッド名“add”を指定します。	add
Parameter			
1	編集するXML文書内の編集基準となるXPathを指定します。		
	name	“esi.XmlOperation.XPath”を指定します。	esi.XmlOperation.XPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)

キー名		説明	指定例
	値	編集するXML文書内の基準となる位置をXPathで指定します。	com.fujitsu.esi.xpath2
2	要素追加か属性追加か指定します。		
	name	“esi.XmlOperation.addType”を指定します。	esi.XmlOperation.addType
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	要素追加か属性追加かを指定します。 ・ element:要素 ・ attribute:属性	element(要素)
3	追加位置を指定します。		
	name	“esi.XmlOperation.position”を指定します。	esi.XmlOperation.position
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	追加位置を指定します。ただし、属性追加を指定した場合は、必ず最後に追加され、指定した値は無効となります。 ・ before:前 ・ last:最後	last(最後)
4	追加する要素名/属性名を指定します。		
	name	“esi.XmlOperation.name”を指定します。	esi.XmlOperation.name
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	追加する要素名/属性名を指定します。	company
5	追加する要素内容/属性値を指定します。		
	name	“esi.XmlOperation.value”を指定します。	esi.XmlOperation.value
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	追加する要素内容/属性値を指定します。	Fujitsu

注意

共通メッセージのヘッダに格納されている値を、追加する要素内容・属性値として追加する場合は、パラメタ“esi.XmlOperation.value”に共通メッセージのヘッダから取得した値を指定してください。

C.2.4.2 追加(XMLデータから情報取得)

XMLデータから情報を取得して、ペイロードに要素/属性を追加するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLOperationの呼出し情報(メソッド(addFromXML)やパラメタ)を定義します。

追加する要素/属性は、XMLデータから取得します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath2”で示される位置の前に、以下の属性を追加しています。

- 属性名:“company”
- 値:シーケンス起動時のペイロードの“com.fujitsu.esi.xpath1”で示される値

```
:
    <StepInformation stepname="addFromXML">
        <MediatorFunction name="XMLOperation">
            <Method type="execute" name="addFromXML">
                <ParameterList>
                    <Parameter name="esi.XmlOperation.XPath" valuetype="esi.header">com.fujitsu.esi.xpath2</
Parameter>
                 <Parameter name="esi.XmlOperation.addFromAnotherXMLType" valuetype="direct">attribute</Parameter>
                    <Parameter name="esi.XmlOperation.position" valuetype="direct">before</Parameter>
                    <Parameter name="esi.XmlOperation.name" valuetype="direct">company</Parameter>
                    <Parameter name="esi.XmlOperation.srcXML" valuetype="esi.payload">@start</Parameter>
                    <Parameter name="esi.XmlOperation.srcXPath" valuetype="esi.header">com.fujitsu.esi.xpath1</
Parameter>
                </ParameterList>
             </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.77 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	addFromXML
MediatorFunction			
name		メディエータファンクション名“XMLOperation”を指定します。	XMLOperation
Method			
type		実行種別“execute”を指定します。	execute
name		XMLOperationのメソッド名“addFromXML”を指定します。	addFromXML
Parameter			
1	編集するXML文書内の編集基準となるXPathを指定します。		
	name	“esi.XmlOperation.XPath”を指定します。	esi.XmlOperation.XPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	編集するXML文書内の基準となる位置をXPathで指定します。	com.fujitsu.esi.xpath2
2	要素追加か属性追加か指定します。		
	name	“esi.XmlOperation.addFromAnotherXMLType”を指定します。	esi.XmlOperation.addFromAnotherXMLType
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	要素追加か属性追加か指定します。 ・ element:要素 ・ attribute:属性	attribute(属性)
3	追加位置を指定します。		
	name	“esi.XmlOperation.position”を指定します。	esi.XmlOperation.position
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct

キー名		説明	指定例
	値	追加位置を指定します。ただし、属性追加を指定した場合は、必ず最後に追加され、指定した値は無効となります。 ・ before:前 ・ last:最後	before(前)
4	追加する要素名/属性名を指定します。		
	name	“esi.XmlOperation.name”を指定します。	esi.XmlOperation.name
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	要素名/属性名を指定します。	company
5	追加する要素/属性が格納されているXML文書を指定します。		
	name	“esi.XmlOperation.srcXML”を指定します。	esi.XmlOperation.srcXML
	valuetype	パラメタに指定する値のタイプ“esi.payload”(共通メッセージ内のペイロード)を指定します。	esi.payload
	値	追加する要素/属性が格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
6	追加する要素/属性が格納されている位置を指すXPath		
	name	“esi.XmlOperation.srcXPath”を指定します。	esi.XmlOperation.srcXPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	追加する要素/属性が格納されている位置を指すXPathを指定します。	com.fujitsu.esi.xpath1

C.2.4.3 挿入

ペイロードに要素を挿入するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLOperationの呼出し情報(メソッド(insert)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath2”で示される位置の前に、シーケンス起動時のペイロードの“com.fujitsu.esi.xpath1”で示される要素内容を追加しています。

```
:
    <StepInformation stepname="insert">
        <MediatorFunction name="XMLOperation">
            <Method type="execute" name="insert">
                <ParameterList>
                    <Parameter name="esi.XmlOperation.XPath" valuetype="esi.header">com.fujitsu.esi.xpath2</Parameter>
                    <Parameter name="esi.XmlOperation.position" valuetype="direct">before</Parameter>
                    <Parameter name="esi.XmlOperation.srcXML" valuetype="esi.payload">@start</Parameter>
                  <Parameter name="esi.XmlOperation.srcXPath" valuetype="esi.header">com.fujitsu.esi.xpath1</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.78 定義項目

キー名	説明	指定例
StepInformation		

キー名		説明	指定例
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	insert
MediatorFunction			
name		メディエータファンクション名“XMLOperation”を指定します。	XMLOperation
Method			
type		実行種別“execute”を指定します。	execute
name		XMLOperationのメソッド名“insert”を指定します。	insert
Parameter			
1	編集するXML文書内の編集基準となるXPathを指定します。		
	name	“esi.XmlOperation.XPath”を指定します。	esi.XmlOperation.XPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	編集するXML文書内の基準となる位置をXPathで指定します。	com.fujitsu.esi.xpath2
2	挿入位置を指定します。		
	name	“esi.XmlOperation.position”を指定します。	esi.XmlOperation.position
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	追加位置を指定します。 ・ before：前 ・ last：最後	before(前)
3	挿入する要素が格納されているXML文書を指定します。		
	name	“esi.XmlOperation.srcXML”を指定します。	esi.XmlOperation.srcXML
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	挿入する要素が格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
4	挿入する要素の位置を指すXPathを指定します。		
	name	“esi.XmlOperation.srcXPath”を指定します。	esi.XmlOperation.srcXPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	挿入する要素が格納されている位置を指すXPathを指定します。	com.fujitsu.esi.xpath1

C.2.4.4 更新(固定値)

ペイロードの要素/属性を更新するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLOperationの呼出し情報(メソッド(modify)やパラメタ)を定義します。

なお、更新する要素/属性は、共通メッセージのヘッダから取得することも可能です。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath2”で示される位置の要素内容/属性値を“Fujitsu”に更新しています。

```
:
    <StepInformation stepname="modifyXML">
        <MediatorFunction name="XMLOperation">
            <Method type="execute" name="modify">
                <ParameterList>
                    <Parameter name="esi.XmlOperation.XPath" valuetype="esi.header">com.fujitsu.esi.xpath2</Parameter>
                    <Parameter name="esi.XmlOperation.value" valuetype="direct">Fujitsu</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.79 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	modifyXML
MediatorFunction			
name		メディエータファンクション名“XMLOperation”を指定します。	XMLOperation
Method			
type		実行種別“execute”を指定します。	execute
name		XMLOperationのメソッド名“modify”を指定します。	modify
Parameter			
1	編集するXML文書内の編集基準となるXPathを指定します。		
	name	“esi.XmlOperation.XPath”を指定します。	esi.XmlOperation.XPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	編集するXML文書内の基準となる位置をXPathで指定します。	com.fujitsu.esi.xpath2
2	更新する要素内容/属性値を指定します。		
	name	“esi.XmlOperation.value”を指定します。	esi.XmlOperation.value
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	更新する要素内容/属性値を指定します。	Fujitsu



共通メッセージのヘッダに格納されている値を、更新する要素内容・属性値として更新する場合は、パラメタ“esi.XmlOperation.value”に共通メッセージのヘッダから取得した値を指定してください。

C.2.4.5 更新(XMLデータから情報取得)

XMLデータから情報を取得して、ペイロードの要素/属性を更新するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLOperationの呼出し情報(メソッド(modifyFromXML)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath2”で示される要素内容/属性値を、シーケンス起動時のペイロードの“com.fujitsu.esi.xpath1”で示される要素内容/属性値に更新しています。

```
:
    <StepInformation stepname="modifyFromXML">
        <MediatorFunction name="XMLOperation">
            <Method type="execute" name="modifyFromXML">
                <ParameterList>
                    <Parameter name="esi.XmlOperation.XPath" valuetype="esi.header">com.fujitsu.esi.xpath2</Parameter>
                    <Parameter name="esi.XmlOperation.srcXML" valuetype="esi.payload">@start</Parameter>
                  <Parameter name="esi.XmlOperation.srcXPath" valuetype="esi.header">com.fujitsu.esi.xpath1</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.80 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	modifyFromXML
MediatorFunction			
name		メディエータファンクション名“XMLOperation”を指定します。	XMLOperation
Method			
type		実行種別“execute”を指定します。	execute
name		XMLOperationのメソッド名“modifyFromXML”を指定します。	modifyFromXML
Parameter			
1	編集するXML文書内の編集基準となるXPathを指定します。		
	name	“esi.XmlOperation.XPath”を指定します。	esi.XmlOperation.XPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	編集するXML文書内の基準となる位置をXPathで指定します。	com.fujitsu.esi.xpath2
2	更新する要素内容/属性値が格納されているXML文書を指定します。		
	name	“esi.XmlOperation.srcXML”を指定します。	esi.XmlOperation.srcXML
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	更新する要素内容/属性値が格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
3	更新する要素内容/属性値が格納されている位置を指すXPath		
	name	“esi.XmlOperation.srcXPath”を指定します。	esi.XmlOperation.srcXPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	更新する要素内容/属性値を指定します。	com.fujitsu.esi.xpath1

C.2.4.6 削除

ペイロードの要素/属性を削除するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLOperationの呼出し情報(メソッド(delete)やパラメタ)を定義します。

注意

削除する要素の子要素も同時に削除されます。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath3”で示される要素/属性を削除しています。

```
:
    <StepInformation stepname="delete">
        <MediatorFunction name="XMLOperation">
            <Method type="execute" name="delete">
                <ParameterList>
                    <Parameter name="esi.XmlOperation.XPath" valuetype="esi.header">com.fujitsu.esi.xpath3</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.81 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	delete
MediatorFunction			
name		メディエータファンクション名“XMLOperation”を指定します。	XMLOperation
Method			
type		実行種別“execute”を指定します。	execute
name		XMLOperationのメソッド名“delete”を指定します。	delete
Parameter			
1	削除する要素内容/属性値の位置を指すXPathを指定します。		
	name	“esi.XmlOperation.XPath”を指定します。	esi.XmlOperation.XPath
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	削除する要素内容/属性値を指定します。	com.fujitsu.esi.xpath3

C.2.5 XML Emergerの定義

XML Emergerを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLEmergerExt)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLEmergerExtの呼出し情報(メソッド(extract)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの“com.fujitsu.esi.xpath1”で示される情報を取り出しています。

```
:
   <StepInformation stepname="extract">
       <MediatorFunction name="XMLEmergerExt">
           <Method type="execute" name="extract">
               <ParameterList>
                   <ParameterArray name="esi.XmlEmerger.headerList">
                     <Parameter valuetype="direct">price</Parameter>
                   </ParameterArray>
                   <ParameterArray name="esi.XmlEmerger.XPathList">
                     <Parameter valuetype="esi.header">com.fujitsu.esi.xpath1</Parameter>
                   </ParameterArray>
               </ParameterList>
           </Method>
       </MediatorFunction>
   </StepInformation>
:
```

表C.82 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	extract
MediatorFunction			
name		メディエータファンクション名“XMLEmergerExt”を指定します。	XMLEmergerExt
Method			
type		実行種別“execute”を指定します。	execute
name		XMLEmergerExtのメソッド名“extract”を指定します。	extract
ParameterArray			
name		“esi.XmlEmerger.headerList”を指定します。取得情報格納先ヘッダキーの配列です。	esi.XmlEmerger.headerList
Parameter			
1	取得した情報の格納先ヘッダキーを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	取得情報格納先ヘッダキー“price”を指定します。	price
ParameterArray			
name		“esi.XmlEmerger.XPathList”を指定します。情報取得対象が格納されている位置を指すXPathの配列です。	esi.XmlEmerger.headerList
Parameter			
1	情報取得対象が格納されている位置を指すXPathを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	情報取得対象が格納されている位置を指すXPathを指定します。	com.fujitsu.esi.xpath1

XPathについて

注意

- XPathは、絶対パスで指定してください。相対パスは使用できません。
- 配列のインデックスを指定しないと、最初に出現した要素が指定されたとみなされます。
- 以下は、ISI 9.0以前のXPath指定を行う場合に注意してください。
  - ワイルドカードは指定できません。
  - 軸指定は“/”と“@”だけ使用できます。

XPathの指定例については、“ISI Studio ヘルプ”を参照してください。
なお、ISI 9.0以前のXPath指定を行うかどうかの設定については、“ISI 導入ガイド”を参照してください。

C.2.6 FLAT Emergerの定義

FLAT Emergerを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(FlatEmerger)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、FlatEmergerの呼出し情報(メソッド(extractFlat)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの11バイト目から256バイト分を、文字コードShift_JISで取り出しています。

```
:
    <StepInformation stepname="extractFlat">
        <MediatorFunction name="FlatEmerger">
            <Method type="execute" name="extractFlat">
                <ParameterList>
                    <Parameter name="esi.FlatEmerger.startPoint" valuetype="direct">10</Parameter>
                    <Parameter name="esi.FlatEmerger.length" valuetype="direct">256</Parameter>
                    <Parameter name="esi.FlatEmerger.charName" valuetype="direct">Shift_JIS</Parameter>
                </ParameterList>
                <Return valuetype="esi.header">com.fujitsu.esi.flat.data</Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.83 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	extractFlat
MediatorFunction		
name	メディエータファンクション名“FlatEmerger”を指定します。	FlatEmerger
Method		
type	実行種別“execute”を指定します。	execute
name	FlatEmergerのメソッド名“extractFlat”を指定します。	extractFlat
Parameter		

キー名		説明	指定例
1	情報の取得開始位置を指定します。		
	name	“esi.FlatEmerger.startPoint”を指定します。	esi.FlatEmerger.startPoint
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	情報の取得開始位置をバイト単位で指定します。	10
2	情報の取得けた数を指定します。		
	name	“esi.FlatEmerger.length”を指定します。	esi.FlatEmerger.length
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	情報の取得けた数をバイト単位で指定します。	256
3	情報の文字コードを指定します。		
	name	“esi.FlatEmerger.charName”を指定します。	esi.FlatEmerger.charName
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	情報の文字コードを指定します。	“Shift_JIS”
Return			
valuetype		復帰値の格納先を指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
値		復帰値の設定先を指定します。	com.fujitsu.esi.flat.data

C.2.7 SOAP Operationの定義

SOAP Operationの定義について説明します。

C.2.7.1 SOAPメッセージの作成

SOAPメッセージを作成するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SOAPOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAPOperationの呼出し情報(メソッド(createMessage)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、シーケンス起動時の共通メッセージのペイロードをSOAPヘッダとし、現在の共通メッセージのペイロードをSOAPボディとするSOAPメッセージを作成しています。

```
:
   <StepInformation stepname="SOAPOperation1" monitoring="false">
       <MediatorFunction name="SOAPOperation">
           <Method type="execute" name="createMessage">
               <ParameterList>
                   <Parameter name="esi.soap.header" valuetype="esi.payload">@start</Parameter>
                   <Parameter name="esi.soap.body" valuetype="esi.payload">@current</Parameter>
               </ParameterList>
               <Return valuetype="esi.payload"></Return>
           </Method>
       </MediatorFunction>
   </StepInformation>
:
```

表C.84 定義項目

キー名	説明	指定例
StepInformation		

キー名		説明	指定例
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SOAPOperation1
MediatorFunction			
name		メディエータファンクション名“SOAPOperation”を指定します。	SOAPOperation
Method			
type		実行種別“execute”を指定します。	execute
name		SOAPOperationのメソッド名“createMessage”を指定します。	createMessage
Parameter			
1	SOAPヘッダが格納されているXML文書を指定します。		
	name	“esi.soap.header”を指定します。	esi.soap.header
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	SOAPヘッダが格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
2	SOAPボディが格納されているXML文書を指定します。		
	name	“esi.soap.body”を指定します。	esi.soap.body
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	SOAPボディが格納されているXML文書を指定します。	@current(現在のペイロード)
Return			
valuetype		復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.7.2 SOAPメッセージの作成(SOAPボディだけ)

SOAPボディだけのSOAPメッセージを作成するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SOAPOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAPOperationの呼出し情報(メソッド(createMessageWithBody)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、シーケンス起動時のペイロードの情報から、SOAPボディだけのSOAPメッセージを作成しています。

```
:
   <StepInformation stepname="SOAPOperation1" monitoring="false">
       <MediatorFunction name="SOAPOperation">
           <Method type="execute" name="createMessageWithBody">
               <ParameterList>
                   <Parameter name="esi.soap.body" valuetype="esi.payload">@start</Parameter>
               </ParameterList>
               <Return valuetype="esi.payload"></Return>
           </Method>
       </MediatorFunction>
   </StepInformation>
:
```

表C.85 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SOAPOperation1
MediatorFunction			
name		メディエータファンクション名“SOAPOperation”を指定します。	SOAPOperation
Method			
type		実行種別“execute”を指定します。	execute
name		SOAPOperationのメソッド名“createMessageWithBody”を指定します。	createMessageWithBody
Parameter			
1	SOAPボディが格納されているXML文書を指定します。		
	name	“esi.soap.body”を指定します。	esi.soap.body
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	SOAPボディが格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
Return			
valuetype		復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.7.3 SOAPヘッダ要素を取得

SOAPヘッダ要素を取得するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SOAPOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAPOperationの呼出し情報(メソッド(getHeader)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、シーケンス起動時のペイロードの情報から、SOAPヘッダだけのXML文書を作成しています。

```
:
   <StepInformation stepname="SOAPOperation1" monitoring="false">
       <MediatorFunction name="SOAPOperation">
           <Method type="execute" name="getHeader">
               <ParameterList>
                   <Parameter name="esi.soap.message" valuetype="esi.payload">@start</Parameter>
               </ParameterList>
               <Return valuetype="esi.payload"></Return>
           </Method>
       </MediatorFunction>
   </StepInformation>
:
```

表C.86 定義項目

キー名	説明	指定例
StepInformation		

キー名		説明	指定例
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SOAPOperation1
MediatorFunction			
name		メディエータファンクション名“SOAPOperation”を指定します。	SOAPOperation
Method			
type		実行種別“execute”を指定します。	execute
name		SOAPOperationのメソッド名“getHeader”を指定します。	getHeader
Parameter			
1	取得するSOAPヘッダ要素が格納されているXML文書を指定します。		
	name	“esi.soap.message”を指定します。	esi.soap.message
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	取得するSOAPヘッダ要素が格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
Return			
valuetype		復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.7.4 SOAPボディ要素を取得

SOAPボディ要素を取得するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SOAPOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAPOperationの呼出し情報(メソッド(getBody)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、シーケンス起動時のペイロードの情報から、SOAPボディだけのXML文書を作成しています。

```
:
    <StepInformation stepname="SOAPOperation1" monitoring="false">
        <MediatorFunction name="SOAPOperation">
            <Method type="execute" name="getBody">
                <ParameterList>
                    <Parameter name="esi.soap.message" valuetype="esi.payload">@current</Parameter>
                </ParameterList>
                <Return valuetype="esi.payload"></Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.87 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SOAPOperation1
MediatorFunction		

キー名		説明	指定例
name		メディエータファンクション名“SOAPOperation”を指定します。	SOAPOperation
Method			
type		実行種別“execute”を指定します。	execute
name		SOAPOperationのメソッド名“getBody”を指定します。	getBody
Parameter			
1	取得するSOAPボディ要素が格納されているXML文書を指定します。		
	name	“esi.soap.message”を指定します。	esi.soap.message
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	取得するSOAPボディ要素が格納されているXML文書を指定します。	@current(現在のペイロード)
Return			
valuetype		復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.7.5 XML文書にSOAPタグを追加

XML文書にSOAPタグを追加するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SOAPOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAPOperationの呼出し情報(メソッド(addSoapTag)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、シーケンス起動時のペイロードの情報から、SOAPボディだけのSOAPメッセージを作成しています。

```
:
    <StepInformation stepname="SOAPOperation1" monitoring="false">
        <MediatorFunction name="SOAPOperation">
            <Method type="execute" name="addSoapTag">
                <ParameterList>
                    <Parameter name="esi.soap.body" valuetype="esi.payload">@start</Parameter>
                </ParameterList>
                <Return valuetype="esi.payload"></Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.88 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SOAPOperation1
MediatorFunction		
name	メディエータファンクション名“SOAPOperation”を指定します。	SOAPOperation
Method		
type	実行種別“execute”を指定します。	execute

キー名		説明	指定例
name		SOAPOperationのメソッド名“addSoapTag”を指定します。	addSoapTag
Parameter			
1	SOAPボディが格納されているXML文書を指定します。		
	name	“esi.soap.body”を指定します。	esi.soap.body
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	SOAPボディが格納されているXML文書を指定します。	@start(シーケンス起動時のペイロード)
Return			
valuetype		復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.7.6 SOAPメッセージからSOAPタグを削除

SOAPメッセージからSOAPタグを削除するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SOAPOperation)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAPOperationの呼出し情報(メソッド(removeSoapTag)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、シーケンス起動時のペイロードの情報から、SOAPボディだけのXML文書を作成しています。

```
:
   <StepInformation stepname="SOAPOperation1" monitoring="false">
       <MediatorFunction name="SOAPOperation">
           <Method type="execute" name="removeSoapTag">
               <ParameterList>
                   <Parameter name="esi.soap.message" valuetype="esi.payload">@current</Parameter>
               </ParameterList>
               <Return valuetype="esi.payload"></Return>
           </Method>
       </MediatorFunction>
   </StepInformation>
:
```

表C.89 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SOAPOperation1
MediatorFunction			
name		メディエータファンクション名“SOAPOperation”を指定します。	SOAPOperation
Method			
type		実行種別“execute”を指定します。	execute
name		SOAPOperationのメソッド名“removeSoapTag”を指定します。	removeSoapTag
Parameter			
1	取得するSOAPボディ要素が格納されているXML文書を指定します。		

キー名		説明	指定例
	name	“esi.soap.message”を指定します。	esi.soap.message
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	取得するSOAPボディ要素が格納されているXML文書を指定します。	@current(現在のペイロード)
Return			
valuetype		復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.8 XML Converterの定義

XML Converterを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(XMLConverter)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、XMLConverterの呼出し情報(メソッド(convertXML)やパラメタ)を定義します。

注意

W

使用するXSLTスタイルシートファイルは、事前に用意し、“<INSDIR>¥usr¥xslt”に格納してください。

S L

使用するXSLTスタイルシートファイルは、事前に用意し、“/opt/FJSVesi/usr/xslt”に格納してください。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、ペイロードに格納されているXML文書をconvert001.xslを利用して、フォーマット変換しています。

```
:
    <StepInformation stepname="convertXML">
        <MediatorFunction name="XMLConverter">
            <Method type="execute" name="convertXML">
                <ParameterList>
                    <Parameter name="esi.XmlConverter.XSLTFile" valuetype="direct">convert001.xsl</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.90 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	convertXML
MediatorFunction		
name	“XMLConverter”を指定します。	XMLConverter
Method		
type	実行種別“execute”を指定します。	execute

キー名		説明	指定例
name		XMLConverterのメソッド名“convertXML”を指定します。	convertXML
Parameter			
1	変換に利用するXSLTファイルを指定します。		
	name	“esi.XmlConverter.XSLTFile”を指定します。	esi.XmlConverter.XSLTFile
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	変換に利用するXSLTファイルを指定します。	convert001.xsl

C.2.9 Formatmanagerの定義

Formatmanagerを使用するには、以下のようにします。

C.2.9.1 通常のフォーマット変換

- シーケンス定義のシーケンスブロックに、メディエータファンクション(Formatmanager)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、Formatmanagerの呼出し情報(メソッド(convertbyTransformationSequenceID)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードを、com.fujitsu.esi.formatmanager.transSeqIDに従ってフォーマット変換しています。

```
:
    <StepInformation stepname="formatConversion1">
        <MediatorFunction name="Formatmanager">
            <Method type="execute" name="convertbyTransformationSequenceID">
                <ParameterList>
                    <Parameter name="esi.FormatManager.transSeqID"
valuetype="esi.header">com.fujitsu.esi.formatmanager.transSeqID</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.91 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	formatConversion1
MediatorFunction			
name		“Formatmanager”を指定します。	Formatmanager
Method			
type		実行種別“execute”を指定します。	execute
name		Formatmanagerのメソッド名“convertbyTransformationSequenceID”を指定します。	convertbyTransformationSequenceID
Parameter			
1	変換パラメタIDを指定します。変換パラメタIDは、Formatmanagerクライアントの変換パラメタ登録で設定されています。詳細については、“ISI Formatmanagerクライアント(FEDIT/FL-TABLE)ヘルプ”を参照してください。		

キー名		説明	指定例
	name	“esi.FormatManager.transSeqID”を指定します。	esi.FormatManager.transSeqID
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	変換パラメタIDを指定します。	com.fujitsu.esi.formatmanager.transSeqID

C.2.9.2 複数フォーマット入力を使用したフォーマット変換

- シーケンス定義のシーケンスブロックに、メディエータファンクション(Formatmanager)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、Formatmanagerの呼出し情報(メソッド(executeTransformations)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、開始と現在のペイロードを、com.fujitsu.esi.formatmanager.transSeqIDに従ってフォーマット変換しています。

```
:
    <StepInformation stepname="formatConversion1">
        <MediatorFunction name="Formatmanager">
            <Method type="execute" name="executeTransformations">
                <ParameterList>
                    <Parameter name="esi.FormatManager.transSeqID" valuetype="direct">outfile7</Parameter>
                    <ParameterArray name="esi.FormatManager.stepName">
                        <Parameter valuetype="direct">@start</Parameter>
                        <Parameter valuetype="direct">@current</Parameter>
                    </ParameterArray>
                    <ParameterArray name="esi.FormatManager.flatId">
                        <Parameter valuetype="direct">MLT01</Parameter>
                        <Parameter valuetype="direct">MLT02</Parameter>
                    </ParameterArray>
                </ParameterList>
                <Return valuetype="esi.payload"></Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.92 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	formatConversion1
MediatorFunction			
name		“Formatmanager”を指定します。	Formatmanager
Method			
type		実行種別“execute”を指定します。	execute
name		Formatmanagerのメソッド名“executeTramsformations”を指定します。	executeTransformations
Parameter			
1	変換パラメタIDを指定します。変換パラメタIDは、Formatmanagerクライアントの変換パラメタ登録で設定されています。詳細については、“ISI Formatmanagerクライアント(FEDIT/FL-TABLE)ヘルプ”を参照してください。		
	name	“esi.FormatManager.transSeqID”を指定します。	esi.FormatManager.transSeqID
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)

キー名		説明	指定例
	値	変換パラメタIDを指定します。	com.fujitsu.esi.formatmanager.transSeqID
ParameterArray			
name		“esi.FormatManager.stepName”を指定します。ステップ名の配列です。	esi.FormatManager.stepName
1	Parameter		
	変換対象にするステップ名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	ステップ名を指定します。	@start
name		“esi.FormatManager.stepName”を指定します。ステップ名の配列です。	esi.FormatManager.stepName
2	Parameter		
	フラットIDを指定します。フラットIDは、Formatmanagerクライアントの一般フォーマット登録で設定されています。詳細については、“ISI Formatmanagerクライアント（FEDIT/FL-TABLE）ヘルプ”を参照してください。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	フラットIDを指定します。	MLT01

C.2.10 BPM Analytics連携の定義

Interstage BPM Analyticsの定義について説明します。

C.2.10.1 XMLデータのモニタリング定義

XMLデータのモニタリングを行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(BPMMonitoring)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、BPMMonitoringの呼出し情報(メソッド(monitor)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、ペイロードの“/data/product/name/text()”で示される位置のXMLデータをモニタリングします。

```
:
   <StepInformation stepname="BPMMonitoring1" monitoring="false">
       <MediatorFunction name="BPMMonitoring">
           <Method type="execute" name="monitor">
               <ParameterList>
                   <Parameter name="esi.monitoring.isXML">true</Parameter>
                   <Parameter name="esi.monitoring.charCode"></Parameter>
                   <ParameterArray name="esi.monitoring.id">
                       <Parameter valuetype="direct">product_name</Parameter>
                   </ParameterArray>
                   <ParameterArray name="esi.monitoring.isHeader">
                       <Parameter valuetype="direct">esi.payload</Parameter>
                   </ParameterArray>
                   <ParameterArray name="esi.monitoring.position">
                       <Parameter valuetype="direct">/data/product/name/text()</Parameter>
                   </ParameterArray>
               </ParameterList>
           </Method>
       </MediatorFunction>
   </StepInformation>:
```

表C.93 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	BPMMonitoring1
MediatorFunction			
name		メディエータファンクション名“BPMMonitoring”を指定します。	BPMMonitoring
Method			
type		実行種別“execute”を指定します。	execute
name		BPMMonitoringのメソッド名“monitor”を指定します。	monitor
Parameter			
1	モニタリングするデータのタイプを指定します。		
	name	“esi.monitoring.isXML”を指定します。	esi.monitoring.isXML
	値	モニタリングするデータのタイプを指定します。 ・ true XML形式 ・ false FLAT形式	true
2	文字コードを指定します。		
	name	“esi.monitoring.charCode”を指定します。	esi.monitoring.charCode
	値	指定は不要です。	
ParameterArray			
name		“esi.monitoring.id”を指定します。モニタリング変数の配列です。	esi.monitoring.id
1	Parameter		
	モニタリング変数名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“product_name”を指定します。	product_name
ParameterArray			
name		“esi.monitoring.isHeader”を指定します。データ取り出し先の配列です。	esi.monitoring.isHeader
1	Parameter		
	共通メッセージのどこからデータを取り出すかを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	取り出し元を指定します。	esi.payload(共通メッセージのペイロード)
ParameterArray			
name		“esi.monitoring.position”を指定します。取り出す位置の配列です。	esi.monitoring.position
1	Parameter		
	取り出す位置を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct

キー名		説明	指定例
	値	取り出す位置をXPathで指定します。	/data/product/name/text()

C.2.10.2 FLATデータのモニタリング定義

FLATデータのモニタリングを行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(BPMMonitoring)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、BPMMonitoringの呼出し情報(メソッド(monitor)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、ペイロードの21バイト目から15バイト分のFLATデータと、ヘッダキーの“product.code”で示される位置のFLATデータをモニタリングしています。

```
:
   <StepInformation stepname="BPMMonitoring1" monitoring="false">
      <MediatorFunction name="BPMMonitoring">
         <Method type="execute" name="monitor">
            <ParameterList>
               <Parameter name="esi.monitoring.isXML">false</Parameter>
               <Parameter name="esi.monitoring.charCode">Shift_JIS</Parameter>
               <ParameterArray name="esi.monitoring.id">
                  <Parameter valuetype="direct">price</Parameter>
                  <Parameter valuetype="direct">code</Parameter>
               </ParameterArray>
               <ParameterArray name="esi.monitoring.isHeader">
                  <Parameter valuetype="direct">esi.payload</Parameter>
                  <Parameter valuetype="direct">esi.header</Parameter>
               </ParameterArray>
               <ParameterArray name="esi.monitoring.position">
                  <Parameter valuetype="direct">20,15</Parameter>
                  <Parameter valuetype="direct">product.code</Parameter>
               </ParameterArray>
            </ParameterList>
         </Method>
      </MediatorFunction>
   </StepInformation>:
```

表C.94 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	BPMMonitoring1
MediatorFunction			
name		メディエータファンクション名“BPMMonitoring”を指定します。	BPMMonitoring
Method			
type		実行種別“execute”を指定します。	execute
name		BPMMonitoringのメソッド名“monitor”を指定します。	monitor
Parameter			
1	モニタリングするデータのタイプを指定します。		
	name	“esi.monitoring.isXML”を指定します。	esi.monitoring.isXML

キー名		説明	指定例
	値	モニタリングするデータのタイプを指定します。 ・ true XML形式 ・ false FLAT形式	false
2	文字コードを指定します。		
	name	“esi.monitoring.charCode”を指定します。	esi.monitoring.charCode
	値	“esi.monitoring.isXML”に“false”を指定した場合、文字コードを指定します。	Shift_JIS
ParameterArray			
name		“esi.monitoring.id”を指定します。モニタリング変数の配列です。	esi.monitoring.id
1	Parameter		
	モニタリング変数名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“price”を指定します。	price
2	Parameter		
	モニタリング変数名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“code”を指定します。	code
ParameterArray			
name		“esi.monitoring.isHeader”を指定します。データ取り出し先の配列です。	esi.monitoring.isHeader
1	Parameter		
	共通メッセージのどこからデータを取り出すかを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“price”の取り出し元を指定します。	esi.payload(共通メッセージのペイロード)
2	Parameter		
	共通メッセージのどこからデータを取り出すかを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“code”の取り出し元を指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
ParameterArray			
name		“esi.monitoring.position”を指定します。取り出す位置の配列です。	esi.monitoring.position
Parameter			
1	Parameter		
	取り出す位置を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“price”を取り出す位置を“開始位置,けた数”の形式で指定します。	20,15

キー名		説明	指定例
2	Parameter		
	取り出す位置を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct
	値	モニタリング変数名“code”を取り出す位置をヘッダのキー名で指定します。	product.code

C.2.11 List Creator連携の定義

List Creator連携を行うには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(ListCreator)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、ListCreatorの呼出し情報(メソッド(doLCWebService)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、“受注集計帳票”を印刷、保存、FAX送信、PDF保存します。Interstage List Creator Webサービスに接続するサービスエンドポイント名は、endpoint_listcです。

```
:
    <StepInformation stepname="ListCreator1" monitoring="false">
        <MediatorFunction name="ListCreator">
            <Method type="execute" name="doLCWebService">
                <ParameterList>
                    <Parameter name="endpoint" valuetype="direct">endpoint_listc</Parameter>
                    <Parameter name="formname" valuetype="direct">受注集計帳票</Parameter>
                    <Parameter name="filetype" valuetype="direct">CODE_SJIS</Parameter>
                    <Parameter name="commandparameter1" valuetype="direct">-atdirect print,stream,file,fax</Parameter>
                    <Parameter name="commandparameter2" valuetype="direct">-lwnote1 帳票の備考1です。 -lwnote2 帳票の備
考2です。</Parameter>
                    <Parameter name="charcode" valuetype="direct">Shift_JIS</Parameter>
                    <Parameter name="crcode" valuetype="direct">0x0D0A</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.95 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	ListCreator1
MediatorFunction			
name		メディエータファンクション名“ListCreator”を指定します。	ListCreator
Method			
type		実行種別“execute”を指定します。	execute
name		ListCreatorのメソッド名“doLCWebService”を指定します。	doLCWebService
Parameter			
1	接続先サービスエンドポイント名を指定します。		
	name	“endpoint”を指定します。	endpoint

キー名		説明	指定例
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	接続先サービスエンドポイント名を指定します。	endpoint_listc
2	帳票定義を識別する名前を指定します。		
	name	“formname”を指定します。	formname
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	帳票定義を識別する名前を指定します。	受注集計帳票
3	実行環境のコード系を指定します。		
	name	“filetype”を指定します。	filetype
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	実行環境のコード系を指定します。	CODE_SJIS
4	出力先を指定します。		
	name	“commandparameter1”を指定します。	commandparameter1
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	出力先を“-atdirect 出力先”の形式で指定します。出力先には、以下を指定します。 ・ print 印刷 ・ stream 電子保存 ・ fax FAX送信 ・ file PDF保存	-atdirect print,stream,fax,file
5	追加するコマンドラインを指定します。		
	name	“commandparameter2”を指定します。	commandparameter2
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	追加するコマンドラインを指定します。	-lwnote1 帳票の備考1です。-lwnote2 帳票の備考2です。
6	帳票データのコード系を指定します。		
	name	“charcode”を指定します。	charcode
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	帳票データのコード系を指定します。	Shift_JIS
7	帳票データの改行コードを指定します。		
	name	“crcode”を指定します。	crcode
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	帳票データのコード系を指定します。	0x0D0A

キー名		説明	指定例
		・ 0x0D0A 改行コードがCRLFの場合(0x0D0A) ・ 0x0D 改行コードがCRの場合(0x0D) ・ 0x0A 改行コードがLFの場合(0x0A)	

C.2.12 Command Launcherの定義

Command Launcherを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(CommandLauncher)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、CommandLauncherの呼出し情報(メソッド(executeProcess)やパラメタ)を定義します。

指定例



実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードをパラメタとして渡し、コマンド“c:¥test.bat”を実行しています。
復帰値が“1”、“2”、“3”以外は、シーケンスはエラーとなります。

```
:
    <StepInformation stepname="CommandLauncher1">
        <MediatorFunction name="CommandLauncher">
            <Method type="execute" name="executeProcess">
                <ParameterList>
                    <Parameter name="esi.cmd.command" valuetype="direct">C:¥test.bat</Parameter>
                    <Parameter name="esi.cmd.normalends" valuetype="direct">1,2,3</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```



実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードをパラメタとして渡し、コマンド“/usr/bin/test”を実行しています。
復帰値が“1”、“2”、“3”以外は、シーケンスはエラーとなります。

```
:
    <StepInformation stepname="CommandLauncher1">
        <MediatorFunction name="CommandLauncher">
            <Method type="execute" name="executeProcess">
                <ParameterList>
                    <Parameter name="esi.cmd.command" valuetype="direct">/usr/bin/test</Parameter>
                    <Parameter name="esi.cmd.normalends" valuetype="direct">1,2,3</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.96 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	CommandLauncher1
MediatorFunction			
name		メディエータファンクション名“CommandLauncher”を指定します。	CommandLauncher
Method			
type		実行種別“execute”を指定します。	execute
name		CommandLauncherのメソッド名“executeProcess”を指定します。	executeProcess
Parameter			
1	実行するコマンドを指定します。		
	name	“esi.cmd.command”を指定します。	esi.cmd.command
	valuetype	パラメタに指定する値のタイプを指定します。	Direct
	値	実行するコマンドを指定します。	W C:¥test.bat S L /usr/bin/test
2	コマンドが正常終了した場合に返す復帰値を指定します。		
	name	“esi.cmd.normalends”を指定します。	esi.cmd.normalends
	valuetype	パラメタに指定する値のタイプを指定します。	Direct
	値	正常とみなす復帰値を指定します。	1,2,3

C.2.13 File Operatorの定義

File Operatorの定義について説明します。

C.2.13.1 ファイル読込み

ファイルの内容をペイロードに読み込むには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(FileOperator)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、FileOperatorの呼出し情報(メソッド(readFile)やパラメタ)を定義します。

指定例

W

実行ステップ情報の指定例は、以下のとおりです。この例では、ファイル“C:¥indata¥file_in1.dat”の内容を、ペイロードに読み込みます。

```
:
    <StepInformation stepname="FileOperator1" monitoring="false">
        <MediatorFunction name="FileOperator">
            <Method type="execute" name="readFile">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage" />
```

```
                <Parameter name="filename" valuetype="direct">C:¥indata¥file_in1.dat</Parameter>
            </ParameterList>
            <Return valuetype="esi.payload" />
        </Method>
    </MediatorFunction>
</StepInformation>
:
```

S L

実行ステップ情報の指定例は、以下のとおりです。この例では、ファイル“/indata/file_in1.dat”の内容を、ペイロードに読み込みます。

```
:
<StepInformation stepname="FileOperator1" monitoring="false">
    <MediatorFunction name="FileOperator">
        <Method type="execute" name="readFile">
            <ParameterList>
                <Parameter name="message" valuetype="esi.esimessage" />
                <Parameter name="filename" valuetype="direct">/indata/file_in1.dat</Parameter>
            </ParameterList>
            <Return valuetype="esi.payload" />
        </Method>
    </MediatorFunction>
</StepInformation>
:
```

表C.97 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	FileOperator1
MediatorFunction			
name		メディエータファンクション名“FileOperator”を指定します。	FileOperator
Method			
type		実行種別“execute”を指定します。	execute
name		FileOperatorのメソッド名“readFile”を指定します。	readFile
Parameter			
1	読込み先のメッセージを指定します。		
	name	“message”を指定します。	message
	valuetype	パラメタに指定する値のタイプ“esi.esimessage”を指定します。	esi.esimessage
2	読み込むファイル名を指定します。		
	name	“filename”を指定します。	filename
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	読み込むファイル名を指定します。	W C:¥indata¥file_in1.dat S L /indata/file_in1.dat

キー名	説明	指定例
Return		
valuetype	復帰値の格納先を指定します。	esi.payload(共通メッセージ内のペイロード)

C.2.13.2 ファイル出力

ペイロードの内容をファイルに出力するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(FileOperator)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、FileOperatorの呼出し情報(メソッド(writeFile)やパラメタ)を定義します。

指定例



実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの内容をファイル“C:¥outdata¥file_out1.dat”に出力します。

```
:
    <StepInformation stepname="FileOperator2" monitoring="false">
        <MediatorFunction name="FileOperator">
            <Method type="execute" name="writeFile">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage" />
                    <Parameter name="payload" valuetype="esi.payload">@current</Parameter>
                    <Parameter name="filename" valuetype="direct">C:¥outdata¥file_out1.dat</Parameter>
                    <Parameter name="append" valuetype="direct">false</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```



実行ステップ情報の指定例は、以下のとおりです。この例では、現在のペイロードの内容をファイル“/outdata/file_out1.dat”に出力します。

```
:
    <StepInformation stepname="FileOperator2" monitoring="false">
        <MediatorFunction name="FileOperator">
            <Method type="execute" name="writeFile">
                <ParameterList>
                    <Parameter name="message" valuetype="esi.esimessage" />
                    <Parameter name="payload" valuetype="esi.payload">@current</Parameter>
                    <Parameter name="filename" valuetype="direct">/outdata/file_out1.dat</Parameter>
                    <Parameter name="append" valuetype="direct">false</Parameter>
                </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.98 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	FileOperator2
MediatorFunction			
name		メディエータファンクション名“FileOperator”を指定します。	FileOperator
Method			
type		実行種別“execute”を指定します。	execute
name		FileOperatorのメソッド名“writeFile”を指定します。	writeFile
Parameter			
1	出力対象のメッセージを指定します。		
	name	“message”を指定します。	message
	valuetype	パラメタに指定する値のタイプ“esi.esimessage”を指定します。	esi.esimessage
2	出力対象のペイロードを指定します。		
	name	“payload”を指定します。	payload
	valuetype	パラメタに指定する値のタイプ“esi.payload”を指定します。	esi.payload
	値	出力対象のペイロード“@current”(現在のペイロード)を指定します。	@current
3	出力先のファイル名を指定します。		
	name	“filename”を指定します。	filename
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	出力先のファイル名を指定します。	W C:¥outdata¥file_out1.dat S L /outdata/file_out1.dat
4	ファイルに追記するかどうかを指定します。		
	name	“append”を指定します。	append
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	ファイルに追記するかどうかを指定します。 ・ true:ファイルに追記する ・ false:先頭からの書き込み(デフォルト)	false

C.2.14 ユーザプログラム呼出しの定義

ユーザプログラム呼出しを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、ユーザプログラムのメディエータファンクションの呼出しを定義します。
- シーケンス定義の実行ステップ情報に、ユーザプログラムの呼出し情報(メソッドやパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、パラメタnumAに“100”、パラメタnumB“200”を指定して、メディエータファンクション“calculator”のメソッド“addition”を呼び出しています。

```
:
    <StepInformation stepname="CalcStep">
        <MediatorFunction name="calculator">
            <Method type="execute" name="addition">
                <ParameterList>
                    <Parameter name="numA" valuetype="direct">100</Parameter>
                    <Parameter name="numB" valuetype="direct">200</Parameter>
                </ParameterList>
                <Return valuetype="esi.header">result</Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.99 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	CalcStep
MediatorFunction			
name		ユーザプログラムのメディエータファンクション名“calculator”を指定します。	calculator
Method			
type		実行種別“execute”を指定します。	execute
name		calculatorのメソッド名“addition”を指定します。	addition
Parameter			
1	パラメタを指定します。		
	name	“numA”を指定します。	numA
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	パラメタの値を指定します。	100
2	パラメタを指定します。		
	name	“numB”を指定します。	numB
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	パラメタの値を指定します。	200
Return			
valuetype		復帰値の格納先を指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
値		復帰値の設定先を指定します。	result

C.2.15 条件分岐ブロック情報の定義

条件分岐を利用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、条件分岐ブロック、分岐経路および条件不成立時の経路情報を定義します。

・ シーケンス定義の条件分岐ブロック情報に、条件分岐ブロックの詳細情報(データのタイプ、ポジションおよび分岐経路の条件など)を定義します。

指定例

条件分岐ブロック情報の指定例は、以下の通りです。この例では、ペイロードをCSV形式として、1行目の8カラム目の値を取得し、以下の判定処理を行います。

・ 取得した値が10000以上の場合、分岐経路“ConditionBranch1”上にあるステップの処理を行います。

・ 取得した値が10000より小さい、かつ5000以上の場合、分岐経路“ConditionBranch2”上にあるステップの処理を行います。

・ どちらの条件も満たさない場合、条件不成立時の経路上にあるステップの処理を行います。

```
:
    <ConditionBranchInformation name="ConditionBranchBlock1">
        <DataType>csv</DataType>
        <PositionList>
            <Position name="point" charset="UTF-8">1,8</Position>
        </PositionList>
        <ConditionInformation name="ConditionBranch1">
            <Description>ゴールド会員の場合</Description>
            <Condition>?point? &gt;= 10000</Condition>
        </ConditionInformation>
        <ConditionInformation name="ConditionBranch2">
            <Description>シルバー会員の場合</Description>
            <Condition>?point? &lt; 10000 AND ?point? &gt;= 5000</Condition>
        </ConditionInformation>
    </ConditionBranchInformation>
:
```

表C.100 定義項目

キー名		説明	指定例
ConditionBranchInformation			
name		条件分岐ブロック名を指定します。シーケンスブロックに指定した条件分岐ブロック名を指定してください。	ConditionBranchBlock1
DataType		判断するデータのタイプを指定します。以下のどれかを指定します。 ・ xml XML形式のペイロード ・ flat FLAT形式のペイロード ・ csv CSV形式のペイロード ・ header 共通メッセージのヘッダパラメタ	csv
Position			
1	ポジションを指定します。		
	name	ポジション名を指定します。	point
	charset	使用しているJavaVMがサポートする文字コードを指定します。	UTF-8
	値	判断するデータの位置を指定します。	1,8
ConditionInformation			

キー名		説明	指定例
1	分岐経路の条件を指定します。		
	name	分岐経路名を指定します。シーケンスブロックに指定した分岐経路名を指定してください。	ConditionBranch1
	Description	分岐経路の説明を指定します。	ゴールド会員の場合
	Condition	分岐経路の条件を指定します。	?point? >= 10000
2	分岐経路の条件を指定します。		
	name	分岐経路名を指定します。シーケンスブロックに指定した分岐経路名を指定してください。	ConditionBranch2
	Description	分岐経路の説明を指定します。	シルバー会員の場合
	Condition	分岐経路の条件を指定します。	?point? < 10000 AND ?point? >= 5000

C.2.16 互換機能の定義

C.2.16.1 メッセージ送信の定義(互換定義)

メッセージ送信の定義について説明します。

ルーティングなし

ルーティング機能を使用しないで、指定したサービスエンドポイントにメッセージ送信するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SendMessage)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SendMessageの呼出し情報(メソッド(send)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、サービスエンドポイント“com.fujitsu.esi.endpoint.outbound2”にメッセージ送信しています。

```
:
    <StepInformation stepname="SendMessage1">
        <MediatorFunction name="SendMessage">
            <Method type="init" name="init" />
            <Method type="execute" name="send">
                <ParameterList>
                 <Parameter name="esi.SendMessage.endpoint" valuetype="esi.header">com.fujitsu.esi.endpoint.outbound2</
Parameter>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
            <Method type="destroy" name="destroy" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.101 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SendMessage1
MediatorFunction		

キー名		説明	指定例
name		メディエータファンクション名“SendMessage”を指定します。	SendMessage
Method			
type		実行種別“init”を指定します。	init
name		SendMessageのメソッド名“init”を指定します。	init
Method			
type		実行種別“execute”を指定します。	execute
name		SendMessageのメソッド名“send”を指定します。	send
Parameter			
1	サービスエンドポイント名を指定します。		
	name	“esi.SendMessage.endpoint”を指定します。	esi.SendMessage.endpoint
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のペイロード)
	値	サービスエンドポイント名を指定します。	com.fujitsu.esi.endpoint.outbound2
Method			
type		実行種別“cancel”を指定します。	cancel
name		SendMessageのメソッド名“cancel”を指定します。	cancel
Method			
type		実行種別“destroy”を指定します。	destroy
name		SendMessageのメソッド名“destroy”を指定します。	destroy

ルーティングあり(ルーティング定義を利用しない)

ルーティング機能を使用して、条件に合ったサービスエンドポイントにメッセージ送信するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SendMessage)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SendMessageの呼出し情報(メソッド(sendWithRouting)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、“$price == 1000”の条件に合えばサービスエンドポイント“endpointA”に転送し、“$company @EQUAL@ "Fujitsu"”の条件に合えばサービスエンドポイント“endpointB”に転送し、その他の場合には、“com.fujitsu.esi.endpoint.outbound2”に転送しています。

```
:
   <StepInformation stepname="SendMessage2">
       <MediatorFunction name="SendMessage">
           <Method type="init" name="init" />
           <Method type="execute" name="sendWithRouting">
               <ParameterList>
                   <ParameterArray name="esi.SendMessage.endCondition">
                       <Parameter valuetype="direct">$price == 1000</Parameter>
                       <Parameter valuetype="direct">$company @EQUAL@ "Fujitsu"</Parameter>
                       <Parameter valuetype="direct">*</Parameter>
                   </ParameterArray>
                   <ParameterArray name="esi.SendMessage.endpoint">
                       <Parameter valuetype="direct">endpointA</Parameter>
                       <Parameter valuetype="direct">endpointB</Parameter>
                       <Parameter valuetype="esi.header">com.fujitsu.esi.endpoint.outbound2</Parameter>
                   </ParameterArray>
               </ParameterList>
```

```
            </Method>
            <Method type="cancel" name="cancel" />
            <Method type="destroy" name="destroy" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.102 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SendMessage2
MediatorFunction			
name		メディエータファンクション名“SendMessage”を指定します。	SendMessage
Method			
type		実行種別“init”を指定します。	init
name		SendMessageのメソッド名“init”を指定します。	init
Method			
type		実行種別“execute”を指定します。	execute
name		SendMessageのメソッド名“sendWithRouting”を指定します。	sendWithRouting
ParameterArray			
name		“esi.SendMessage.endCondition”を指定します。条件リストの配列です。	esi.SendMessage.endCondition
1	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	$price == 1000
2	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	$company @EQUAL@ "Fujitsu"
3	Parameter		
	条件式を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	条件式を指定します。詳細は、“条件式”を参照してください。	*
ParameterArray			
name		“esi.SendMessage.endpoint”を指定します。サービスエンドポイントリストの配列です。	esi.SendMessage.endpoint
1	Parameter		
	条件式1のサービスエンドポイントを指定します。		

キー名		説明	指定例
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointA
2	Parameter		
	条件式2のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointB
3	Parameter		
	条件式3のサービスエンドポイントを指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	esi.header(共通メッセージ内のヘッダパラメタ)
	値	サービスエンドポイントを指定します。	com.fujitsu.esi.endpoint.outbound2
Method			
type		実行種別“cancel”を指定します。	cancel
name		SendMessageのメソッド名“cancel”を指定します。	cancel
Method			
type		実行種別“destroy”を指定します。	destroy
name		SendMessageのメソッド名“destroy”を指定します。	destroy

条件式

ここでは、条件式で使用できる演算子と文法について説明します。

演算子

条件式で使用できる演算子を以下に示します。

表C.103 演算子の一覧

演算子	説明
>	数値比較を行います。左辺の値が右辺の値よりも大きい場合に真となります。
<	数値比較を行います。左辺の値が右辺の値よりも小さい場合に真となります。
>=	数値比較を行います。左辺の値が右辺の値以上の場合に真となります。
<=	数値比較を行います。左辺の値が右辺の値以下の場合に真となります。
==	数値比較を行います。左辺の値が右辺の値と等しい場合に真となります。
!=	数値比較を行います。左辺の値が右辺の値と異なる場合に真となります。
@EQUAL@	文字列比較を行います。左辺の文字列が右辺の文字列と等しい場合に真となります。
@NOTEQUAL@	文字列比較を行います。左辺の文字列が右辺の文字列と異なる場合に真となります。

条件式の文法

条件式には、以下の文法があります。

- 文字列を表す場合は、“ " ”(ダブルクォーテーション)でくくります。
  “ " ”がない場合は、数値とみなされます。
- 条件式に括弧は使用できません。
- 論理演算子は、ANDだけ使用でき、“&&”と記述します。
- 共通メッセージのヘッダパラメタを指定する場合は、ヘッダパラメタキー名の先頭に“$”を付けます。
- 比較対象値、比較演算子、および論理演算子の間には、デリミタとして必ず半角スペースを入れます。

- 条件式に“*”(半角のアスタリスク)だけを指定すると、無条件に判定結果が真となります。

注意

XML文書中には直接“ " ”、“&”、“<”および“>”を記述することができません。条件式でこれらの文字を使用する場合は、CDATAセクションまたは実体参照を使用して記述してください。

$price >= 1000を定義する場合

- CDATAセクションを使用する場合の定義例
  <![CDATA[$price >= 1000]]>
- 実体参照を使用する場合の定義例
  $price >= 1000

ルーティングあり(ルーティング定義を利用)

送信するメッセージ内の値や条件によって、動的に送信先サービスエンドポイントを設定するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SendMessage)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SendMessageの呼出し情報(メソッド(sendWithContentsRouting)やパラメタ)を定義します。
- パラメタにOUTBOUND用のルーティング定義名を指定します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、OUTBOUND用のルーティング定義“routingA”にメッセージを送信しています。

```
:
    <StepInformation stepname="SendMessage3">
        <MediatorFunction name="SendMessage">
            <Method type="init" name="init" />
            <Method type="execute" name="sendWithContentsRouting">
                <ParameterList>
                    <Parameter name="esi.SendMessage.routing.definition">routingA</Parameter>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
            <Method type="destroy" name="destroy" />
        </MediatorFunction>
    </StepInformation>
:
```

表C.104 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SendMessage3
MediatorFunction		
name	メディエータファンクション名“SendMessage”を指定します。	SendMessage
Method		
type	実行種別“init”を指定します。	init
name	SendMessageのメソッド名“init”を指定します。	init

キー名		説明	指定例
Method			
type		実行種別“execute”を指定します。	execute
name		SendMessageのメソッド名“sendWithContentsRouting”を指定します。	sendWithContentsRouting
Parameter			
1	ルーティング定義名を指定します。		
	name	“esi.SendMessage.routing.definition”を指定します。	esi.SendMessage.routing.definition
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	ルーティング定義名を指定します。	routingA
Method			
type		実行種別“cancel”を指定します。	cancel
name		SendMessageのメソッド名“cancel”を指定します。	cancel
Method			
type		実行種別“destroy”を指定します。	destroy
name		SendMessageのメソッド名“destroy”を指定します。	destroy



注意

ルーティング定義名で指定するルーティング定義は、送信メッセージルーティング(OUTBOUND)を指定してください。

同報送信

複数のアウトバウンドサービスエンドポイントや複数のシーケンスに同報送信するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(SendMessage)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SendMessageの呼出し情報(メソッド(multicast)やパラメタ)を定義します。
- パラメタに送信先のサービスエンドポイント名を複数指定します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、サービスエンドポイント“endpointA”、“endpointB”、“endpointC”に同報でメッセージを送信しています。

```
:
    <StepInformation stepname="SendMessage3">
        <MediatorFunction name="SendMessage">
            <Method type="init" name="init" />
            <Method type="execute" name="multicast">
                <ParameterList>
                    <Parameter name="esi.SendMessage.error.send.action">continue</Parameter>
                    <Parameter name="esi.SendMessage.error.sequence.action">break</Parameter>
                    <ParameterArray name="esi.SendMessage.endpoint">
                          <Parameter>endpointA</Parameter>
                          <Parameter>endpointB</Parameter>
                          <Parameter>endpointC</Parameter>
                    </ParameterArray>
                </ParameterList>
            </Method>
            <Method type="cancel" name="cancel" />
            <Method type="destroy" name="destroy" />
```

```
        </MediatorFunction>
    </StepInformation>
:
```

表C.105 定義項目

キー名		説明	指定例
StepInformation			
stepname		ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	SendMessage3
MediatorFunction			
name		メディエータファンクション名“SendMessage”を指定します。	SendMessage
Method			
type		実行種別“init”を指定します。	init
name		SendMessageのメソッド名“init”を指定します。	init
Method			
type		実行種別“execute”を指定します。	execute
name		SendMessageのメソッド名“multicast”を指定します。	multicast
Parameter			
1	エラー発生時に同報送信を継続するかどうかを指定します。		
	name	“esi.SendMessage.error.send.action”を指定します。	esi.SendMessage.error.send.action
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	エラー発生時に同報送信を継続するかどうかを指定します。 ・ continue 継続(デフォルト) ・ break 中断	continue
2	エラー発生時にシーケンス処理を継続するかどうかを指定します。		
	name	“esi.SendMessage.error.sequence.action”を指定します。	esi.SendMessage.error.sequence.action
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	エラー発生時にシーケンス処理を継続するかどうかを指定します。 ・ continue 継続(デフォルト) ・ break 中断	continue
ParameterArray			
name		“esi.SendMessage.endpoint”を指定します。サービスエンドポイントリストの配列です。	esi.SendMessage.endpoint
1	Parameter		
	送信先1のサービスエンドポイント名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointA

キー名		説明	指定例
2	Parameter		
	送信先2のサービスエンドポイント名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointB
3	Parameter		
	送信先3のサービスエンドポイント名を指定します。		
	valuetype	パラメタに指定する値のタイプを指定します。	direct(固定値)
	値	サービスエンドポイントを指定します。	endpointC
Method			
type		実行種別“cancel”を指定します。	cancel
name		SendMessageのメソッド名“cancel”を指定します。	cancel
Method			
type		実行種別“destroy”を指定します。	destroy
name		SendMessageのメソッド名“destroy”を指定します。	destroy

C.2.16.2 SOAP Gatewayの定義(互換定義)

SOAP Gatewayを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、SOAP Gatewayのメディエータファンクションの呼出しを定義します。
- シーケンス定義の実行ステップ情報に、SOAP Gatewayの呼出し情報(メソッドやパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、共通メッセージ本体をパラメタに指定して、メディエータファンクション“ESISoapGw”のメソッド“sendMessage”を呼び出しています。

```
:
    <StepInformation stepname="GwStep">
        <MediatorFunction name="ESISoapGw">
            <Method type="execute" name="sendMessage">
            <ParameterList>
                <Parameter name="esi.SoapGateway.endpoint" valuetype="direct">endpoint01</Parameter>
            </ParameterList>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.106 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	GwStep
MediatorFunction		
name	SOAP Gatewayのメディエータファンクション名“ESISoapGw”を指定します。	ESISoapGw
Method		

キー名	説明	指定例
type	実行種別“execute”を指定します。	execute
name	ESISoapGwのメソッド名“sendMessage”を指定します。	sendMessage
Parameter		
name	“esi.SoapGateway.endpoint”を指定します。	esi.SoapGateway.endpoint
valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
値	サービスエンドポイント名を指定します。	endpoint01

ポイント

Parameterの“esi.SoapGateway.endpoint”を省略した場合は、ESIHeaderのパラメタ“com.fujitsu.esi.endpoint.outbound”の値が使用されます。

C.2.16.3 JCA Adapterの定義(互換定義)

JCA Adapterを使用するには、以下のようにします。

- シーケンス定義のシーケンスブロックに、メディエータファンクション(JCAAdapter)の呼出しを定義します。
- シーケンス定義の実行ステップ情報に、JCAAdapterの呼出し情報(メソッド(execute)やパラメタ)を定義します。

指定例

実行ステップ情報の指定例は、以下のとおりです。この例では、サービスエンドポイント名“esi.JcaAdapter.endpoint”に定義されたアダプタ製品または企業情報システムを呼び出します。

```
:
    <StepInformation stepname="JcaAdapter1" monitoring="false">
        <MediatorFunction name="JCAAdapter">
            <Method type="execute" name="execute">
                <ParameterList>
                    <Parameter name="esi.JcaAdapter.payload" valuetype="esi.payload">@current</Parameter>
                    <Parameter name="esi.JcaAdapter.endpoint" valuetype="direct">jca_endpoint</Parameter>
                    <Parameter name="esi.JcaAdapter.charcode" valuetype="direct">UTF-8</Parameter>
                    <Parameter name="esi.JcaAdapter.isTransaction">true</Parameter>
                </ParameterList>
                <Return valuetype="esi.payload"></Return>
            </Method>
        </MediatorFunction>
    </StepInformation>
:
```

表C.107 定義項目

キー名	説明	指定例
StepInformation		
stepname	ステップ名を指定します。シーケンスブロックに指定したステップ名を指定してください。	JcaAdapter1
MediatorFunction		
name	メディエータファンクション名“JCAAdapter”を指定します。	JCAAdapter
Method		

キー名		説明	指定例
type		実行種別“execute”を指定します。	execute
name		JCAAdapterのメソッド名“execute”を指定します。	execute
Parameter			
1	ペイロードを指定します。		
	name	“esi.JcaAdapter.payload”を指定します。	esi.JcaAdapter.payload
	valuetype	パラメタに指定する値のタイプを指定します。	esi.payload(共通メッセージ内のペイロード)
	値	JCAアダプタで渡すペイロードを指定します。	@current
2	サービスエンドポイント名を指定します。		
	name	“esi.JcaAdapter.endpoint”を指定します。	esi.JcaAdapter.endpoint
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	サービスエンドポイント名を指定します。	jca_endpoint
3	ペイロードの文字コードを指定します。		
	name	“esi.JcaAdapter.charcode”を指定します。	esi.JcaAdapter.charcode
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	ペイロードの文字コードを指定します。	UTF-8
4	トランザクション制御を指定します。		
	name	“esi.JcaAdapter.isTransaction”を指定します。	esi.JcaAdapter.isTransaction
	valuetype	パラメタに指定する値のタイプ“direct”(固定値)を指定します。	direct
	値	トランザクション制御を指定します。 ・ true する ・ false しない	true
Method			
type		実行種別“cancel”を指定します。	cancel
name		JCAAdapterのメソッド名“cancel”を指定します。	cancel
Method			
type		実行種別“destroy”を指定します。	destroy
name		JCAAdapterのメソッド名“destroy”を指定します。	destroy

C.3 メディエータファンクション定義

メディエータファンクション定義の詳細について説明します。

C.3.1 メディエータファンクション定義の構成

メディエータファンクション定義の構成と指定例は、以下のとおりです。

メディエータファンクション定義

```
MediatorFunctionList
      MediatorFunction
```

```
        Description
        DependJar
              Jar
        Method
              ParameterList
                      Parameter
                      ParameterArray
              Return
```

指定例

メディエータファンクション定義:mediatorfunction.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<MediatorFunctionList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
mediatorfunction.xsd">
    <MediatorFunction name="calculator"
                jar="userfunction.jar"
                class="user.function.Calculator">
        <Method name="addition" methodname="addition">
            <ParameterList>
                <Parameter name="numA" type="long" />
                <Parameter name="numB" type="long" />
            </ParameterList>
            <Return type="long"/>
        </Method>
    </MediatorFunction>
</MediatorFunctionList>
```

表C.108 定義項目

キー名		説明	指定例
MediatorFunction			
name		ユーザプログラムのメディエータファンクション名を指定します。ユーザが任意に指定できます。	calculator
jar		メディエータファンクションのjarファイル名を指定します。ユーザが任意に指定できます。	userfunction.jar
class		メディエータファンクションのクラス名を指定します。	user.function.Calculator
Method			
name		定義上のメソッド名を指定します。	addition
methodname		実メソッド名を指定します。	addition
Parameter			
1	パラメタを指定します。		
	name	パラメタ名を指定します。ユーザが任意に指定できます。	numA
	type	パラメタのデータ型を指定します。	long
2	パラメタを指定します。		
	name	パラメタ名を指定します。ユーザが任意に指定できます。	numB
	type	パラメタのデータ型を指定します。	long
Return			
type		復帰値のデータ型を指定します。	long

C.4 ルーティング定義

ルーティング定義の詳細について説明します。

C.4.1 ルーティング定義の構成

ルーティング定義の構成と指定例は、以下のとおりです。

ルーティング定義

```
MessageDistribution
      Description
      PositionList
            Position
      DistributionList
            Distribution
                  Condition
                  Endpoint
                  Sequence
```

指定例

ルーティング定義の指定例は、以下のとおりです。この例では、条件文により受信したデータを判断し、以下のようにインバウンドサービスエンドポイントへ転送しています。

データの位置

- bookid
  XML形式のデータで、XPath“/booklist/book/@bookid”にデータが格納されています。
- price
  XML形式のデータで、XPath“/booklist/book/price”にデータが格納されています。
- division
  FLAT形式のデータで、先頭から3バイトまでにデータが格納されています。

条件

- bookidが“B01”ならば、サービスエンドポイント“endpoint01”に転送します。
- bookidが“B02”、かつpriceが“1000未満”ならば、サービスエンドポイント“endpoint02”に転送します。
- bookidが“B02”、かつpriceが“1000以上”ならば、サービスエンドポイント“endpoint03”に転送します。
- divisionが“999”ならば、サービスエンドポイント“endpoint11”に転送します。
- 上記のどの条件にも合わない場合は、サービスエンドポイント“endpoint21”に転送します。

ルーティング定義:soap.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<MessageDistribution name="soap-distribution" routing="INBOUND" inboundtype="SOAP-MSG" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-endpointdistribution.xsd">
    <Description>受信用振分け定義</Description>
    <PositionList>
        <Position name="bookid" type="xml">/booklist/book/@bookid</Position>
        <Position name="price" type="xml">/booklist/book/price</Position>
        <Position name="division" type="flat" code="Shift-jis">0,3</Position>
    </PositionList>
    <DistributionList>
        <Distribution>
            <Condition>$bookid @EQUAL@ "B01"</Condition>
            <Endpoint>endpoint01</Endpoint>
        </Distribution>
        <Distribution>
            <Condition>$bookid @EQUAL@ "B02" &amp;amp;&amp;amp; $price &amp;lt; 1000</Condition>
```

```
            <Endpoint>endpoint02</Endpoint>
            <Condition>$bookid @EQUAL@ "B02" &amp;amp;&amp;amp; $price &gt;= 1000</Condition>
            <Endpoint>endpoint03</Endpoint>
        </Distribution>
        <Distribution>
            <Condition>$division @EQUAL@ "999"</Condition>
            <Endpoint>endpoint11</Endpoint>
        </Distribution>
        <Distribution>
            <Condition>*</Condition>
            <Endpoint>endpoint21</Endpoint>
        </Distribution>
    </DistributionList>
</MessageDistribution>
```

表C.109 定義項目

キー名	説明	指定例
MessageDistribution		
name	ルーティング定義名を指定します。ユーザが任意に指定できます。	soap-distribution
routing	ルーティング種別を指定します。 ・ INBOUND 受信メッセージを各インバウンドサービスエンドポイントへルーティングするための定義です。“inboundtype”属性で指定するサービスで使用します。 ・ OUTBOUND 送信メッセージを各アウトバウンドサービスエンドポイントへルーティングするための定義です。メッセージ送信メディエータファンクションで使用します。 ・ SEQUENCE インバウンドサービスエンドポイントから各シーケンスへルーティングするための定義です。	INBOUND
inboundtype	ルーティング種別が“INBOUND”の場合、接続方式を指定します。以下のどれかを指定します。 ・ SOAP-MSG SOAP接続でのルーティングを行います。 ・ CORBA EE CORBA接続でのルーティングを行います。 ・ MQ-MSG EE MQ連携でのルーティングを行います。 ・ QUECONNECTED EE キュー間転送でのルーティングを行います。 ・ MQD S EE MQD接続でのルーティングを行います。 ・ COMMON カスタムアダプタ接続でのルーティングを行います。	SOAP-MSG
outboundtype	ルーティング種別が“OUTBOUND”のときに同期か非同期かを指定します。指定しない場合は互換定義になります。	SYNC

キー名		説明	指定例
		・ SYNC 同期サービスエンドポイントのルーティングを行います。 ・ ASYNC 非同期サービスエンドポイントのルーティングを行います。	
Description		ルーティング定義についての説明を指定します。	受信用振分け定義
Parameter			
1	条件に合うかどうかを判断するデータの位置を指定します。		
	name	パラメタ名を指定します。ユーザが任意に指定できます。	bookid
	type	データのタイプを指定します。	xml
	値	データの位置をXPathで指定します。	/booklist/book/@bookid
2	条件に合うかどうかを判断するデータの位置を指定します。		
	name	パラメタ名を指定します。ユーザが任意に指定できます。	price
	type	データのタイプを指定します。	xml
	値	データの位置をXPathで指定します。	/booklist/book/price
3	条件に合うかどうかを判断するデータの位置を指定します。		
	name	パラメタ名を指定します。ユーザが任意に指定できます。	division
	type	データのタイプを指定します。	flat
	code	使用している文字コードを指定します。	Shift-jis
	値	データの位置を“開始位置,けた数”で指定します。	0,3
DistributionList			
Distribution			
1	Condition		
	データを振り分ける条件を指定します。		
	値	条件式を指定します。	$bookid @EQUAL@ "B01"
	Endpoint		
	振分け先のサービスエンドポイント名を指定します。		
	値	振分け先のサービスエンドポイント名を指定します。	endpoint01
2	Condition		
	データを振り分ける条件を指定します。		
	値	条件式を指定します。	$bookid @EQUAL@ "B02" && $price < 1000
	Endpoint		
	振分け先のサービスエンドポイント名を指定します。		
	値	振分け先のサービスエンドポイント名を指定します。	endpoint02
	Condition		
	データを振り分ける条件を指定します。		
	値	条件式を指定します。	$bookid @EQUAL@ "B02" && $price >= 1000
	Endpoint		
	振分け先のサービスエンドポイント名を指定します。		

キー名		説明	指定例
	値	振分け先のサービスエンドポイント名を指定します。	endpoint03
3	Condition		
	データを振り分ける条件を指定します。		
	値	条件式を指定します。	$division @EQUAL@ "999"
	Endpoint		
	振分け先のサービスエンドポイント名を指定します。		
	値	振分け先のサービスエンドポイント名を指定します。	endpoint11
4	Condition		
	データを振り分ける条件を指定します。		
	値	条件式を指定します。	*
	Endpoint		
	振分け先のサービスエンドポイント名を指定します。		
	値	振分け先のサービスエンドポイント名を指定します。	endpoint21

付録D　互換機能

ここでは、ISIの互換機能についてサンプルプログラムを交えて説明します。

ポイント

S L

ISIクライアント(Windows)を対象に例を記述しています。ISIサーバ(Solaris、またはLinux)をクライアントとして利用する場合は、利用する環境に合わせて記述されているファイルパスを読み替えてください。

D.1　サービス提供側のアプリケーションの設計

サービス提供側のアプリケーションの開発について説明します。

D.1.1　非同期通信を行うアプリケーション(JMSアプリケーション)

ISIのシーケンスから外部JMSを介して、非同期通信を行うアプリケーションにメッセージを送信します。
非同期通信を行うアプリケーション(JMSアプリケーション)の設計は、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”の“JMS編”を参照してください。

D.1.2　MQDを使用するアプリケーション S EE

ISIのシーケンスからMQDを使用するアプリケーションへメッセージを送信します。
MQDを使用するアプリケーションの設計は、“MessageQueueDirector説明書”を参照してください。

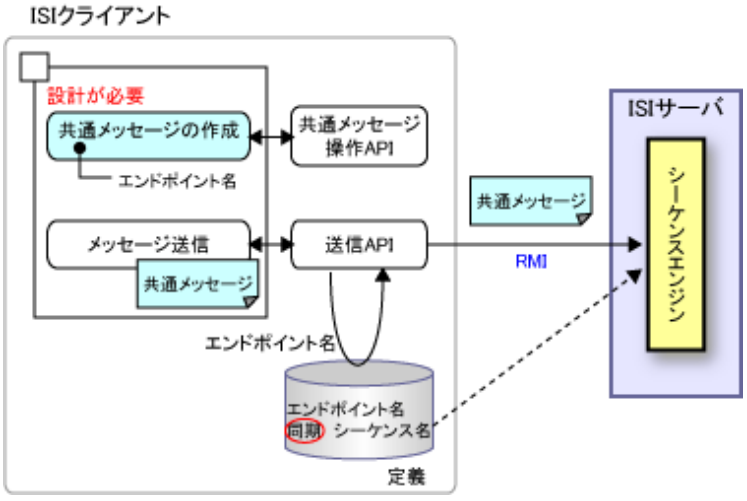
D.2　サービス利用側のアプリケーションの設計（共通APIを使用する場合）

サービス利用側のアプリケーションの開発について説明します。

D.2.1　RMIの利用

共通メッセージ操作APIを使用して、共通メッセージを作成します。作成した共通メッセージを、メッセージ送受信APIのパラメタとしてセットし、ISIと通信します。送信APIは、定義の内容を参照して、同期(RMI)と非同期(JMS)を切り替えます。
共通メッセージに設定する内容は、事前に確認してください。

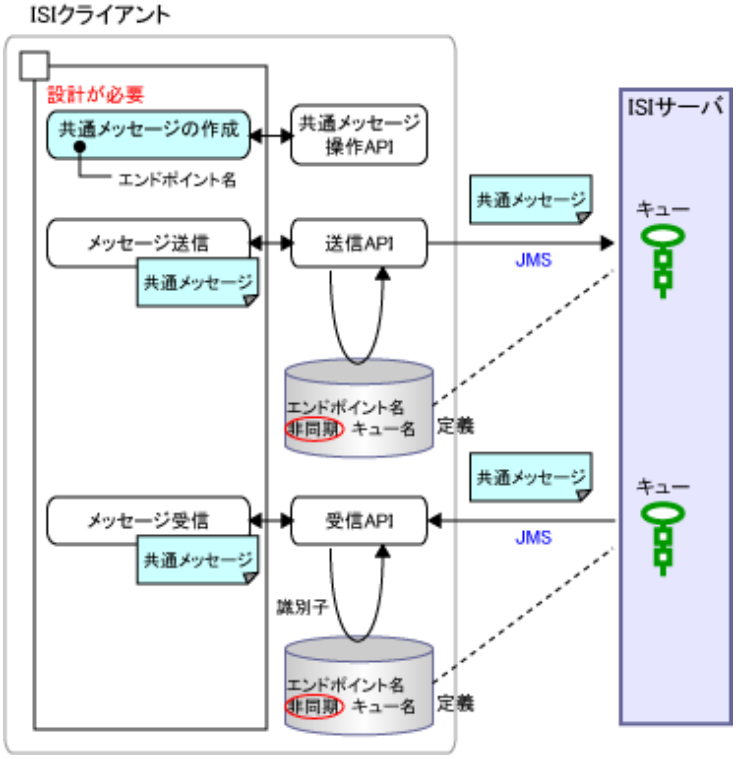
図D.1 RMI利用時のイメージ



D.2.2 JMSの利用

共通メッセージ操作APIを使用して、共通メッセージを作成します。作成した共通メッセージを、メッセージ送受信APIの送信APIのパラメタとしてセットし、ISIと通信します。送信APIは、定義の内容を参照して、JMSとJAX-RPCを切り替えます。
結果を受信する場合は、受信APIを使用してください。
共通メッセージに設定する内容は、事前に確認してください。

図D.2 JMS利用時のイメージ



D.2.3 SOAPのRPC機能の利用

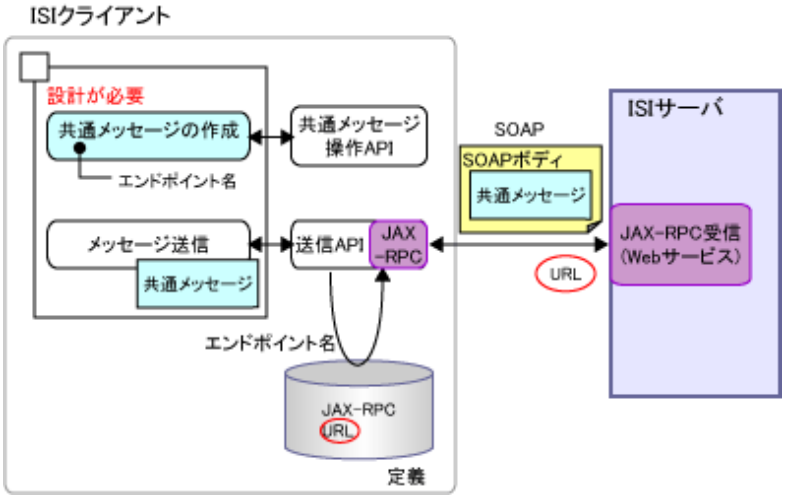
SOAPのRPC機能(JAX-RPC)を利用したアプリケーションを作成する場合、以下の方法があります。

- 共通APIを使用する方法
- スタブを使用する方法

共通APIを使用する方法

共通メッセージ操作APIを使用して、共通メッセージを作成します。作成した共通メッセージを、メッセージ送受信APIのパラメタとしてセットし、ISIと通信します。共通メッセージは、SOAPメッセージのSOAPボディに格納されます。
共通メッセージに設定する内容とJAX-RPCのURLは、事前に確認してください。

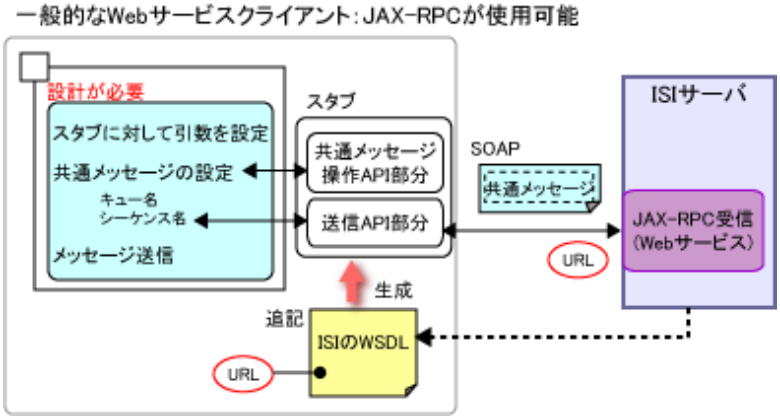
図D.3 共通API使用時のイメージ



スタブを使用する方法

アプリケーション開発時は、スタブに対して必要な項目を設定します。スタブは、ISIが準備したWSDLから生成します。
設計に必要な項目は、共通メッセージに設定するシーケンス名などです。また、ISIサーバのJAX-RPCのURLを確認します。URLは、WSDLファイルの“soap:address要素のlocation属性”に設定されています。
スタブ(共通メッセージ)に設定する内容およびURLは、事前に確認してください。

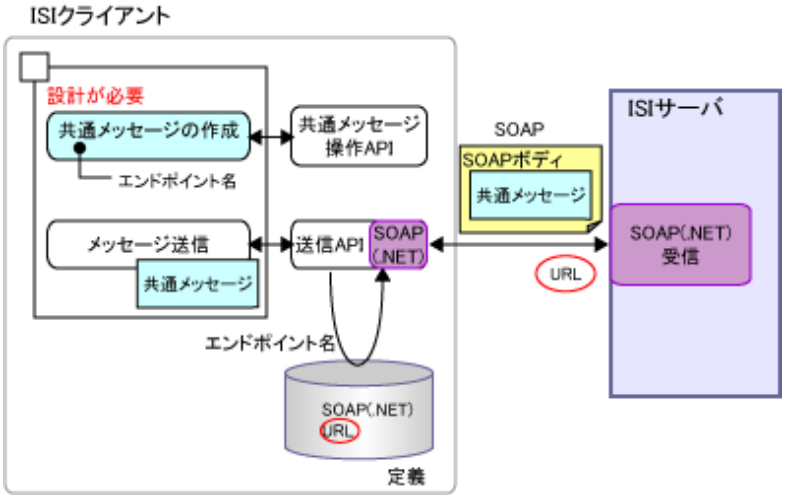
図D.4 スタブ使用時のイメージ



D.2.4 SOAP(.NET)の利用

共通メッセージ操作APIを使用して、共通メッセージを作成します。作成した共通メッセージを、メッセージ送受信APIのパラメタとしてセットし、ISIと通信します。共通メッセージは、SOAPメッセージのSOAPボディに格納されます。
共通メッセージに設定する内容とSOAP(.NET)のURLは、事前に確認してください。

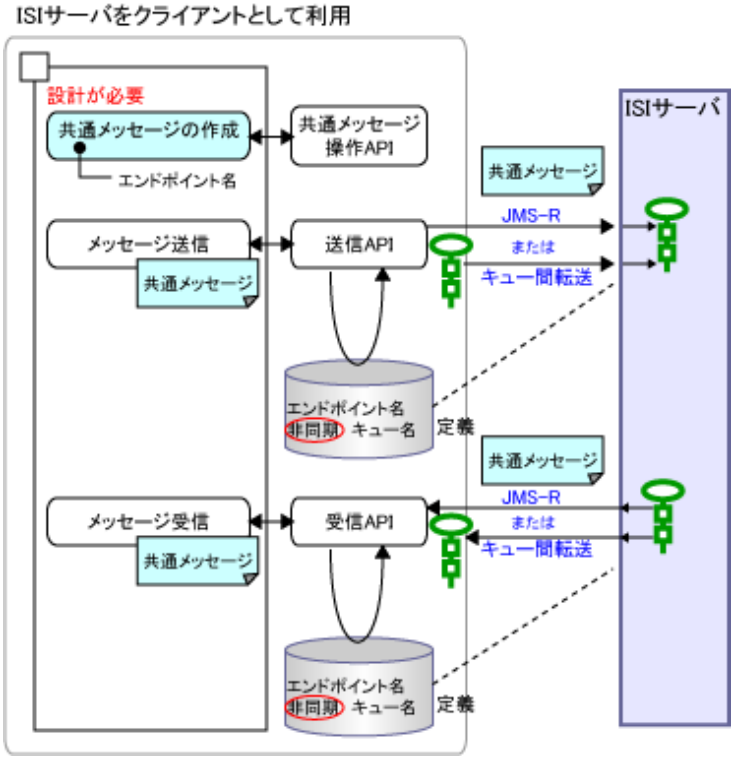
図D.5 SOAP(.NET)利用時のイメージ



D.2.5 JMS-Rの利用 EE

共通メッセージ操作APIを使用して、共通メッセージを作成します。作成した共通メッセージを、メッセージ送受信APIの送信APIのパラメタとしてセットし、ISIと通信します。送信APIは、定義の内容を参照して、JMS-Rと判断し、キューにメッセージを格納します。
結果を受信する場合は、受信APIを使用してください。
共通メッセージに設定する内容は、事前に確認してください。

図D.6 JMS-R利用時のイメージ



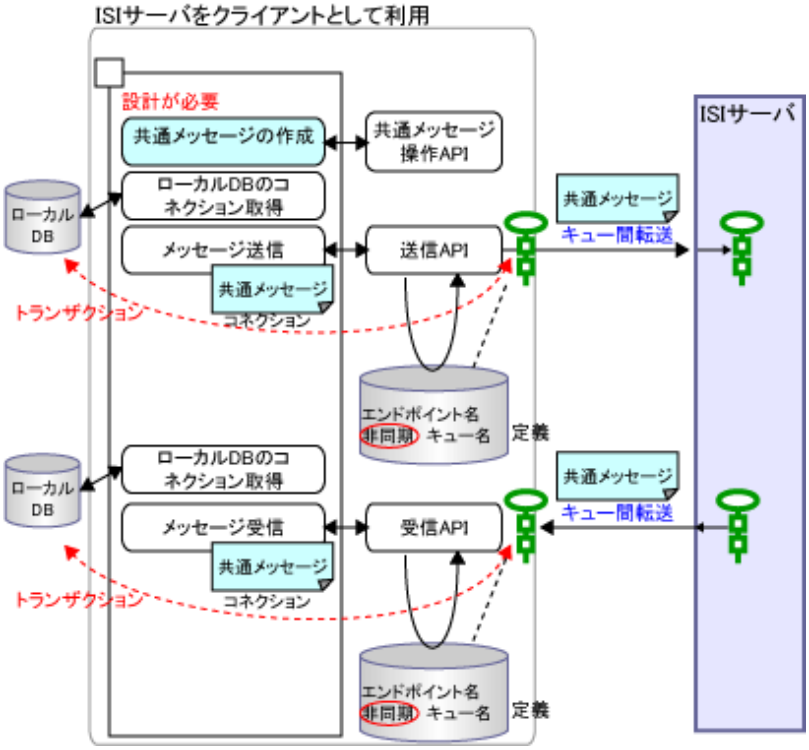
以下では、送信API、受信APIとも作成するアプリケーションは同じです。

- ISIサーバのJMS-Rキューに書き込む場合
- ISIサーバをクライアントとして利用しているサーバのJMS-Rキューに書込み、キュー間転送する場合

D.2.6　ローカルDBとのトランザクション機能の利用

共通メッセージ操作APIを使用して、共通メッセージを作成します。また、事前にローカルDBのコネクションを取得しておきます。
共通メッセージとコネクションをローカルDBとのトランザクション用送信APIのパラメタとしてセットし、ISIと通信します。APIは定義の内容を参照して、キュー間転送と判断し、キューにメッセージを格納します。
結果を受信する場合は、ローカルDBとのトランザクション用受信APIを使用してください。事前にローカルDBのコネクションを取得しておきます。
共通メッセージに設定する内容は、事前に確認してください。

図D.7 ローカルDBとのトランザクション利用時のイメージ



注意

本製品に同梱されるSymfoware/RDBは業務用のデータベースとして使用できません。
ローカルDBとのトランザクション機能で業務用のデータベースを作成する場合は、製品版のSymfoware Serverを使用してください。

D.2.7　MQDの利用 S EE

ISIのシーケンスにMQDを使用するアプリケーションからメッセージを送信します。
MQDを使用するアプリケーションの設計は、“MessageQueueDirector説明書”を参照してください。

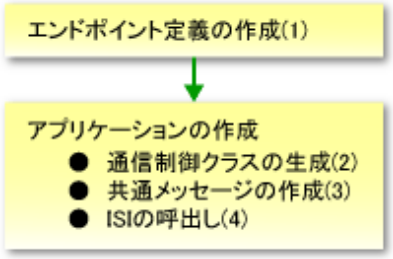
D.3　共通APIを利用したサービス利用側アプリケーションの開発

共通メッセージ操作APIを利用し共通メッセージを作成・操作するアプリケーションの開発について説明します。

D.3.1　RMIの利用

例をもとに、RMIを利用したアプリケーションの開発手順を説明します。
アプリケーションの開発は、以下の手順で行います。

図D.8 開発の手順



前提

サービスエンドポイント名：endpointRMI

手順

1. サービスエンドポイント定義を作成します。
   メッセージ送受信APIを利用するためのサービスエンドポイント定義にサービスエンドポイント名“endpointRMI”を定義します。
   作成方法については、“ISI Studio ヘルプ”を参照してください。
2. 手順1で作成したサービスエンドポイント名をパラメタに設定し、通信制御クラス(ESIService)を生成します。
3. 共通メッセージ(ESIMessage)を作成します。
   共通メッセージの作成方法については、“D.5 共通メッセージの作成・操作”を参照してください。
4. ISIを呼び出し、共通メッセージを送受信します。

アプリケーションの例

```
import com.fujitsu.esi.client.ESIAPIException;
import com.fujitsu.esi.client.ESIService;
import com.fujitsu.esi.client.ESIServiceFactory;
import com.fujitsu.esi.message.ESIMessage;

public class ESISend {
    public static void main(String[] args) {
        try {
            //(2)通信制御クラス(ESIService)を生成
            ESIService service = ESIServiceFactory.getService("endpointRMI");
            //(3)ESIMessageを作成
            ESIMessage message = new ESIMessage();
                        :
            //(4)ISIを呼び出す
            ESIMessage returnMessage = service.send(message); ……(a)
                        :
        } catch (ESIAPIException e) {
            e.printStackTrace();
        }
    }
}
```

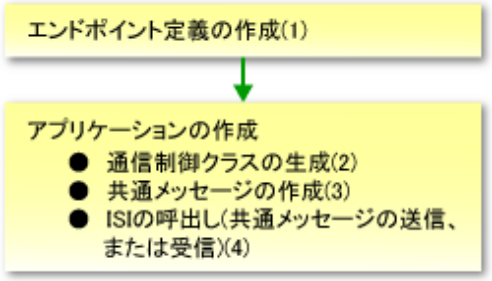
a. メッセージ送受信APIの同期APIとして提供しているメソッドは、以下のとおりです。

   ー 同期送信呼出し：send(ESIMessage)

D.3.2 JMS,JMS-Rの利用

例をもとに、JMSを利用したアプリケーションの開発手順を説明します。
アプリケーションの開発は、以下の手順で行います。

図D.9 開発の手順



前提

サービスエンドポイント名：endpointJMSIN、endpointJMSOUT

手順

1. サービスエンドポイント定義を作成します。
   メッセージ送受信APIを利用するためのサービスエンドポイント定義にサービスエンドポイント名“endpointJMSIN”と“endpointJMSOUT”を定義します。
   作成方法については、“ISI Studio ヘルプ”を参照してください。
2. 手順1で作成したサービスエンドポイント名をパラメタに設定し、通信制御クラス(ESIService)を生成します。
3. 共通メッセージ(ESIMessage)を作成します。
   共通メッセージの作成方法については、“D.5 共通メッセージの作成・操作”を参照してください。
4. ISIを呼び出し、共通メッセージを送信、または受信します。

アプリケーションの例(送信)

```
import com.fujitsu.esi.client.ESIAPIException;
import com.fujitsu.esi.client.ESIService;
import com.fujitsu.esi.client.ESIServiceFactory;
import com.fujitsu.esi.message.ESIHeaderConstants;
import com.fujitsu.esi.message.ESIMessage;
import com.fujitsu.esi.message.ESIParameter;

public class ESISend {
    public static void main(String[] args) {
        try {
            //(2)通信制御クラス(ESIService)を生成
            ESIService service = ESIServiceFactory.getService("endpointJMSIN");
            //(3)ESIMessageを作成
            ESIMessage message = new ESIMessage();
            //JMS用パラメタの設定
            ESIParameter parameter = message.getHeader().getParameter();
            //JMSCorrelationIDの設定
            parameter.put(ESIHeaderConstants.ESI_HEADER_KEY_JMS_HEADER_JMS_CORRELATION_ID, "ID_0123456789");
            //JMS Stringプロパティ “category” の設定
            parameter.put(ESIHeaderConstants.ESI_HEADER_KEYPREFIX_JMS_STRING + "category", "A01");
                        :
            //(4)ISIを呼び出す(共通メッセージの送信)
            ESIMessage returnMessage = service.send(message); ……(a)
                        :
        } catch (ESIAPIException e) {
            e.printStackTrace();
        }
    }
}
```

アプリケーションの例(受信)

```
import com.fujitsu.esi.client.ESIAPIException;
import com.fujitsu.esi.client.ESIClientUtil;
import com.fujitsu.esi.client.ESIFilter;
import com.fujitsu.esi.client.ESIService;
import com.fujitsu.esi.client.ESIServiceFactory;
import com.fujitsu.esi.message.ESIMessage;

public class ESIReceive {
    public static void main(String[] args) {
        try {
            //(2)通信制御クラス(ESIService)を生成
            ESIService service = ESIServiceFactory.getService("endpointJMSOUT");
            //フィルタの設定
            ESIFilter filter =
       ESIClientUtil.createJMSReceiveEqualFilter("JMSCorrelationID", "ID_0123456789"); ……(b)
                        :
            //(4)ISIを呼び出す(共通メッセージの受信)
            ESIMessage returnMessage = service.receive(filter); ……(a)
                        :
        } catch (ESIAPIException e) {
            e.printStackTrace();
        }
    }
}
```

a. メッセージ送受信APIの送信API、受信APIとして提供しているメソッドは、以下のとおりです。

- 非同期送信呼出し:send(ESIMessage)
- 非同期受信呼出し(フィルタ付き):receive(ESIFilter)
- 非同期受信呼出し(フィルタなし):receive()

b. 送信で指定したJMS Stringプロパティ“category”で検索する場合は、以下のようになります。

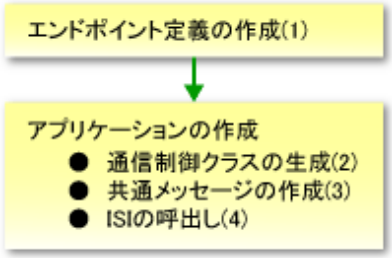
```
ESIFilter filter =
      ESIClientUtil.createJMSReceiveEqualFilter("category", "A01");
```

ESIFilterにクエリを設定しない場合は、キューの先頭に格納されている共通メッセージを受信します。

D.3.3 SOAPのRPC機能(JAX-RPC)の利用

例をもとに、SOAPのRPC機能(JAX-RPC)を、共通APIで利用するアプリケーションの開発手順を説明します。
アプリケーションの開発は、以下の手順で行います。

図D.10 開発の手順



前提

サービスエンドポイント名:endpointRPC

手順

1. サービスエンドポイント定義を作成します。
   SOAPのRPC機能を利用するためのサービスエンドポイント定義にサービスエンドポイント名“endpointRPC”を定義します。
   作成方法については、“ISI Studio ヘルプ”を参照してください。
2. 手順1で作成したサービスエンドポイント名をパラメタに設定し、通信制御クラス(ESIService)を生成します。
3. 共通メッセージ(ESIMessage)を作成します。
   共通メッセージの作成方法については、“D.5 共通メッセージの作成・操作”を参照してください。
4. ISIを呼び出します。

アプリケーションの例

```
import com.fujitsu.esi.client.ESIAPIException;
import com.fujitsu.esi.client.ESIService;
import com.fujitsu.esi.client.ESIServiceFactory;
import com.fujitsu.esi.message.ESIMessage;

public class ESISend {

    public static void main(String[] args) {
        try {
          //(2)通信制御クラスを生成
          ESIService service = ESIServiceFactory.getService("endpointRPC");

          //(3)ESIMessageを作成
          ESIMessage message= new ESIMessage();
                :
          //(4)ISIを呼び出す
          ESIMessage returnMessage = service.send(message);……(a)
                :
          } catch (ESIAPIException e) {
                    e.printStackTrace();
          }
        }
}
```

a. ISIの呼出しとして提供しているメソッドは、以下のとおりです。

   ー 同期/非同期送信呼出し:send(ESIMessage)
   同期、非同期の設定は手順1のサービスエンドポイント定義で指定してください。

   ー 非同期受信呼出し:receive(ESIFilter)

注意

非同期受信メソッド“receive(ESIFilter)”を使用する場合は、引数としてESIFilterが必要です。ESIFilterにクエリを設定しない場合は、キューの先頭に格納されている共通メッセージを受信します。

ポイント

同期送信アプリケーションの作成方法は、RMI利用時と同じです。詳細については、“D.3.1 RMIの利用”を参照してください。非同期送信・受信アプリケーションの作成方法は、JMS利用時と同じです。詳細については、“D.3.2 JMS,JMS-Rの利用”を参照してください。
RMIの利用・JMSの利用/JAX-RPCの利用は、サービスエンドポイント定義のサービス情報で切り替えます。

■Proxy、Basic認証、接続タイムアウト時間の設定

プロキシ情報、Basic認証、接続タイムアウト時間の設定は、サービスエンドポイント定義で設定します。設定方法は、“ISI Studio ヘルプ”を参照してください。

■SSLの設定

◆Interstage Application Serverを利用した場合

クライアント認証なしの場合、以下の手順で設定します。

1. 動作環境の設定
2. 証明書環境の構築
    a. 証明書の登録・管理用ファイルを配置するディレクトリの作成
    b. 証明書管理ファイルの作成
    c. サーバのサイト証明書を登録している認証局からの認証局証明書取得
    d. 認証局証明書の登録
    e. 証明書管理ファイルの作成
3. クライアントアプリケーションの設定

クライアント認証ありの場合以下の手順で設定します。

1. 動作環境の設定
2. 証明書環境の構築
    a. 証明書の登録・管理用ファイルを配置するディレクトリの作成
    b. 証明書管理ファイルの作成
    c. 証明書発行要求の作成
    d. 証明書の発行依頼
    e. 認証局から証明書の取得
    f. 認証局から発行された証明書の登録
    g. 認証局から発行されたサイト証明書の登録
3. クライアントアプリケーションの設定

SSLの設定方法の詳細については、“Interstage Application Server セキュリティシステム運用ガイド”を参照してください。

◆AXISを利用した場合

SSL情報は、アプリケーション実行時のJavaVMオプションからシステムプロパティに設定します。

例:ESISend.java

```
# java
-Djavax.net.ssl.trustStore=C:¥cert¥truststore¥keystore
-Djavax.net.ssl.trustStorePassword=trustpass
-Djavax.net.ssl.trustStoreType=jks
-Djavax.net.ssl.keyStore=C:¥cert¥keystore¥keystore
-Djavax.net.ssl.keyStorePassword=keystorepass
-Djavax.net.ssl.keyStoreType=jks
: ……(注)
ESISend
```

注:コンパイル・実行時に必要な設定を確認してください。必要な設定については、“D.6 アプリケーションのコンパイルと実行”を参照してください。

表D.1 プロパティ値

プロパティ名	値
javax.net.ssl.trustStore	SSLサーバ認証用のキーストアのパス名 例)C:¥cert¥truststore¥keystore
javax.net.ssl.trustStorePassword	SSLサーバ認証用のキーストアのパスワード 例)trustpass
javax.net.ssl.trustStoreType	SSLサーバ認証用のキーストアのタイプ 例)jks
javax.net.ssl.keyStore	SSLクライアント認証用のキーストアのパス名 例)C:¥cert¥keystore¥keystore
javax.net.ssl.keyStorePassword	SSLクライアント認証用のキーストアのパスワード 例)keystorepass
javax.net.ssl.keyStoreType	SSLクライアント認証用のキーストアのタイプ 例)jks

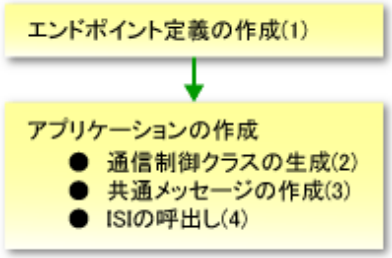
D.3.4 .NETの利用

例をもとに、.NET機能を利用したアプリケーションの開発手順を説明します。
アプリケーションの開発は、以下の手順で行います。

注意

- .NETメッセージ送受信APIでは、エンドポイントのISIサーバ名情報が無視され、すべてデフォルトのISIサーバが指定されたものとみなされます。
- .NETメッセージ送受信APIからルーティングを用いたエンドポイントは使用できません。

図D.11 開発の手順



前提

サービスエンドポイント名 : endpointNet

手順

1. サービスエンドポイント定義を作成します。
   メッセージ送受信APIを利用するためのサービスエンドポイント定義にサービスエンドポイント名“endpointNet”を定義します。
   作成方法については、“ISI Studio ヘルプ”を参照してください。
2. 手順1で作成したサービスエンドポイント名をパラメタに設定し、通信制御クラス(IESIService)を生成します。
3. 共通メッセージ(ESIMessage)を作成します。
   共通メッセージの作成方法については、“D.5.4 共通メッセージの作成・操作(.NET)”を参照してください。
4. ISIを呼び出し、共通メッセージを送受信します。

C#アプリケーションの例

```
using System;
using Com.Fujitsu.Esi.Client;
using Com.Fujitsu.Esi.Message;

namespace sample
{
    class ESISend
    {
        static void Main(string[] args)
        {
            try {
                //(2)通信制御クラス(IESIService)を生成
                IESIService service = ESIServiceFactory.GetService("endpointNet");
                //(3)ESIMessageを作成
                ESIMessage message = new ESIMessage();

                //(4)ISIを呼び出す
                ESIMessage returnMessage = service.Send(message);
            } catch (ESIAPIException e) {
                Console.WriteLine(e.StackTrace);
            }
        }
    }
}
```

a. メッセージ送受信APIの同期APIとして提供しているメソッドは、以下のとおりです。

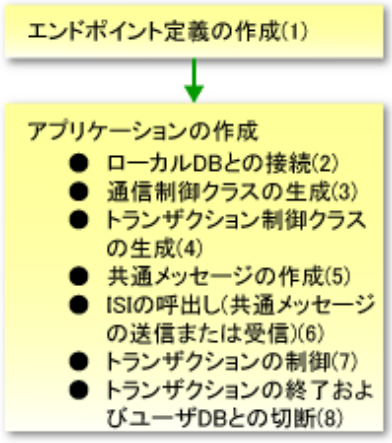
- 同期送信呼出し:Send(ESIMessage)

D.3.5 ローカルDBとのトランザクション機能の利用

例をもとに、ローカルDBとのトランザクション機能を利用したアプリケーションの開発手順を説明します。

アプリケーションの開発は、以下の手順で行います。

図D.12 開発の手順



前提

サービスエンドポイント名:endpointJMSRIN、endpointJMSROUT
DBの種類:Symfoware Server
DBの接続先:jdbc:symford://isiserv:2050/ESIMESSAGEDB

DBのユーザID:UID
DBのパスワード:PWD

手順

1. サービスエンドポイント定義を作成します。
   メッセージ送受信APIを利用するためのサービスエンドポイント定義にサービスエンドポイント名“endpointJMSRIN”と“endpointJMSROUT”を定義します。
   作成方法については、“ISI Studio ヘルプ”を参照してください。
2. ローカルDBと接続します。
   ローカルDBと接続し、自動コミットをオフにします。
   接続先のDBは、環境に合わせて変更してください。
3. 手順1で作成したサービスエンドポイント名をパラメタに設定し、通信制御クラス(ESIService)を生成します。
4. トランザクション制御クラス(ESILocalTransaction)を生成します。手順2で作成したローカルDBとの接続情報をパラメタに設定します
5. 共通メッセージ(ESIMessage)を作成します。
   共通メッセージの作成方法については、“D.5 共通メッセージの作成・操作”を参照してください。
6. ISIを呼出し、共通メッセージを送信、または受信します。
7. 送信または受信結果によってコミットまたはロールバックを行います。
8. トランザクションを終了し、ローカルDBから切断します。



ESILocalTransactionクラスのclose()メソッドでは、ローカルDBとのコネクションのクローズは行いません。別途、Connectionクラスのclose()メソッドを実行してください。

アプリケーションの例(送信)

```
import java.sql.Connection;
import java.sql.DriverManager;
import com.fujitsu.esi.client.ESIAPIException;
import com.fujitsu.esi.client.ESILocalTransaction;
import com.fujitsu.esi.client.ESIService;
import com.fujitsu.esi.client.ESIServiceFactory;
import com.fujitsu.esi.message.ESIHeaderConstants;
import com.fujitsu.esi.message.ESIMessage;

public class ESISend {
    public static void main(String[] args) {

        ESILocalTransaction localTran = null;
        Connection con = null;
        try {
            // (2)ローカルDBとの接続
            // JDBCドライバをロードします。
            Class.forName("com.fujitsu.symfoware.jdbc.SYMDriver");
            // DBと接続します。
            con = DriverManager.getConnection(
                        "jdbc:symford://isiserv:2050/ESIMESSAGEDB","UID", "PWD");
            // 自動コミットのオフ
            con.setAutoCommit(false);

            // (3)通信制御クラス(ESIService)を生成
            ESIService service = ESIServiceFactory.getService("endpointJMSRIN");

            // (4)トランザクション制御クラスを生成
            localTran = service.createLocalTransaction(con);

            // (5)ESIMessageを作成
```

```
            ESIMessage message = new ESIMessage();
                                :
            // (6)ISIを呼び出す(共通メッセージの送信)
            ESIMessage returnMessage = service.send(message, localTran); ……(a)
                                :
            // (7)送信結果に合わせてトランザクションの制御を行う
            String status = returnMessage.getHeader().getParameter().
                                                    get("com.fujitsu.esi.status");
            if ( status.equals("SUCCESS") ) {
                 // 送信が成功
                 // SQL文の発行を行う。
                                :
                 // SQL文の発行まで、正常に終了したのでコミット
                 localTran.commit();
            }else{
                 // エラーが発生したのでロールバック
                 localTran.rollback();
            }
                                :
        } catch (Exception e) {
            // (SQL発行を含む)エラーが発生したのでロールバック
            try {
               if (localTran != null) {
                  localTran.rollback();
               }
            }catch(ESIAPIException apie){
               apie.printStackTrace();
            }
            e.printStackTrace();
        }finally{
            // (8)トランザクション終了およびDBとの切断
            try{
               if ( localTran != null ) {
                  localTran.close();
               }
               if ( con != null ) {
                  con.close();
               }
            } catch (Exception e) {
               e.printStackTrace();
            }
        }
    }
}
```

アプリケーションの例(受信)

```
import java.sql.Connection;
import java.sql.DriverManager;
import com.fujitsu.esi.client.ESIAPIException;
import com.fujitsu.esi.client.ESILocalTransaction;
import com.fujitsu.esi.client.ESIService;
import com.fujitsu.esi.client.ESIServiceFactory;
import com.fujitsu.esi.message.ESIHeaderConstants;
import com.fujitsu.esi.message.ESIMessage;

public class ESIReceive {
    public static void main(String[] args) {

        ESILocalTransaction localTran = null;
        Connection con = null;
        try {
```

```
            // (2) ローカルDBとの接続
            // JDBCドライバをロードします。
            Class.forName("com.fujitsu.symfoware.jdbc.SYMDriver");
            // データベースと接続します。
            con = DriverManager.getConnection(
                        "jdbc:symford://isiserv:2050/ESIMESSAGEDB","UID", "PWD");
            // 自動コミットのオフ
            con.setAutoCommit(false);

            // (3)通信制御クラス(ESIService)を生成
            ESIService service = ESIServiceFactory.getService("endpointJMSROUT");

            // (4)トランザクション制御クラスを生成
            localTran = service.createLocalTransaction(con);

            // (6)ISIを呼び出す(共通メッセージの受信)
            ESIMessage returnMessage = service.receive(localTran); ……(a)
                              :
            // (7)受信結果に沿ってトランザクションの制御を行う
            if ( returnMessage != null ) {
                 // 受信メッセージをもとにローカルDBに対してSQLを発行する。
                              :
             }
             // 処理が終了したのでコミットする。
             localTran.commit();

        } catch (Exception e) {
            // (SQL発行を含む)エラーが発生したのでロールバック
            try {
               if (localTran != null) {
                  localTran.rollback();
               }
            }catch(ESIAPIException apie){
               apie.printStackTrace();
            }
            e.printStackTrace();
        }finally{
            // (8)トランザクション終了およびDBとの切断
            try{
               if ( localTran != null ) {
                  localTran.close();
               }
               if ( con != null ) {
                  con.close();
               }
            } catch (Exception e) {
               e.printStackTrace();
            }
        }
      }
}
```

ローカルDBとのトランザクション機能を利用するためのメッセージ送受信APIの送信API、受信APIとして提供しているメソッドは、以下のとおりです。

非同期送信呼出し:send(ESIMessage,ESILocalTransaction)

非同期受信呼出し(フィルタ付き):receive(ESIFilter,ESILocalTransaction)

非同期受信呼出し(フィルタなし):receive(ESILocalTransaction)

D.4 スタブを利用したサービス利用側アプリケーションの開発

SOAPのRPC機能(JAX-RPC)をスタブで利用するアプリケーションの開発について説明します。
スタブを利用する場合、ISIクライアントがインストールされていない環境からでもISIサービスを利用することができます。

D.4.1 Interstage Application Server

例をもとに、Interstage Application Server利用時にSOAPのRPC機能(JAX-RPC)を、スタブで利用するアプリケーションの開発手順を説明します。
アプリケーションの開発は、以下の手順で行います。

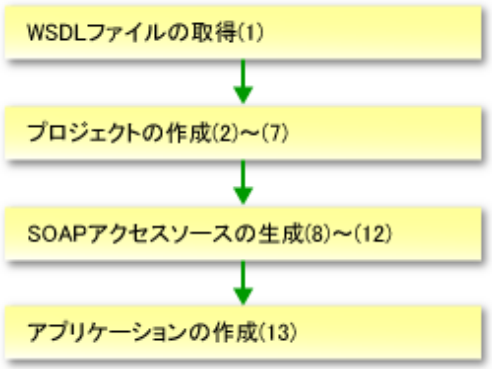
注意

スタブを作成するためには、Interstage Studioが利用できる環境が必要となります。
Windowsシステム上でワークベンチを使って、アプリケーションを開発します。

S L

開発したアプリケーションの実行資源を、実行環境であるSolaris、またはLinuxシステムに配備し、運用してください。詳細は、“Interstage Studio ユーザーズガイド”を参照してください。

図:開発の手順



手順

1. アクセス先のISIサーバで用意されているWSDLファイル(ESIService.wsdl)を取得し、任意のディレクトリに格納します。

   例)“C:¥WSDL¥ESIService.wsdl”

   ### 注意

   ISIのWSDLファイルの取得方法は、ISIサーバの運用管理者に問い合わせてください。

2. ワークベンチを起動し、メニューから[ファイル]－[新規]－[プロジェクト]を選択します。
   [新規プロジェクト]ウィザードが表示されます。

3. [Javaアプリケーションプロジェクト]を選択し、[次へ]ボタンをクリックします。
   [新規Javaアプリケーションプロジェクト]画面が表示されます。

4. [プロジェクト名]に任意のプロジェクト名(例:“ESIClientSampleProject”)を指定し、[内容]に[ワークスペース内に新規プロジェクトを作成]をチェックします。
   [JRE]に[デフォルトJREの使用]にチェックし、[終了]ボタンをクリックします。
   [関連付けられたパースペクティブを開きますか?]ダイアログが表示されます。
   [いいえ]を選択します。初期状態が、J2EEパースペクティブ以外の場合は、J2EEパースペクティブに変更してください。

5. Webサービスクライアントを開発するためのクラスパスを設定するため、作成したプロジェクト(ESIClientSampleProject)を選択して、右クリックでコンテキストメニューを表示し、[ビルドパス]－[ライブラリの追加]を選択します。
[ライブラリの追加]ウィザードが表示されます。

6. [Interstage J2EEライブラリ]を選択し、[次へ]ボタンをクリックします。

7. 表示された画面で[J2EEアプリケーションからWebサービスを呼び出す]をチェックし、[終了]ボタンをクリックします。

8. メニューから[ファイル]－[新規]－[Webサービスクライアント]を選択します。
J2EEパースペクティブ以外の場合には、[ファイル]－[新規]－[その他]を選択し、新規ウィザードから、[J2EE]－[Webサービスクライアント]を選択します。
[Webサービスクライアント生成]ウィザードが表示されます。

9. 以下を指定し、[終了]ボタンをクリックします。

   － ソースディレクトリ:ESIClientSampleProject/src

   － WSDLファイル:手順1で取得したWSDLファイル“C:¥WSDL¥ESIService.wsdl”

   － ユーザ定義クラスを生成する:(チェック)

10. メニューから[ファイル]－[新規]－[クラス]を選択します。
[Javaクラス]ウィザードが表示されます。

11. 以下を指定し、[終了]ボタンをクリックします。

   － ソースディレクトリ:ESIClientSampleProject/src

   － パッケージ:sample

   － エンクロージング型:チェックしない

   － 名前:ESIClientSample

   － 修飾子:public

   － スーパークラス:java.lang.Object

   － インタフェース:指定しない

   － 作成するメソッドスタブの選択:[public static void main(String[] args)]をチェック、[継承された抽象メソッド]をチェック

   － 現在のプロジェクトのプロパティで構成したとおりコメントを追加:チェックしない

12. 以下のファイルが生成されたことを確認します。

   － “sample¥ESIClientSample.java”

   － “ESIServer¥_isws_ESIServerPortSoapBindingStub.java”

   － “ESIServer¥_isws_ESIServerServiceLocator.java”

   － “ESIServer¥ESIServerPortType.java”

   － “ESIServer¥ESIServerService.java”

   － “com¥fujitsu¥esi¥client¥_isws_ESIJMSReceiveFilter_Helper.java”

   － “com¥fujitsu¥esi¥client¥ESIJMSReceiveFilter.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIAttachment_Helper.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIBytePayload_Helper.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIFunction_Helper.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIHeader_Helper.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIHistoryInfo_Helper.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIMessage_Helper.java”

   － “com¥fujitsu¥esi¥message¥_isws_ESIParameter_Helper.java”

- “com¥fujitsu¥esi¥message¥_isws_ESIParameterItem_Helper.java”
- “com¥fujitsu¥esi¥message¥_isws_ESIPayload_Helper.java”
- “com¥fujitsu¥esi¥message¥_isws_ESISequence_Helper.java”
- “com¥fujitsu¥esi¥message¥_isws_ESISequenceLogInfo_Helper.java”
- “com¥fujitsu¥esi¥message¥ESIAttachment.java”
- “com¥fujitsu¥esi¥message¥ESIBytePayload.java”
- “com¥fujitsu¥esi¥message¥ESIFunction.java”
- “com¥fujitsu¥esi¥message¥ESIHeader.java”
- “com¥fujitsu¥esi¥message¥ESIHistoryInfo.java”
- “com¥fujitsu¥esi¥message¥ESIMessage.java”
- “com¥fujitsu¥esi¥message¥ESIParameter.java”
- “com¥fujitsu¥esi¥message¥ESIParameterItem.java”
- “com¥fujitsu¥esi¥message¥ESIPayload.java”
- “com¥fujitsu¥esi¥message¥ESISequence.java”
- “com¥fujitsu¥esi¥message¥ESISequenceLogInfo.java”

13. アプリケーションを作成します。

アプリケーションの例：サンプルプログラム“ESIClientSample.java”を例にあげて説明します。



注意

ISI 8.0以降では、WSDLファイルが変更されています。ISI 7.0で提供しているWSDLファイルを用いてアプリケーションを作成している場合、アプリケーションを修正する必要があります。詳細は、“D.4.3 アプリケーションの移行”を参照してください。

以下のサンプルプログラム“ESIClientSample.java”の斜体部分は、同期送信サービスの呼出し例です。

```
package sample;

import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;

import javax.xml.rpc.ServiceFactory;
import com.fujitsu.esi.client.*;
import com.fujitsu.esi.message.*;
import ESIServer.ESIServerPortType;
import ESIServer.ESIServerService;

public class ESIClientSample{
      public ESIClientSample() {
            super();
      }

       public void run(String[] args) {
       //(13)アプリケーション実行時に実行されるコードを下に追加します
         try {
           ServiceFactory serviceFactory = ServiceFactory.newInstance();
           ESIServerService service
           = (ESIServerService) serviceFactory.loadService(ESIServerService.class);
           ESIServerPortType esiSv = service.getESIServerPort(); ……(a)

           ESIMessage reqmsg = new ESIMessage(); ……(b)
```

```
            ESIParameter param = reqmsg.getHeader().getParameter();
            //シーケンス名設定
            ESIParameterItem[] items = new ESIParameterItem[1];
            items[0] = new ESIParameterItem();
            items[0].setKey("com.fujitsu.esi.sequence.name");
            items[0].setValue("sample_sequence");

            param.setItems(items);
             :
            //Webサービスの呼出し
            ESIMessage rspmsg = esiSv.sendESIMsgSync(reqmsg ); ……(c)

            //戻り値ESIMessageの処理結果、シーケンス名取得
            String status = null;
            String sequenceName = null;
            ESIParameterItem[] resultItems
                  = rspmsg.getHeader().getParameter().getItems();

            for( int i= 0; i < resultItems.length; i++){
                if(resultItems[i].getKey().equals("com.fujitsu.esi.status") ){ ……(d)
                 status = resultItems[i].getValue();
               } else
                if( resultItems[i].getKey().equals("com.fujitsu.esi.sequence.name") ){
                sequenceName = resultItems[i].getValue();
                    break;
               }
            }
             :

        }
         catch (ServiceException e) {
              e.printStackTrace();
         } catch (RemoteException e) {
              e.printStackTrace();
         }
       }

        public static void main(String[] args) {
              ESIClientSample object = new ESIClientSample();
              object.run(args);
        }
}
```

a. 送信先WebサービスのURLの指定には、2通りの方法があります。

- 手順1で取得したWSDLファイルに記述されている送信先WebサービスのURLに送信する場合、パラメタなしでスタブオブジェクトを作成します。(サンプルプログラム“ESIClientSample.java”の方法)
- 手順1で取得したWSDLファイルに記述されている送信先WebサービスのURLとは異なるURLに送信する場合、スタブオブジェクトに対してプロパティで設定します。
  例:
  ESIServerPortType esiSv = service.getESIServerPort();
  ((Stub) esiSv)._setProperty("javax.xml.rpc.service.endpoint.address", "送信先URL");
  ※ javax.xml.rpc.Stubをimportする必要があります。

b. 注意事項を参照してください。

c. ISIがWebサービスとして提供しているメソッドは、以下のとおりです。

- 同期送信サービス:sendESIMsgSync(ESIMessage)
- 非同期送信サービス:sendESIMsgAsync(ESIMessage)
- 非同期受信サービス:receiveESIMsgAsync(ESIJMSReceiveFilter, Queue名)

ポイント

送信先URLには、ISIのJAX-RPC用のURLを指定します。
SOAPのRPC機能(JAX-RPC)の設定については、“ISI 導入ガイド”を参照してください。

注意

メッセージ送受信APIを利用した送信用メソッドは同一ですが、スタブを利用した送信用のメソッドは異なります。

各メソッドを呼び出す場合、共通メッセージ(ESIMessage)に設定が必要な項目を以下に示します。

表D.2 共通メッセージ(ESIMessage)に設定が必要な項目

メソッド名	キュー名	シーケンス名
sendESIMsgSync(ESIMessage)	―	○
sendESIMsgAsync(ESIMessage)	△	○

○:ESIMessageに設定必須
△:ESIMessageに設定可(省略時は、デフォルト値が有効となります。)
―:ESIMessageに設定不要(設定は無効となります。)

共通メッセージの作成方法については、“D.5 共通メッセージの作成・操作”を参照してください。

a. 以下のキー名を指定して値を取得すると、ISI呼出し結果情報やエラーが発生した原因情報を取得できます。

表D.3 キー名と取得情報

キー名	情報
com.fujitsu.esi.status	ISIの処理結果です。 SUCCESS 処理が正常終了した場合 FAILURE 処理中にエラーが発生した場合
com.fujitsu.esi.error.code	ISIの処理中に最初に発生したエラーのエラーコードです。
com.fujitsu.esi.error.stepname	エラーが発生したステップです。 ISIの初期処理中にエラーが発生した場合は、nullになります。
com.fujitsu.esi.error.methodtype	エラーが発生したメソッド種別です。 ISIの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。
com.fujitsu.esi.error.handle.status	ISI処理のエラー発生時に呼び出したシーケンスの処理結果です。 SUCCESS 処理が正常終了した場合 FAILURE 処理中にエラーが発生した場合
com.fujitsu.esi.error.handle.code	エラー発生時に呼び出したシーケンスの中で、最初に発生したエラーのエラーコードです。
com.fujitsu.esi.error.handle.stepname	エラー発生時に呼び出したシーケンスの中で、エラーが発生したステップです。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。

キー名	情報
com.fujitsu.esi.error.handle.methodtype	エラー発生時に呼び出したシーケンスの中で、エラーが発生したメソッド種別です。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。

注意事項

ESIMessageの要素を初期化するためには、以下の設定が必要です。

```
ESIMessage reqmsg = new ESIMessage();

ESIHeader header = new ESIHeader();
ESIParameter parameter = new ESIParameter();
ESIParameterItem[] items= new ESIParameterItem[0];
parameter.setItems(items);
header.setParameter(parameter);

ESISequence sequence = new ESISequence();
ESISequenceLogInfo[] sequenceLogList = new ESISequenceLogInfo[0];
sequence.setSequenceLogInfoList(sequenceLogList);
ESIFunction[] fuctionList = new ESIFunction[0];
sequence.setFunctionList(fuctionList);

ESIBytePayload objectPayload = new ESIBytePayload();

ESIPayload payload = new ESIPayload();
payload.setBytePayload(objectPayload);
ESIAttachment attachment = new ESIAttachment();
ESIBytePayload[] bytePayload = new ESIBytePayload[0];
attachment.setBytePayloadList(bytePayload);
payload.setAttachment(attachment);
header.setSequence(sequence);

reqmsg.setHeader(header);
reqmsg.setPayload(payload);
```

サンプルプログラム“ESIClientSample.java”で呼出し例をあげた同期送信サービス以外の各メソッドの呼出し例を、以下に示します。
サンプルプログラム“ESIClientSample.java”の斜体の部分を置き換えてください。

非同期送信サービス:sendESIMsgAsync(ESIMessage)

```
ESIMessage reqmsg = new ESIMessage();

ESIParameter param = reqmsg.getHeader().getParameter();
//キュー名、シーケンス名設定

ESIParameterItem[] items= new ESIParameterItem[2];
items[0] = new ESIParameterItem();
items[0].setKey("com.fujitsu.esi.queue.name");
items[0].setValue("sampleQueue");

items[1] = new ESIParameterItem();
items[1].setKey("com.fujitsu.esi.sequence.name");
items[1].setValue("sample_sequence");

param.setItems(items);
```

```
        :
//Webサービスの呼出し
ESIMessage rspmsg = esiSv.sendESIMsgAsync(reqmsg);

//戻り値ESIMessageのシーケンス名、キュー名取得
String sequenceName = null;
String queueName  = null;
ESIParameterItem[] resultItems = rspmsg.getHeader().getParameter().getItems();

for( int i= 0; i < resultItems.length; i++){
         if( resultItems[i].getKey().equals("com.fujitsu.esi.sequence.name") ){
                        sequenceName = resultItems[i].getValue();
         }else if( resultItems[i].getKey().equals("com.fujitsu.esi.queue.name") ){
                        queueName = resultItems[i].getValue();
         }
}
```

非同期受信サービス:receiveESIMsgAsync(ESIJMSReceiveFilter, Queue名)

```
ESIJMSReceiveFilter fltr= new ESIJMSReceiveFilter();
//クエリ設定
fltr.setQuery("sampleKey='SampleValue'"); ……(a)
//キュー名設定
String queue = "sampleQueue";

        :
//Webサービスの呼出し
ESIMessage rspmsg = esiSv.receiveESIMsgAsync(fltr , queue);

//戻り値ESIMessageのキュー名取得
String queueName = null;
ESIParameterItem[] resultItems = rspmsg.getHeader().getParameter().getItems();

for( int i= 0; i < resultItems.length; i++){
         if( resultItems[i].getKey().equals("com.fujitsu.esi.queue.name") ){
                         queueName = resultItems[i].getValue();
                         break;
          }
}
```

ESIJMSReceiveFilterにクエリを設定しない場合は、キューの先頭に格納されている共通メッセージを受信します。

■Basic認証、接続タイムアウト時間、セション管理の設定

Basic認証、接続タイムアウト時間、セション管理の設定は、スタブオブジェクトのプロパティで設定します。

例:サンプルプログラム"ESIClientSample.java"をもとに、各設定方法を以下に示します。

```
                    :
import javax.xml.rpc.Stub;

public void run(String[] args){
       //アプリケーション実行時に実行されるコードを下に追加します
         try {
           ServiceFactory serviceFactory = ServiceFactory.newInstance();
           ESIServerService service
           = (ESIServerService) serviceFactory.loadService(ESIServerService.class);
           ESIServerPortType esiSv = service.getESIServerPort;
            //Basic認証：ユーザ名、パスワード
            ((Stub) esiSv)._setProperty("javax.xml.rpc.security.auth.username", "username");
             ((Stub) esiSv)._setProperty("javax.xml.rpc.security.auth.password", "password");
```

```
//接続タイムアウト時間の設定
((Stub) esiSv)._setProperty("com.fujitsu.interstage.isws.client.connect.timeout", new Integer(6000));

//セション管理の設定
((Stub) esiServer)._setProperty("javax.xml.rpc.session.maintain", Boolean.TRUE);
        :
```

表D.4 プロパティ値

プロパティ名	値	備考
javax.xml.rpc.security.auth.username	(java.lang.String) Basic認証ユーザ名 例)username	
javax.xml.rpc.security.auth.password	(java.lang.String) Basic認証パスワード 例)password	
com.fujitsu.interstage.isws.client.connect.timeout	(java.lang.Integer) 接続先から応答がない場合、接続を切断するタイムアウト時間(ミリ秒)(※1) 例)60000	デフォルト値:600000(10分)
javax.xml.rpc.session.maintain	(java.lang.Boolean) 以下のどちらかを指定します。 Boolean.TRUE セション管理を利用する Boolean.FALSE セション管理を利用しない	デフォルト値: Boolean.FALSE

※1:タイムアウト時間がISIサーバの処理時間より短い場合、接続タイムアウトが発生します。接続タイムアウトが発生すると、クライアント側でエラーになったり、データが消失したりする可能性があるため、タイムアウト時間は、ISIサーバの処理時間より長めに設定してください。

■Proxyの設定

プロキシ情報は、アプリケーション実行時のJavaVMオプションからシステムプロパティに設定します。

例:ESIClientSample.java

```
java
-Dhttp.proxyHost=proxy.proxyhost.com
-Dhttp.proxyPort=8080
-Dhttp.nonProxyHosts=localhost|*.myhost.com
-Dhttp.proxyUser=proxyUserName
-Dhttp.proxyPassword=proxyPassword
ESIClientSample
```

表D.5 プロパティ値

プロパティ名	値	備考
http.proxyHost	プロキシサーバのホスト名 例)proxy.proxyhost.com	値が設定されていない場合は、プロキシを経由せずに接続を行います。
http.proxyPort	プロキシサーバのポート番号 例)8080	SSLを使用しない場合 値が設定されていない場合は、プロキシを経由せずに接続を行います。 SSLを使用する場合 省略値は“80”です。(http.proxyHostが指定された場合)

プロパティ名	値	備考
http.nonProxyHosts	プロキシを経由せずに接続するホスト名 例)localhost|*.myhost.com	“|”区切りで複数のホスト名を指定できます。
http.proxyUser	プロキシサーバのログインユーザ名 例)proxyUserName	プロキシがBasic認証を行っている場合に設定してください。
http.proxyPassword	プロキシサーバのログインパスワード 例)proxyPassword	プロキシがBasic認証を行っている場合に設定してください。

■SSLの設定

クライアント認証なしの場合、以下の手順で設定します。

1. 動作環境の設定
2. 証明書環境の構築
    a. 証明書の登録・管理用ファイルを配置するディレクトリの作成
    b. 証明書管理ファイルの作成
    c. サーバのサイト証明書を登録している認証局からの認証局証明書取得
    d. 認証局証明書の登録
    e. 証明書管理ファイルの作成
3. SSL定義の作成
4. クライアントアプリケーションの設定

クライアント認証ありの場合、以下の手順で設定します。

1. 動作環境の設定
2. 証明書環境の構築
    a. 証明書の登録・管理用ファイルを配置するディレクトリの作成
    b. 証明書管理ファイルの作成
    c. 証明書発行要求の作成
    d. 証明書の発行依頼
    e. 認証局から証明書の取得
    f. 認証局から発行された証明書の登録
    g. 認証局から発行されたサイト証明書の登録
3. SSL定義の作成
4. クライアントアプリケーションの設定

SSLの設定方法の詳細については、“Interstage Application Server セキュリティシステム運用ガイド”、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”を参照してください。

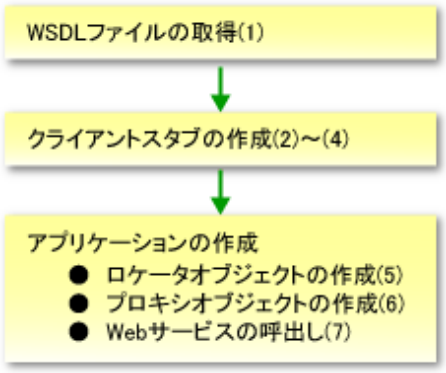
D.4.2 AXIS

例をもとに、AXIS利用時にSOAPのRPC機能(JAX-RPC)を、スタブで利用するアプリケーションの開発手順を説明します。
アプリケーションの開発は、以下の手順で行います。

注意

スタブを作成するためには、AXISが利用できる環境が必要となります。AXISのバージョン1.3(説明:Final Version 1.3 日付:October 5, 2005)を使用してください。

図D.13 開発の手順



手順

1. アクセス先のISIサーバで用意されているWSDLファイル(ESIService.wsdl)を取得し、任意のディレクトリに格納します。

   例)“D:¥WSDL¥ESIService.wsdl”

   注意

   ISIのWSDLファイルの取得方法は、ISIサーバの運用管理者に問い合わせてください。

2. AXISのコマンドを使用し、手順1で取得したWSDLファイルを利用して、クライアントスタブを自動生成します。
   なお、コマンドを実行するためには、org.apache.axis.wsdl.WSDL2Javaが実行できる環境の設定が必要です。

   ```
   java  org.apache.axis.wsdl.WSDL2Java  D:¥WSDL¥ESIService.wsdl ……(a)
   ```

   a. 手順1で取得したWSDLファイルを指定してください。

3. 以下のファイルが生成されたことを確認します。

   - “com¥fujitsu¥esi¥client¥ESIJMSReceiveFilter.java”
   - “com¥fujitsu¥esi¥message¥ESIAttachment.java”
   - “com¥fujitsu¥esi¥message¥ESIBytePayload.java”
   - “com¥fujitsu¥esi¥message¥ESIFunction.java”
   - “com¥fujitsu¥esi¥message¥ESIHeader.java”
   - “com¥fujitsu¥esi¥message¥ESIHistoryInfo.java”
   - “com¥fujitsu¥esi¥message¥ESIMessage.java”
   - “com¥fujitsu¥esi¥message¥ESIParameter.java”
   - “com¥fujitsu¥esi¥message¥ESIParameterItem.java”
   - “com¥fujitsu¥esi¥message¥ESIPayload.java”
   - “com¥fujitsu¥esi¥message¥ESISequence.java”
   - “com¥fujitsu¥esi¥message¥ESISequnceLogInfo.java”
   - “ESIServer¥ESIServerPortSoapBinding.java”
   - “ESIServer¥ESIServerPortType.java”
   - “ESIServer¥ESIServerService.java”
   - “ESIServer¥ESIServerServiceLocator.java”

4. 自動生成された上記のファイルをすべて、コンパイルします。
5. ロケータオブジェクトを作成します。
6. プロキシオブジェクトを作成します。
7. Webサービスを呼び出します。

アプリケーションの例：サンプルプログラム“ESIClientSample.java”を例にあげて説明します。



ISI 8.0以降では、WSDLファイルが変更されています。ISI 7.0で提供しているWSDLファイルを用いてアプリケーションを作成している場合、アプリケーションを修正する必要があります。詳細は、“D.4.3 アプリケーションの移行”を参照してください。

以下のサンプルプログラム“ESIClientSample.java”の斜体部分は、同期送信サービスの呼出し例です。

```
package sample;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;

import com.fujitsu.esi.client.*;
import com.fujitsu.esi.message.*;
import ESIServer.ESIServerPortType;
import ESIServer.ESIServerService;
import ESIServer.ESIServerServiceLocator;

public class ESIClientSample {
       public static void main(String[] args){
         try {
           //(5)ロケータオブジェクトの作成
           ESIServerService serviceLocator = new ESIServerServiceLocator();
           //(6)プロキシオブジェクトの作成
           ESIServerPortType serviceProxy = serviceLocator.getESIServerPort(); ……(a)
           ESIMessage reqmsg = new ESIMessage(); ……(b)
           ESIParameter param = reqmsg.getHeader().getParameter();
           //シーケンス名設定
           ESIParameterItem[] items=  new ESIParameterItem[1];
           items[0] = new ESIParameterItem();
           items[0].setKey("com.fujitsu.esi.sequence.name");
           items[0].setValue("sample_sequence");

           param.setItems(items);
              ;
           //(7)Webサービスの呼出し
           ESIMessage rspmsg = serviceProxy.sendESIMsgSync(reqmsg); ……(c)
           //戻り値ESIMessageの処理結果、シーケンス名取得
           String status = null;
           String sequenceName = null;
           ESIParameterItem[] resultItems
                     = rspmsg.getHeader().getParameter().getItems();

           for( int i= 0; i < resultItems.length; i++){
               if(resultItems[i].getKey().equals("com.fujitsu.esi.status") ){ ……(d)
                status = resultItems[i].getValue();
              } else
           if( resultItems[i].getKey().equals("com.fujitsu.esi.sequence.name") ){
                          sequenceName = resultItems[i].getValue();
                          break;
           }
```

```
            /
                :
          } catch (ServiceException e) {
            e.printStackTrace();
          } catch (RemoteException e) {
            e.printStackTrace();
          } catch (MalformedURLException e) {
            e.printStackTrace();
          }
        }
}
```

a. 送信先Webサービスの URLの指定には、2通りの方法があります。

- 手順1で取得したWSDLファイルに記述されている送信先Webサービスの URLに送信する場合は、パラメタなしでプロキシオブジェクトを作成します。(サンプルプログラム“ESIClientSample.java”の方法)
  例:ESIServerPortType serviceProxy = serviceLocator.getESIServerPort();
- 手順1で取得したWSDLファイルに記述されている送信先Webサービスの URLとは異なるURLに送信する場合、送信先URLをパラメタに設定し、プロキシオブジェクトを作成します。
  例:ESIServerPortType serviceProxy = serviceLocator.getESIServerPort(new URL("送信先URL"));

b. 注意事項を参照してください。

c. ISIがWebサービスとして提供しているメソッドは、以下のとおりです。

- 同期送信サービス:sendESIMsgSync(ESIMessage)
- 非同期送信サービス:sendESIMsgAsync(ESIMessage)
- 非同期受信サービス:receiveESIMsgAsync(ESIJMSReceiveFilter, Queue名)

注意

メッセージ送受信APIを利用した送信用メソッドは同一ですが、スタブを利用した送信用のメソッドは異なります。

各メソッドを呼び出す場合、共通メッセージ(ESIMessage)に設定が必要な項目を以下に示します。

表D.6 共通メッセージ(ESIMessage)に設定が必要な項目

メソッド名	キュー名	シーケンス名
sendESIMsgSync(ESIMessage)	−	○
sendESIMsgAsync(ESIMessage)	△	○

○:ESIMessageに設定必須
△:ESIMessageに設定可(省略時は、デフォルト値が有効となります。)
−:ESIMessageに設定不要(設定は無効となります。)

共通メッセージの作成方法については、“D.5 共通メッセージの作成・操作”を参照してください。

d. 以下のキー名を指定して値を取得すると、ISI呼出し結果情報やエラーが発生した原因情報を取得できます。

表D.7 キー名と取得情報

キー名	情報
com.fujitsu.esi.status	ISIの処理結果です。 ・ SUCCESS 処理が正常終了した場合 ・ FAILURE 処理中にエラーが発生した場合
com.fujitsu.esi.error.code	ISIの処理中に最初に発生したエラーのエラーコードです。

キー名	情報
com.fujitsu.esi.error.stepname	エラーが発生したステップです。 ISIの初期処理中にエラーが発生した場合は、nullになります。
com.fujitsu.esi.error.methodtype	エラーが発生したメソッド種別です。 ISIの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。
com.fujitsu.esi.error.handle.status	ISI処理のエラー発生時に呼び出したシーケンスの処理結果です。 ・ SUCCESS 処理が正常終了した場合 ・ FAILURE 処理中にエラーが発生した場合
com.fujitsu.esi.error.handle.code	エラー発生時に呼び出したシーケンスの中で、最初に発生したエラーのエラーコードです。
com.fujitsu.esi.error.handle.stepname	エラー発生時に呼び出したシーケンスの中で、エラーが発生したステップです。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。
com.fujitsu.esi.error.handle.methodtype	エラー発生時に呼び出したシーケンスの中で、エラーが発生したメソッド種別です。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。

注意事項

ESIMessageの要素を初期化するためには、以下の設定が必要です。

```
ESIMessage reqmsg = new ESIMessage();

ESIHeader header = new ESIHeader();
ESIParameter parameter = new ESIParameter();
ESIParameterItem[] items= new ESIParameterItem[0];
parameter.setItems(items);
header.setParameter(parameter);

ESISequence sequence = new ESISequence();
ESISequenceLogInfo[] sequenceLogList = new ESISequenceLogInfo[0];
sequence.setSequenceLogInfoList(sequenceLogList);
ESIFunction[] fuctionList = new ESIFunction[0];
sequence.setFunctionList(fuctionList);

ESIBytePayload objectPayload = new ESIBytePayload();

ESIPayload payload = new ESIPayload();
payload.setBytePayload(objectPayload);
ESIAttachment attachment = new ESIAttachment();
ESIBytePayload[] bytePayload  = new ESIBytePayload[0];
attachment.setBytePayloadList(bytePayload);
payload.setAttachment(attachment);
header.setSequence(sequence);

reqmsg.setHeader(header);
reqmsg.setPayload(payload);
```

サンプルプログラム“ESIClientSample.java”で呼出し例をあげた同期送信サービス以外の各メソッドの呼出し例を、以下に示します。
サンプルプログラム“ESIClientSample.java”の斜体の部分を置き換えてください。

非同期送信サービス:sendESIMsgAsync(ESIMessage)

```
ESIMessage reqmsg = new ESIMessage();

ESIParameter param = reqmsg.getHeader().getParameter();
//キュー名、シーケンス名設定

ESIParameterItem[] items= new ESIParameterItem[2];
items[0] = new ESIParameterItem();
items[0].setKey("com.fujitsu.esi.queue.name ");
items[0].setValue("sampleQueue");

items[1] = new ESIParameterItem();
items[1].setKey("com.fujitsu.esi.sequence.name");
items[1].setValue("sample_sequence");

param.setItems(items);

       :
//(7)Webサービスの呼出し
ESIMessage rspmsg = serviceProxy.sendESIMsgAsync(reqmsg);

//戻り値ESIMessageのシーケンス名、キュー名取得
String sequenceName = null;
String queueName  = null;
ESIParameterItem[] resultItems = rspmsg.getHeader().getParameter().getItems();

for( int i= 0; i < resultItems.length; i++){

        if( resultItems[i].getKey().equals("com.fujitsu.esi.sequence.name") ){
                        sequenceName = resultItems[i].getValue();
         }else if( resultItems[i].getKey().equals("com.fujitsu.esi.queue.name") ){
                        queueName = resultItems[i].getValue();
         }
}
```

非同期受信サービス:receiveESIMsgAsync(ESIJMSReceiveFilter, Queue名)

```
ESIJMSReceiveFilter fltr = new ESIJMSReceiveFilter();
//クエリ設定
fltr.setQuery("sampleKey='SampleValue'"); ……(a)
//キュー名設定
String queue = "sampleQueue";

       :
//(7)Webサービスの呼び出し
ESIMessage rspmsg = serviceProxy.receiveESIMsgAsync(fltr , queue);

//戻り値ESIMessageのキュー名取得
         String queueName = null;
         SIParameterItem[] resultItems = rspmsg.getHeader().getParameter().getItems();

         if( resultItems[i].getKey().equals("com.fujitsu.esi.queue.name") ){
                         queueName = resultItems[i].getValue();
                         break;
                 }
}
```

a. ESIJMSReceiveFilterにクエリを設定しない場合は、キューの先頭に格納されている共通メッセージを受信します。

■Basic認証、接続タイムアウト時間、セション管理の設定

Basic認証の設定は、スタブオブジェクトのプロパティで設定します。
接続タイムアウト時間、セション管理の設定は、スタブオブジェクトのメソッドで設定します。

例:サンプルプログラム“ESIClientSample.java”をもとに、各設定方法を以下に示します。

```
        :
import org.apache.axis.client.Stub;

public class ESIClientSample {

    public static void main(String[] args){
      try {
          //(1)ロケータオブジェクトの作成
          ESIServerService serviceLocator = new ESIServerServiceLocator();
          //(2)プロキシオブジェクトの作成
          ESIServerPortType serviceProxy = serviceLocator.getESIServerPort();

          //Basic認証：ユーザ名、パスワード設定
           ((Stub) serviceProxy)._setProperty(Stub.USERNAME_PROPERTY, "username");
           ((Stub) serviceProxy)._setProperty(Stub.PASSWORD_PROPERTY, "password");

          //接続タイムアウトの設定
           ((Stub) serviceProxy).setTimeout(50000); ……(a)

          //セション管理
           ((Stub) serviceProxy).setMaintainSession(true); ……(b)
        :
```

a. AXISが提供しているStubクラスの以下のメソッドを使用し、接続を切断するタイムアウト時間(ミリ秒)を設定します。
設定しない場合は、デフォルト値“60000(1分)”が有効となります。

- public void setTimeout(int timeout)

注意

- AXISのバージョン1.3を利用した場合です。
- タイムアウト時間がISIサーバの処理時間より短い場合、接続タイムアウトが発生します。接続タイムアウトが発生すると、クライアント側でエラーになったり、データが消失したりする可能性があるため、タイムアウト時間は、ISIサーバの処理時間より長めに設定してください。

b. AXISが提供しているStubクラスの以下のメソッドを使用し、HTTPセション継続を行うかどうか設定します。
設定しない場合は、デフォルト値“false”が有効となります。

- public void setMaintainSession(boolean session)

注意

AXISのバージョン1.3を利用した場合です。

■Proxy、SSLの設定

プロキシ情報、SSL情報は、アプリケーション実行時のJavaVMオプションからシステムプロパティに設定します。

例:ESIClientSample.java

```
java
     -Dhttp.proxySet=true
     -Dhttp.proxyHost=proxy.proxyhost.com
```

```
-Dhttp.proxyPort=8080
-Dhttp.nonProxyHosts =localhost|*.myhost.com
-Dhttp.proxyUser=proxyUserName
-Dhttp.proxyPassword=proxyPassword
-Djavax.net.ssl.trustStore=C:¥cert¥truststore¥keystore
-Djavax.net.ssl.trustStorePassword=trustpass
-Djavax.net.ssl.trustStoreType=jks
-Djavax.net.ssl.keyStore=C:¥cert¥keystore¥keystore
-Djavax.net.ssl.keyStorePassword=keystorepass
-Djavax.net.ssl.keyStoreType=jks
ESIClientSample
```

表D.8 プロパティ値

プロパティ名	値	備考
http.proxySet	以下のどちらかを指定します。 ・ true プロキシを介する ・ false プロキシを介さない	
http.proxyHost	プロキシサーバのホスト名 例)proxy.proxyhost.com	
http.proxyPort	プロキシサーバのポート番号 例)8080	
http.nonProxyHosts	プロキシを経由せずに接続するホスト名 例)localhost|*.myhost.com	“|”区切りで複数のホストを指定できます。
http.proxyUser	プロキシサーバのユーザ名 例)proxyUserName	プロキシがBasic認証を行っている場合に設定してください。
http.proxyPassword	プロキシサーバのパスワード 例)proxyPassword	プロキシがBasic認証を行っている場合に設定してください。
javax.net.ssl.trustStore	SSLサーバ認証用のキーストアのパス名 例)C:¥cert¥truststore¥keystore	
javax.net.ssl.trustStorePassword	SSLサーバ認証用のキーストアのパスワード 例)trustpass	
javax.net.ssl.trustStoreType	SSLサーバ認証用のキーストアのタイプ 例)jks	
javax.net.ssl.keyStore	SSLクライアント認証用のキーストアのパス名 例)C:¥cert¥keystore¥keystore	
javax.net.ssl.keyStorePassword	SSLクライアント認証用のキーストアのパスワード 例)keystorepass	
javax.net.ssl.keyStoreType	SSLクライアント認証用のキーストアのタイプ 例)jks	

D.4.3 アプリケーションの移行

ISI V7.0L10に対して、共通メッセージにESIParameterItemクラスが追加され、WSDLファイルが変更されています。
ISI V7.0L10で提供しているWSDLファイルを用いてクライアントアプリケーションを作成している場合、以下の作業が必要です。

S L

ISI 7.0に対して、共通メッセージにESIParameterItemクラスが追加され、WSDLファイルが変更されています。
ISI 7.0で提供しているWSDLファイルを用いてクライアントアプリケーションを作成している場合、以下の作業が必要です。

手順 W

1. ISI V7.0L10のWSDLファイルを用いて作成されているクライアントスタブを削除します。
2. ISI 8.0以降のWSDLファイルを用いて、クライアントスタブを生成し直します。
3. ISI V7.0L10で作成したクライアントアプリケーションの共通メッセージ(ESIMessage)操作を修正します。

手順 S L

1. ISI 7.0のWSDLファイルを用いて作成されているクライアントスタブを削除します。
2. ISI 8.0以降のWSDLファイルを用いて、クライアントスタブを生成し直します。
3. ISI 7.0で作成したクライアントアプリケーションの共通メッセージ(ESIMessage)操作を修正します。

修正が必要な共通メッセージ(ESIMessage)操作と修正例は、以下のとおりです。

[ISI V7.0L10] W

[ISI 7.0] S L

表D.9 修正が必要な共通メッセージ(ESIMessage)操作

クラス名	メソッド名	引数	戻り値
ESIParameter	getItems	なし	String[][]
	setItems	String[][]	なし

[ISI 8.0以降]

表D.10 修正が必要な共通メッセージ(ESIMessage)操作

クラス名	メソッド名	引数	戻り値
ESIParameter	getItems	なし	ESIParameterItem[]
	setItems	ESIParameterItem[]	なし

例:

[ISI V7.0L10] W

[ISI 7.0] S L

```
ESIMessage message = new ESIMessage();
ESIParameter parameter = message.getHeader().getParameter();

//プロパティの設定
String[][] items = {{"sample.key1","samplevalue1"},
{"sample.key2","samplevalue2"}};

parameter.setItems(items);

//プロパティの取得
String[][] items2 = parameter.getItems();
```

[ISI 8.0以降]

```
ESIMessage message = new ESIMessage();
ESIParameter parameter = message.getHeader().getParameter();

//プロパティの設定
ESIParameterItem[] items= new ESIParameterItem[2];
items[0] = new ESIParameterItem();
items[0].setKey("sample.key1");
items[0].setValue("sample_sequence");

items[1] = new ESIParameterItem();
items[1].setKey("sample.key2");
items[1].setValue("sample_sequence2");

parameter.setItems(items);

//プロパティの取得
ESIParameterItem[] items2 = parameter.getItems();
```

D.5 共通メッセージの作成・操作

共通メッセージ操作APIを利用し共通メッセージを作成・操作するアプリケーションの開発について説明します。

D.5.1 ヘッダの作成・操作

共通メッセージのヘッダにパラメタを設定したり、ヘッダから情報を取得したりするアプリケーションを作成できます。
パラメタの設定、情報取得には、以下のメソッドを使用します。

- ESIParameter#put(“ヘッダパラメタキー名”、“ヘッダパラメタ値”)
- ESIParameter#get(“ヘッダパラメタキー名”)

ヘッダの設定

共通メッセージのヘッダにパラメタを設定するアプリケーションについて、例をもとに説明します。
この例では、以下の条件を満たすアプリケーションを作成します。

- 非同期通信を利用する場合の共通メッセージを作成
- パラメタはヘッダに設定
- シーケンス“sample_sequence”を利用
- JMSCorrelationID“id000012345”を使用
- JMSキューに“jms_string_key”をキーとしたストリングプロパティとして“jmsstringvalue”を設定
- ユーザプロパティの“sample.key”に“samplevalue”を設定

指定例

共通メッセージのヘッダにパラメタを設定する場合の指定例は、以下のとおりです。

```
ESIMessage message = new ESIMessage();
ESIParameter parameter = message.getHeader().getParameter();

//(1)シーケンス名を指定
parameter.put(ESIHeaderConstants.ESI_HEADER_KEY_SEQUENCE_NAME, "sample_sequence"); ……(a)
//(2)JMSCorrelationIDを指定
parameter.put(ESIHeaderConstants.ESI_HEADER_KEY_JMS_HEADER_JMS_CORRELATION_ID, "id000012345"); ……(b)
//(3)JMSストリングプロパティの設定
parameter.put(ESIHeaderConstants.ESI_HEADER_KEYPREFIX_JMS_STRING + "jms_string_key", "jmsstringvalue"); ……(c)
```

```
//(4)ユーザプロパティの設定
parameter.put("sample.key", "samplevalue");
```

a. プログラム中にシーケンス名を直接指定した場合、サービスエンドポイント定義で指定したシーケンス名よりも優先されます。

b. JMSCorrelationIDには、以下の文字を指定しないでください。

- < (不等号(より小))
- > (不等号(より大))
- & (アンパサンド)
- ' (シングルクォーテーション)
- " (ダブルクォーテーション)

c. JMSキューには、“jms_string_key”をキーとしたストリングプロパティとして設定されます。

ヘッダからの情報取得

共通メッセージのヘッダから情報を取得するアプリケーションについて、例をもとに説明します。
この例では、以下の条件を満たすアプリケーションを作成します。

- 非同期通信を利用する場合の共通メッセージから情報を取得
- ヘッダから情報を取得
- シーケンス名を取得
- JMSCorrelationIDを取得
- JMSキューの“jms_string_key”をキーとしたストリングプロパティを取得
- ユーザプロパティの“sample.key”をキーとしたプロパティを取得

指定例

共通メッセージのヘッダのパラメタから情報を取得する場合の指定例は、以下のとおりです。

```
ESIService service = ESIServiceFactory.getService("endpointJMSReceive");
ESIMessage receiveMessage = service.receive();
ESIParameter receiveParam = receiveMessage.getHeader().getParameter();

//(1)シーケンス名を取得
String sequence = receiveParam.get(ESIHeaderConstants.ESI_HEADER_KEY_SEQUENCE_NAME);
//(2)JMSCorrelationIDを取得
String jmsCorrelationID = receiveParam.get(ESIHeaderConstants.ESI_HEADER_KEY_JMS_HEADER_JMS_CORRELATION_ID);
//(3)JMSストリングプロパティの取得
String jmskey = receiveParam.get(ESIHeaderConstants.ESI_HEADER_KEYPREFIX_JMS_STRING + "jms_string_key");
//(4)ユーザプロパティの取得
String samplekey = receiveParam.get("sample.key");
```

D.5.2 ISI処理結果の確認

ESIParameter#get(“ヘッダパラメタキー名”)メソッドに特定のヘッダパラメタキー名を指定すると、ISI呼出し結果情報やエラーが発生した原因情報を取得できます。

表D.11 ヘッダパラメタキー名と取得情報

ヘッダパラメタキー名	情報
ESIHeaderConstants.ESI_HEADER_KEY_STATUS	ISIの処理結果です。 ・ SUCCESS 処理が正常終了した場合

ヘッダパラメタキー名	情報
	・ FAILURE 処理中にエラーが発生した場合
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_CODE	ISIの処理中に最初に発生したエラーのエラーコードです。
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_STEP_NAME	エラーが発生したステップです。 ISIの初期処理中にエラーが発生した場合は、nullになります。
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_METHOD_TYPE	エラーが発生したメソッド種別です。 ISIの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_HANDLE_STATUS	ISI処理のエラー発生時に呼び出したシーケンスの処理結果です。 ・ SUCCESS 処理が正常終了した場合 ・ FAILURE 処理中にエラーが発生した場合
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_HANDLE_CODE	エラー発生時に呼び出したシーケンスの中で、最初に発生したエラーのエラーコードです。
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_HANDLE_STEP_NAME	エラー発生時に呼び出したシーケンスの中で、エラーが発生したステップです。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。
ESIHeaderConstants.ESI_HEADER_KEY_ERROR_HANDLE_METHOD_TYPE	エラー発生時に呼び出したシーケンスの中で、エラーが発生したメソッド種別です。 エラー発生時、シーケンスの初期処理中にエラーが発生した場合は、nullになります。 その他の場合はinit、execute、cancel、destroyのどれかになります。

注意

正常終了した場合、エラー情報は格納されません。

D.5.3 ペイロードの作成・操作

共通メッセージのペイロードに情報を設定したり、ペイロードから情報を取得したりするアプリケーションを作成できます。
情報の設定、情報取得には、以下のメソッドを使用します。

- ESIBytePayload#getObject()
- ESIBytePayload#setObject(byte[])

ペイロードの設定

ペイロードに情報を設定するアプリケーションについて、例をもとに説明します。
この例では、以下の条件を満たすアプリケーションを作成します。

・ 共通メッセージを作成

・ XML文書“contents”をペイロードに格納

指定例

ペイロードに情報を設定する場合の指定例は、以下のとおりです。

```
ESIMessage message = new ESIMessage();

ESIBytePayload bytePayload = message.getPayload().getBytePayload();
byte[] object = "<xmlcontents>contents</xmlcontents>".getBytes();
bytePayload.setObject(object);
```

SOAPメッセージを設定する場合は、SOAPのエンベロープ全体を設定します。SOAPの仕様に従った整形式XML文書を設定してください。特に以下の文字については、エンティティ参照の指定をしてください。

表D.12 置換する文字

置換する文字	定義済み実体参照
<	<
>	>
&	&
'	'
"	"

SOAPメッセージの指定例

```
ESIMessage message = new ESIMessage();

ESIBytePayload bytePayload = message.getPayload().getBytePayload();
byte[] object = "<?xml version=¥"1.0¥" encoding=¥"UTF-8¥"?><soapenv:Envelope soapenv:encodingStyle=¥"http://
schemas.xmlsoap.org/soap/encoding/¥" xmlns:soapenv=¥"http://schemas.xmlsoap.org/soap/envelope/¥" xmlns:xsd=¥"http://
www.w3.org/2001/XMLSchema¥" xmlns:xsi=¥"http://www.w3.org/2001/XMLSchema-instance¥" xmlns:SOAP-ENC=¥"http://
schemas.xmlsoap.org/soap/encoding/¥"><soapenv:Body><ns1:countWord xmlns:ns1=¥"urn:WordServer¥"><countWord_1_1
xsi:type=¥"xsd:string¥">12345</countWord_1_1><countWord_1_2 xsi:type=¥"xsd:string¥"></countWord_1_2></
ns1:countWord></soapenv:Body></soapenv:Envelope>".getBytes();
bytePayload.setObject(object);
```

ペイロードからの情報取得

ペイロードから情報を取得するアプリケーションについて、例をもとに説明します。
この例では、以下の条件を満たすアプリケーションを作成します。

・ サービスエンドポイント定義のサービスエンドポイント名“endpointRMI”に定義されたあて先に送信するデータを取得

指定例

ペイロードから情報を取得する場合の指定例は、以下のとおりです。

```
ESIService service = ESIServiceFactory.getService("endpointRMI");
ESIMessage returnMessage = service.send(message);

ESIBytePayload returnBytePayload = returnMessage.getPayload().getBytePayload();
byte[] returnObject = returnBytePayload.getObject();
```

D.5.4 共通メッセージの作成・操作(.NET)

Java版と同様の操作で共通メッセージのヘッダ、ペイロードの操作ができます。
共通メッセージのヘッダにパラメタを設定したり、ヘッダから情報を取得したりするアプリケーションを作成できます。
パラメタの設定、情報取得には、以下のメソッドを使用します。

- ESIParameter#Put(“ヘッダパラメタキー名”、“ヘッダパラメタ値”)
- ESIParameter#Get(“ヘッダパラメタキー名”)

ペイロードの設定、情報取得には、以下の属性を使用します。

- ESIBytePayload.Object

指定例

C#により共通メッセージのヘッダ・ペイロードを操作する例は、以下のとおりです。

```
ESIMessage message = new ESIMessage();
ESIParameter parameter = message.Header.Parameter;

//(1)ヘッダの設定
parameter.Put("sample.key", "samplevalue");

//(2)ペイロードの設定
ESIBytePayload bytePayload = message.Payload.BytePayload;
byte[] data = Encoding.UTF8.GetBytes("<xmlcontents>contents</xmlcontents>");
bytePayload.Object = data;

//通信処理
IESIService service = ESIServiceFactory.GetService("endpointNet");
ESIMessage receiveMessage = service.Send(message);
ESIParameter receiveParam = receiveMessage.Header.Parameter;

//(3)ヘッダの取得
string samplekey = parameter.Get("sample.key");

//(4)ペイロードの取得
ESIBytePayload returnBytePayload = receiveMessage.Payload.BytePayload;
byte[] returnObject = returnBytePayload.Object;
```

D.6 アプリケーションのコンパイルと実行

アプリケーションのコンパイルと実行方法について説明します。

アプリケーションのコンパイルと実行は、管理者権限を所有するユーザ、またはInterstage運用者で動作させてください。メッセージ送信APIのメッセージ保証機能を利用する場合、常に同じユーザでアプリケーションを実行してください。

共通APIを使用したアプリケーションの場合

作成したアプリケーションをコンパイルし、実行します。コンパイル・実行時には、クラスパスに以下を含めてください。

- %ESI_HOME%¥lib¥esicore.jar
- %ESI_HOME%¥lib¥esiejb_jar_client.jar
- %ESI_HOME%¥lib¥log4j-1.2.15.jar
- %CLASSPATH%

SOAPのRPC機能(JAX-RPC)を使用した場合、以下のクラスパスを追加してください。

- <Interstageインストールディレクトリ>¥J2EE¥lib¥isws.jar

- <Interstageインストールディレクトリ>¥J2EE¥lib¥isws-lib.jar

実行時には、以下をVM引数に指定してください。

- -Dcom.fujitsu.esi.home=%ESI_HOME%
- -Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB
- -Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB
- -Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl
- -Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl
- -Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl



- /opt/FJSVesi/lib/esicore.jar
- /opt/FJSVesi/lib/esiejb_jar_client.jar
- /opt/FJSVesi/lib/log4j-1.2.15.jar
- $CLASSPATH

SOAPのRPC機能(JAX-RPC)を使用した場合、以下のクラスパスを追加してください。

- /opt/FJSVj2ee/lib/isws.jar
- /opt/FJSVj2ee/lib/isws-lib.jar

実行時には、以下をVM引数に指定してください。

- -Dcom.fujitsu.esi.home=/opt/FJSVesi
- -Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB
- -Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB
- -Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl
- -Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl
- -Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl

RMI、JMSを利用した場合



コンパイルの例

```
javac -classpath "%ESI_HOME%¥lib¥esicore.jar;%ESI_HOME%¥lib¥esiejb_jar_client.jar;%ESI_HOME%¥lib¥log4j-1.2.15.jar;
%CLASSPATH%;." ESISend.java
```

実行例

```
java -classpath "%ESI_HOME%¥lib¥esicore.jar;%ESI_HOME%¥lib¥esiejb_jar_client.jar;%ESI_HOME%¥lib¥log4j-1.2.15.jar;
%CLASSPATH%;." -Dcom.fujitsu.esi.home=%ESI_HOME% -
Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

S L

コンパイルの例

```
javac -classpath "/opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar:
$CLASSPATH:." ESISend.java
```

実行例

```
java -classpath "/opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar:
$CLASSPATH:." -Dcom.fujitsu.esi.home=/opt/FJSVesi -
Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

JMS-Rを利用した場合

W

コンパイルの例

```
javac -classpath "%ESI_HOME%¥lib¥esicore.jar;%ESI_HOME%¥lib¥esiejb_jar_client.jar;%ESI_HOME%¥lib¥log4j-1.2.15.jar;
%CLASSPATH%;." ESISend.java
```

実行例

```
java -Dcom.fujitsu.ObjectDirector.DeterredMessage=YES
-Djava.naming.factory.initial=com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient
-classpath "%ESI_HOME%¥lib¥esicore.jar;%ESI_HOME%¥lib¥esiejb_jar_client.jar;%ESI_HOME%¥lib¥log4j-1.2.15.jar;
%IS_HOME%¥BAS¥lib¥apfwasyncapi.jar;%IS_HOME%¥BAS¥lib¥apfwibscm.jar;%IS_HOME%¥lib¥isjmxcommon.jar;
%IS_HOME%¥APC¥lib¥uji.jar;%IS_HOME%¥APC¥lib¥ujief.jar;%IS_HOME%¥APC¥lib¥ujilog.jar;
%IS_HOME%¥ODWIN¥etc¥class¥ODjava4.jar;%IS_HOME%¥eswin¥lib¥esnotifyjava4.jar;%IS_HOME%¥J2EE¥lib¥isj2ee.jar;
%IS_HOME%¥jms¥lib¥fjmsprovider.jar;<Symfoware Serverクライアントインストールディレクトリ>¥JDBC¥fjjdbc¥lib
¥fjsymjdbc2.jar;%IS_HOME%¥F3FMjs5¥common¥lib¥servlet-api.jar;%IS_HOME%¥ots¥lib¥fjtsclient.jar;%CLASSPATH%"
-Dcom.fujitsu.esi.home=%ESI_HOME% -Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

S L

コンパイルの例

```
javac -classpath
/opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar
:.:$CLASSPATH ESISend.java
```

実行例 S

```
java -Dcom.fujitsu.ObjectDirector.DeterredMessage=YES
-Djava.naming.factory.initial=com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient
-classpath /opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar
:/opt/FJSVibs/lib/apfwasyncapi.jar:/opt/FJSVibs/lib/apfwibscm.jar:/opt/FJSVisjmx/lib/isjmxcommon.jar
:/opt/FJSVwebc/lib/uji.jar:/opt/FJSVapcef/lib/ujief.jar:/opt/FJSVapclg/lib/ujilog.jar
:/opt/FSUNod/etc/class/ODjava4.jar:/opt/FJSVes/lib/esnotifyjava4.jar:/opt/FJSVj2ee/lib/isj2ee.jar
:/opt/FJSVjms/lib/fjmsprovider.jar:/opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar
:/opt/FJSVjs5/common/lib/servlet-api.jar:/opt/FJSVibs/lib/apfwsetsecurity.jar
:.:$CLASSPATH -Dcom.fujitsu.esi.home=/opt/FJSVesi -
Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

実行例 L

```
java -Dcom.fujitsu.ObjectDirector.DeterredMessage=YES
-Djava.naming.factory.initial=com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient
-classpath /opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar
:/opt/FJSVibs/lib/apfwasyncapi.jar:/opt/FJSVibs/lib/apfwibscm.jar:/opt/FJSVisjmx/lib/isjmxcommon.jar
:/opt/FJSVwebc/lib/uji.jar:/opt/FJSVapcef/lib/ujief.jar:/opt/FJSVapclg/lib/ujilog.jar
:/opt/FJSVod/etc/class/ODjava4.jar:/opt/FJSVes/lib/esnotifyjava4.jar:/opt/FJSVj2ee/lib/isj2ee.jar
:/opt/FJSVjms/lib/fjmsprovider.jar:/opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar
:/opt/FJSVjs5/common/lib/servlet-api.jar:/opt/FJSVibs/lib/apfwsetsecurity.jar
:.:$CLASSPATH -Dcom.fujitsu.esi.home=/opt/FJSVesi -
Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

注意

- ISIサーバ(Solaris、またはLinux)を対象に例を記述しています。
- LD_LIBRARY_PATHに、以下のパスを指定してください。

  S

  - /opt/FJSVesi/lib
  - /opt/FSUNrdb2b/lib
  - /opt/FJSVsymjd/fjjdbc/bin
  - /opt/FJSVibs/lib
  - /opt/FSUNod/lib
  - /opt/FJSVjms/lib

  L

  - /opt/FJSVesi/lib
  - /opt/FJSVrdb2b/lib
  - /opt/FJSVsymjd/fjjdbc/bin

ー /opt/FJSVibs/lib

ー /opt/FJSVod/lib

ー /opt/FJSVjms/lib

SOAPのRPC機能(JAX-RPC)を利用した場合

W

コンパイルの例

```
javac -classpath "%ESI_HOME%¥lib¥esicore.jar;%ESI_HOME%¥lib¥esiejb_jar_client.jar;%ESI_HOME%¥lib
¥log4j-1.2.15.jar;<Interstageインストールディレクトリ>¥J2EE¥lib¥isws.jar;<Interstageインストールディレクトリ>¥J2EE¥lib¥isws-
lib.jar;%CLASSPATH%;." ESISend.java
```

実行例

```
java -classpath "%ESI_HOME%¥lib¥esicore.jar;%ESI_HOME%¥lib¥esiejb_jar_client.jar;%ESI_HOME%¥lib
¥log4j-1.2.15.jar;<Interstageインストールディレクトリ>¥J2EE¥lib¥isws.jar;<Interstageインストールディレクトリ>¥J2EE¥lib¥isws-
lib.jar;%CLASSPATH%;." -Dcom.fujitsu.esi.home=%ESI_HOME% -
Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

S L

コンパイルの例

```
javac classpath "/opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar:/opt/
FJSVj2ee/lib/isws.jar:/opt/FJSVj2ee/lib/isws-lib.jar:$CLASSPATH:." ESISend.java
```

実行例

```
java classpath "/opt/FJSVesi/lib/esicore.jar:/opt/FJSVesi/lib/esiejb_jar_client.jar:/opt/FJSVesi/lib/log4j-1.2.15.jar:/opt/
FJSVj2ee/lib/isws.jar:/opt/FJSVj2ee/lib/isws-lib.jar:$CLASSPATH:." -Dcom.fujitsu.esi.home=/opt/FJSVesi -
Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB -
Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB -
Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl -
Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl -
Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
ESISend
```

注意

・ メッセージがイベントチャネルにない場合、イベントログに以下の内容が出力されることがあります。

OD: 情報: od60003:OD-Javaランタイムが例外を受け取りました。IDL:CORBA/COMM_FAILURE:1.0
com_fujitsu_CORBA_Request.cpp [1191]

出力を抑止する場合は、実行時のJavaVMオプションに、以下を追加してください。

ー -Dcom.fujitsu.ObjectDirector.DeterredMessage=YES

※本メッセージが出力されても、メッセージの受信に問題はありません。

- SOAPのRPC機能（JAX-RPC）による送受信を、スタブを利用して開発した場合、コンパイル・実行時に上記クラスパスやVM引数を設定する必要はありません。
- 環境変数CLASSPATHに適切な値が設定されている必要があります。詳細については、“ISI 導入ガイド”を参照してください。

.NET機能を利用した場合

作成したアプリケーションをコンパイルし、実行します。コンパイル・実行時には、参照設定に以下を含めてください。

- “Com.Fujitsu.Esi.Client.dll”
- “Com.Fujitsu.Esi.Common.dll”
- “Com.Fujitsu.Esi.Definition.dll”
- “Com.Fujitsu.Esi.Message.dll”
- “log4net.dll”

コンパイル・実行方法については、ご利用の開発環境、またはCSCコマンドの使用方法を確認してください。

注意

Windows VistaのSP1を適用している環境で、.NET機能を利用するアプリケーションの実行時に以下のエラーが発生する場合、以下に示す対処を一度だけ実施してください。

エラーの内容：

log4net:ERROR XmlHierarchyConfigurator: Could not create Appender [EventLogAppender] of type [log4net.Appender.EventLogAppender]. Reported error follows.

対処方法：

1. 対象の.NETアプリケーションを右クリックします。
2. “管理者として実行”を選択し、実行します。

コマンドプロンプトから対象の.NETアプリケーションを実行する場合は、以下に示す対処を一度だけ実行してください。

1. [スタート]メニューの[すべてのプログラム]－[アクセサリ]－[コマンド プロンプト]を右クリックします。
2. “管理者として実行”を選択し、コマンドプロンプトを開きます。
3. .NETアプリケーションを実行します。

スタブを利用したアプリケーションの場合

Interstage Application Serverを利用した場合

作成したアプリケーションをコンパイルし、実行します。コンパイル・実行時には、クラスパスに以下を含めてください。

- <Interstageインストールディレクトリ>¥J2EE¥lib¥isws.jar
- <Interstageインストールディレクトリ>¥J2EE¥lib¥isws-lib.jar

注意

上記の“isws-lib.jar”をクラスパスに設定することで、以下のjarファイルも自動的にクラスパスに設定されます。詳細については、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”を参照してください。

- <Interstageインストールディレクトリ>¥J2EE¥lib¥xerces¥xercesImpl.jar
- <Interstageインストールディレクトリ>¥J2EE¥lib¥xerces¥xml-apis.jar
- <Interstageインストールディレクトリ>¥J2EE¥lib¥isj2ee.jar

SSLを設定した場合、以下のクラスパスを追加してください。

- <Interstageインストールディレクトリ>¥lib¥isadmin_scs.jar

コンパイルの例

```
javac -classpath "<Interstageインストールディレクトリ>¥J2EE¥lib¥isws.jar;<Interstageインストールディレクトリ>¥J2EE¥lib¥isws-lib.jar;%CLASSPATH%;." ESIClientSample.java
```

実行例

```
java -classpath "<Interstageインストールディレクトリ>¥J2EE¥lib¥isws.jar;<Interstageインストールディレクトリ>¥J2EE¥lib¥isws-lib.jar;%CLASSPATH%;." ESIClientSample
```

AXISを利用した場合

作成したアプリケーションをコンパイルし、実行します。コンパイル・実行時には、クラスパスにAXISのライブラリ(基本的に<Axisインストールディレクトリ>¥libのjarファイルすべて)を含めてください。

AXIS1.3を使用した場合の例

- <Axisインストールディレクトリ>¥lib¥axis.jar
- <Axisインストールディレクトリ>¥lib¥axis-ant.jar
- <Axisインストールディレクトリ>¥lib¥axis-schema.jar
- <Axisインストールディレクトリ>¥lib¥commons-discovery-0.2.jar
- <Axisインストールディレクトリ>¥lib¥commons-logging-1.0.4.jar
- <Axisインストールディレクトリ>¥lib¥jaxrpc.jar
- <Axisインストールディレクトリ>¥lib¥log4j-1.2.8.jar
- <Axisインストールディレクトリ>¥lib¥saaj.jar
- <Axisインストールディレクトリ>¥lib¥wsdl4j-1.5.1.jar

コンパイルの例

```
javac -classpath "<Axisインストールディレクトリ>¥lib¥axis.jar;<Axisインストールディレクトリ>¥lib¥axis-ant.jar;<Axisインストールディレクトリ>¥lib¥axis-schema.jar;<Axisインストールディレクトリ>¥lib¥commons-discovery-0.2.jar;<Axisインストールディレクトリ>¥lib¥commons-logging-1.0.4.jar;<Axisインストールディレクトリ>¥lib¥jaxrpc.jar;<Axisインストールディレクトリ>¥lib¥log4j-1.2.8.jar;<Axisインストールディレクトリ>¥lib¥saaj.jar;<Axisインストールディレクトリ>¥lib¥wsdl4j-1.5.1.jar;%CLASSPATH%;." ESIClientSample.java
```

実行例

```
java -classpath "<Axisインストールディレクトリ>¥lib¥axis.jar;<Axisインストールディレクトリ>¥lib¥axis-ant.jar;<Axisインストールディレクトリ>¥lib¥axis-schema.jar;<Axisインストールディレクトリ>¥lib¥commons-discovery-0.2.jar;<Axisインストールディレクトリ>¥lib¥commons-logging-1.0.4.jar;<Axisインストールディレクトリ>¥lib¥jaxrpc.jar;<Axisインストールディレクトリ>¥lib¥log4j-1.2.8.jar;<Axisインストールディレクトリ>¥lib¥saaj.jar;<Axisインストールディレクトリ>¥lib¥wsdl4j-1.5.1.jar;%CLASSPATH%;." ESIClientSample
```

CORBAを利用した場合

アプリケーションのコンパイルと実行方法については、“Interstage Application Server アプリケーション作成ガイド(CORBAサービス編)”を参照してください。

D.7 メディエータファンクションの開発

D.7.1 .NETユーザプログラム呼出しの概要

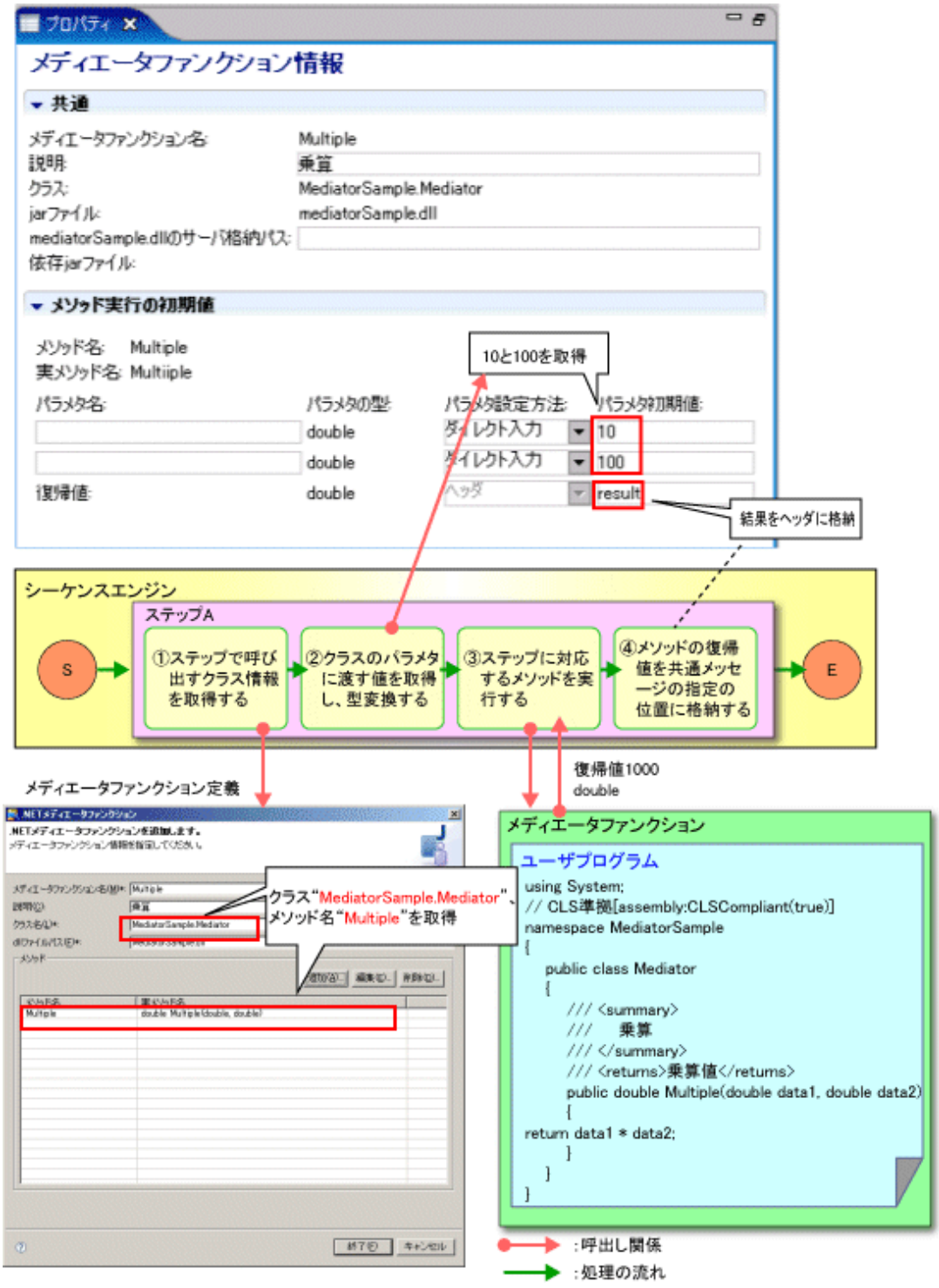
例をもとに、.NET Framework上で動作するユーザプログラム呼出しの概要について説明します。
この例では、以下の条件を満たすプログラムを作成します。

- 言語は、C#
- メソッドの種別は、実行処理メソッド
- 名前空間は、“MediatorSample”
- クラス名は、“Mediator”
- メソッド名は、“Multiple”
- メディエータファンクション名は、“Multiple”
- パラメタは、data1とdata2
- data1に“10”、data2に“100”を指定
- 2つのパラメタの乗算値“1000”を返却
- 結果は、共通メッセージのヘッダに格納

シーケンスエンジン内の処理の流れ

1. メディエータファンクション定義から呼び出すプログラムのクラス情報を取得
2. シーケンス定義からパラメタの値を取得し、型変換
3. メソッド“Multiple”を実行(data1×data2)
4. 復帰値“1000”を共通メッセージのヘッダに格納

図D.14 .NETプログラムの呼出しイメージ



D.7.2 .NETユーザプログラムの作成 W

.NET Framework上で動作するユーザプログラムを呼び出す場合、メディエータファンクションのユーザプログラムは、以下の手順で開発します。

1. メディエータファンクションクラスを作成します。
2. 作成したユーザプログラムの情報を、メディエータファンクション定義ファイルおよびシーケンス定義ファイルに定義します。
3. 作成したクラスが含まれるDLLファイルを固定のディレクトリに配備します。

“.NETユーザプログラム呼出しの概要”の例に従って、作成手順を以下に説明します。

クラスの作成

シーケンスエンジンから実行されるクラスを作成します。

表D.13 クラスの機能一覧

メソッドの種別	名前空間	クラス名	メソッド名	パラメタ	復帰値
-	MediatorSample	Mediator	-	-	-
execute	-	-	Multiple	double data1 double data2	double

1. クラスとメソッドを作成します。以下のサンプルコードを作成します。
   指定例

```
using System;

// CLS準拠
[assembly:CLSCompliant(true)]
namespace MediatorSample
{
    public class Mediator
    {
        /// <summary>
        ///     乗算
        /// </summary>
        /// <returns>乗算値</returns>
        public double Multiple(double data1, double data2)
        {
            return data1 * data2;
        }
    }
}
```

2. 作成したクラスをコンパイルします。DLLファイルが生成されます。
   コンパイル方法は、ご使用の開発環境、またはCSCコマンドの使用方法を確認してください。

メディエータファンクション定義の設定

作成したユーザプログラムの情報を、ISI Studioを利用しメディエータファンクションとして定義します。
ここでは、“.NETユーザプログラムの作成”で作成したユーザプログラム“MediatorSample.dll”を例に、メディエータファンクション定義とシーケンス定義の作成までを説明します。

.NETユーザプログラムをメディエータファンクションとしてシーケンスから呼び出すため、.NETメディエータファンクションの定義を作成します。

1. ISIナビゲータービューで以下のどちらかを操作します。

   ー メディエータファンクションのツリーをダブルクリック、手順2へ進みます。

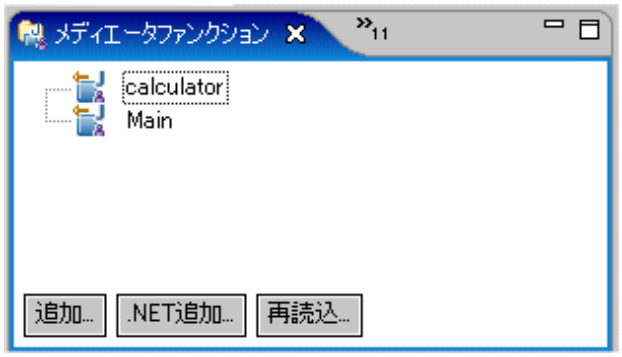
ー ISIプロジェクト配下のツリーを選択し、ポップアップメニューの[新規.NETメディエータファンクション]をクリックし、手順3へ進みます。

図D.15 ISIナビゲータービューの画面例



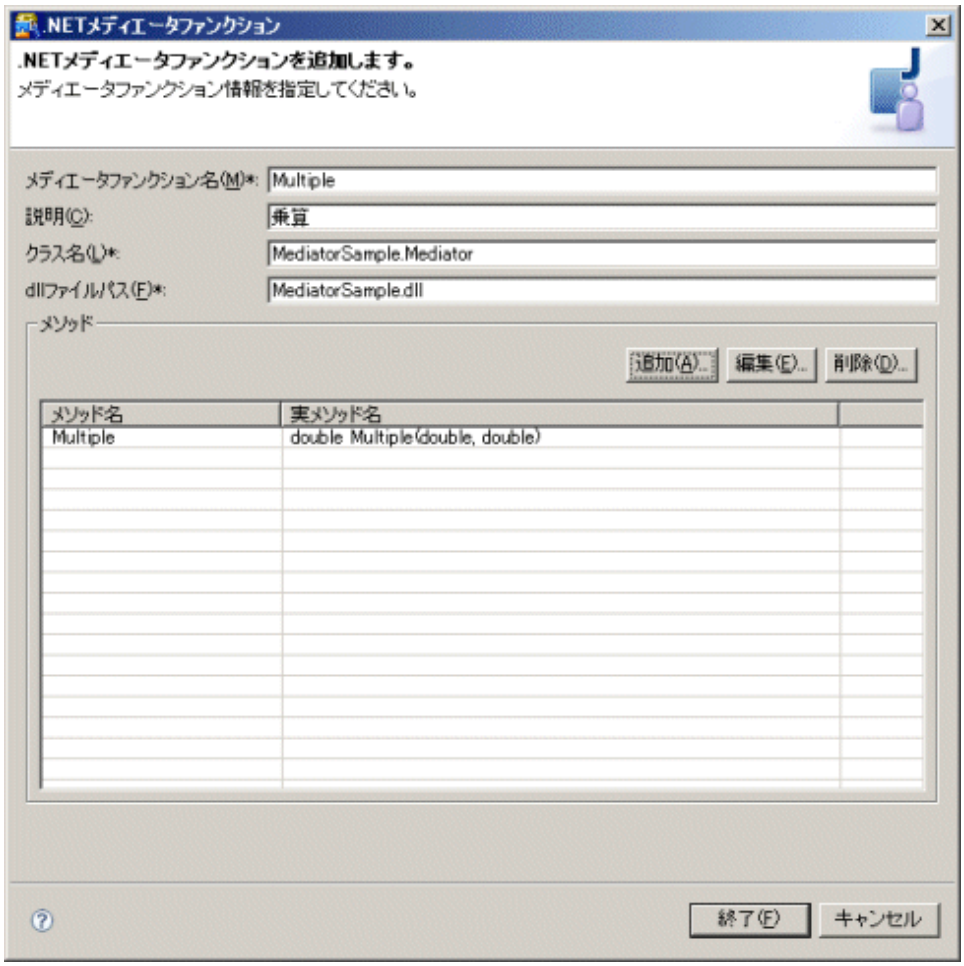
2. メディエータファンクションエディタで、[.NET追加]ボタンをクリックし、手順3へ進みます。

図D.16 メディエータファンクションエディタの画面例

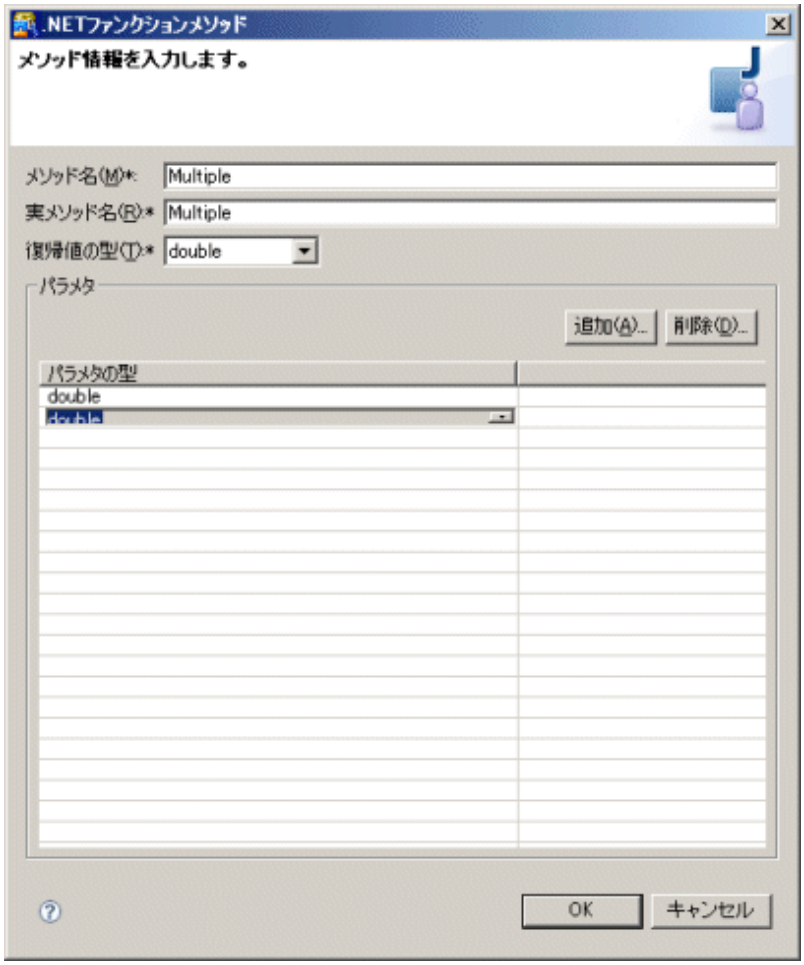


3. メディエータファンクション情報とメソッドの情報を指定し、[終了]ボタンをクリックします。

図D.17 [.NETメディエータファンクション]ダイアログボックスの画面例

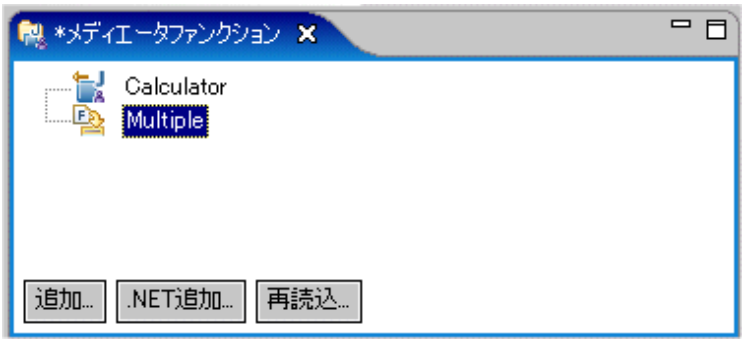


図D.18 [.NETファンクションメソッド]ダイアログボックスの画面例

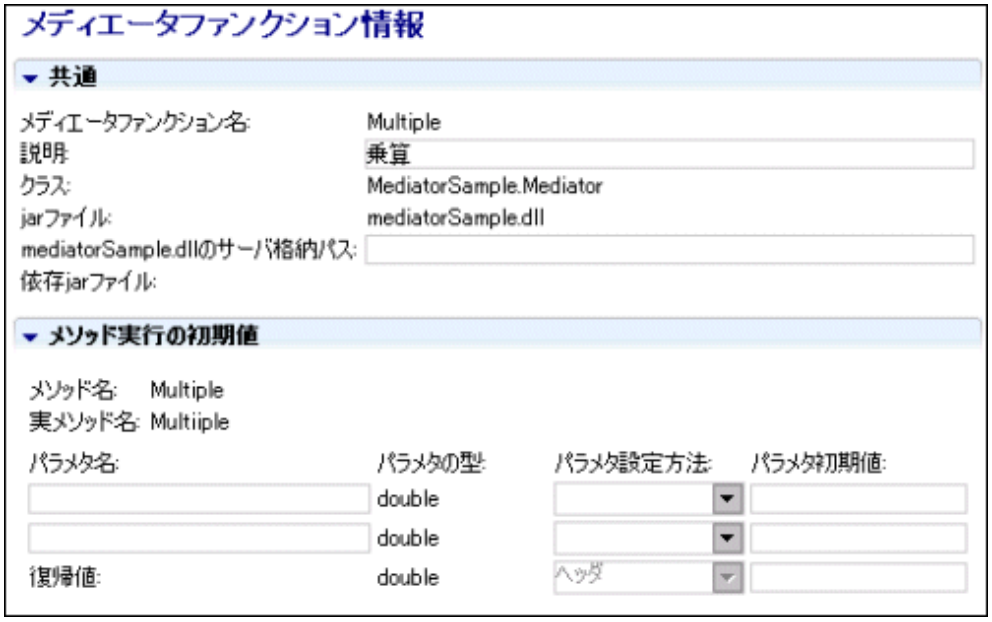


4. 作成されたメディエータファンクションのプロパティを確認します。
ユーザプログラムへの初期値を指定する場合は、プロパティを変更してください。
以下の例では、プロパティの変更は不要です。

図D.19 メディエータファンクションの定義例(メディエータファンクションエディタ)



図D.20 メディエータファンクションの定義例(プロパティビュー)



5. メディエータファンクションエディタを保存します。
   ショートカットキーの[Ctrl+S]、またはメニューの[ファイル]－[保存]を選択し、メディエータファンクションエディタを保存します。

シーケンスの作成

.NETユーザプログラムを呼び出すシーケンスを作成します。
シーケンスの作成方法については、“定義の設定”を参照してください。

ISI Studioの操作や画面の詳細については、“ISI Studio ヘルプ”を参照してください。
定義の詳細については、“付録B 定義の詳細”を参照してください。

DLLファイルの配置

定義で指定したdllファイル格納パスにdllファイルを配置します。
定義のサーバ格納パスの指定を省略した場合は、以下に配置します。

```
<INSDIR>¥usr¥lib
```

D.8 サンプルプログラム

ここでは、互換機能のサンプルプログラムについて説明します。

以下のサンプルプログラムを用意しています。

- ESIAPISendTest.java
  サービス利用側アプリケーションにおいて、共通APIを使った“送信処理”を行います。
- ESIAPISendTest.cs
  .NET Framework上で動作するサービス利用側アプリケーションにおいて、共通APIを使った“送信処理”を行います。
- ESIAPIReceiveTest.java
  サービス利用側アプリケーションにおいて、共通APIを使った“非同期受信処理”を行います。

サンプルプログラムは、以下に格納されています。

```
<INSDIR>¥sample¥App¥src
```

S L

```
/opt/FJSVesi/sample/App/src
```

クライアント

```
<INSDIR>¥sample¥App¥src
```

Javaのサンプルプログラムは、JDK1.4互換ソースのため、JDK5.0以上でコンパイルした場合に警告が発生することがありますが、動作に問題はありません。

D.8.1 ESIAPISendTest.java(送信処理)

共通APIを使用した送信処理のサービス利用側アプリケーションのサンプルプログラムです。

```
        :
        :
public class ESIAPISendTest {

        public static void main(String[] args) throws ESIAPIException {
                // (1) サービスエンドポイント名の指定
                String strEndpoint = "inboundtest";
                if (args.length >= 1) {
                        strEndpoint = args[0];
                }
                try {
                        System.out.println("java.home=" + System.getProperty("java.home"));

                        // (2) Serviceクラスの生成
                        ESIService es = ESIServiceFactory.getService(strEndpoint);

                        // (3) 共通メッセージの生成
                        ESIMessage msg = new ESIMessage();

                        // (4) ペイロードの設定
                        msg.getPayload().getBytePayload().setObject("<name>test-payload</name>".getBytes());

                        // (5) ヘッダの設定
                        SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");
                        msg.getHeader().getParameter().put("user.createdate", sdf1.format(new Date()));
                        // (6) 送信APIの呼出し
                        ESIMessage retMessage = es.send(msg);

                        // (7) ペイロードの取り出し
                        byte[] bytePayload = retMessage.getPayload().getBytePayload().getObject();
                        if (bytePayload != null) {
                                String byteString = new String(bytePayload);
                                System.out.println("objectPayload=[" + byteString + "]");
                        }

                        // (8) ヘッダの取り出し
                        System.out.println("Status=" +
retMessage.getHeader().getParameter().get("com.fujitsu.esi.status"));
                        System.out.println("Errorcode=" +
retMessage.getHeader().getParameter().get("com.fujitsu.esi.error.code"));
                        System.out.println("stepname=" +
retMessage.getHeader().getParameter().get("com.fujitsu.esi.error.stepname"));
                        System.out.println("methodtype=" +
retMessage.getHeader().getParameter().get("com.fujitsu.esi.error.methodtype"));
                        System.out.println("user.result=" + retMessage.getHeader().getParameter().get("result"));
```

```
                        :
                        :
                } catch (ESIAPIException e) {
                        e.printStackTrace();
                }
        }
        :
        :
}
```

1. サービスエンドポイント名の設定
   サービスエンドポイント名“inboundtest”を設定します。
   サービスエンドポイント定義で指定するサービスエンドポイント名と一致させます。

   W

   サービスエンドポイント定義:<INSDIR>¥sample¥App¥def¥endpoint.xml

   S L

   サービスエンドポイント定義:/opt/FJSVesi/sample/App/def/endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
    <Endpoint name="inboundtest" sync="true"> ……サービスエンドポイント名
        <Description>Sync Send Endpoint</Description>
        <SequenceName>calctest</SequenceName>
    </Endpoint>
        :
```

   ポイント

   サービスエンドポイント定義の設定により、同期処理で送信するか、非同期処理で送信するかが決まります。

2. Serviceクラスの作成
   サービスエンドポイント名をもとに、送信のServiceクラスを作成します。
3. 共通メッセージの作成
   共通メッセージを作成します。共通メッセージ操作APIを使用します。
4. ペイロードの設定
   ペイロードに作業用のデータとして“<name>test-payload</name>”を設定します。
5. ヘッダの設定
   共通メッセージのヘッダに作業用のデータとしてキー:“user.createdate”、値:“<現在日時>”を設定します。
6. 送信APIの呼出し
   共通メッセージを設定して、メッセージ送受信API“ESIService#send()”を呼び出します。
7. ペイロードの取出し
   共通メッセージの中から、ペイロードを取り出します。
8. ヘッダの取出し
   共通メッセージの中から、ヘッダパラメタキー名と値を取り出します。
   サンプルプログラムは、共通メッセージのヘッダのすべてのパラメタキー名と値を取り出すように作成されています。

D.8.2 ESIAPISendTest.cs(送信処理)

共通APIを使用した送信処理の.NET Framework上で動作するサービス利用側アプリケーションのサンプルプログラムです。

```
using System;
using System.Collections.Generic;
using System.Text;
using Com.Fujitsu.Esi.Client;
using Com.Fujitsu.Esi.Message;

namespace test.api
{
    public class ESIAPISendTest
    {
        public static void Main(string[] args)
        {
            // (1) サービスエンドポイント名の指定
            string strEndpoint = "soapnet";
            if (args.Length >= 1)
            {
                strEndpoint = args[0];
            }

            try {
                // (2) Serviceクラスの生成
                IESIService es = ESIServiceFactory.GetService(strEndpoint);

                // (3) 共通メッセージの生成
                ESIMessage msg = new ESIMessage();

                // (4) ペイロードの設定
                msg.Payload.BytePayload.Object =
                    Encoding.UTF8.GetBytes("<name>test-payload</name>");

                // (5) ヘッダの設定
                msg.Header.Parameter.Put("user.createdate", DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss"));

                // (6) 送信APIの呼出し
                ESIMessage retMessage = es.Send(msg);

                // (7) ペイロードの取出し
                byte[] bytePayload = retMessage.Payload.BytePayload.Object;

                if (bytePayload != null) {
                    string byteString = Encoding.UTF8.GetString(bytePayload);
                    Console.WriteLine("objectPayload=[" + byteString + "]");
                }

                // (8) ヘッダの取出し
                foreach(ESIParameterItem item in retMessage.Header.Parameter.Items) {
                    Console.WriteLine("key=[" + item.Key + "] value=[" + item.Value + "]");
                }

            } catch (ESIAPIException e) {
                Console.WriteLine(e.StackTrace);
            }
        }
    }
}
```

1. サービスエンドポイント名の指定
   サービスエンドポイント名“soapnet”を指定します。
   サービスエンドポイント定義で指定するサービスエンドポイント名と一致させます。サービスエンドポイント定義の作成方法については、“ISI Studio ヘルプ”を参照してください。

2. Serviceクラスの作成
   サービスエンドポイント名をもとに、送信のServiceクラスを作成します。

3. 共通メッセージの作成
   共通メッセージを作成します。共通メッセージ操作APIを使用します。

4. ペイロードの設定
   ペイロードに作業用のデータとして“<name>test-payload</name>”を設定します。

5. ヘッダの設定
   共通メッセージのヘッダに作業用のデータとしてキー:“user.createdate”、値:“<現在日時>”を設定します。

6. 送信APIの呼出し
   共通メッセージを設定して、メッセージ送受信API“IESIService#Send()”を呼び出します。

7. ペイロードの取出し
   共通メッセージの中から、ペイロードを取り出します。

8. ヘッダの取出し
   共通メッセージの中から、ヘッダパラメタキー名と値を取り出します。
   サンプルプログラムは、共通メッセージのヘッダのすべてのパラメタキー名と値を取り出すように作成されています。

D.8.3 ESIAPIReceiveTest.java(非同期受信処理)

共通APIを使用した非同期受信処理のサービス利用側アプリケーションのサンプルプログラムです。

```
        :
        :
public class ESIAPIReceiveTest {

        public static void main(String[] args) {
                // (1) サービスエンドポイント名の指定
                String strEndpoint = "receivetest";
                if (args.length >= 1) {
                        strEndpoint = args[0];
                }
                try {
                        System.out.println("java.home=" +
                                System.getProperty("java.home"));

                        // (2) Serviceクラスの生成
                        ESIService es = ESIServiceFactory.getService(strEndpoint);

                        // (3) 受信APIの呼出し
                        ESIMessage retMessage = es.receive();

                        if (retMessage != null) {
                                // (4) ペイロードの取り出し
                                byte[] bytePayload =
                                        retMessage.getPayload().getBytePayload().getObject();
                                String byteString = new String(bytePayload);
                                System.out.println("objectPayload=[" + byteString + "]");

                                // (5) ヘッダの取り出し
                              System.out.println("user.result=" + retMessage.getHeader().getParameter().get("result"));
                                        :
                                        :
                                }
                        } else {
                                // (6) 受信するメッセージがなかったときの処理
                                System.out.println("ESIMessage was not queuing.");
                        }
```

```
                } catch (ESIAPIException e) {
                        e.printStackTrace();
                }
        }
        :
        :
}
```

1. サービスエンドポイント名の設定
   サービスエンドポイント名“receivetest”を設定します。
   サービスエンドポイント定義で指定するサービスエンドポイント名と一致させます。

   W

   サービスエンドポイント定義:<INSDIR>¥sample¥App¥def¥endpoint.xml

   S L

   サービスエンドポイント定義:/opt/FJSVesi/sample/App/def/endpoint.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<EndpointList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="esi-
endpoint.xsd">
        :
    <Endpoint name="receivetest" sync="false"> ……サービスエンドポイント名
        <Description>Async Receive Endpoint</Description>
        <QueueName>ESIOutboundQueue</QueueName>
    </Endpoint>
</EndpointList>
```

2. Serviceクラスの作成
   サービスエンドポイント名をもとに、受信のServiceクラスを作成します。
3. 受信APIの呼出し
   メッセージ送受信API“ESIService#receive()”を呼び出します。
4. ペイロードの取出し
   共通メッセージの中から、ペイロードを取り出します。
5. ヘッダの取出し
   共通メッセージの中から、ヘッダパラメタキー名と値を取り出します。
   サンプルプログラムは、共通メッセージのヘッダのすべてのパラメタキー名と値を取り出すように作成されています。
6. 受信するメッセージがなかったときの処理
   JMSキューに受信するメッセージがなかったときの処理です。

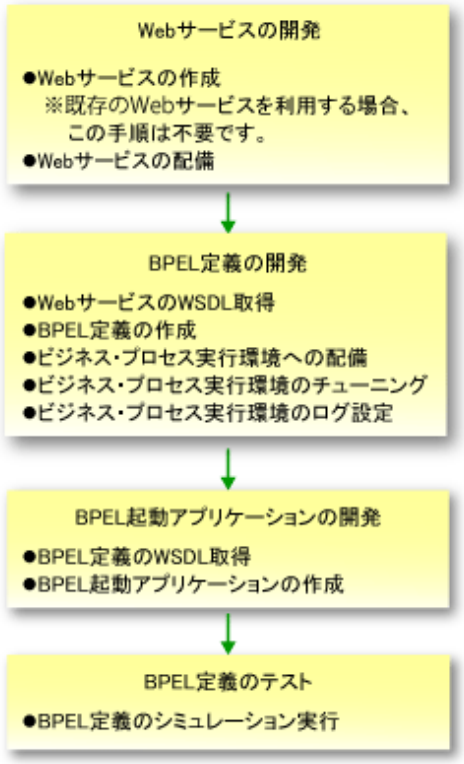
付録E　BPEL機能のアプリケーション開発

BPEL機能を使用するには、BPEL定義、ビジネス・プロセス実行機能より呼び出すWebサービス、およびビジネス・プロセス実行機能を起動するBPEL起動アプリケーションを開発する必要があります。ここでは、それぞれの開発方法について説明します。

E.1　アプリケーション開発の流れ

以下の手順で開発します。

図E.1 アプリケーション開発の流れ



次に各手順について説明します。

E.2　Webサービスの開発

ビジネス・プロセス実行機能より呼び出すWebサービスを開発し、Webサービスを実行する環境(IJServerクラスタ、またはIJServer)に配備します。呼び出すサービスがCORBAアプリケーションなどWebサービスではない場合は、ISIサーバによりサービスをWebサービス化しておきます。次に各方法について説明します。

E.2.1　Webサービスの作成

Interstage StudioでWebサービスアプリケーションを作成します（Webサービスを呼び出すビジネス・プロセス実行機能がWebサービスクライアントの位置づけとなります）。

IJServerクラスタに配備するWebサービスの作成方法については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発”の“Webサービスアプリケーションの開発” を参照してください。

IJServerに配備するWebサービスの作成方法については、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”の“Webサービス編”の“Webサービスの開発”の“Webサービス（サーバ機能）の開発”を参照してください。

なお、すでに開発済のWebサービスを利用する場合、本手順は不要です。また、CORBAアプリケーションなどWebサービスではないサービスを利用する場合は、“E.2.3 WebサービスではないサービスのWebサービス化”を参照してください。

E.2.2 Webサービスの配備

ビジネス・プロセス実行機能より呼び出すWebサービスをIJServerクラスタ、またはIJServerに配備します。

IJServerクラスタにWebサービスを配備する場合には、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの運用”の“Webサービスを利用する場合の手順”を参照してください。

IJServerにWebサービスを配備する場合には、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”の“J2EEアプリケーションの運用”の“J2EEアプリケーションの配備と設定”を参照してください。

E.2.3 WebサービスではないサービスのWebサービス化

CORBAアプリケーションなどWebサービスではないサービスを利用する場合は、ISIサーバによりサービスをWebサービス化し、ビジネス・プロセス実行機能からはWebサービスとして呼び出します。

ビジネス・プロセス実行機能がサービス利用側アプリケーション、ビジネス・プロセス実行機能から呼び出すサービスがサービス提供側アプリケーションとなります。ビジネス・プロセス実行機能に対するWebサービスとしてのインタフェースを決定したあと、ISIのフォーマット定義・変換定義、およびISI定義の作成、必要に応じてアダプタ、およびメディエータファンクションを開発します。

ISIのフォーマット定義・変換定義については“第5章 変換定義の開発”、ISI定義については“ISI Studio ヘルプ”、アダプタの開発については“第4章 アダプタの開発”、メディエータファンクションの開発については“第3章 メディエータファンクションの開発”をそれぞれ参照してください。

E.3 BPEL定義の開発

BPEL定義の開発では、ビジネス・プロセス実行機能より呼び出すWebサービスのWSDL取得、BPELエディタによるBPEL定義の作成、およびBPEL定義のビジネス・プロセス実行環境への配備を行います。次に各方法について説明します。

E.3.1 WebサービスのWSDL取得

ビジネス・プロセス実行機能より呼び出すWebサービスの公開用WSDLを取得します。

- IJServerクラスタに配備されたWebサービスの公開用WSDLを取得する場合

  WSDLは、ファイルとして取得します。取得方法については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発 ”の“アプリケーションの作成方法”の“Webサービスクライアントアプリケーションの開発”を参照してください。

  ポイント

  BPEL定義を開発する環境から、取得したWSDL一式のwsdl:import要素やxs:import要素などの参照URLにアクセスすることができない場合は、wsdl:import要素やxs:import要素を、適切な相対パスに修正するなど、BPEL定義を開発する環境で利用可能にするための措置をしたWSDLを、再取得してください。WSDLの修正方法については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発 ”の“アプリケーションの作成方法”の“Webサービスアプリケーションの開発”の“Webサービスクライアントへの WSDLの提供”の“ijwsimportコマンドで取得保管したWSDL一式をファイルで提供する”を参照してください。
  Webサービスが Java EEで作成されている場合、WSDL一式のファイルの修正が必要な場合があります。使用しているデータの型により、XMLスキーマファイルのelement定義部分に、minOccurs="0"が存在する場合があります。この場合これを削除してください。修正例を以下に示します

  修正前

```
<xs:element name="arg0" type="xs:string" minOccurs="0" />
```

  修正後

```
<xs:element name="arg0" type="xs:string" />
```

- IJServerに配備されたWebサービスの公開用WSDLを取得する場合

  “Interstage Application Server J2EE ユーザーズガイド(旧版互換)”の“Webサービス編”の“Webサービスの運用”の “Webサービス(サーバ機能)の運用方法”の“公開用WSDLの取得”を参照してください。

E.3.2 BPEL定義の作成

BPEL定義の作成方法については、“E.6 BPELエディタによるBPEL定義の作成方法”を参照してください。

E.3.3 ビジネス・プロセス実行環境への配備

作成したBPEL定義をビジネス・プロセス実行環境に配備します。

配備手順

1. BPEL定義の転送

   BPEL定義を作成したマシンからビジネス・プロセス実行環境が作成されているマシンへBPEL定義を転送してください。

   BPELエディタで作成したBPEL定義のプロジェクトには、以下のBPEL定義が含まれています。定義はすべて同一ディレクトリ内に格納してください。

   - BPEL定義(.bpel)
   - WSDL定義(.wsdl)
   - deploy.xml

   また、WSDL定義で他の外部ファイルをインポートしている場合は、インポートしている外部ファイルも同一ディレクトリ内に格納してください。

2. BPEL定義の配備

   ビジネス・プロセス実行環境が作成されているマシン上で、isibpeladminコマンドのdeployサブコマンドを使用してBPEL定義を配備します。

   配備は、手順の1で転送したBPEL定義を格納したディレクトリを指定します。

   isibpeladminコマンドのdeployサブコマンドについては、“ISI リファレンス”の“BPEL機能”の“BPEL定義の配備/配備解除/一覧表示コマンド”の“deployサブコマンド”を参照してください。

   なお、以下の場合はビジネス・プロセス実行環境を複数作成して配備してください。

   - 同じBPELプロセス名のBPEL定義を複数配備する場合
   - BPEL定義からBPEL定義で作成したサービスをInvokeで呼出す場合

BPEL定義の反映

BPEL定義はビジネス・プロセス実行環境の運用中に配備を行うことができます。

ビジネス・プロセス実行環境が起動している状態でBPEL定義を配備した場合、BPEL定義は自動的にビジネス・プロセス実行環境に配備されます。

また、ビジネス・プロセス実行環境が停止している状態でBPEL定義を配備した場合、配備したBPEL定義は配備待ちとなり、ビジネス・プロセス実行環境の起動時に配備されます。

ポイント

BPEL定義を配備した場合、ビジネス・プロセス実行環境へのBPEL定義の配備状態により以下の2つの状態があります。

- 配備済み状態

  ビジネス・プロセス実行環境への配備が完了した状態です。

- 配備待ち状態

  ビジネス・プロセス実行環境への配備が未完了で、配備待ちの状態です。

  配備待ち状態のBPEL定義は次回ビジネス・プロセス実行環境を起動したときに、ビジネス・プロセス実行環境に配備されます。

BPEL定義の配備結果の確認

以下に、配備結果を確認する手順を説明します。

ビジネス・プロセス実行環境が起動している状態でBPEL定義を配備した場合

コマンドの実行結果を確認してください。

－ WSBPEL-52501が出力された場合

配備が完了しました。

使用例

```
isibpeladmin deploy --target ode BPEL_Sample001
INFO: WSBPEL-52501: 配備処理が完了しました: BPELプロセス名=BPELSample001
```

－ WSBPEL-52011が出力された場合

配備に失敗しました。イベントログまたはシステムログにエラーメッセージが出力されます。出力されているエラーメッセージから原因を特定し、対処を行ってください。

使用例

```
isibpeladmin deploy --target ode BPEL_Sample002
ERROR: WSBPEL-52011: 配備に失敗しました: BPELプロセス名=BPELSample002 ID=BPEL_Sample002_1297058215971
```

－ WSBPEL-52551が出力された場合

ビジネス・プロセス実行環境の起動後の最初の配備、またはビジネス・プロセス実行環境が高負荷な状態では、本メッセージが出力される場合があります。

使用例

```
isibpeladmin deploy --target ode BPEL_Sample001
WARNING: WSBPEL-52551: 配備結果を取得できませんでした: BPELプロセス名=BPELSample001
ID=BPEL_Sample001_1297057600598
```

isibpeladminコマンドのlistサブコマンドを実行し、BPELプロセス名の一覧を表示し、配備状態を確認してください。表示された場合、配備は完了しており配備済み状態です。表示されない場合、ビジネス・プロセス実行環境への定義の配備に時間がかかっている、または配備に失敗している可能性があります。イベントログまたはシステムログで配備結果を確認してください。

isibpeladminコマンドのlistサブコマンドについては、“ISIリファレンス”の“BPEL機能”の“BPEL定義の配備/配備解除/一覧表示コマンド”の“listサブコマンド”を参照してください。

ビジネス・プロセス実行環境が停止している状態でBPEL定義を配備した場合

コマンドの実行時に以下のメッセージが表示されます。

－ WSBPEL-52551: 配備結果を取得できませんでした

isibpeladminコマンドのlistサブコマンドを“--nodeploy”オプションを指定して実行します。配備したBPEL定義のBPELプロセス名が一覧に表示されることを確認してください。

使用例

```
isibpeladmin deploy --target ode BPEL_Sample001
WARNING: WSBPEL-52551: 配備結果を取得できませんでした: BPELプロセス名=BPELSample001
ID=BPEL_Sample001_1297057600598
isibpeladmin list --nodeploy ode
BPELSample001
```

ビジネス・プロセス実行環境を起動すると、BPEL定義がビジネス・プロセス実行環境に反映され、配備済み状態となります。

isibpeladminコマンドのlistサブコマンドを実行し、BPELプロセス名の一覧を表示し、配備状態を確認してください。表示された場合、配備は完了しており配備済み状態です。表示されない場合、ビジネス・プロセス実行環境への定義の反映に時間がかかっている、または配備に失敗している可能性があります。イベントログまたはシステムログで配備結果を確認してください。

イベントログ/システムログの確認方法

コマンドが出力したメッセージのパラメタの“BPELプロセス名”と“ID”は、ビジネス・プロセス実行環境へのBPEL定義の配備状態の確認に使用します。配備状態の確認を行うときは、イベントログまたはシステムログから、これらの値と同一の値がパラメタとして出力されているメッセージを確認してください。

ビジネス・プロセス実行環境へのBPEL定義の配備が完了している場合は、WSBPEL-44402の情報メッセージが出力されます。失敗している場合は、エラーメッセージが出力されます。出力されているエラーメッセージから原因を特定し、対処を行ってください。

配備が完了後、BPEL起動アプリケーション開発者に、BPEL定義の公開用WSDL定義を提供します。公開用WSDL定義は、isibpelwsdlexportコマンドを使用して作成します。

isibpelwsdlexportコマンドは、WSDL定義取得対象のBPEL定義が配備されているビジネス・プロセス実行環境が存在するマシン上で実行します。

isibpelwsdlexportの詳細は、“ISI リファレンス”の“BPEL機能”の“セットアップコマンド”の“isibpelwsdlexport”を参照してください。

E.3.4 ビジネス・プロセス実行環境のチューニング

配備したBPEL定義に応じて、ビジネス・プロセス実行環境のチューニングを行います。BPEL定義、Webサービスが要求/応答型の場合、タイムアウトを設定できます。以下にチューニングの方法を示します。

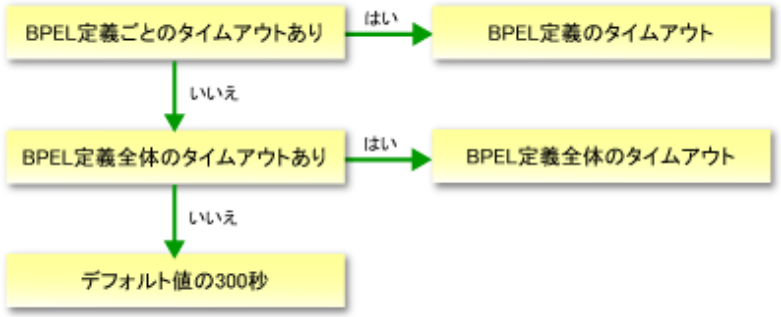
BPEL定義の実行時間のタイムアウト

BPEL定義の実行時間のタイムアウトの設定には、以下の2通りの方法があります。

- BPEL定義ごとに、タイムアウトを設定
- BPEL定義全体で共通のタイムアウトを設定

BPEL定義の実行時間のタイムアウトは以下のように判定します。

図E.2 BPEL定義のタイムアウト判定の流れ



ポイント

BPEL定義の実行時間のタイムアウトが発生した場合は、BPEL起動アプリケーションにSOAP Faultを通知します。BPEL定義は最後まで実行し、システムログにWSBPEL-40208のメッセージを出力します。

■BPEL定義ごとのタイムアウトの設定

BPEL定義ごとのタイムアウトの設定は、定義ファイルを使用して行います。

定義ファイルのファイル名を“bpeltimeout.properties”として、以下の位置に格納してください。

W

```
[Java EE共通ディレクトリ]¥domains¥interstage¥applications¥j2ee-modules¥ode¥WEB-INF¥conf
```

S L

```
[Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/conf
```

定義ファイルは、プロパティファイル形式(“キー”:“値”の形式)となっており、キーがBPEL定義のプロセス名で、値が秒単位の時間です。指定可能な値は、1から32767であり、それ以外の値が指定された場合には、指定されていないものとして扱われます。

以下に定義ファイルの例を示します。

```
BPEL001: 10
BPEL002: 10
BPEL003: 10
BPEL004: 20
```

タイムアウト時間は、BPEL定義の設計時に決めた値とする方法と、実際に動作させて測定した値からタイムアウトを決定する方法があります。

後者は以下の方法で求めることができます。

- ビジネス・プロセス実行機能に、測定対象のBPEL定義を配備します。
- 性能ログを採取するように設定します。性能ログについては、“BPEL機能のアプリケーション開発”の“BPEL定義の開発”の“E.3.5 ビジネス・プロセス実行環境のログ設定”を参照してください。
- BPEL定義を複数回実行します。
- 性能ログから以下の値を求めます。

  Webサービス呼び出しの最大時間

  BPEL定義実行時間から、Webサービス呼び出し時間を除いた時間の最大時間
- 上記の最大時間の和をタイムアウト時間とします。

定義ファイルはビジネス・プロセス実行環境の起動時と、BPEL定義の配備時に読み込まれますので、定義ファイルを作成、更新した場合には、適宜ビジネス・プロセス実行環境の再起動を行ってください。

S L

強化セキュリティモードの場合は、他のInterstage運用グループのユーザーが操作できるよう所属するグループをInterstage運用グループに変更します。

```
chgrp isgroup [Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/conf/
bpeltimeout.properties
```

上記は以下の場合の例です。

- Interstage運用グループがisgroupの場合。
- ビジネス・プロセス実行機能のアプリケーション名がodeの場合

■BPEL定義全体で共通のタイムアウトの設定

BPEL定義全体で共通のタイムアウトの設定は、定義ファイルを使用して行います。

定義ファイルのファイル名を“bpeltimeoutall.properties”として、以下の位置に格納してください。

W

```
[Java EE共通ディレクトリ]¥domains¥interstage¥applications¥j2ee-modules¥ode¥WEB-INF¥conf
```

S L

```
[Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/conf
```

定義ファイルは、プロパティファイル形式(“キー”:“値”の形式)となっており、キーが”TIMEOUT”で、値が秒単位の時間です。指定可能な値は、1から32767であり、それ以外の値が指定された場合には、指定されていないものとして扱われます。

以下に定義ファイルの例を示します。

```
TIMEOUT: 20
```

本タイムアウトは、BPEL定義の設計時に決めた値が、複数のBPEL定義で同じ場合に使用してください。

定義ファイルはビジネス・プロセス実行環境の起動時に読み込まれますので、定義ファイルを作成、更新した場合には、適宜ビジネス・プロセス実行環境の再起動を行ってください。

S L

強化セキュリティモードの場合は、他のInterstage運用グループのユーザーが操作できるよう所属するグループをInterstage運用グループに変更します。

```
chgrp isgroup [Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/conf/
bpeltimeoutall.properties
```

上記は以下の場合の例です。

- Interstage運用グループがisgroupの場合。
- ビジネス・プロセス実行機能のアプリケーション名がodeの場合

Webサービスのタイムアウト

BPEL定義のInvoke要素が呼び出すWebサービスの実行時間のタイムアウトをWebサービスごとに設定します。タイムアウトの設定は、定義ファイルを使用して行います。

定義ファイルのファイル名を“webtimeout.properties”として、以下の位置に格納してください。

W

```
[Java EE共通ディレクトリ]¥domains¥interstage¥applications¥j2ee-modules¥ode¥WEB-INF¥conf
```

S L

```
[Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/conf
```

定義ファイルは、プロパティファイル形式(“キー”:“値”の形式)となっており、キーがWebサービスのURL、値が秒単位の時間です。指定可能な値は、1から32767であり、それ以外の値が指定された場合には、指定されていないものとして扱われます。タイムアウトを設定していない場合のタイムアウトは、600秒です。

以下に定義ファイルの例を示します。

```
http¥://webservice.fujitsu.com/service/G01_U001: 5
http¥://webservice.fujitsu.com/service/G01_U002: 5
http¥://webservice.fujitsu.com/service/G01_U010: 10
```

定義ファイルはビジネス・プロセス実行環境の起動時に読み込まれますので、定義ファイルを作成、更新した場合には、適宜ビジネス・プロセス実行環境の再起動を行ってください。

S L

強化セキュリティモードの場合は、他のInterstage運用グループのユーザーが操作できるよう所属するグループをInterstage運用グループに変更します。

```
chgrp isgroup [Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/conf/
webtimeout.properties
```

上記は以下の場合の例です。

- Interstage運用グループがisgroupの場合。
- ビジネス・プロセス実行機能のアプリケーション名がodeの場合

ビジネス・プロセス実行環境の運用時に必要なメモリ容量

ビジネス・プロセス実行環境の運用時に必要なメモリ容量は以下のとおりです。

```
メモリ所要量
=IJServerクラスタのサーバインスタンスの動作に必要なメモリ
+ビジネス・プロセス実行環境が必要とするメモリ
+BPEL定義実行に必要とするメモリ
```

このメモリ所要量が、デフォルトのヒープサイズ(512MB)より多い場合は、IJServerクラスタの設定のJVMオプションで、ヒープサイズを指定してください。詳細は、Interstage Application Serverの“Java EE運用ガイド”の“Java EE機能のチューニング”の“IJServerクラスタのチューニング”の“Java EE機能のチューニング”を参照してください。

■IJServerクラスタのサーバインスタンスの動作に必要なメモリ

226MB必要となります。

■ビジネス・プロセス実行環境が必要とするメモリ

21MB必要となります。

■BPEL定義実行に必要とするメモリ

以下の要因により必要なメモリの量は変化します。

- BPEL定義中の要素や変数などの定義の数
- BPEL定義実行時の入力パラメタ、出力パラメタのサイズ
- BPEL定義から呼び出すWebサービスに受け渡すメッセージ、受け渡されるメッセージのサイズ

例として以下のBPEL定義を10多重で実行した場合、メモリは377Mバイト必要になります。このときの入力パラメタのサイズは348バイト、出力パラメタのサイズは388バイト、BPEL定義から呼び出すWebサービスに受け渡すメッセージのサイズは295バイト、受け渡されるメッセージのサイズは388バイトです。

```
<!-- PerformanceBPEL BPEL Process [Generated by the Eclipse BPEL Designer] -->
<bpel:process name="PerformanceBPEL"
         targetNamespace="http://sample.bpel.org/bpel/sample"
         suppressJoinFailure="yes"
         xmlns:tns="http://sample.bpel.org/bpel/sample"
         xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
         xmlns:ns1="http://ws1.performance.bpel.interstage.fujitsu.com" xmlns:ns2="http://
ws2.performance.bpel.interstage.fujitsu.com">

    <!-- Import the client WSDL -->
    <bpel:import namespace="http://ws2.performance.bpel.interstage.fujitsu.com" location="PerformanceWebService2.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import namespace="http://ws1.performance.bpel.interstage.fujitsu.com" location="PerformanceWebService1.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
    <bpel:import location="PerformanceBPELArtifacts.wsdl" namespace="http://sample.bpel.org/bpel/sample"
                importType="http://schemas.xmlsoap.org/wsdl/" />

    <!-- ================================================================= -->
    <!-- PARTNERLINKS                                                      -->
    <!-- List of services participating in this BPEL process               -->
    <!-- ================================================================= -->
    <bpel:partnerLinks>
        <!-- The 'client' role represents the requester of this service. -->
        <bpel:partnerLink name="client"
                    partnerLinkType="tns:PerformanceBPEL"
                    myRole="PerformanceBPELProvider"
                    />
        <bpel:partnerLink name="WS1PLink" partnerLinkType="tns:WS1PLinkType" partnerRole="WS1Role"></bpel:partnerLink>
        <bpel:partnerLink name="WS2PLink" partnerLinkType="tns:WS2PLinkType" partnerRole="WS2Role"></bpel:partnerLink>
    </bpel:partnerLinks>

    <!-- ================================================================= -->
    <!-- VARIABLES                                                         -->
    <!-- List of messages and XML documents used within this BPEL process  -->
    <!-- ================================================================= -->
    <bpel:variables>
        <!-- Reference to the message passed as input during initiation -->
        <bpel:variable name="input"
                  messageType="tns:PerformanceBPELRequestMessage"/>

        <!--
```

```
         Reference to the message that will be returned to the requester
         -->
        <bpel:variable name="output"
                  messageType="tns:PerformanceBPELResponseMessage"/>
        <bpel:variable name="WS1Request" messageType="ns1:apprun1Request"></bpel:variable>
        <bpel:variable name="WS1Response" messageType="ns1:apprun1Response"></bpel:variable>
        <bpel:variable name="WS2Request" messageType="ns2:apprun2Request"></bpel:variable>
        <bpel:variable name="WS2Response" messageType="ns2:apprun2Response"></bpel:variable>
    </bpel:variables>

    <!-- ================================================================= -->
    <!-- ORCHESTRATION LOGIC                                               -->
    <!-- Set of activities coordinating the flow of messages across the    -->
    <!-- services integrated within this business process                  -->
    <!-- ================================================================= -->
    <bpel:sequence name="main">

        <!-- Receive input from requester.
             Note: This maps to operation defined in PerformanceBPEL.wsdl
             -->
        <bpel:receive name="receiveInput" partnerLink="client"
                 portType="tns:PerformanceBPEL"
                 operation="process" variable="input"
                 createInstance="yes"/>

        <!-- Generate reply to synchronous request -->
        <bpel:assign validate="no" name="Assign">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:apprun1 xmlns:impl="http://
ws1.performance.bpel.interstage.fujitsu.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:in0></impl:in0>
</impl:apprun1>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="WS1Request" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:id_in]]></
bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="WS1Request">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[ns1:in0]]></
bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Invoke" partnerLink="WS1PLink" operation="apprun1" portType="ns1:PerformanceWebService1"
inputVariable="WS1Request" outputVariable="WS1Response"></bpel:invoke>
        <bpel:assign validate="no" name="Assign1">
            <bpel:copy>
                <bpel:from>
                    <bpel:literal xml:space="preserve"><impl:apprun2 xmlns:impl="http://
ws2.performance.bpel.interstage.fujitsu.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <impl:in0></impl:in0>
</impl:apprun2>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="WS2Request" part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
```

```
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:id_in]]></
bpel:query>
                </bpel:from>
                <bpel:to part="parameters" variable="WS2Request">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[ns2:in0]]></
bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Invoke1" partnerLink="WS2PLink" operation="apprun2" portType="ns2:PerformanceWebService2"
inputVariable="WS2Request" outputVariable="WS2Response"></bpel:invoke>
        <bpel:assign validate="no" name="Assign2">
            <bpel:copy>
                <bpel:from>
                  <bpel:literal xml:space="preserve"><tns:PerformanceBPELResponse xmlns:tns="http://sample.bpel.org/bpel/
sample" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:result></tns:result>
  <tns:id_out></tns:id_out>
</tns:PerformanceBPELResponse>
</bpel:literal>
                </bpel:from>
                <bpel:to variable="output" part="payload"></bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from>
                  <![CDATA[concat($WS1Response.parameters/ns1:apprun1Return,$WS2Response.parameters/ns2:apprun2Return)]]>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:result]]></
bpel:query>
                </bpel:to>
            </bpel:copy>
            <bpel:copy>
                <bpel:from part="payload" variable="input">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:id_in]]></
bpel:query>
                </bpel:from>
                <bpel:to part="payload" variable="output">
                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:id_out]]></
bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:reply name="replyOutput"
               partnerLink="client"
               portType="tns:PerformanceBPEL"
               operation="process"
               variable="output"
               />
    </bpel:sequence>
</bpel:process>
```

E.3.5 ビジネス・プロセス実行環境のログ設定

ビジネス・プロセス実行環境のログについて説明します。ビジネス・プロセス実行環境では、BPEL定義の実行時の状態をログファイルに出力します。

ログは、以下のような場合に有効です。

- BPEL定義の実行時間の把握
- ビジネス・プロセス実行環境を停止するときの、動作中のBPEL定義の確認
- ビジネス・プロセス実行環境が強制停止したときの、動作中だったBPEL定義の確認

・ シミュレーション結果の参照

ログは、採取する情報を変更することができます。変更は、JavaVMのオプションで指定します。

JavaVMオプションの設定は、Interstage Java EE管理コンソール、またはasadminコマンドを使用して行います。詳細はInterstage Application Serverの“Interstage Java EE管理コンソールヘルプ”、または“Java EE運用ガイド”の”Java EE運用コマンド”の“asadmin”を参照してください。

JavaVMオプションの指定方法を以下に示します。

IJServerクラスタのJavaVMオプションとして、以下のように指定してください。

```
-Dcom.fujitsu.interstage.isi.bpel.loglevel={0|1|2|3}
```

本指定がない、または無効な値を指定した場合は、0を指定したものとして動作します。

指定に対応する、ログの名称と用途は以下のとおりです。

表E.1 ログの種類

loglevel	名称	用途
0	実行ログ	実行中のBPEL定義の確認
1	性能ログ	タイムアウト値設定の基礎情報の採取
2	詳細な実行ログ	実行中のBPEL定義の詳細な確認
3	シミュレーションログ	シミュレーション結果の確認

ログは、ビジネス・プロセス実行機能を配備したIJServerクラスタのカレントディレクトリに出力されます。IJServerクラスタのカレントディレクトリについては、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの提供機能”の“IJServerクラスタ”にある“IJServerクラスタのファイル構成”を参照してください。

ログファイル名は、“isibpel.log”です。ログは、初期状態では日単位でローテーションしています。ローテーションを変更するときは、ログ定義ファイルを修正してください。ログ定義ファイルについては、“ログ定義ファイル”を参照してください。ログファイルは自動的に削除されないため、定期的にローテーション後のログファイルを別のディスクに退避してください。

ログの出力内容は以下のとおりです。

表E.2 ログの出力内容

項番	ログの項目	ログ種別			
		実行ログ	性能ログ	詳細な実行ログ	シミュレーションログ
1	日時	○	○	○	○
2	ラベル	○	○	○	○
3	エラー種別	○	○	○	○
4	メッセージID	○	○	○	○
5	BPEL定義のプロセス名	○	○	○	○
6	通番（※1）	○	○	○	○
7	実行識別名（※2）	－	－	－	－
8	要素種別	Receive Reply	Receive Reply Invoke	すべて	すべて
9	要素の名前	－	○	○	○
10	開始/終了	○	○	○	○
11	詳細情報	－	－	－	○

※1 通番は、BPEL定義の実行開始ごとに増加する値です。2147483647を超えた場合0に戻ります。またIJServerクラスタのサーバインスタンスを停止した場合、0に戻ります。

※2 実行識別名は、本バージョンでは未対応であり、“-----”と表示されます。

実行ログ

実行ログでは、BPEL定義の実行の開始(Receiveの開始)と終了(Replyの終了)を採取します。要求/応答型のBPEL定義を実行した場合のログの例を以下に示します。

```
2010-11-09 14:26:41.475 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Receive ----- START -----
2010-11-09 14:26:42.460 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Reply ----- END -----
```

性能ログ

性能ログでは、BPEL定義の実行の開始(Receiveの開始)と終了(Replyの終了)、およびWebサービスの実行(Invokeの開始と終了)を採取します。要求/応答型のBPEL定義を実行した場合のログの例を以下に示します。

```
2010-11-09 14:26:41.475 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Receive ----- START -----
2010-11-09 14:26:41.865 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Invoke Invoke01 START -----
2010-11-09 14:26:42.422 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Invoke Invoke01 END -----
2010-11-09 14:26:42.460 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Reply ----- END -----
```

詳細な実行ログ

詳細な実行ログでは、BPEL定義の実行の開始(Receiveの開始)と終了(Replyの終了)、およびその他の要素(開始と終了)を採取します。Ifの場合、終了のログは実行した条件判定の数だけ出力します。要求/応答型のBPEL定義を実行した場合のログの例を以下に示します。

```
2010-11-09 14:26:41.475 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Receive ----- START -----
2010-11-09 14:26:41.737 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Sequence main START -----
2010-11-09 14:26:41.760 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Assign Assign01 START -----
2010-11-09 14:26:41.810 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Assign Assign01 END -----
2010-11-09 14:26:41.865 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Invoke Invoke01 START -----
2010-11-09 14:26:42.422 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Invoke Invoke01 END -----
2010-11-09 14:26:42.431 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Assign Assign02 START -----
2010-11-09 14:26:42.433 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Assign Assign02 END -----
2010-11-09 14:26:42.455 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Sequence main END -----
2010-11-09 14:26:42.460 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Reply ----- END -----
```

シミュレーションログ

シミュレーションログでは、詳細な実行ログに加え、各要素の開始と終了時の詳細な情報を採取します。詳細な情報は、1つの要素を実行するごとにシミュレーションログファイルを作成して出力します。シミュレーションログファイルは、詳細な実行ログと同じ位置に通番でサブディレクトリを作成し、その下に作成します。シミュレーションログファイルのファイル名は、以下のルールで作成します。

BPEL定義実行開始からの通番 + "_" + BPEL定義名 + "_" + 要素名 + "_" + 要素の名前 + ".log"

シミュレーションログのファイル名が、詳細な実行ログの詳細情報の項目に入ります。

シミュレーションログは自動的に削除されないため、定期的に別のディスクに退避してください。

シミュレーションログのファイル名やフルパスの長さがOSの最大長を超過した場合、イベントログまたはシステムログにエラーを出力して、シミュレーションログを採取しません。BPELプロセス名や要素の名前を短くしてください。



ファイル名の最大長は255文字、フォルダまでのフルパス(ドライブ名を含みます)の最大長は247文字、ファイルまでのフルパス(ドライブ名を含みます)の最大長は259文字です。



ファイル名の最大長は255バイト、フルパスの最大長は1023バイトです。



ファイル名の最大長は255バイト、フルパスの最大長は4095バイトです。

採取する情報は以下のとおりです。

表E.3 シミュレーションログの採取情報一覧

分類	要素名	要素の実行前	要素の実行後
基本アクティビティ	Invoke	○ 呼び出し時のパラメタ	○ 復帰値/Faultの情報(要求/応答型の場合)
	Receive	○ 受信したメッセージ情報	-
	Reply	-	○ 返信するメッセージ情報
	Assign	○ 代入前の値	○ 代入後の値
	Throw	-	○ Faultの情報
構造化アクティビティ	Sequence	-	-
	Flow	-	-
	If	-	○ 条件式の結果
	While		
その他の要素	Variables	-	-
	PartnerLinks	-	-
	FaultHandlers	○ Faultの情報	-
	CorrelationSets	-	-

○:出力あり

ー:出力なし

BPEL定義とWebサービスの両方とも要求/応答型の場合の各要素のシミュレーションログファイルの例を以下に示します。例はBPEL起動アプリケーション、Webアプリケーションともに Interstage Studio Java EEで作成した場合を記載しています。なお、SOAPメッセージ形式の場合の例は、整形して記載しています。

■Invoke

呼び出すサービスに受け渡すメッセージ、サービスから受け渡されるメッセージをSOAPメッセージ形式で出力します。

実行前

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <getCountryData xmlns="http://sample/">
      <arg0>11.0</arg0>
    </getCountryData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

実行後

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:getCountryDataResponse xmlns:ns2="http://sample/"/>
  </S:Body>
</S:Envelope>
```

■Receive

BPEL実行開始時の入力パラメタをSOAPメッセージ形式で出力します。

実行前

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <BPELprocessRequest xmlns="http://sample.bpel.org/bpel/sample">
      <input>hello</input>
    </BPELprocessRequest>
  </S:Body>
</S:Envelope>
```

■Reply

BPEL実行完了時の出力パラメタをSOAPメッセージ形式で出力します。

実行後

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <BPELprocessResponse xmlns="http://sample.bpel.org/bpel/sample">
      <tns:result xmlns:tns="http://sample.bpel.org/bpel/sample">test</tns:result>
    </BPELprocessResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

■Assign

1行に1つの変数の情報を採取します。“=”の左側に変数の名前、右側に値を出力します。変数の型が“要素宣言”または“メッセージ”の場合、変数の名前は型の構造を“/”で区切る形式となります。変数の値は、数値、文字列、またはXML形式となります。

実行前

```
WS1Request/parameters=
WS1Request/parameters/ns1:in0=
Variable001=aaa
```

実行後

```
WS1Request/parameters=
WS1Request/parameters/ns1:in0=aaaaa
Variable001=bbb
```

■Throw

1行目に障害名、2行目に障害変数の名前を採取します。

実行後

```
{http://docs.oasis-open.org/wsbpel/2.0/process/executable}xsltInvalidSource
MyMessageVariable
```

■If / While

条件判定の処理を行うごとにファイルを分けて採取します。If要素に、Else/Else If が存在する場合、その数に対応して条件判定を行うため、その数だけファイルを採取します。条件判定の結果と、条件判定の式を採取します。

実行後

```
False, {OXPath10Expression $input.payload/tns:input = 'A'}
```

■FaultHandler

1行目に障害名、2行目に変数名を採取します。障害ハンドラを追加していない場合でも、採取します。

実行後

```
{http://docs.oasis-open.org/wsbpel/2.0/process/executable}xsltInvalidSource
Variable001
```

ログ定義ファイル

ログ定義ファイルでは、ビジネス・プロセス実行機能のログ(実行ログ・性能ログ・詳細な実行ログ・シミュレーションログ)の動作を定義しています。本ファイルを修正することでログの動作を変更することができます。本ファイルは、log4jのログ定義です。ログ定義ファイルは以下の位置にあります。



```
[Java EE共通ディレクトリ]¥domains¥interstage¥applications¥j2ee-modules¥ode¥WEB-INF¥classes¥log4j.xml
```



```
[Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/classes/log4j.xml
```

ログ定義ファイルの、ビジネス・プロセス実行機能のログの動作を定義している部分を以下に示します。

```
<appender name="FILEEXEC" class="org.apache.log4j.DailyRollingFileAppender">
  <param name="file" value="isibpel.log" />
  <param name="threshold" value="info"/>
  <param name="append" value="true" />
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss.SSS} FSP_INTS-ESI_WSBPEL-SV INFO %m%n" />
  </layout>
</appender>
```

E.3.6 ビジネス・プロセス実行環境からの配備解除

配備したBPEL定義が不要となった場合、または配備したBPEL定義を新しいものに置き換える場合は、不要となったBPEL定義を配備解除します。配備解除は以下の手順で実施します。

1. 配備解除するBPEL定義を利用した業務の終了確認

   配備解除するBPEL定義を利用した業務が実行されていないことを確認してください。BPEL機能を利用した業務が実行中か否かは、実行ログを参照してください。実行ログについては、“BPEL機能のアプリケーション開発”の“BPEL定義の開発”の“ビジネス・プロセス実行機能のログ設定”の“実行ログ”を参照してください。

2. BPEL定義の配備解除

   isibpeladminコマンドのundeployサブコマンドを使用してBPEL定義を配備解除します。

   isibpeladminコマンドのundeployサブコマンドについては、“ISIリファレンス”の“BPEL機能”の“BPEL定義の配備/配備解除/一覧表示コマンド”の“undeployサブコマンド”を参照してください。

BPEL定義の反映

BPEL定義はビジネス・プロセス実行環境の運用中に配備解除を行うことができます。

ビジネス・プロセス実行環境が起動している状態でBPEL定義を配備解除した場合、BPEL定義は自動的にビジネス・プロセス実行環境から配備解除されます。

また、ビジネス・プロセス実行環境が停止している状態でBPEL定義を配備解除した場合、配備解除したBPEL定義は配備解除待ちとなり、ビジネス・プロセス実行環境の起動時に配備解除されます。



BPEL定義を配備解除した場合、ビジネス・プロセス実行環境からのBPEL定義の配備解除状態により以下の2つの状態があります。

ー 配備解除済み状態

ビジネス・プロセス実行環境からの配備解除が完了した状態です。

ー 配備解除待ち状態

ビジネス・プロセス実行環境からの配備解除が未完了で、配備解除待ちの状態です。

配備解除待ち状態のBPEL定義は次回ビジネス・プロセス実行環境を起動したときに、ビジネス・プロセス実行環境から配備解除されます。

BPEL定義の配備解除結果の確認

以下に配備解除結果を確認する手順を説明します。

ビジネス・プロセス実行環境が起動している状態でBPEL定義を配備解除した場合

コマンドの実行結果を確認してください。

- WSBPEL-52502が出力された場合

  配備解除が完了しました。

- WSBPEL-52552が出力された場合

  ビジネス・プロセス実行環境が高負荷な状態では、本メッセージが出力される場合があります。

  isibpeladminコマンドのlistサブコマンドを実行し、BPELプロセス名の一覧を表示し、配備状態を確認してください。表示されない場合、配備解除は完了しており配備解除済み状態です。表示された場合、ビジネス・プロセス実行環境への定義の配備解除に時間がかかっている、または配備解除に失敗している可能性があります。イベントログまたはシステムログで配備解除結果を確認してください。

  isibpeladminコマンドのlistサブコマンドについては、“ISI リファレンス”の“BPEL機能”の“BPEL定義の配備/配備解除/一覧表示コマンド”の“listサブコマンド”を参照してください。

ビジネス・プロセス実行環境が停止している状態でBPEL定義を配備解除した場合

コマンドの実行時に以下のメッセージが表示されます。

- WARNING: WSBPEL-52552: 配備解除結果を取得できませんでした

  ビジネス・プロセス実行環境を起動すると、BPEL定義がビジネス・プロセス実行環境に反映され、配備解除済み状態となります。

  isibpeladminコマンドのlistサブコマンドを実行し、BPELプロセス名の一覧を表示し、配備状態を確認してください。表示されない場合、配備解除は完了しており配備解除済み状態です。表示された場合、ビジネス・プロセス実行環境への定義の反映に時間がかかっている、または配備解除に失敗している可能性があります。イベントログまたはシステムログで配備解除結果を確認してください。

コマンド実行例

コマンドが出力したメッセージのパラメタの“BPELプロセス名”と“ID”は、ビジネス・プロセス実行環境からのBPEL定義の配備解除状態の確認に使用します。配備解除状態の確認を行うときは、イベントログまたはシステムログから、これらの値と同一の値がパラメタとして出力されているメッセージを確認してください。

ビジネス・プロセス実行環境からのBPEL定義の配備解除が完了している場合は、WSBPEL-44401の情報メッセージが出力されます。失敗している場合は、エラーメッセージが出力されます。出力されているエラーメッセージから原因を特定し、対処を行ってください。

```
isibpeladmin undeploy --target ode BPEL_Sample001
WARNING: WSBPEL-52552: 配備解除結果を取得できませんでした: BPELプロセス名=BPEL_Sample001
ID=BPEL_Sample001_128955855828
```

E.4 BPEL起動アプリケーションの開発

BPEL起動アプリケーションの開発では、BPEL定義のWSDL作成、BPEL起動アプリケーションの作成、およびBPEL起動アプリケーションの配備を行います。次に各方法について説明します。

E.4.1 BPEL定義のWSDL取得

BPEL起動アプリケーションが呼び出すBPEL定義のWSDLを取得します。

以下のどちらかの方法で、BPEL起動アプリケーションが呼び出すBPEL定義のWSDL定義を取得します。

- ビジネス・プロセス実行環境管理者からBPEL定義の公開用WSDL定義を受け取る。
- BPEL定義の作成者からBPEL定義のWSDL定義を受け取る。

  BPEL起動アプリケーションを実行する場合は、ビジネス・プロセス実行環境管理者から公開用WSDLを受け取り、スタブを生成しなおしてください。

E.4.2 BPEL起動アプリケーションの作成

BPEL起動アプリケーションは、Webサービスクライアントとして作成します(ビジネス・プロセス実行機能はWebサービスアプリケーションの位置づけとなります)。

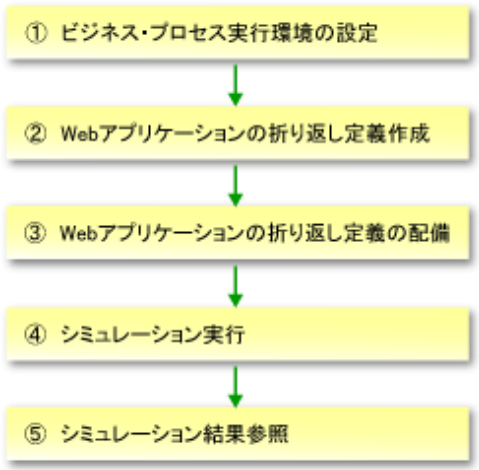
Java EE環境で動作させるWebサービスクライアントの作成方法については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発”の“アプリケーションの作成方法” の“Webサービスクライアントアプリケーションの開発”を参照してください。一方向型(one-way)のBPEL定義に対する起動アプリケーションは、Java EE環境のみ実行できます。

J2EE環境で動作させるWebサービスクライアントの作成方法については、“Interstage Application Server J2EE ユーザーズガイド(旧版互換)”の“Webサービス編”の“Webサービスの開発” の“Webサービスを呼び出す場合(クライアント機能)の開発”を参照してください。作成の際に使用するWebサービスのインタフェース情報は、“E.4.1 BPEL定義のWSDL取得”で作成したWSDLを使用してください。

E.5 BPEL定義のシミュレーション実行

BPEL定義のシミュレーションは、以下の手順で行います。

図E.3 BPEL定義シミュレーション実行の流れ



ビジネス・プロセス実行環境の設定

BPEL定義のシミュレーションを行う場合、ビジネス・プロセス実行機能を配備したIJServerクラスタの設定を変更します。JavaVMのオプションを以下のように指定します。

```
-Dcom.fujitsu.interstage.isi.bpel.loglevel=3
```

IJServerクラスタの設定変更は、Interstage Java EE管理コンソール、またはasadminコマンドを使用して行います。詳細はInterstage Application Serverの”Interstage Java EE管理コンソールヘルプ”、または”Java EE運用ガイド”の”Java EE運用コマンド”の”asadmin”を参照してください。

Webアプリケーションの折り返し定義作成

Webアプリケーションの折り返し定義とは、Invoke要素がWebアプリケーションを呼び出すことなく、あたかも呼び出したように振舞うための定義です。Webアプリケーションの折り返し定義は、Webアプリケーションごとに作成します。本定義を作成していない場合は、本来のWebアプリケーションの呼び出しとなります。isibeplgenturnback コマンドを使用して、WSDL定義から折り返し定義のひな型を作成します。

isibeplgenturnback コマンドについては、“ISI リファレンス”の“BPEL機能”の“セットアップコマンド”の“isibpelgenturnback”を参照してください。

コマンド実行すると、カレントディレクトリに折り返し定義を出力します。折り返し定義のファイル名は、Webアプリケーションのサービスエンドポイントの URLをURLエンコードした文字列に、”@”、オペレーション名、および”.xml”を連結したものです。WSDL定義に複数のオペレーションが定義されている場合には、オペレーションごとに折り返し定義のファイルが生成されます。Webアプリケーションが一方向型(one-way)のサービスの場合、折り返し定義のファイルは空ファイルとなります。

実行例



```
isibpelgenturnback C:¥WebService001.wsdl
Retrieving document at 'C:¥WebService001.wsdl'.
Generating a file for 'execute' operation in 'http://localhost/NOPService001/services/NOPService001'.
Generated the file of the name 'C:¥http%3A%2F%2Flocalhost%2FNOPService001%2Fservices%2FNOPService001@execute.xml'.
```



```
isibpelgenturnback /tmp/WebService001.wsdl
Retrieving document at '/tmp/WebService001.wsdl'.
Generating a file for 'execute' operation in 'http://localhost/NOPService001/services/NOPService001'.
Generated the file of the name '/tmp/http%3A%2F%2Flocalhost%2FNOPService001%2Fservices%2FNOPService001@execute.xml'.
```

折り返し定義の内容は、SOAPメッセージそのものです。ひな型では値が設定されていないので、テキストエディタなどで値を設定してください。WSDL定義と、生成した折り返し定義ファイルの例を示します。

WSDL定義

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://sample" xmlns:impl="http://sample" xmlns:intf="http://sample"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Interstage Web Service version: 1.2 (r6010:6011)
Built on Jun 04, 2009 (02:22:19 JST)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://sample" xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="execute">
    <complexType>
     <sequence>
      <element name="in0" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="executeResponse">
    <complexType>
     <sequence>
      <element name="executeReturn" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>
 <wsdl:message name="executeRequest">
  <wsdl:part element="impl:execute" name="parameters"/>
 </wsdl:message>
```

```
<wsdl:message name="executeResponse">
 <wsdl:part element="impl:executeResponse" name="parameters"/>
</wsdl:message>
<wsdl:portType name="NOPService001">
 <wsdl:operation name="execute">
  <wsdl:input message="impl:executeRequest" name="executeRequest"/>
  <wsdl:output message="impl:executeResponse" name="executeResponse"/>
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="NOPService001PortSoapBinding" type="impl:NOPService001">
 <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <wsdl:operation name="execute">
  <wsdlsoap:operation soapAction="execute"/>
  <wsdl:input name="executeRequest">
   <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="executeResponse">
   <wsdlsoap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:service name="NOPService001Service">
 <wsdl:port binding="impl:NOPService001PortSoapBinding" name="NOPService001Port">
  <wsdlsoap:address location="http://localhost/NOPService001/services/NOPService001"/>
 </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

折り返し定義

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <executeResponse xmlns="http://sample">
         <executeReturn></executeReturn>
      </executeResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Webアプリケーションの折り返し定義の配備

ビジネス・プロセス実行環境への配備は、以下の手順で行います。

以下の位置に折り返し定義をコピーしてください。



```
[Java EE共通ディレクトリ]¥domains¥interstage¥applications¥j2ee-modules¥ode¥WEB-INF¥simulation
```



```
[Java EE共通ディレクトリ]/domains/interstage/applications/j2ee-modules/ode/WEB-INF/simulation
```

コピーしたファイルは、強化セキュリティモードの場合は、他のInterstage運用グループのユーザーが操作できるよう以下のとおり変更します。

- 所属するグループをInterstage運用グループに変更

```
chgrp isgroup http%3A%2F%2Flocalhost%2FNOPService001%2Fservices%2FNOPService001.xml
```

上記は以下の場合の例です。

ー 折り返し定義のファイル名が、http%3A%2F%2Flocalhost%2FNOPService001%2Fservices%2FNOPService001.xmlの場合。

ー Interstage運用グループがisgroupの場合。

Webアプリケーションの折り返し定義の内容を頻繁に更新しテストを行う場合には、以下の設定を行うことを推奨します。

・ Interstage運用グループの書込み権限を付加

```
chmod g+w http%3A%2F%2Flocalhost%2FNOPService001%2Fservices%2FNOPService001.xml
```

上記は以下の場合の例です。

ー 折り返し定義のファイル名が、http%3A%2F%2Flocalhost%2FNOPService001%2Fservices%2FNOPService001.xmlの場合。

■シミュレーション実行

シミュレーションの実行は、BPEL起動アプリケーションの実行で行います。

シミュレーションを実行後、折り返し定義が不要となったときには、配備先の折り返し定義のファイルを削除してください。削除後、必要ならばビジネス・プロセス実行環境を再起動してください。

■シミュレーション結果参照

シミュレーション結果は、ログとして出力します。“BPEL機能のアプリケーション開発”の“BPEL定義の開発”の“ビジネス・プロセス実行機能のログ設定”の”シミュレーションログ”を参照してください。

ポイント

Webアプリケーションの折り返し定義ファイルを配備したときは、ログ設定をシミュレーションログに指定してください。シミュレーションログ以外の値を指定している場合、ビジネス・プロセス実行基盤の動作が異常となる場合があります。以下の動作となります。

ー IJServerクラスタの起動時

イベントログまたはシステムログに以下のメッセージが出力されます。

```
WARN:WSBPEL-47218: Simulation file was found when the loglevel is not 3.
```

ー BPEL定義を起動し、折り返し定義に対応するInvokeを実行した時

BPEL定義の実行は中断されます。

イベントログまたはシステムログに以下のメッセージが出力されます。

```
WARN:WSBPEL-47219: Simulation file was found when the loglevel is not 3: BPEL Process Name={%1}%2 ItemName=%3
FileName=%4
```

詳細は、“ISI メッセージ集”を参照してください。

ログファイル(isibpel.log)に以下のようなログが出力されます。ログの項目で開始/終了を出力する箇所に“###WARNING_SIMULATION_IS_IGNORE###”が出力されます。

```
2011-01-14 19:08:08.852 FSP_INTS-ESI_WSBPEL-SV INFO 00001 BPEL001 0000000000000000000 ----- Invoke Invoke01
###WARNING_SIMULATION_IS_IGNORE### -----
```

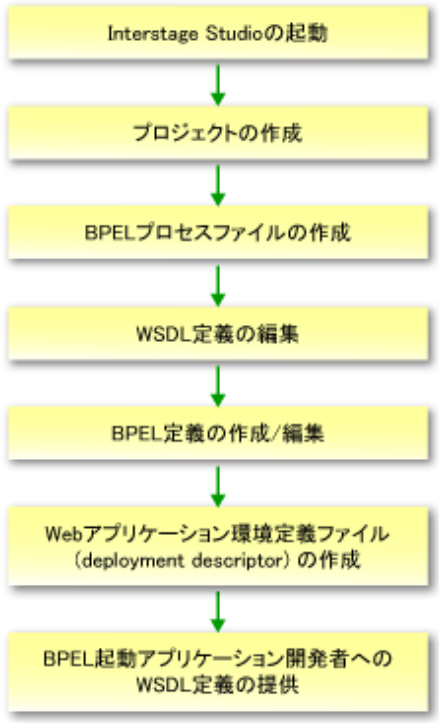
E.6 BPELエディタによるBPEL定義の作成方法

BPELエディタによるBPEL定義の作成方法について説明します。

E.6.1 BPEL定義作成の流れ

BPEL定義は、以下の手順で作成します。

図E.4 BPEL定義作成の流れ



次に各手順について説明します。

E.6.2 Interstage Studioの起動

Interstage StudioのJava EEワークベンチを起動します。以下の手順で起動します。

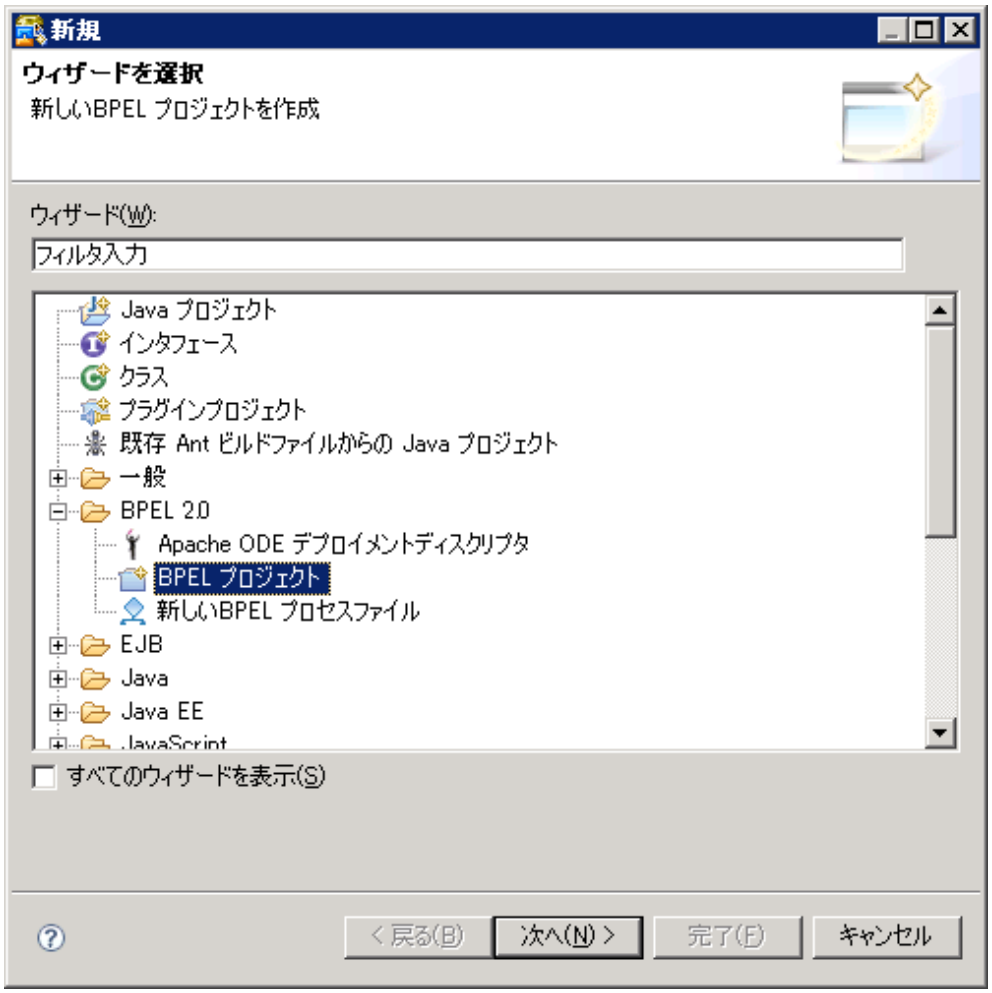
1. Interstage Studio V10.0 の場合は、[スタート]メニューの[すべてのプログラム]－[Interstage]－[Studio V10.0]－[Interstage Studio]を選択します。
2. Interstage Studio V10.1 の場合は、[スタート]メニューの[すべてのプログラム]－[Interstage]－[Studio V10.1]－[Interstage Studio]を選択します。
3. Interstage Studio V11.0 の場合は、[スタート]メニューの[すべてのプログラム]－[Interstage Studio V11.0]－[Interstage Studio]を選択します。
4. ワークスペースの選択
   ワークスペースランチャの画面が表示されますので、ワークスペースを指定します。
   ワークスペースについては、Interstage Studioの“Java EEワークベンチユーザーズガイド”の“ワークベンチの概要”を参照してください。

E.6.3 プロジェクトの作成

BPEL定義を作成するプロジェクトを作成します。

1. メニューバーから[ファイル]－[新規]－[その他]を選択すると、[新規]ウィザードが表示されます。
[BPELプロジェクト]を選択し、[次へ]をクリックしてください。

図E.5 [新規]ウィザード

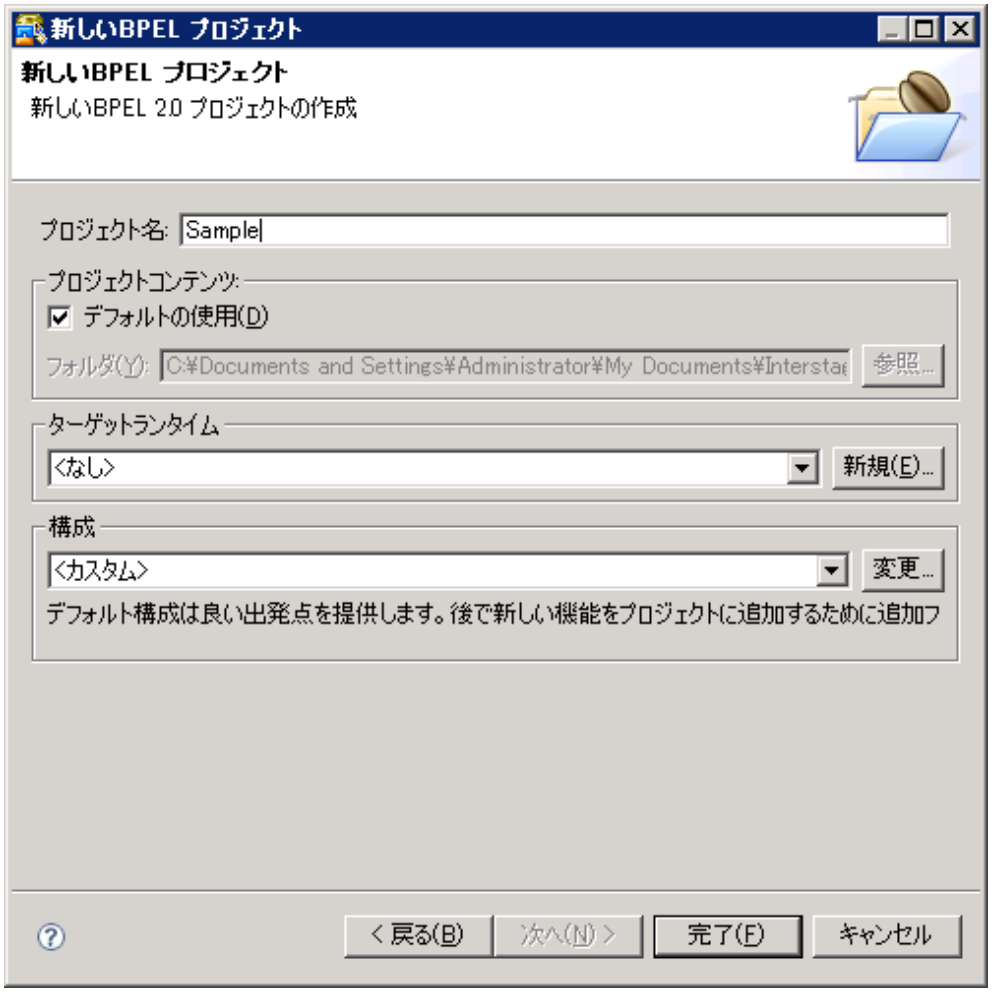


2. 以下の設定項目を確認、入力し、[完了]をクリックしてください。

表E.4 [新しいBPELプロジェクト]の設定内容

設定項目	設定内容
プロジェクト名	他のプロジェクトと重複しない任意の名前
プロジェクトコンテンツ	初期設定のままとしてください(デフォルトの使用にチェック)。
ターゲットランタイム	初期設定のままとしてください(なし)。
構成	初期設定のままとしてください(カスタム)。

図E.6 [新しいBPELプロジェクト]画面



ポイント

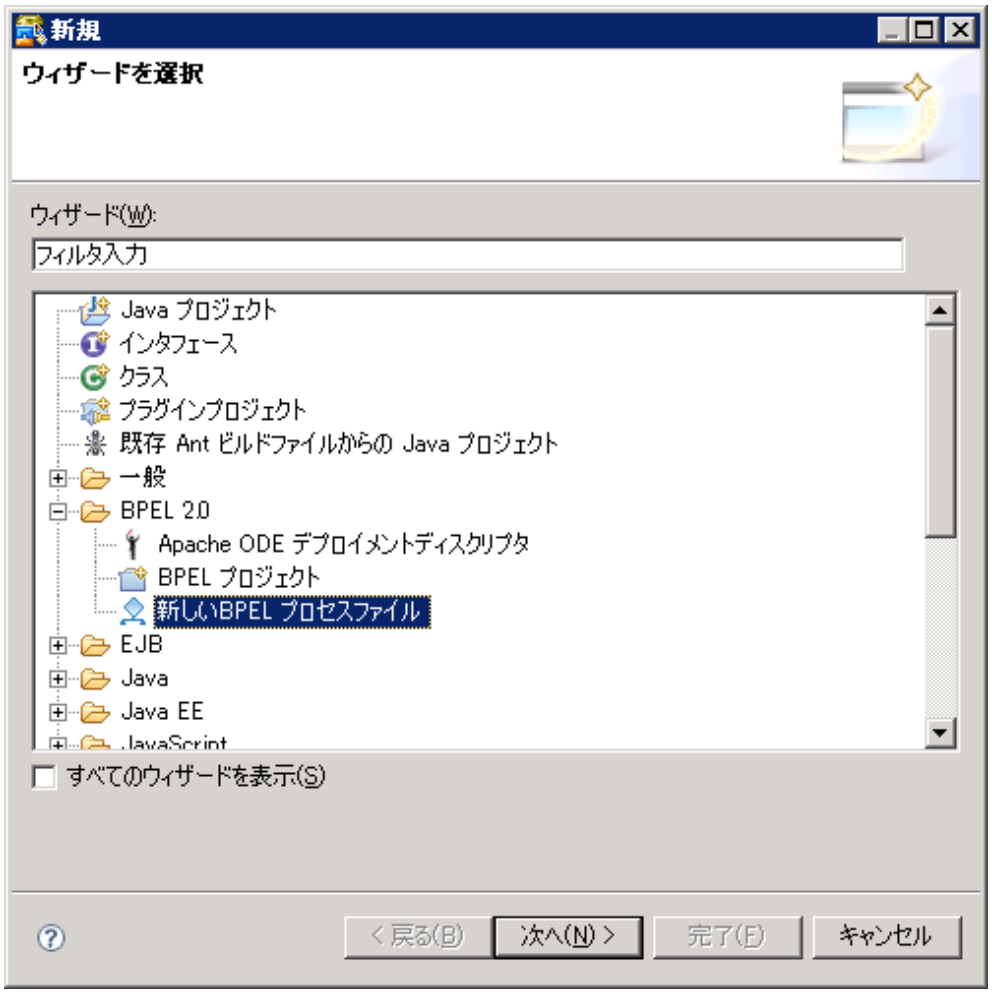
1つのプロジェクトに作成できるBPELプロセスファイルの数は1つです。

E.6.4 BPELプロセスファイルの作成

BPELプロセスファイルを作成します。

1. メニューバーから[ファイル]－[新規]－[その他]を選択すると、[新規]ウィザードが表示されます。
[新しいBPELプロセスファイル]を選択し、[次へ]をクリックしてください。

図E.7 [新規]ウィザード

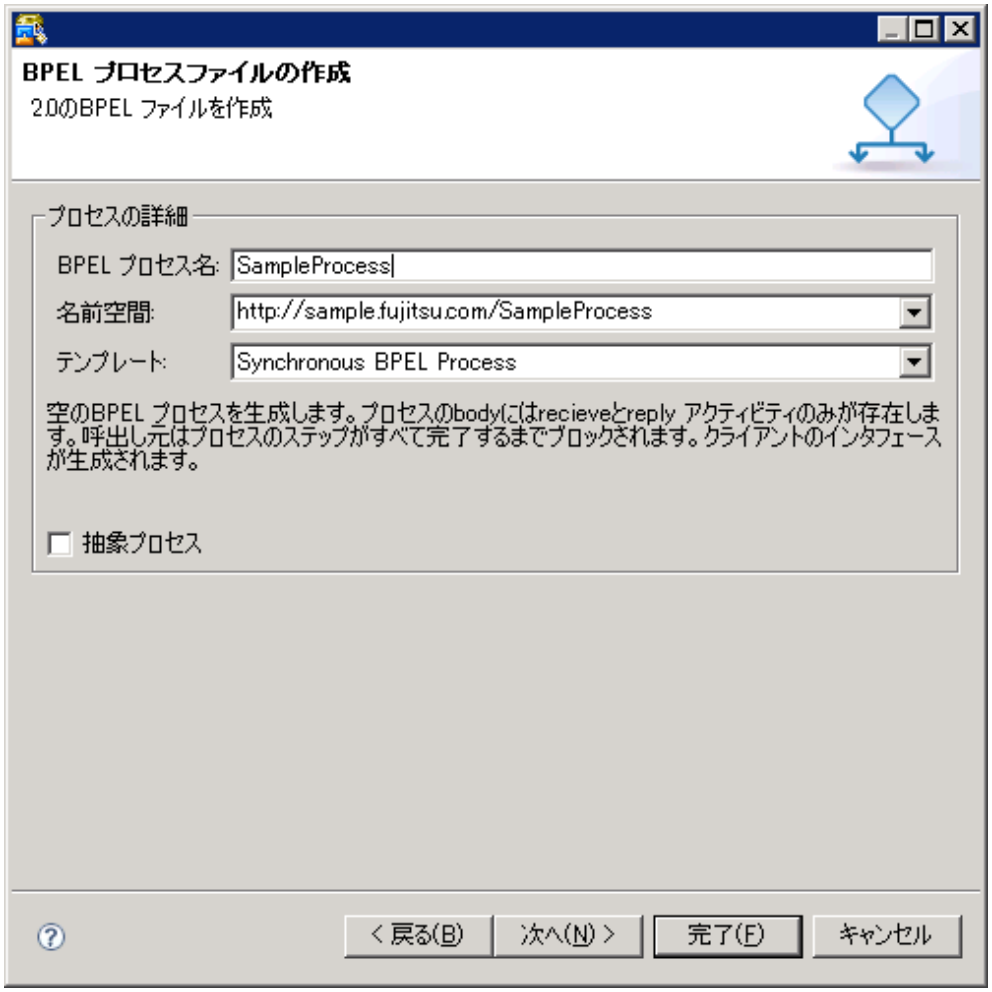


2. 以下の設定項目を確認、入力し、[次へ]をクリックしてください。

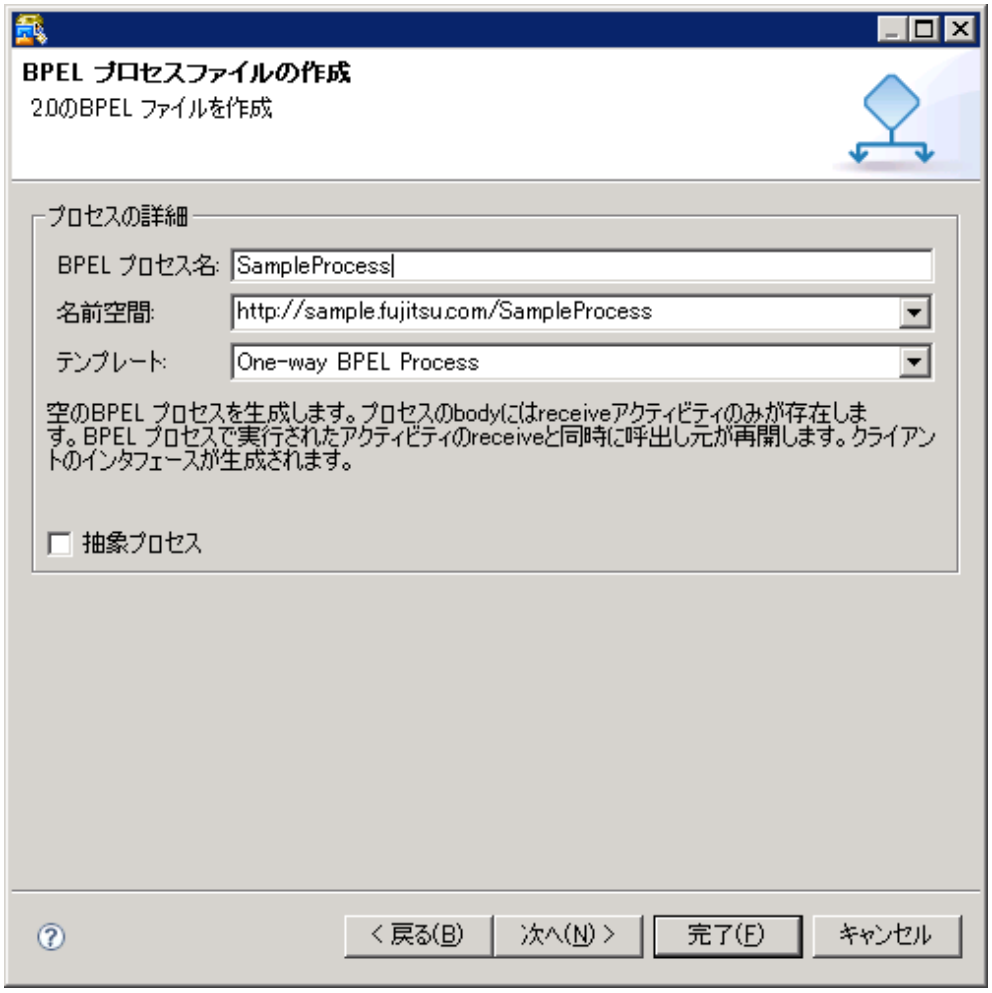
表E.5 [BPELプロセスファイルの作成]の設定内容

設定項目	設定内容
BPELプロセス名	他のBPELプロセスファイルと重複しない任意の名前
名前空間	BPELプロセスの名前空間
テンプレート	要求/応答型の場合は「Synchronous BPEL Process」、一方向型(one-way)の場合は「One-way BPEL Process」を選択してください。 その他のテンプレートはサポートしていません。
抽象プロセス	初期設定のままとしてください(チェックなし)。 抽象プロセスはサポートしていません。

図E.8 要求/応答型の[BPELプロセスファイルの作成]画面

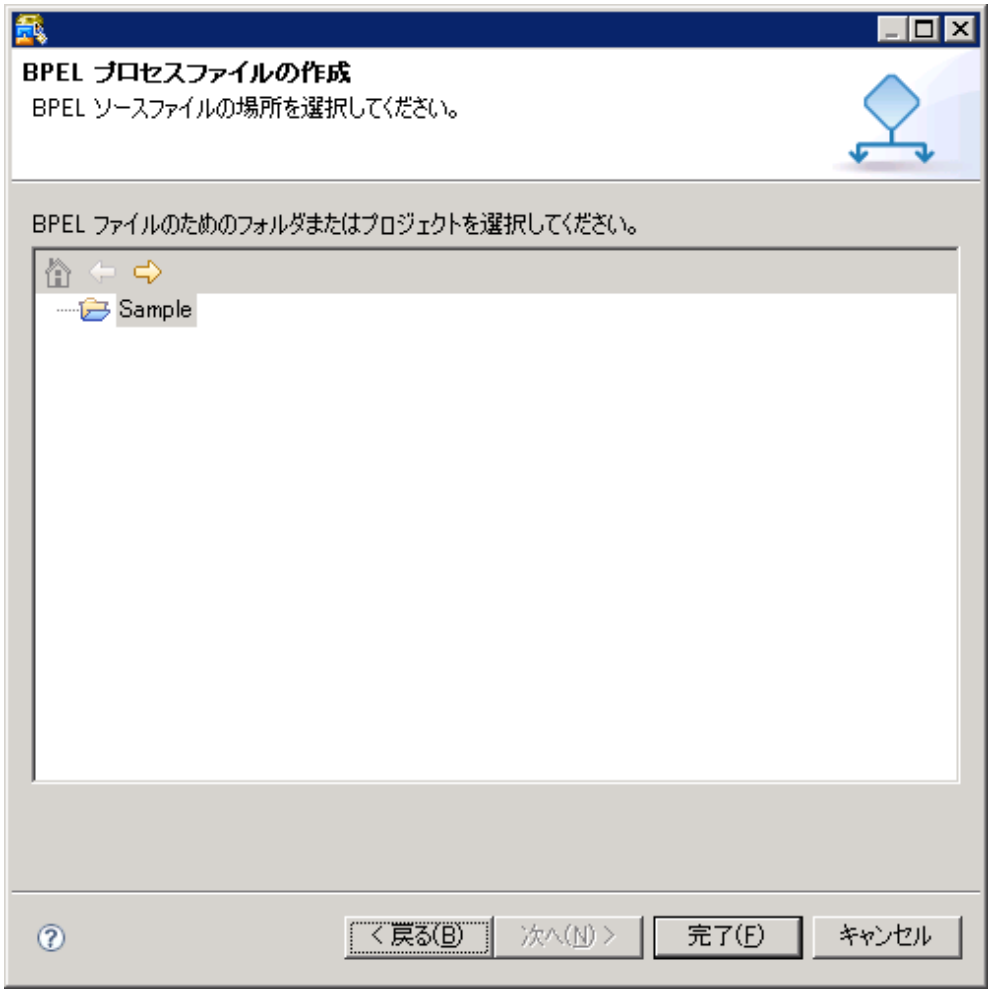


図E.9 一方向型(onw-way)の[BPELプロセスファイルの作成]画面



3. BPELプロセスファイルを作成するプロジェクトを選択し、[完了]をクリックしてください。

図E.10 [BPELプロセスファイルの作成]の完了画面



4. 以下に示すひな型のBPEL定義とWSDL定義が生成されます。

   ー BPEL定義

表E.6 要求/応答型の場合のひな型のBPEL定義

要素種別	要素	名前	概要
アクティビティ	Receive	receiveInput	BPEL起動アプリケーションからの要求受付
	Reply	replyOutput	BPEL起動アプリケーションへの応答
その他	PartnerLinks	client	receiveInput/repluOutputに対応した本BPEL定義のPartnerLink
	Variables	input	receiveInputで受信したメッセージを格納する変数
		output	replyOutputで返信する応答メッセージを格納する変数

表E.7 一方向型(one-way)の場合のひな型のBPEL定義

要素種別	要素	名前	概要
アクティビティ	Receive	receiveInput	BPEL起動アプリケーションからの要求受付

要素種別	要素	名前	概要
その他	PartnerLinks	client	receiveInputに対応した本BPEL定義のPartnerLink
	Variables	input	receiveInputで受信したメッセージを格納する変数

－ WSDL定義
上記BPEL定義のアクティビティに対応したインタフェースが定義されたWSDL定義ファイル。

BPELプロセス名に関する注意事項

BPELプロセス名は、BPELプロセスファイル、WSDL定義のファイル名に使用されます。このためBPELプロセス名が長い場合、配備が失敗する可能性があります。詳細は、“ISI リファレンス”の“BPEL機能”の“BPEL定義の配備/配備解除/一覧表示コマンド”の“deployサブコマンド”の“注意事項”を参照してください。

E.6.5 WSDL定義の編集

本BPEL定義のWSDL定義を編集します。以下の手順でWSDL定義を編集します。

なお、WSDLエディタによるWSDL定義の編集中は、アウトラインビューを最大化しないでください。

- ひな型のWSDL定義の編集
- 追加するReceive/Reply用のWSDL定義の追加
- 外部ファイルの参照に関する注意事項
- ポートタイプのMessageのPartに設定する型に関する注意事項

ひな型のWSDL定義の編集

ひな型のWSDL定義を以下の手順で編集します。ここでは、要求/応答型の場合について説明します。一方向型(one-way)の場合も手順は同じです。

1. WSDLエディタの起動
[プロジェクトエクスプローラ]ビューで編集するWSDL定義を右クリックし、ポップアップメニューの[開く]をクリックします。WSDLエディタについては、Interstage Studioの“Java EEワークベンチユーザーズガイド”の“Webサービスのアプリケーションを開発する”の“タスク”の“WSDLを作成する”の“WSDLエディタ”、またはInterstage Studioのヘルプを参照してください。

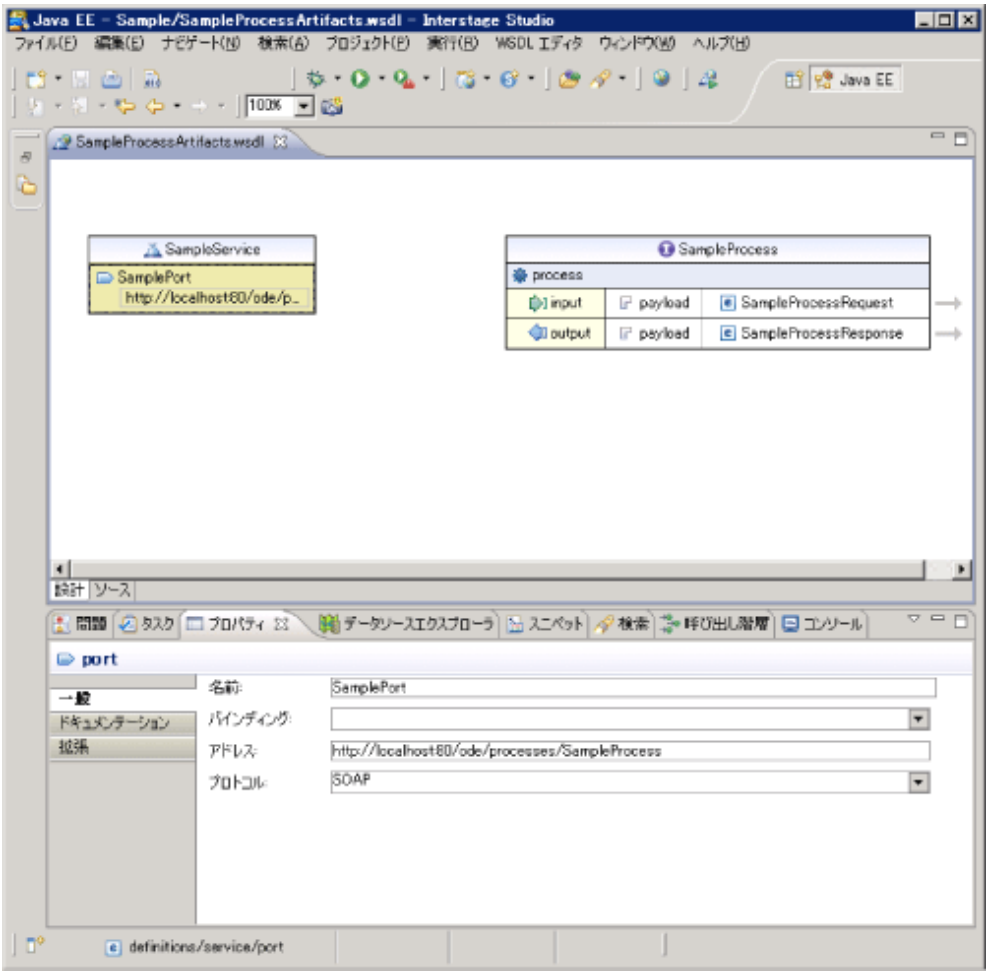
2. サービスの作成
定義されているポートタイプに対応するサービスを作成します。
サービスの作成は、WSDLエディタの[設計]タブの空白部分で右クリックし、ポップアップメニューの[サービスの追加]をクリックすることで行います。追加後は必要に応じてサービスの名前を設定ください。

3. ポートの設定
作成したサービスに、ポートを設定します。
WSDLエディタの[設計]タブのサービス定義の中にあるポート（NewPortを書かれた部分）をクリックし、プロパティビューの[一般]で以下の設定を行います。

表E.8 ポートの設定内容

設定項目	設定内容
名前	他のポート名と重複しない任意の名前。
バインディング	次の手順で設定しますので、ここでは設定しないでください。
アドレス	以下のアドレスを設定してください。 http://localhost:80/ode/processes/[システムで一意の名前]
プロトコル	SOAPを選択してください。

図E.11 ポートの設定画面

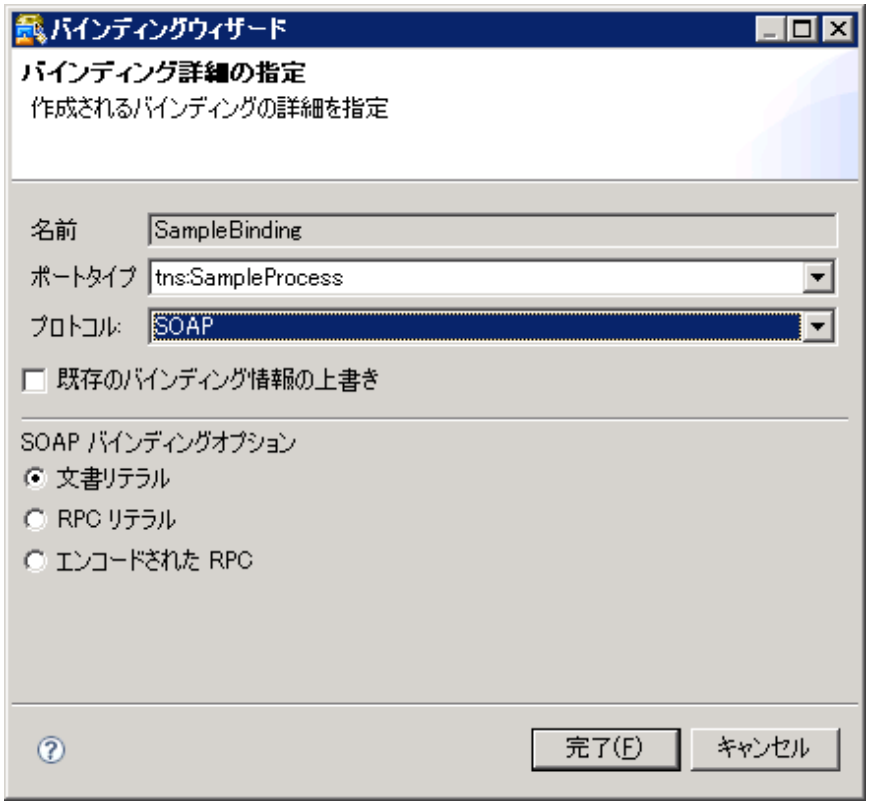


4. バインディングの作成
ポートタイプとサービスを関連付けるバインディングを作成します。
手順3.のポートのプロパティのバインディングで[新規]を選択します。[新規バインディング]ダイアログボックスが表示されますので、任意のバインディング名を設定し、[OK]をクリックします。
WSDLエディタの[設計]タブにバインディングが表示されますので、プロパティビューの[一般]で[バインディングコンテンツの生成]ボタンをクリックし、次に表示される[バインディングウィザード]で以下の設定を行い、[完了]をクリックします。

表E.9 [バインディングウィザード]の設定内容

設定項目	設定内容
ポートタイプ	関連付けるポートタイプを選択してください。
プロトコル	SOAPを選択してください。
既存のバインディング情報の上書き	初期設定のままとしてください(チェックなし)。
SOAPバインディングオプション	初期設定のままとしてください(文書リテラル)。

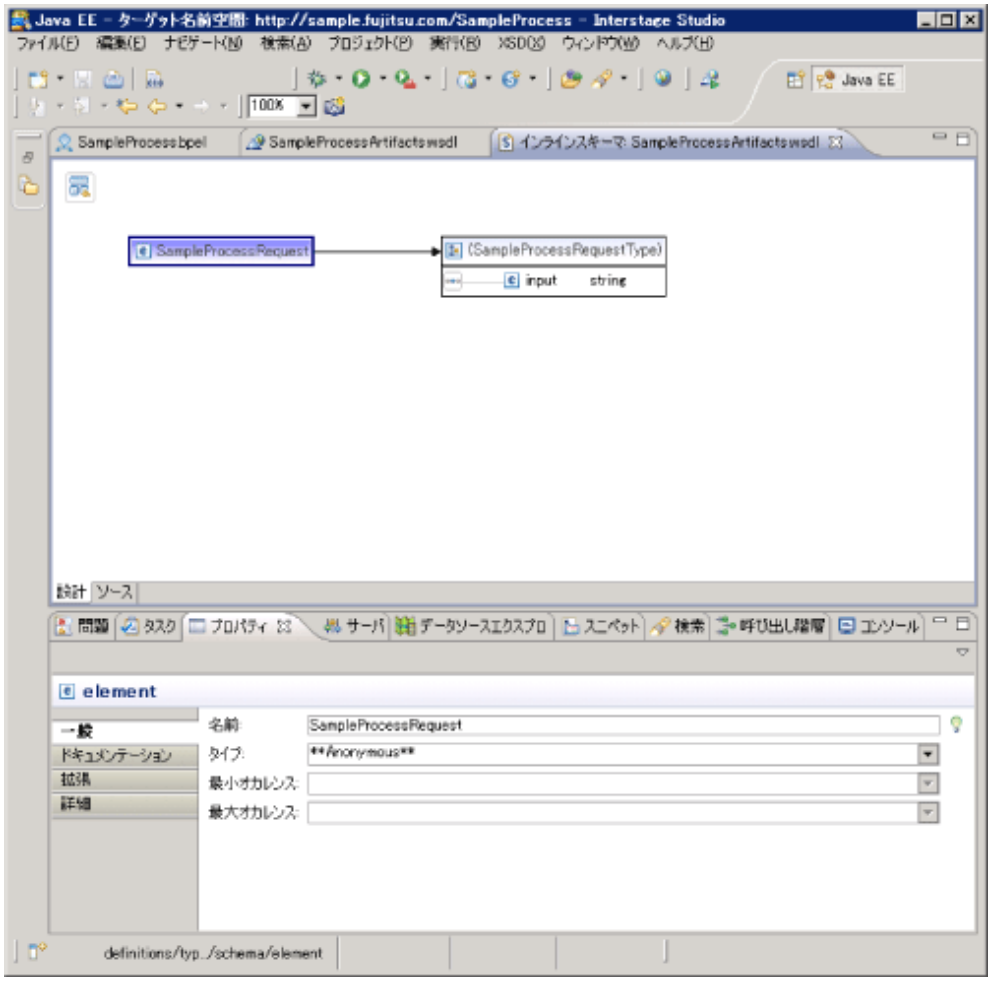
図E.12 [バインディングウィザード]画面



5. メッセージの定義
   ポートタイプのメッセージを定義します。
   WSDLエディタの[設計]タブで、ポートタイプ中のメッセージの右側に表示されている矢印をクリックします。インラインスキーマの画面が表示されますので、メッセージの型を定義します。定義方法については、Interstage Studioの“Java EEワークベンチユー

ザーズガイド”の“Webサービスのアプリケーションを開発する”の“タスク”の“WSDLを作成する”の“WSDLエディタ”、またはInterstage Studioのヘルプを参照してください。

図E.13 メッセージの定義画面



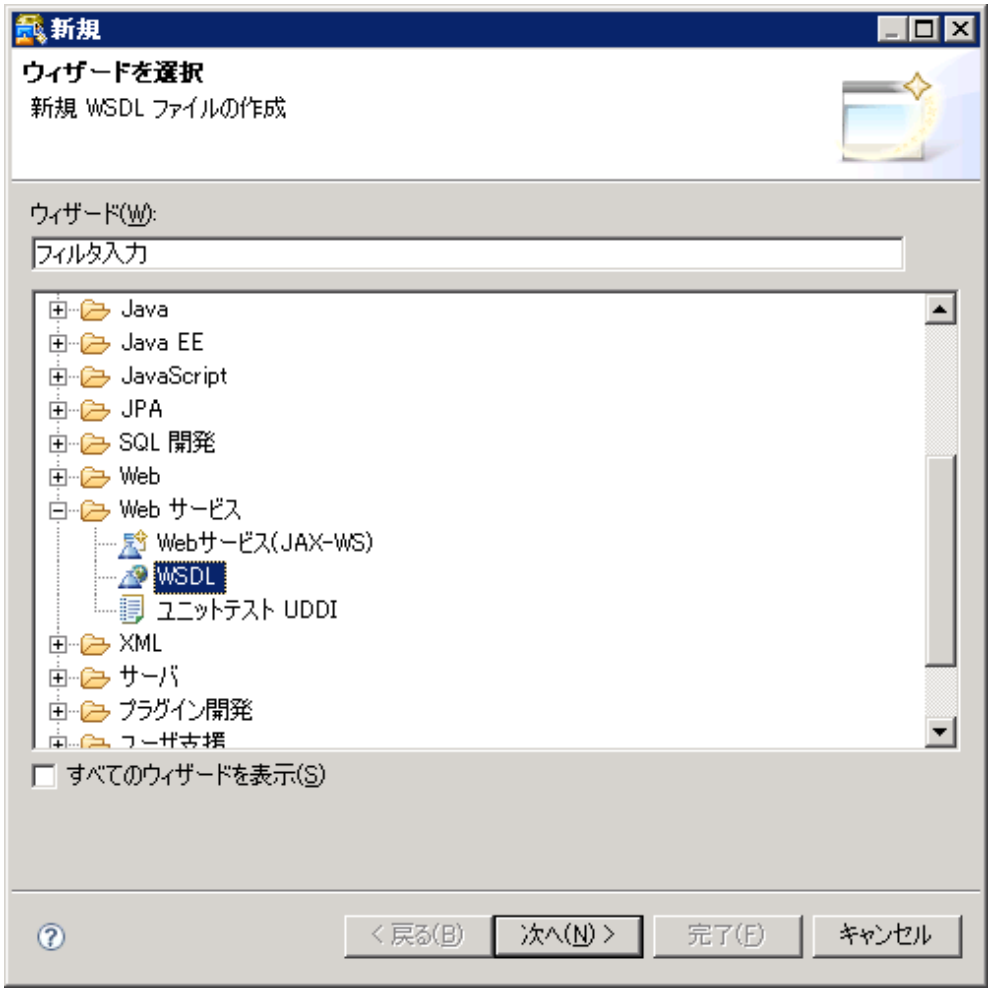
6. WSDL定義の保存
   メニューバーから[ファイル]－[保存]を選択し、WSDL定義を保存してください。
7. WSDLエディタの終了
   メニューバーから[ファイル]－[閉じる]を選択し、WSDLエディタを終了してください。

追加するReceive/Reply用のWSDL定義の追加

ひな型のBPEL定義にReceive/Replyのペアを追加する場合は、対応するWSDL定義を追加するペアの数分、追加します。以下の手順で追加します。

1. WSDL定義の追加
   メニューバーから[ファイル]－[新規]－[その他]を選択すると、[新規]ウィザードが表示されます。
   [WSDL]を選択し、[次へ]をクリックしてください。

図E.14 [新規]ウィザード



2. 以下の設定項目を確認、入力し、[次へ]をクリックしてください。

表E.10 [新規WSDLファイル]画面の設定内容

設定項目	設定内容
親フォルダを入力または選択	WSDL定義を作成する親ディレクトリを設定してください。
ファイル名	追加するWSDL定義ファイルの名前を設定してください。

図E.15 [新規WSDLファイル]画面

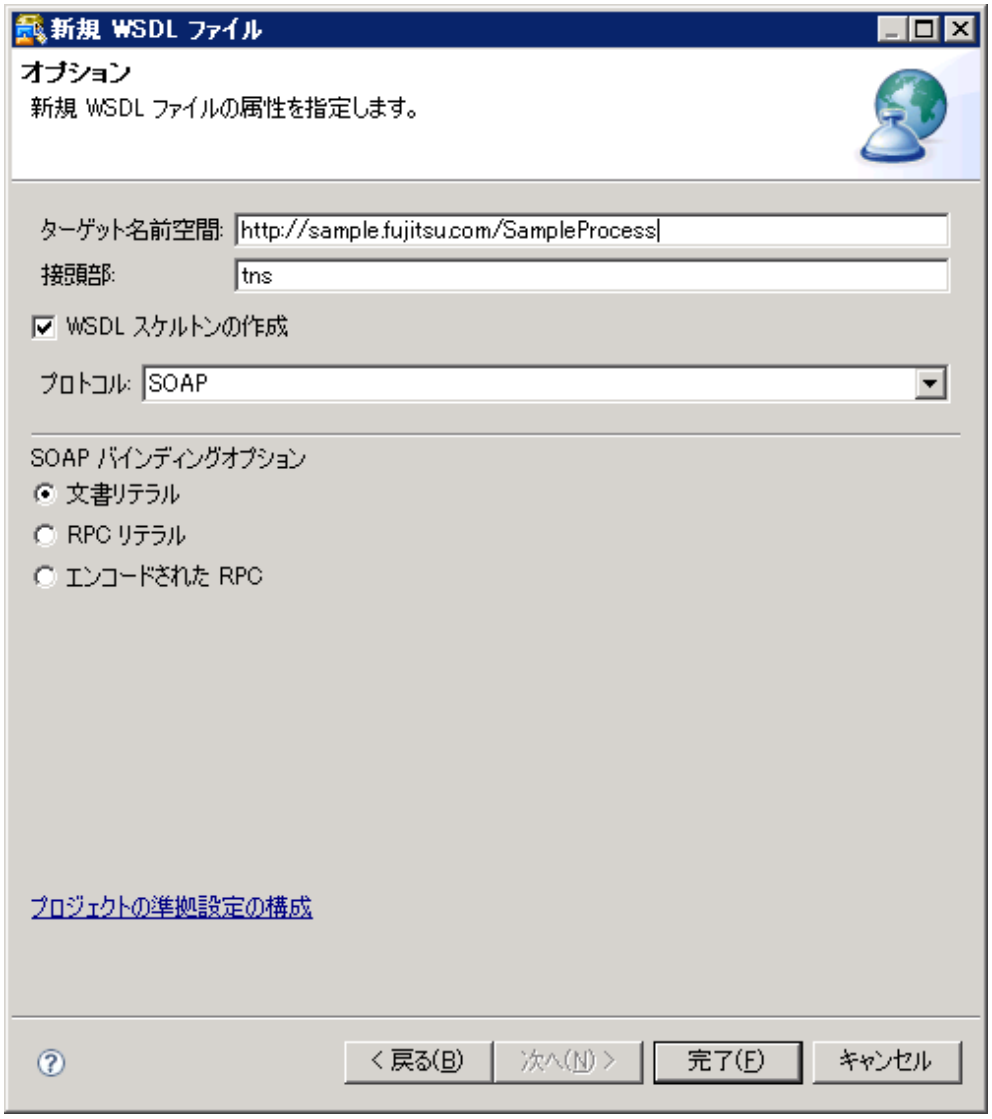


3. 以下の設定項目を確認、入力し、[完了]をクリックしてください。

表E.11 [新規WSDLファイル]オプション画面の設定内容

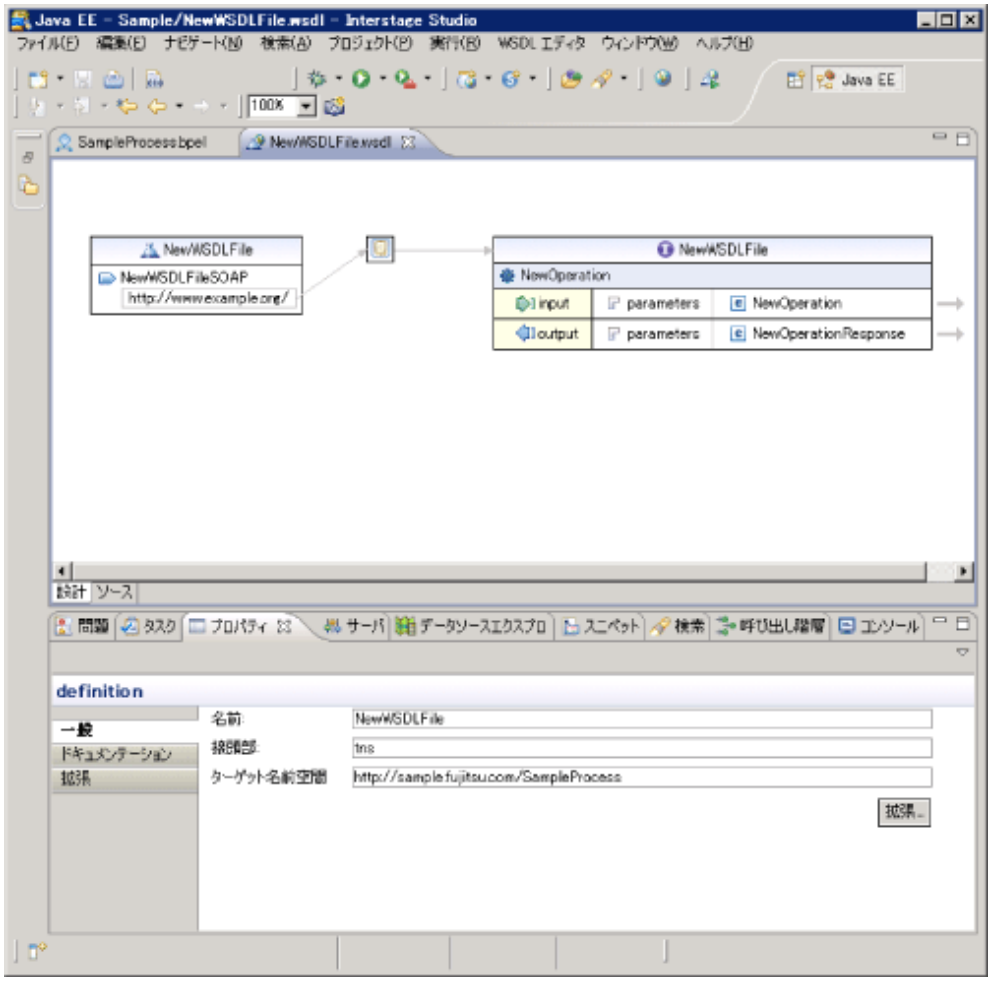
設定項目	設定内容
ターゲット名前空間	BPELプロセスのNamespace
接頭部	ターゲット名前空間に対応する接頭部
WSDLスケルトンの作成	初期設定のままとしてください(チェックあり)。
プロトコル	初期設定のままとしてください(SOAP)。
SOAPバインディングオプション	初期設定のままとしてください(文書リテラル)。

図E.16 [新規WSDLファイル]オプション画面



4. 以下に示すWSDL定義が生成されます。

図E.17 WSDL定義画面



5. ポートタイプの設定
   生成されたWSDL定義のポートタイプを設定します。必要に応じてポートタイプの名前や、input/outputパラメタを設定してください。

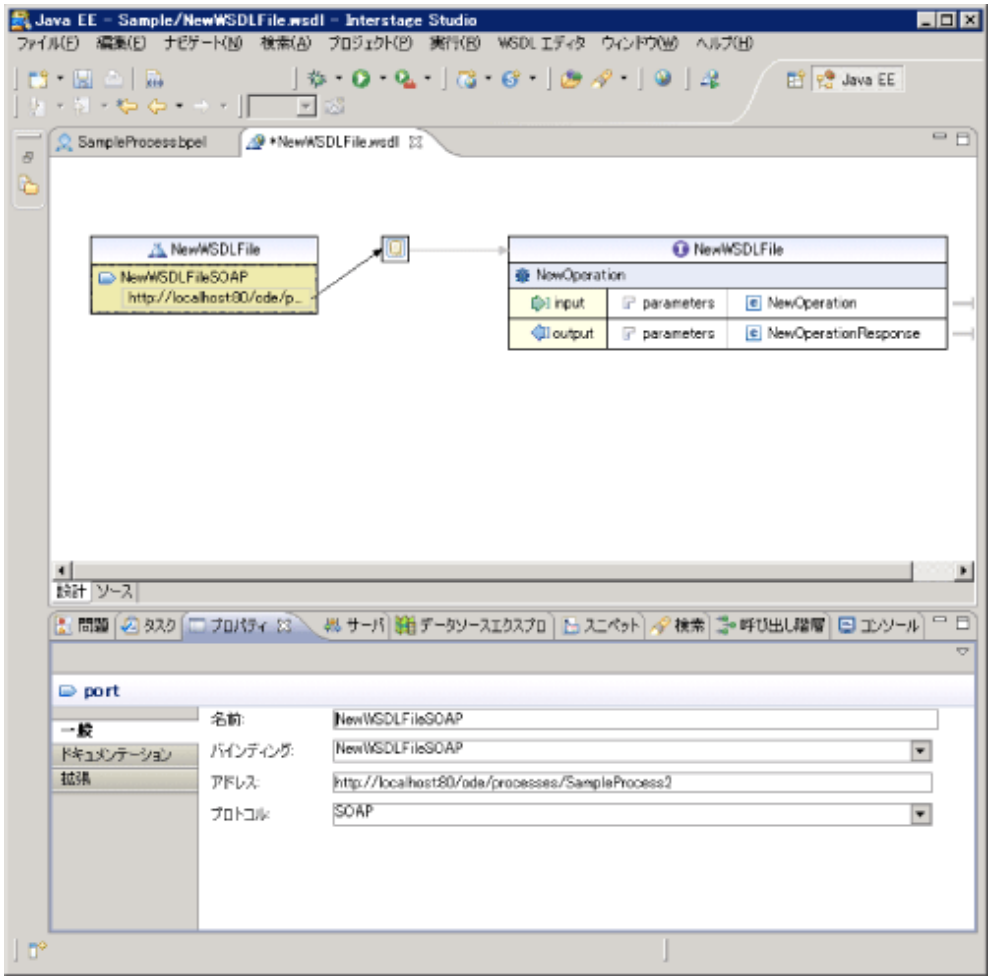
6. サービスの設定
   生成されたWSDL定義のサービスを設定します。必要に応じてサービスの名前を設定ください。

7. ポートの設定
   生成されたWSDL定義のポートを設定します。
   WSDLエディタの[設計]タブのサービス定義の中にあるポートをクリックし、プロパティビューの[一般]で以下の設定を行います。

表E.12 ポートの設定内容

設定項目	設定内容
名前	他のポート名と重複しない任意の名前。
バインディング	次の手順で設定しますので、ここでは設定しないでください。
アドレス	以下のアドレスを設定してください。 http://localhost:80/ode/processes/[システムで一意の名前]
プロトコル	SOAPを選択してください。

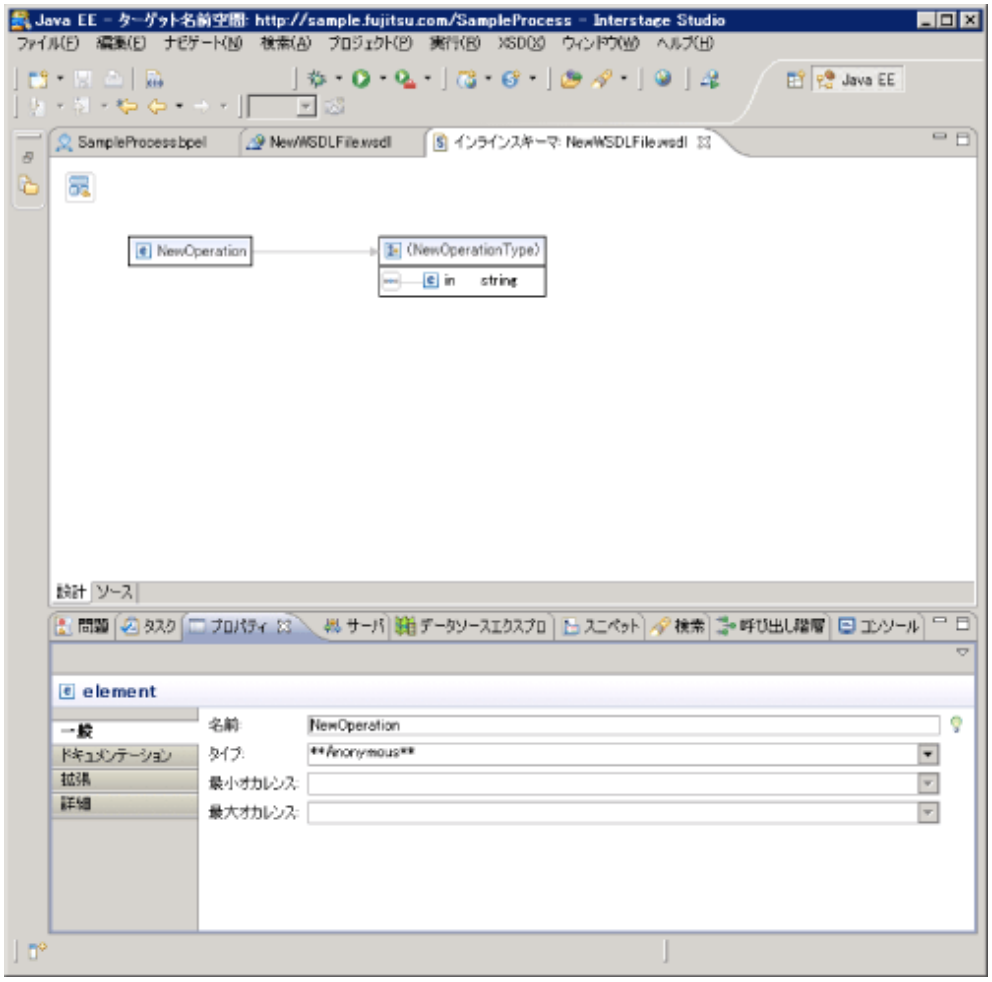
図E.18 ポートの設定画面



8. メッセージの定義
   ポートタイプのメッセージを定義します。
   WSDLエディタの[設計]タブで、ポートタイプ中のメッセージの右側に表示されている矢印をクリックします。インラインスキーマの画面が表示されますので、メッセージの型を定義します。定義方法については、Interstage Studioの“Java EEワークベンチユー

ザーズガイド”の“Webサービスのアプリケーションを開発する”の“タスク”の“WSDLを作成する”の“WSDLエディタ”、またはInterstage Studioのヘルプを参照してください。

図E.19 メッセージの定義画面



9. WSDL定義の保存
   メニューバーから[ファイル]－[保存]を選択し、WSDL定義を保存してください。
10. WSDLエディタの終了
    メニューバーから[ファイル]－[閉じる]を選択し、WSDLエディタを終了してください。

外部ファイルの参照に関する注意事項

WSDL定義から外部ファイルを参照する場合、および外部ファイルが他の外部ファイルを参照する場合、外部ファイルの拡張子は“.xsd”にしてください。外部ファイルは、BPELプロジェクトの直下に格納してください。

importやincludeなどに設定する、外部ファイルのロケーションまたはスキーマロケーションに設定できる値は、BPELプロジェクトの直下に格納したファイルの相対パスだけです。URLは定義できません。

ポートタイプのMessageのPartに設定する型に関する注意事項

ポートタイプのMessageのPartに設定する型は、必ずElement型としてください。

E.6.6 BPEL定義の作成/編集

BPELエディタでBPEL定義を作成/編集します。以下に要素ごとの作成/編集方法を説明します。

BPELエディタは、[プロジェクトエクスプローラ]ビューで編集するBPEL定義を右クリックし、ポップアップメニューの[開く]をクリックします。

ここでは、以下に示す項目ごとに説明します。

表E.13 BPEL定義の要素

分類	要素名	概要
BPEL定義のプロパティの設定		BPEL定義のプロパティを設定します。
基本アクティビティ	Invoke	サービスを呼び出します。
	Receive	送信されたメッセージを受信します。
	Reply	受信したメッセージに対して応答を返します。
	Assign	変数に値を設定します。
	Throw	障害を生成し、エラー情報を通知します。
構造化アクティビティ	Sequence	Sequence内に定義されたアクティビティを1つずつ順に実行します。
	Flow	Flow内に定義された複数のアクティビティを同時並列に実行します。
	If	条件付き動作を定義し、条件に従ってアクティビティを実行します。
	While	While内に定義されたアクティビティを条件が真の間、繰返し実行します。
その他の要素	Variables	メッセージや変数を定義します。
	PartnerLinks	Invoke/Receive/Replyで連携するサービスを定義します。
	FaultHandlers	エラー情報を受け取ります。
	CorrelationSets	Receiveで受信するメッセージとビジネス・プロセスのインスタンスを関連付けます。

BPEL定義のプロパティの設定

BPEL定義のプロパティを以下のとおり設定します。

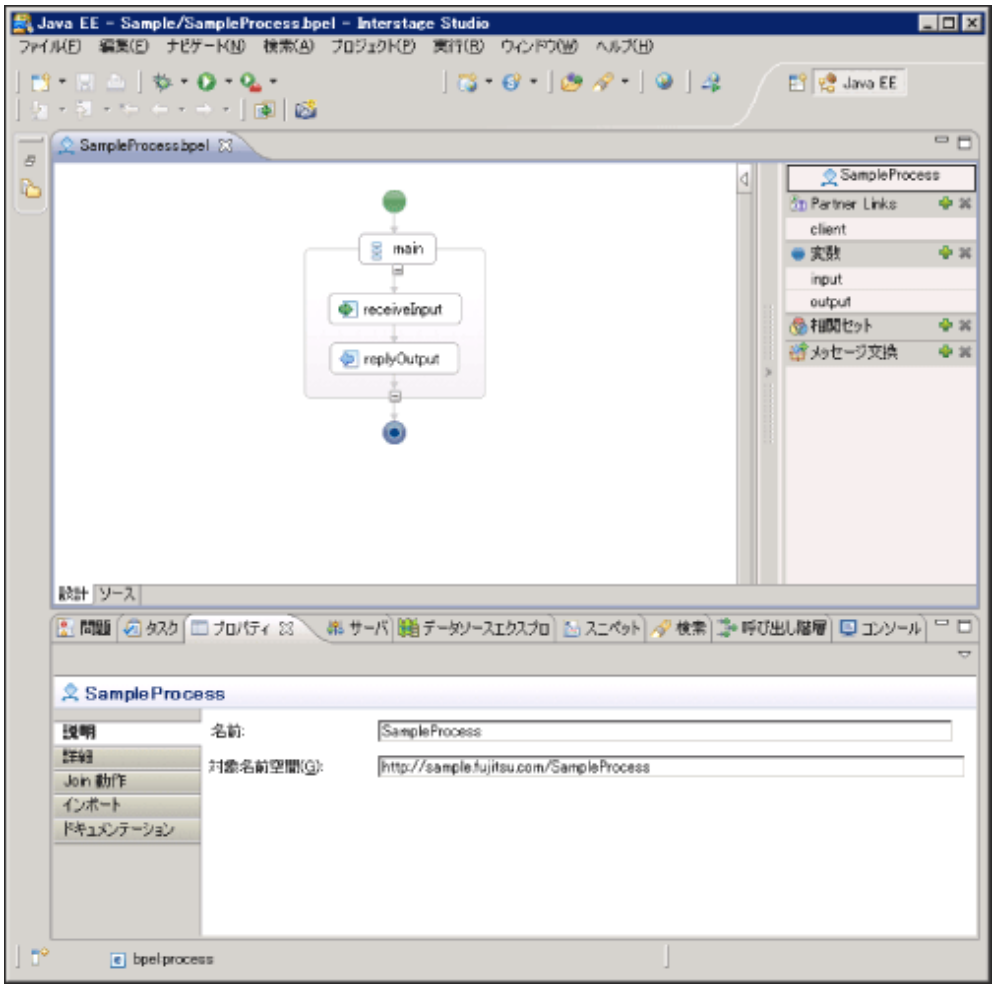
1. プロパティビューの表示
   BPELエディタの[設計]タブの空白部分をクリックし、BPEL定義のプロパティビューを表示します。
2. プロパティビューの[説明]、[詳細]、[Join動作]、および[インポート]で以下の設定を行います。

表E.14 BPEL定義のプロパティの設定内容

設定場所	設定項目	設定内容
説明	名前	BPELプロセスファイル名を設定します。 初期状態では、“E.6.4 BPELプロセスファイルの作成”で設定したBPELプロセスファイル名となっています。
	対象名前空間	BPELプロセスの名前空間を設定します。 初期状態では、“E.6.4 BPELプロセスファイルの作成”で設定した名前空間となっています。
詳細	式言語	条件式の形式を設定します。必ず[XPath 1.0 in BPEL 2.0]を選択し、その他は選択しないでください。
	クエリ言語	XML参照方式を設定します。必ず[XPath 1.0 in BPEL 2.0]を選択し、その他は選択しないでください。
	標準障害による終了	BPEL標準障害が発生した場合の動作を設定します。以下のどちらかを選択してください。初期設定ではNoとなっています。

設定場所	設定項目	設定内容
		・ する BPEL標準障害が発生した場合、即、exitする。 ・ しない BPEL標準障害が発生した場合、exitしない(障害ハンドラでエラー処理を行う)。
Join動作	―	各アクティビティのプロパティの[Join動作]の[Join Failureの抑制]の初期値を設定します。以下のどちらかを選択します。 ・ する bpel:joinFailure(障害)を通知しない。 ・ しない bpel:joinFailure(障害)を通知する。
インポート	―	初期設定のままとしてください。

図E.20 BPEL定義のプロパティの設定画面



Invoke

Invokeの追加、削除、および編集は、以下のとおり行います。

・ 追加

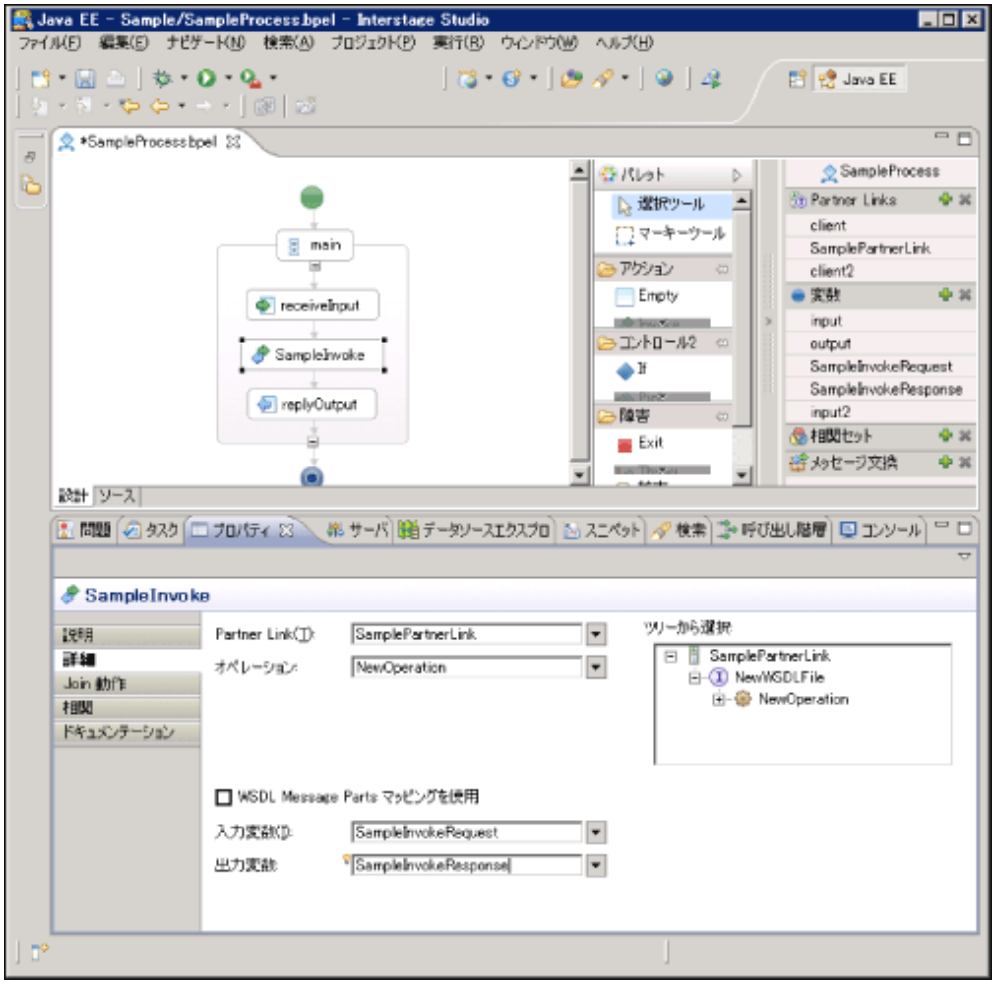
1. Invokeの追加
   BPELエディタの[設計]タブの[パレット][アクション]内の[Invoke]を、Invokeを追加する位置にドラッグ＆ドロップします。

2. プロパティの設定
   BPELエディタの[設計]タブで、追加したInvokeをクリックし、プロパティビューの[説明]で[名前]にInvokeの名前を設定します。
   次にプロパティビューの[詳細]で以下の設定を行います。

表E.15 Invokeの設定内容

設定項目	設定内容
Partner Link	入力フィールド右にある▼をクリックして表示された一覧から、Invokeで呼び出すサービスのPartner Linkをダブルクリックで選択します。Partner Linkはあらかじめ作成しておく必要があります。
オペレーション	入力フィールド右にある▼をクリックして表示された一覧から、Invokeで呼び出すサービスのオペレーションをダブルクリックで選択します。
WSDL Message Parts マッピングを使用	初期設定のままとしてください（チェックなし）。
入力変数	入力フィールド右にある▼をクリックして表示された一覧から、Invokeで呼び出すサービスに受け渡すメッセージを格納する変数をダブルクリックで選択します。変数はあらかじめ作成しておく必要があります。
出力変数	入力フィールド右にある▼をクリックして表示された一覧から、Invokeで呼び出すサービスから受け渡されるメッセージを格納する変数をダブルクリックで選択します。変数はあらかじめ作成しておく必要があります。呼び出すサービスが一方向型(one-way)の場合、本項目は入力しないでください。

図E.21 Invokeの設定画面



- 削除

  BPELエディタの[設計]タブで、削除するInvokeをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するInvokeをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するInvokeをクリックし、プロパティビューで変更する設定値を編集します。

注意

- Invokeの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。
- Invokeを追加する場合、呼び出すサービスに受け渡すメッセージを格納する変数、および呼び出すサービスから受け渡されるメッセージを格納する変数をあらかじめ作成しておく必要があります。あらかじめ作成していないと、プロパティビューの[詳細]で[入力変数]、および[出力変数]の入力フィールドが表示されない場合があります。この場合は、そのInvokeをいったん削除し、各変数を作成後に、再度Invokeの追加を行ってください。

・ 以下の場合は呼出し元のBPEL定義と呼出し先のBPEL定義を異なるビジネス・プロセス実行環境に配備してください。ビジネス・プロセス実行環境へのBPEL定義の配備については、“E.3.3 ビジネス・プロセス実行環境への配備”を参照してください。

　ー BPEL定義からBPEL定義で作成したサービスをInvokeで呼出す場合。

Receive

Receiveの追加、削除、および編集は、以下のとおり行います。

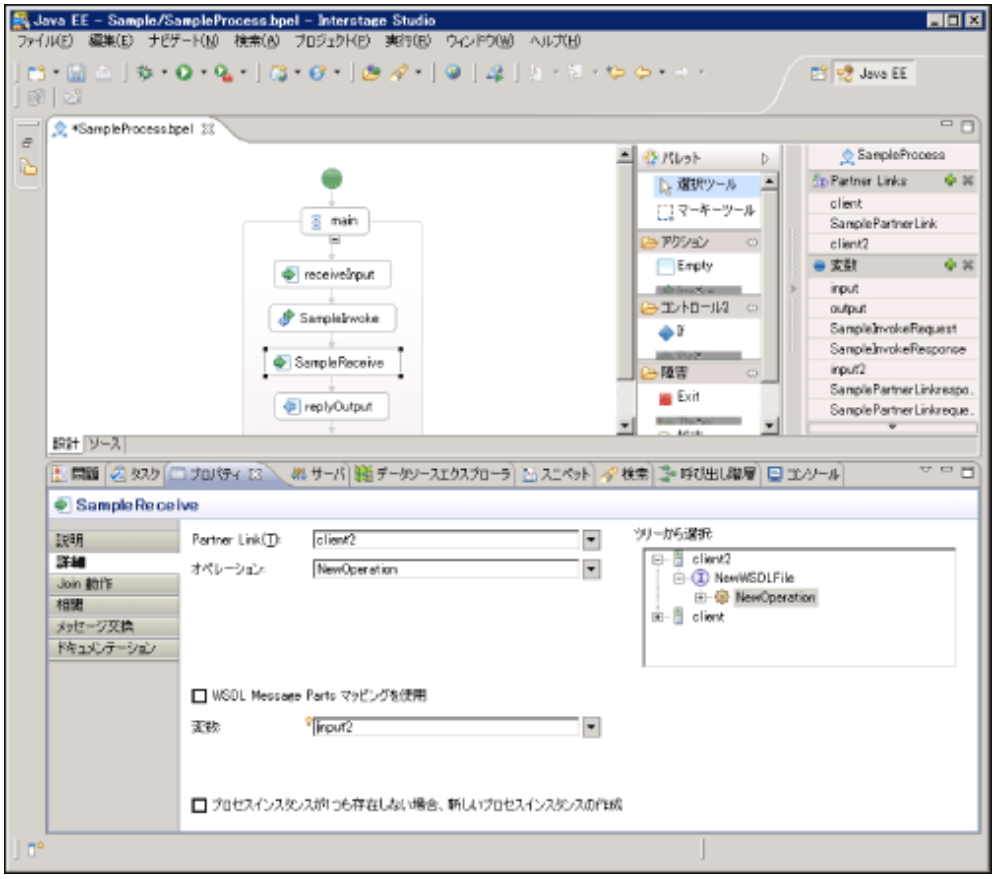
・ 追加

1. Receiveの追加
   BPELエディタの[設計]タブの[パレット][アクション]内の[Receive]を、Receiveを追加する位置にドラッグ＆ドロップします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加したReceiveをクリックし、プロパティビューの[説明]で[名前]にReceiveの名前を設定します。
   次にプロパティビューの[詳細]で以下の設定を行います。

表E.16 Receiveの設定内容

設定項目	設定内容
Partner Link	入力フィールド右にある▼をクリックして表示された一覧から、Receiveに対応するPartner Linkをダブルクリックで選択します。Partner Linkはあらかじめ作成しておく必要があります。
オペレーション	入力フィールド右にある▼をクリックして表示された一覧から、Receiveに対応するオペレーションをダブルクリックで選択します。
WSDL Message Parts マッピングの使用	チェックなしの設定としてください。
変数	入力フィールド右にある▼をクリックして表示された一覧から、Receiveで受信したメッセージを格納する変数をダブルクリックで選択します。変数はあらかじめ作成しておく必要があります。
プロセスインスタンスが1つも存在しない場合、新しいプロセスインスタンスの作成	BPEL定義中の最初のReceiveの場合はチェックしてください。 その他のReceiveではチェックをはずしてください。

図E.22 Receiveの設定画面



- 削除

  BPELエディタの[設計]タブで、削除するReceiveをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するReceiveをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するReceiveをクリックし、プロパティビューで変更する設定値を編集します。

注意

- BPEL定義が要求/応答型の場合、Receiveは、必ずReplyとペアで使用してください。
- Receiveの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

Reply

Replyの追加、削除、および編集は、以下のとおり行います。

・ 追加

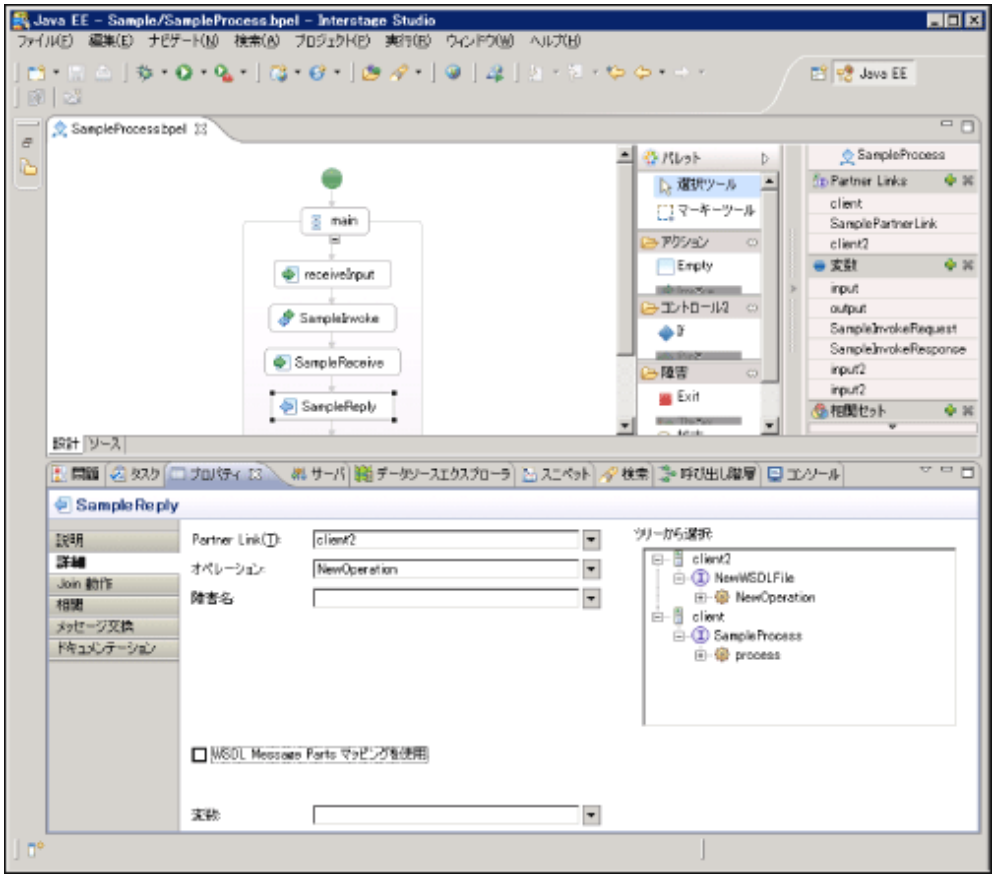
1. Replyの追加
   BPELエディタの[設計]タブの[パレット][アクション]内の[Reply]を、Replyを追加する位置にドラッグ＆ドロップします。

2. プロパティの設定
   BPELエディタの[設計]タブで、追加したReplyをクリックし、プロパティビューの[説明]で[名前]にReplyの名前を設定します。
   次にプロパティビューの[詳細]で以下の設定を行います。

表E.17 Replyの設定内容

設定項目	設定内容
Partner Link	入力フィールド右にある▼をクリックして表示された一覧から、Replyに対応するPartner Linkをダブルクリックで選択します。Partner Linkはあらかじめ作成しておく必要があります。
オペレーション	入力フィールド右にある▼をクリックして表示された一覧から、Replyに対応するオペレーションをダブルクリックで選択します。
障害名	障害を通知する場合は、入力フィールド右にある▼をクリックして表示された一覧から、通知する障害名をダブルクリックで選択します。障害はあらかじめ"E.6.5 WSDL定義の編集"で作成しておく必要があります。
WSDL Message Parts マッピングの使用	チェックなしの設定としてください。
変数	障害を通知しない場合は、Replyで通知するメッセージを格納する変数を設定します。 入力フィールド右にある▼をクリックして表示された一覧から、Replyで通知するメッセージを格納する変数をダブルクリックで選択します。変数はあらかじめ作成しておく必要があります。 障害を通知する場合（障害名を設定した場合）、変数には何も設定しないでください。設定してしまった場合は、入力フィールド右にある▼をクリックして表示された一覧から[出力変数の消去]を選択し、設定を消去してください。

図E.23 Replyの設定画面



- 削除

  BPELエディタの[設計]タブで、削除するReplyをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するReplyをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するReplyをクリックし、プロパティビューで変更する設定値を編集します。

注意

- BPEL定義が要求/応答型の場合、Replyは、必ずReceiveとペアで使用してください。
- Replyは、ビジネス・プロセスの最後で実行し、Replyのあとにアクティビティを実行しないようBPEL定義を作成してください。
- Replyでメッセージを通知する場合は、プロパティビューの[詳細]の[障害名]を設定しないでください。障害を通知する場合、メッセージを通知することはできません。
- Replyの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

Assign

Assignの追加、削除、および編集は、以下のとおり行います。

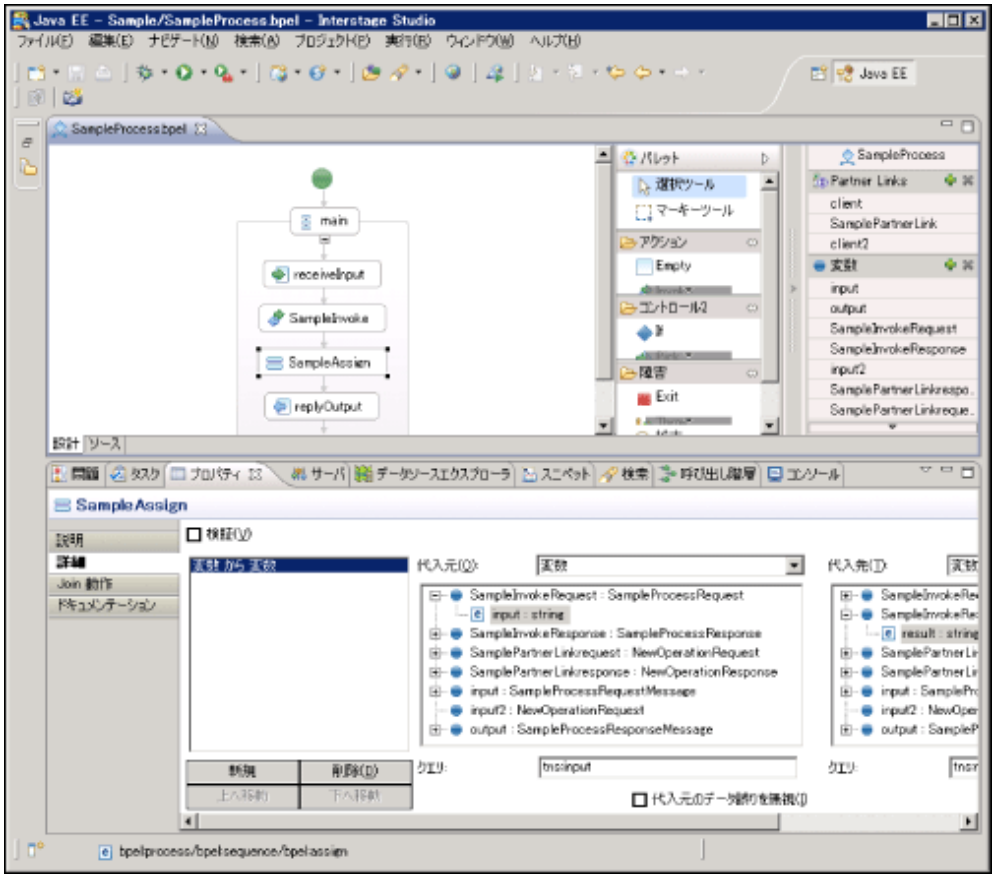
・ 追加

1. Assignの追加
   BPELエディタの[設計]タブの[パレット][アクション]内の[Assign]を、Assignを追加する位置にドラッグ&ドロップします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加したAssignをクリックし、プロパティビューの[説明]で[名前]にAssignの名前を設定します。
   次にプロパティビューの[詳細]で以下の設定を行います。

表E.18 Assignの設定内容

設定項目	設定内容
検証	初期設定のままとしてください(チェックなし)。
[新規]ボタン	新たなコピー定義を作成する場合にクリックしてください。
[削除]ボタン	定義済のコピー定義を削除する場合、削除するコピー定義を選択のうえクリックしてください。
[上へ移動]ボタン	コピー定義の実行順を他のコピー定義より先にする場合、実行順を変更するコピー定義を選択のうえクリックし、実行する位置に移動してください。
[下へ移動]ボタン	コピー定義の実行順を他のコピー定義より後にする場合、実行順を変更するコピー定義を選択のうえクリックし、実行する位置に移動してください。
代入元	入力フィールド右にある▼をクリックして表示された一覧から、コピー元種別をダブルクリックで選択します。選択は以下の3つのどれかを選択し、その他は選択しないでください。 ・[変数] ・[式] ・[定数値] 各種別の設定方法については、以下の“3.各種別の設定方法”を参照してください。 また、[代入元]と[代入先]で組合せ可能な変数型については、以下の“4. 組合せ可能な変数型”を参照してください。
代入先	入力フィールド右にある▼をクリックして表示された一覧から、コピー先種別をダブルクリックで選択します。選択は以下の2つのどちらかを選択し、その他は選択しないでください。 ・[変数] ・[式] 各種別の設定方法については、以下の“3.各種別の設定方法”を参照してください。 また、[代入元]と[代入先]で組合せ可能な変数型については、以下の“4. 組合せ可能な変数型”を参照してください。
代入元のデータ誤りを無視	コピー処理でエラーが発生しても無視する場合にチェックしてください。チェックしたときは、コピー処理でエラーが発生した場合でも、bpel:selectionFailure(障害)は通知されません。
代入元のエレメント名を維持	値だけではなく、要素名もコピー先にコピーする場合にチェックしてください。

図E.24 Assignの設定画面



3. 各種別の設定方法
プロパティビューの[詳細]の[代入元]、および[代入先]について種別ごとに設定方法を説明します。

表E.19 Assignの代入元/代入先の設定方法

種別	設定方法
変数	1. [代入元]、または[代入先]で入力フィールド右にある▼をクリックして表示された一覧から[変数]ダブルクリックで選択します。 2. 直下に変数の一覧が表示されますので、代入元、または代入先とする変数を選択します。 3. 代入先に選択した変数が設定中のAssign内で初期化されていない場合は、初期化処理を自動生成するか確認するメッセージが表示されます。自動生成する場合は[はい]を、自動生成しない場合は[いいえ]をクリックしてください（他のAssignプロパティで初期化済の場合は[いいえ]を選択してください）。
式	1. [代入元]、または[代入先]で入力フィールド右にある▼をクリックして表示された一覧から[式]をダブルクリックで選択します。 2. [式言語]で入力フィールド右にある▼をクリックして表示された一覧から[プロセス(XPath 1.0 in BPEL 2.0)と同じ]をダブルクリックで選択します。その他は選択しないでください。 3. 直下にXPath 1.0形式でコピー元、またはコピー先を定義します。
定数値	変数を初期化します（構造の初期化）。

種別	設定方法
※代入元にだけ指定できます。	1. [代入元]で入力フィールド右にある▼をクリックして表示された一覧から[定数値]をダブルクリックで選択します。 2. 直下にXML形式で初期化を定義します。 例） <tns:ColTest1Request xmlns:tns=”http://sample.bpel.org/bpel> <tns:input_bp></tns:input_bp> <tns:CID_BPELIN></tns:CID_BPELIN> </tns:ColTest1Request>

4. 組合せ可能な変数型
   代入元と代入先で組合せ可能な変数型は以下のとおりです。以下に示すうちのどれかを使用してください。

   - 代入元、代入先ともに同じ変数型
   - 代入元と代入先で異なる変数型の場合は、以下の組合せだけ可能です。その他の組合せは使用しないでください。

表E.20 Assignの代入元と代入先で組合せ可能な変数型

代入元	代入先
integer	decimal
long	decimal
	integer
int	decimal
	integer
	Long
short	decimal
	integer
	Long
	Int
byte	decimal
	integer
	Long
	Int
	short
nonNegativeInteger	decimal
	integer
positiveInteger	decimal
	integer
	nonNegativeInteger
unsignedLong	decimal
	integer
	nonNegativeInteger
unsignedInt	decimal
	integer

代入元	代入先
	nonNegativeInteger
	unsignedLong
unsignedShort	decimal
	integer
	nonNegativeInteger
	unsignedLong
	unsignedInt
unsignedByte	decimal
	integer
	nonNegativeInteger
	unsignedLong
	unsignedInt
	unsignedShort
nonPositiveInteger	decimal
	integer
	Long
negativeInteger	integer
	Long
	nonPositiveInteger
normalizedString	string
token	string
	normalizedString
language	string
	normalizedString
	token
Name	string
	normalizedString
	token
NCName	string
	normalizedString
	token
	Name
ID	string
	normalizedString
	token
	Name
	NCName
NMTOKEN	string
	normalizedString

代入元	代入先
	token

- 削除

  BPELエディタの[設計]タブで、削除するAssignをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するAssignをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するAssignをクリックし、プロパティビューで変更する設定値を編集します。

  なお、障害ハンドラ内のAssignの場合は、Assignが配置されているCatch、またはCatch Allをクリックしたあと、Assignをクリックしてプロパティの変更を行ってください。

注意

- 検証はサポートしていません。チェックしても無視されます。
- 代入元、および代入先に指定できる種別は以下のとおりです。その他は指定しないでください。

表E.21 Assignの代入元/代入先で指定できる種別

指定位置	指定できる種別
代入元	・[変数] ・[式] ・[定数値]
代入先	・[変数] ・[式]

- [代入先]に設定する変数の[データ型]が[要素宣言]または[メッセージ]の場合、変数の中の整数型のElementに値をコピーするときは、[代入元]に[式]を指定しないでください。コピーするときは、[式]の結果を[データ型]が[プリミティブ型]または[Simple Types]の変数にいったんコピーしたあと、本来のコピー先にコピーしてください。
- Xpath1.0形式で指定する場合、以下の関数は使用しないでください。
  - lang()
  - doXslTransform()
- [代入先]に不当なXPath1.0関数を指定しても、bpel:selectionFailureは通知されません。
- Assignの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

Throw

Throwの追加、削除、および編集は、以下のとおり行います。

- 追加

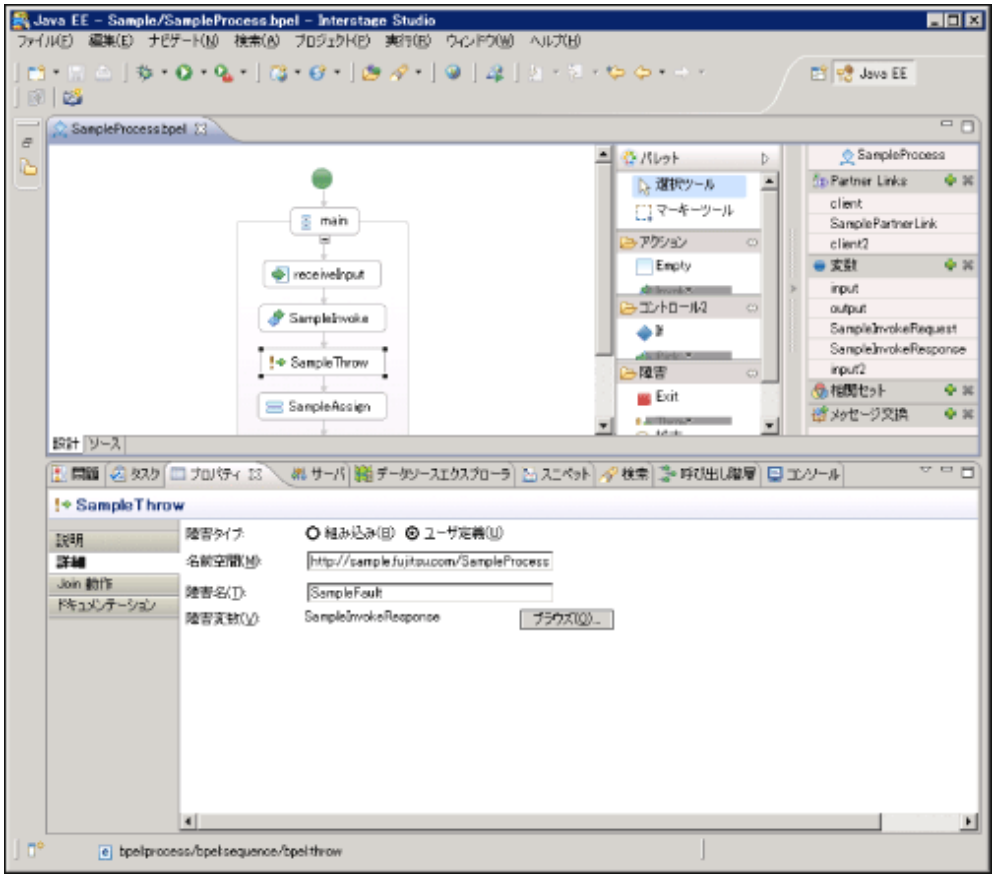
  1. Throwの追加
     BPELエディタの[設計]タブの[パレット][障害]内の[Throw]を、Throwを追加する位置にドラッグ＆ドロップします。

2. プロパティの設定
BPELエディタの[設計]タブで、追加したThrowをクリックし、プロパティビューの[説明]で[名前]にThrowの名前を設定します。
次にプロパティビューの[詳細]で以下の設定を行います。

表E.22 Throwの設定内容

設定項目	設定内容
障害タイプ	以下のどちらかを選択します。 ・ [組み込み] BPEL標準の障害をThrowする場合。 ・ [ユーザ定義] ユーザ定義の障害をThrowする場合。
名前空間	[ユーザ定義]を選択した場合にだけ設定できます。 名前空間に設定した障害が属する名前空間を設定します。 名前空間は必ず設定してください。
障害名	Throwする障害名を設定します。 [組み込み]を選択した場合は、入力フィールド右にある▼をクリックして表示された一覧から、Throwする障害をダブルクリックで選択します。 [ユーザ定義]を選択した場合は、Throwする障害を設定します。
障害変数	障害名とともに情報を通知する場合、通知する情報が格納された変数を指定します。 [ブラウズ]ボタンをクリックし、表示された一覧から選択してください。

図E.25 Throwの設定画面



- 削除

  BPELエディタの[設計]タブで、削除するThrowをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するThrowをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するThrowをクリックし、プロパティビューで変更する設定値を編集します。

注意

- Throwの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

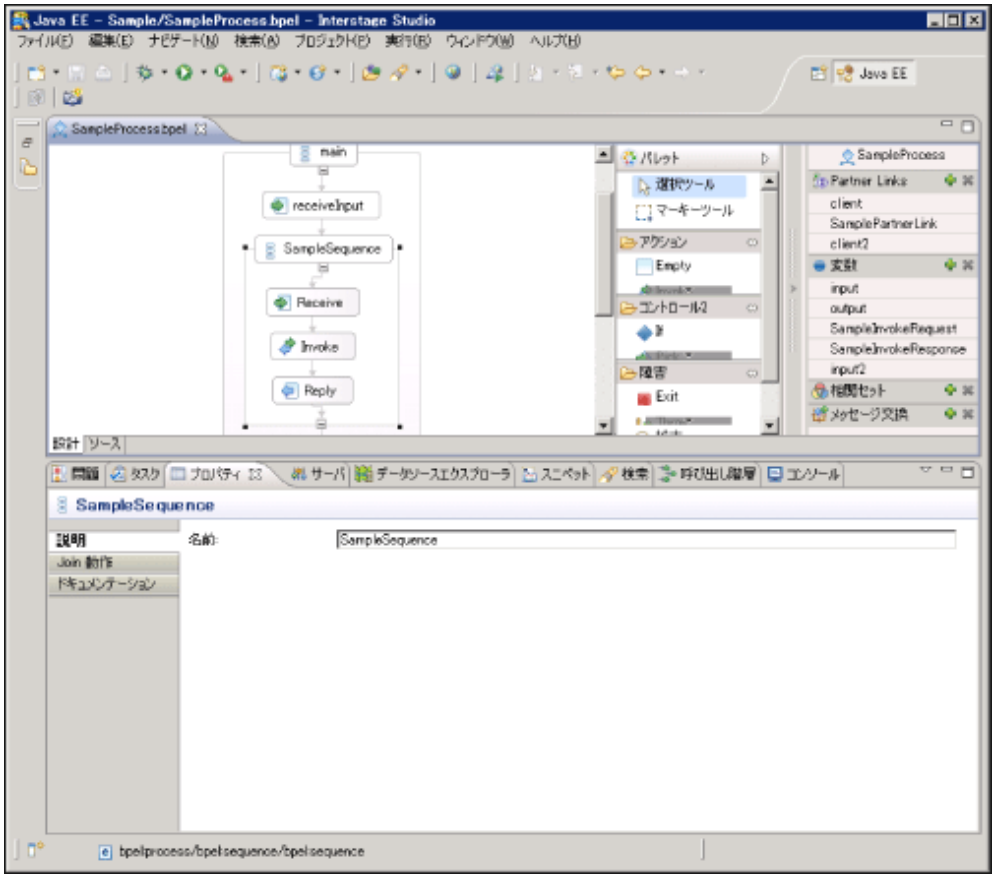
Sequence

Sequenceの追加、削除、および編集は、以下のとおり行います。

・ 追加

1. Sequenceの追加
   BPELエディタの[設計]タブの[パレット][コントロール2]内の[Sequence]を、Sequenceを追加する位置にドラッグ&ドロップします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加したSequenceをクリックし、プロパティビューの[説明]で[名前]にSequenceの名前を設定します。
3. Sequence内にアクティビティを配置
   BPELエディタの[設計]タブで、順次実行したいアクティビティをSequenceの枠の中に追加します。

図E.26 Sequenceの設定画面



・ 削除

BPELエディタの[設計]タブで、削除するSequenceをクリックし、キーボードの[Delete]キーを押下します。

・ 編集

場所を移動する場合は、BPELエディタの[設計]タブで、移動するSequenceをクリックし、移動先までドラッグ&ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するSequenceをクリックし、プロパティビューで変更する設定値を編集します。

注意

- Sequenceの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

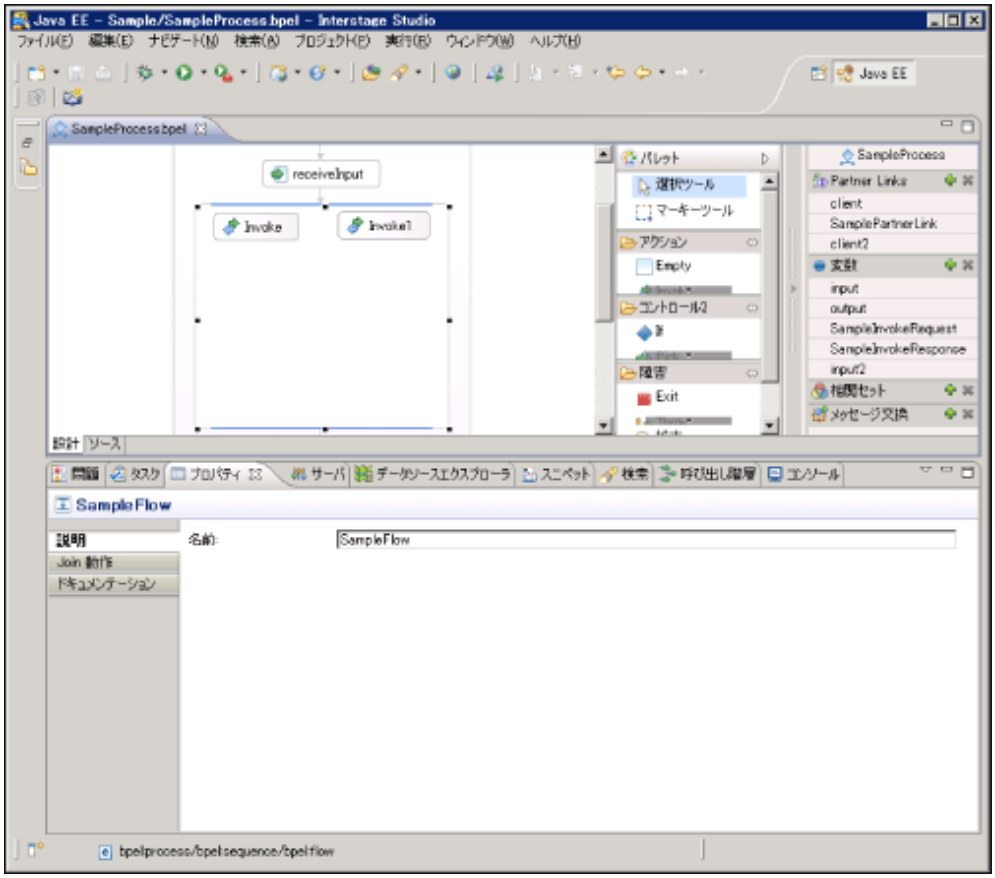
Flow

Flowの追加、削除、および編集は、以下のとおり行います。

- 追加

1. Flowの追加
   BPELエディタの[設計]タブの[パレット][コントロール2]内の[Flow]を、Flowを追加する位置にドラッグ＆ドロップします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加したFlowをクリックし、プロパティビューの[説明]で[名前]にFlowの名前を設定します。
3. Flow内にアクティビティを配置
   BPELエディタの[設計]タブで、並列実行したいアクティビティをFlowの枠の中に追加します。

図E.27 Flowの設定画面



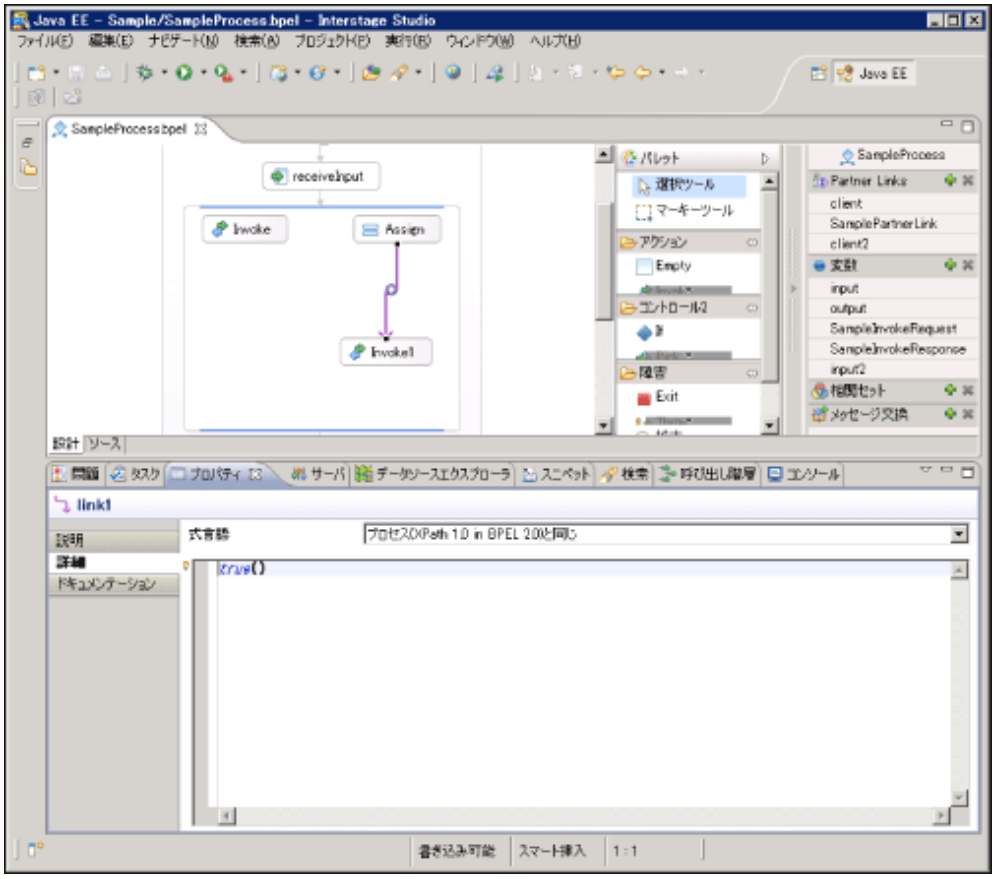
4. Linkの追加
   Flow中のアクティビティで順次実行したいアクティビティがある場合は、先に実行するアクティビティと次に実行するアクティビティをLinkで結合します。Linkは1つのアクティビティに対して複数設定することもできます。Linkは以下の手順で作成します。

1. BPELエディタの[設計]タブで、Flow内の先に実行するアクティビティを右クリックし、ポップアップメニューの[Linkを追加]をクリックします。
2. 次に実行するアクティビティをクリックすると、Link元のアクティビティからLink先のアクティビティに矢印（Link）が追加されます。
3. BPELエディタの[設計]タブで、追加したLinkをクリックし、プロパティビューの[詳細]で以下の設定を行います。
なお、プロパティビューの[説明]の[名前]は変更しないでください。初期値のLink名を、そのまま使用してください。

表E.23 Linkの設定内容

設定項目	設定内容
式言語	以下のどちらかを選択します。その他は選択しないでください。 ・ [式がありません] 無条件に次のアクティビティに遷移する場合。 ・ [プロセス(XPath 1.0 in BPEL 2.0)と同じ] 本選択を行った場合、直下にXPath1.0の条件式を定義します。条件式の結果（TrueまたはFalse）は、Link変数（変数名はLink名と同じ）に格納され、Link先のアクティビティの実行条件判定に利用することができます。Link先のアクティビティの実行条件の設定については、以下の“5.Link先のアクティビティの実行条件の設定”を参照してください。
[新しい条件の作成]ボタン	クリックすると[式言語]で[プロセス(XPath 1.0 in BPEL 2.0)と同じ]が選択されます。

図E.28 Linkの設定画面

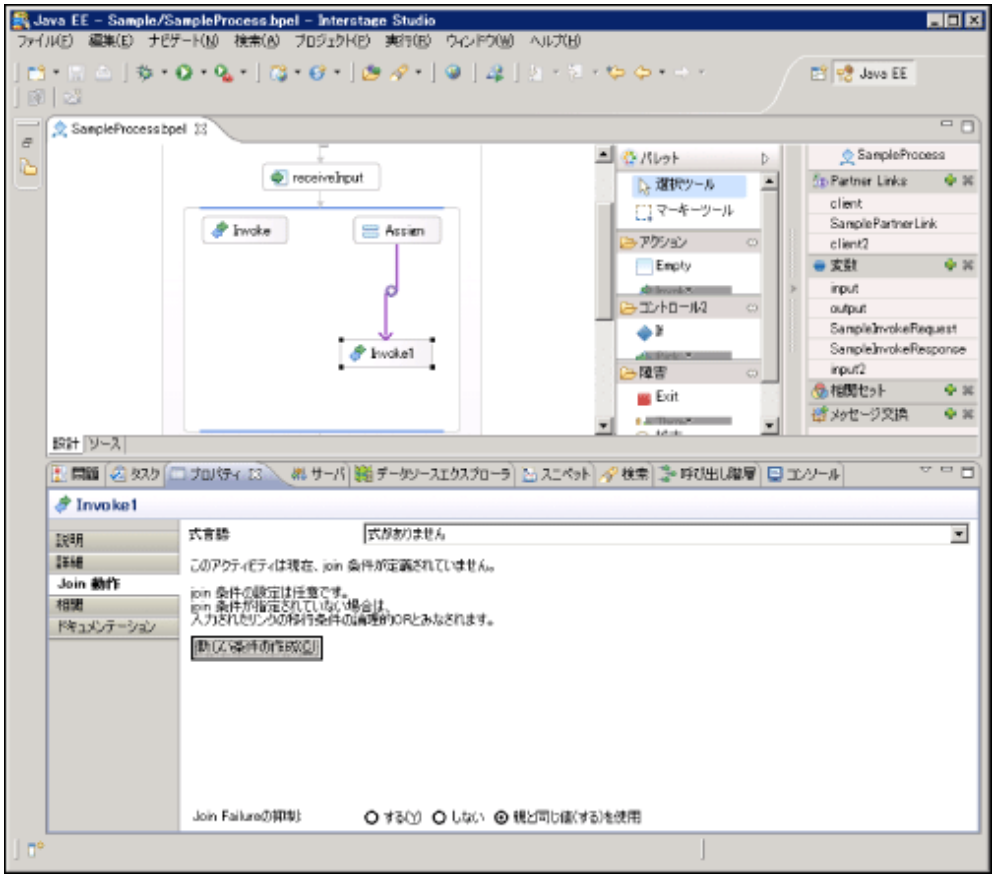


5. Link先のアクティビティの実行条件の設定
   Link先のアクティビティを実行する条件を設定することができます。
   設定する場合は、BPELエディタの[設計]タブで、実行条件を設定するアクティビティをクリックし、プロパティビューの[Join動作]で以下の設定を行います。

表E.24 アクティビティの実行条件の設定内容

設定項目	設定内容
式言語	以下のどちらかを選択します。その他は選択しないでください。 ・ [式がありません] 無条件にアクティビティを実行する場合。 ・ [プロセス(XPath 1.0 in BPEL 2.0)と同じ] アクティビティを実行する条件を指定します。条件は直下にXPath1.0の条件式で定義します。条件式には本アクティビティ前のLinkのLink変数（変数名はLink名と同じ）を利用することができます。条件式の結果がTrueの場合、アクティビティを実行し、Falseの場合はbpel:joinFailure（障害）が通知されます。なお、以下のSuppress Join Failureの設定が[する]の場合は、bpel:joinFailure（障害）は通知しません。 Link変数の設定については、上記の“4.Linkの追加”を参照してください。
[新しい条件の作成]ボタン	クリックすると[式言語]で[プロセス(XPath 1.0 in BPEL 2.0)と同じ]が選択されます。
Join Failureの抑制	以下のどれかを選択します。その他は選択してないでください。 ・ する 実行条件がFalseの場合でもbpel:joinFailure（障害）を通知しない。 ・ しない 実行条件がFalseの場合bpel:joinFailure（障害）を通知する。 ・ 親と同じ値を使用 BPEL定義のプロパティ設定に従う。

図E.29 アクティビティの実行条件の設定内容



- 削除

  BPELエディタの[設計]タブで、削除するFlowをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するFlowをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するFlowをクリックし、プロパティビューで変更する設定値を編集します。

注意

- Linkで無限ループする定義は作成しないでください。作成するとビジネス・プロセス実行時に無限ループし、ビジネス・プロセスの実行がタイムアウトとなります。
- BPELエディタの[設計]タブでSequence、If、Whileなどの折りたたみ表示（各アクティビティ内に定義したアクティビティの表示を省略する表示）を行うと、Linkの矢印表示が異常となります。折りたたみ表示から元の表示に戻すことで矢印は正常な表示に戻ります。
- Flowの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。
- Linkのプロパティビューの[説明]の[名前]は変更しないでください。初期値のLink名を、そのまま使用してください。

If

Ifの追加、削除、および編集は、以下のとおり行います。

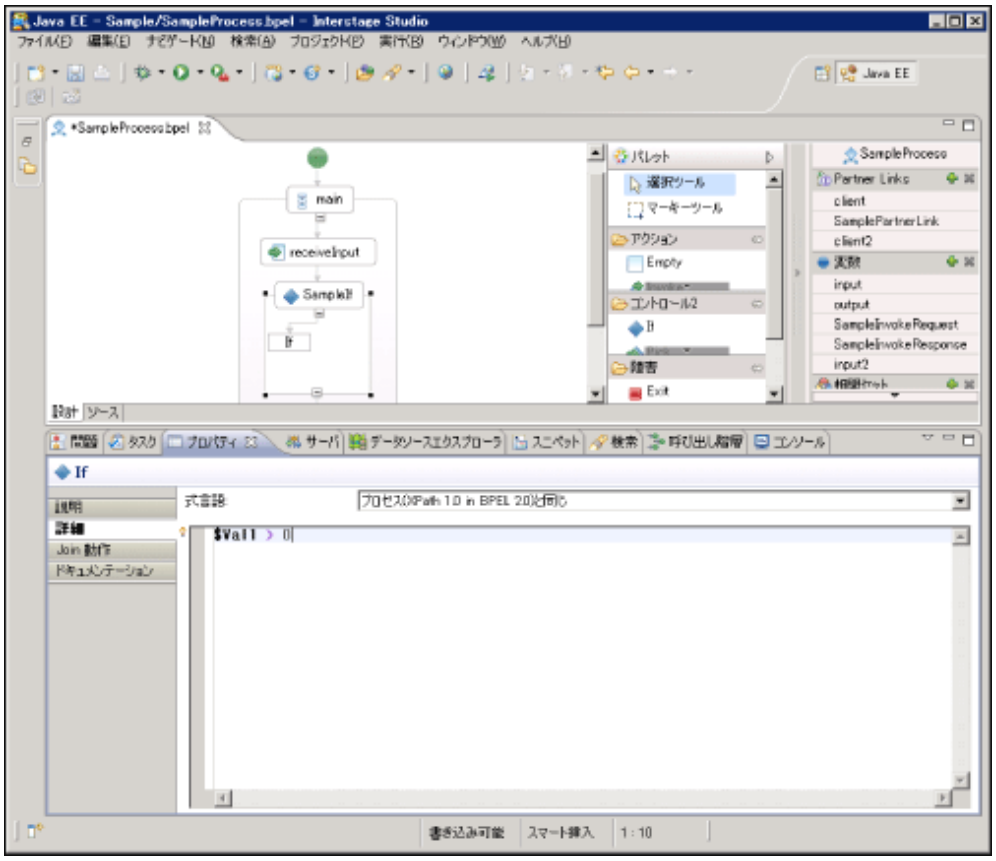
- 追加

1. Ifの追加
   BPELエディタの[設計]タブの[パレット][コントロール2]内の[If]を、Ifを追加する位置にドラッグ＆ドロップします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加したIfをクリックし、プロパティビューの[説明]で[名前]にIfの名前を設定します。
   次にプロパティビューの[詳細]で以下の設定を行います。

表E.25 Ifの設定内容

設定項目	設定内容
式言語	以下を選択します。その他は選択しないでください。 ・ [プロセス(XPath 1.0 in BPEL 2.0)と同じ] 次のアクティビティに遷移する条件を指定します。条件は直下にXPath1.0の条件式で定義します。条件式の結果がTrueの場合、次のアクティビティに遷移し、Falseの場合は遷移しません。
[新しい条件を作成]ボタン	クリックすると[式言語]で[プロセス(XPath 1.0 in BPEL 2.0)と同じ]が選択されます。

図E.30 Ifの設定画面



3. Ifで実行するアクティビティを配置
   BPELエディタの[設計]タブで、Ifで実行したいアクティビティをIfのあとに追加します。必ず何らかのアクティビティを配置してください。

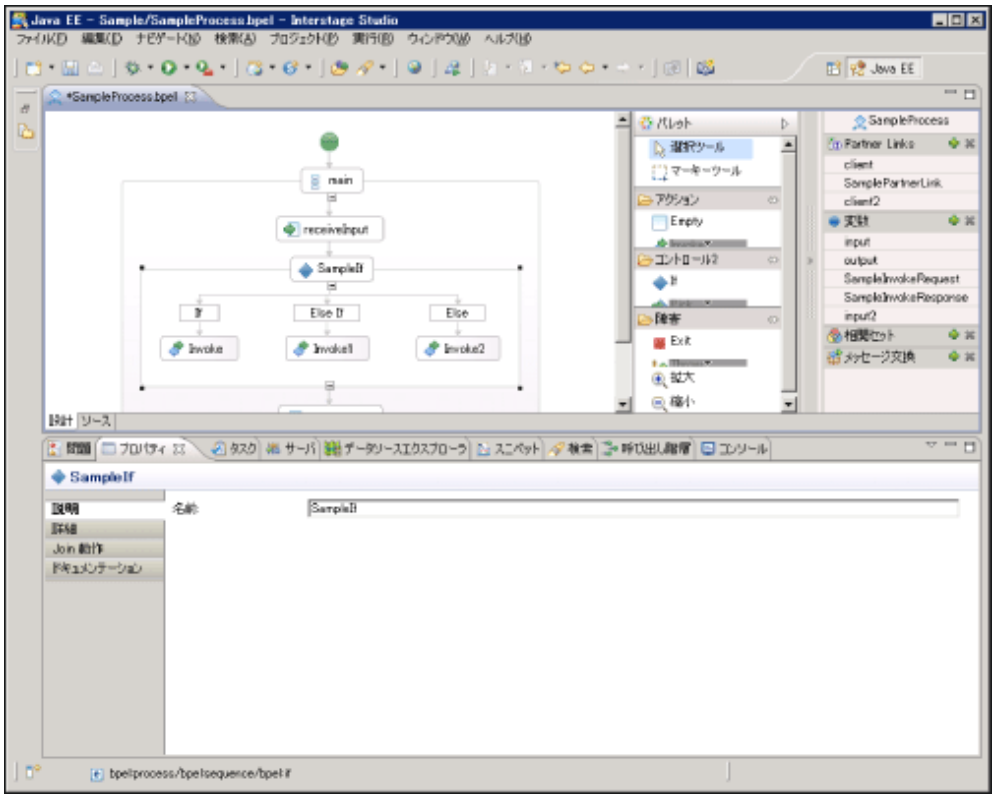
4. Else/Else Ifの追加
   Ifの条件に合致しなかったときにアクティビティを実行したい場合はElseを、更に条件を判定のうえアクティビティを実行したい場合は、Else Ifを使用します。Else/Else Ifは次の手順で追加します。

   1. BPELエディタの[設計]タブで、追加したIfをクリックし、ポップアップメニューの[Elseを追加]、または[Else Ifを追加]をクリックします。
   2. BPELエディタの[設計]タブで、追加したElse、またはElse Ifをクリックし、プロパティビューの [詳細]で以下の設定を行います。

表E.26 Else/Else Ifの設定内容

設定項目	設定内容
式言語	以下を選択します。その他は選択しないでください。 ・ [プロセス(XPath 1.0 in BPEL 2.0)と同じ] 次のアクティビティに遷移する条件を指定します。条件は直下にXPath1.0の条件式で定義します。条件式の結果がTrueの場合、次のアクティビティに遷移し、Falseの場合は遷移しません。
[新しい条件を作成]ボタン	クリックすると[式言語]で[プロセス(XPath 1.0 in BPEL 2.0)と同じ]が選択されます。

図E.31 Else/Else Ifの設定画面



3. BPELエディタの[設計]タブで、Else、またはElse Ifで実行したいアクティビティをElse、またはElse Ifのあとに追加します。必ず何らかのアクティビティを配置してください。

- 削除

  BPELエディタの[設計]タブで、削除するIfをクリックし、キーボードの[Delete]キーを押下します。Else、およびElse Ifも同様に削除できます。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するIfをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するIf、Else、またはElse Ifをクリックし、プロパティビューで変更する設定値を編集します。

注意

- Ifの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

While

Whileの追加、削除、および編集は、以下のとおり行います。

- 追加

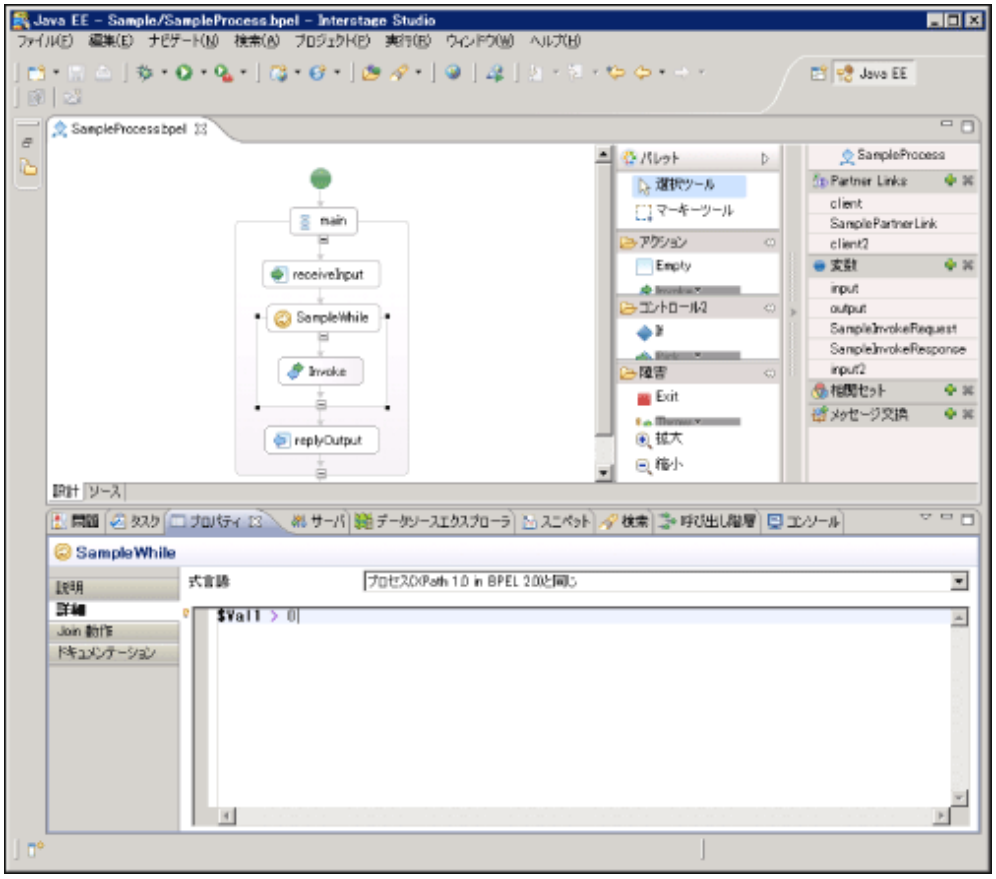
  1. Whileの追加
     BPELエディタの[設計]タブの[パレット][コントロール2]内の[While]を、Whileを追加する位置にドラッグ＆ドロップします。
  2. プロパティの設定
     BPELエディタの[設計]タブで、追加したWhileをクリックし、プロパティビューの[説明]で[名前]にWhileの名前を設定します。
     次にプロパティビューの[詳細]で以下の設定を行います。

表E.27 Whileの設定内容

設定項目	設定内容
式言語	以下を選択します。その他は選択しないでください。 ・ [プロセス(XPath 1.0 in BPEL 2.0)と同じ] 次のアクティビティに遷移する条件指定します。条件は直下にXPath1.0の条件式で定義します。条件式の結果がTrueの場合、次のアクティビティに遷移し、Falseの場合は遷移しません。
[新しい条件の作成]ボタン	クリックすると[式言語]で[プロセス(XPath 1.0 in BPEL 2.0)と同じ]が選択されます。

3. Whileで実行するアクティビティを配置
   BPELエディタの[設計]タブで、Whileで実行したいアクティビティをWhileの枠の中に追加します。必ず何らかのアクティビティを配置してください。

図E.32 Whileの設定画面



- 削除

  BPELエディタの[設計]タブで、削除するWhileをクリックし、キーボードの[Delete]キーを押下します。

- 編集

  場所を移動する場合は、BPELエディタの[設計]タブで、移動するWhileをクリックし、移動先までドラッグ＆ドロップします。また、プロパティを変更する場合は、BPELエディタの[設計]タブで、変更するWhileをクリックし、プロパティビューで変更する設定値を編集します。

注意

- While内に定義されたアクティビティを条件が真の間、繰返し実行します。条件判定は事前に行うため、条件判定の結果が偽の場合は、While内のアクティビティは一度も実行されずにWhileを終了します。
- Whileの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

Variables

変数の追加、削除、および編集は、以下のとおり行います。

- 追加

1. 変数の追加
   BPELエディタの[設計]タブの[変数]横の[＋]をクリックします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加された変数をクリックし、プロパティビューの[説明]で[名前]に変数の名前を設定します。
   次にプロパティビューの[詳細]、および[初期化]で以下の設定を行います。

表E.28 変数の設定内容

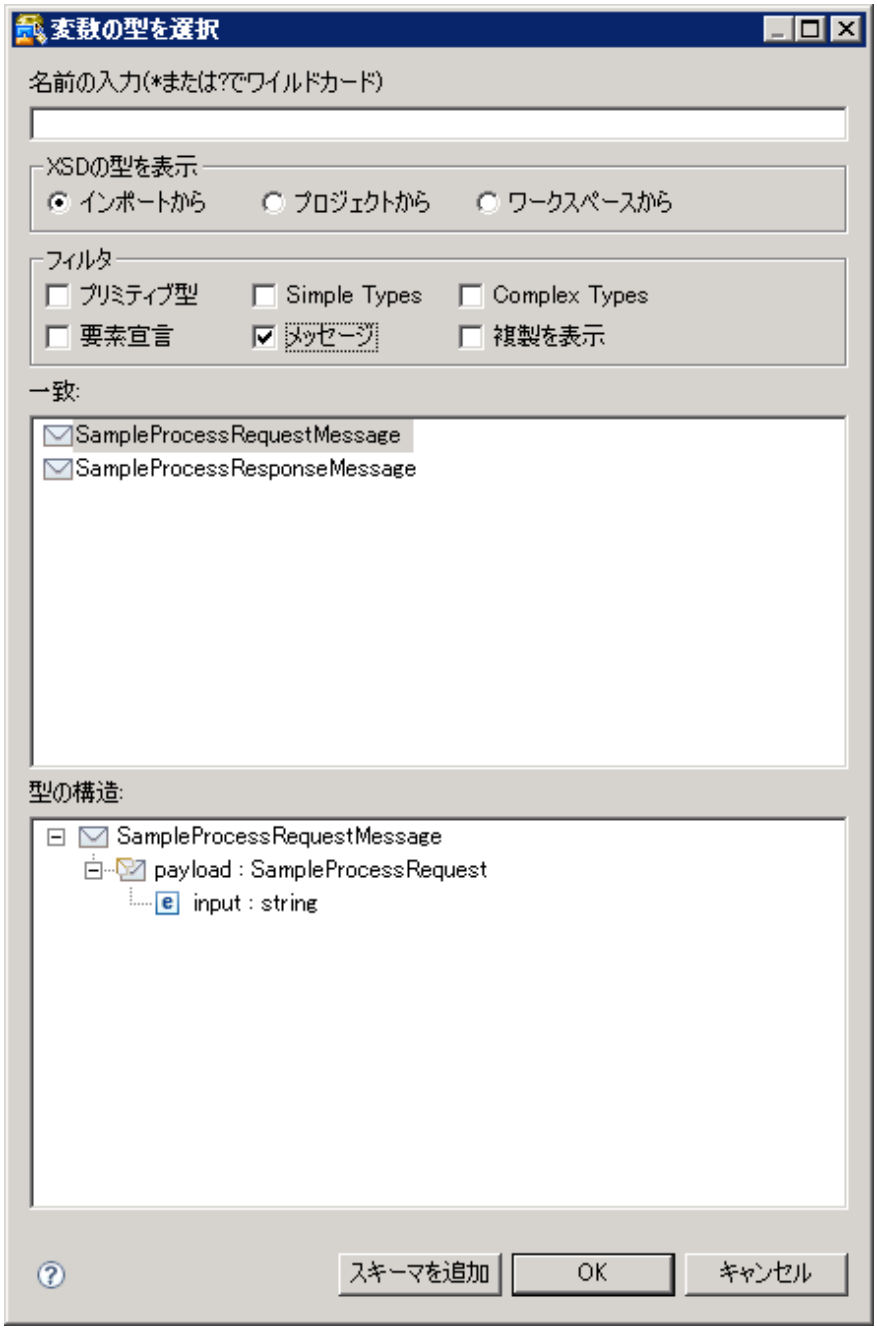
設定場所	設定項目	設定内容
詳細	データ型	[ブラウズ]ボタンをクリックし、[変数の型を選択]ウインドウでデータ型を指定します。 [変数の型を選択]ウインドウについては、以下の“3.[変数の型を選択]の設定”を参照してください。
	構造	設定した[データ型]の構造を表示します。
初期化	代入元	初期設定のままとしてください（Noneを選択）。

3. [変数の型を選択]の設定
   以下の設定を行います。

表E.29 [変数の型を選択]の設定内容

設定項目	設定内容
名前の入力	[一致]に表示されるデータ型を絞り込むためのキーワードを入力します。
XSDの型を表示	[一致]に表示するデータ型の範囲を指定します。以下のどれかを選択します。 ・ [インポートから] BPELプロセスでインポートした定義 ・ [プロジェクトから] プロジェクト内の定義 ・ [ワークスペースから] ワークスペース内の定義
フィルタ	[一致]に表示するデータ型の種別を指定します。表示したい種別をチェックしてください。必ず必要なもの1つだけチェックしてください。
一致	追加する変数のデータ型を選択してください。 BPEL機能で使用できるデータ型については、以下の“注意事項”を参照してください。
型の構造	[一致]で選択したデータ型の構造を表示します。
[スキーマを追加]ボタン	データ型を追加したい場合にクリックしてください。
[OK]ボタン	[一致]で選択したデータ型の選択を確定します。
[キャンセル]ボタン	データ型の選択を取り消します。

図E.33 [変数の型を選択]画面



- 削除

  BPELエディタの[設計]タブで、[変数]下の削除する変数をクリックし、[変数]横の[×]をクリックします。

  なお、変数を削除する場合は、事前に削除する変数を使用している定義(Assignの代入元/代入先定義やInvokeの入力変数/出力変数など)を編集して削除する変数を使用しないようにしてください。

- 編集

  プロパティを変更する場合は、BPELエディタの[設計]タブで、[変数]下の変更する変数をクリックし、プロパティビューで変更する設定値を編集します。

なお、変数の名前の変更は行わないでください。名前の変更を行う場合は、該当変数をいったん削除し、新たな名前の変数を新規に作成してください。

注意

- 変数のプロパティの[初期化]での初期化定義は未サポートです。変数の初期化はAssignで実施してください。
- 変数の名前として使用できる文字列は、NCNameに準拠します。ただし、“.”(ピリオド)は使用しないでください。
- BPEL機能で使用できるデータ型は、Interstage Application Serverに準拠していますので、その他のデータ型は使用しないでください。使用できるデータ型については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発”の“アプリケーションの作成方法”の“Webサービスの通信で利用できるデータ型”を参照してください。
  また、Complex Type、QName、およびAttributeは使用しないでください。
  要素宣言については、WSDLで定義したものだけ使用してください。
- 変数の名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

PartnerLinks

PartnerLinkの追加、削除、および編集は、以下のとおり行います。

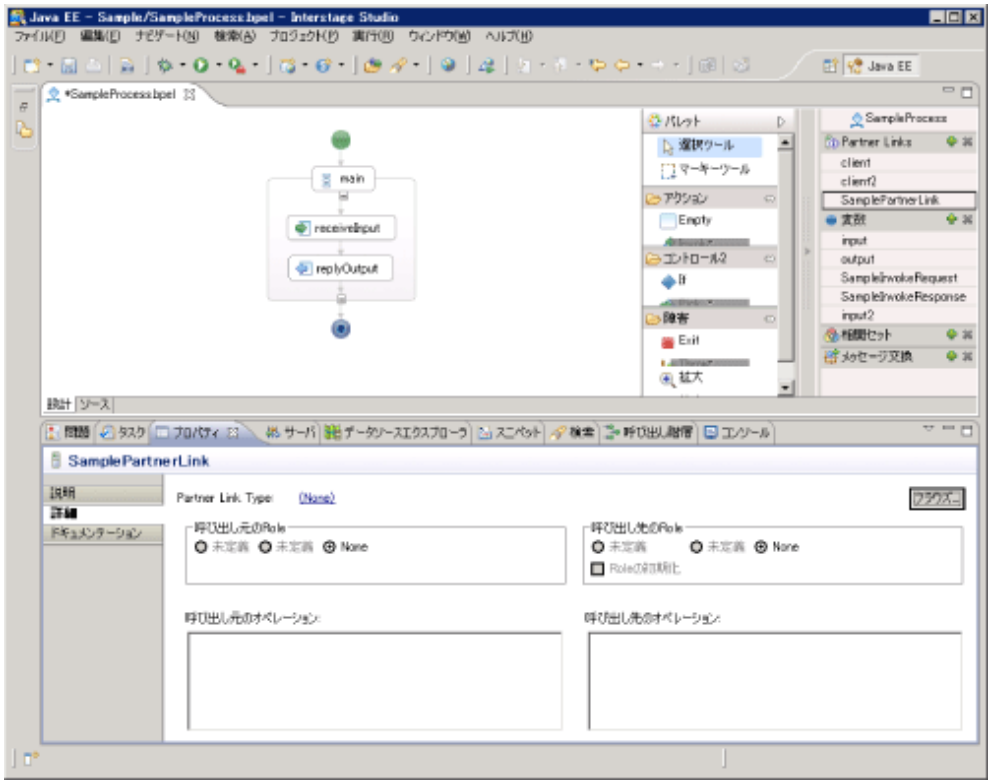
- 追加

  1. WSDL定義のの格納
     BPEL定義から呼び出すサービスのWSDL定義を、BPELプロジェクト配下に格納してください。
  2. PartnerLinkの追加
     BPELエディタの[設計]タブの[PartnerLinks]横の[＋]をクリックします。
  3. プロパティの設定
     BPELエディタの[設計]タブで、追加されたPartnerLinkをクリックし、プロパティビューの[説明]で[名前]にPartnerLinkの名前を設定します。
     次にプロパティビューの[詳細]で以下の設定を行います。

表E.30 PartnerLinksの設定内容

設定項目	設定内容
Partner Link Type	[ブラウズ]ボタンをクリックし、[Partner Link typeを選択]ウインドウでPartner Link Typeを指定します。 [Partner Link typeを選択]ウインドウについては、以下の“3.[Partner Link typeを選択]の設定”を参照してください。
呼び出し元のRole	BPEL定義(Receive/Replyで使用)用のRoleが表示されます。
呼び出し先のRole	呼び出すサービス(Invokeで使用)用のRoleが表示されます。
呼び出し元のオペレーション	BPEL定義(Receive/Replyで使用)用のPartner Linkのオペレーションが表示されます。
呼び出し先のオペレーション	呼び出すサービス(Invokeで使用)用のPartner Linkのオペレーションが表示されます。

図E.34 PartnerLinksの設定画面



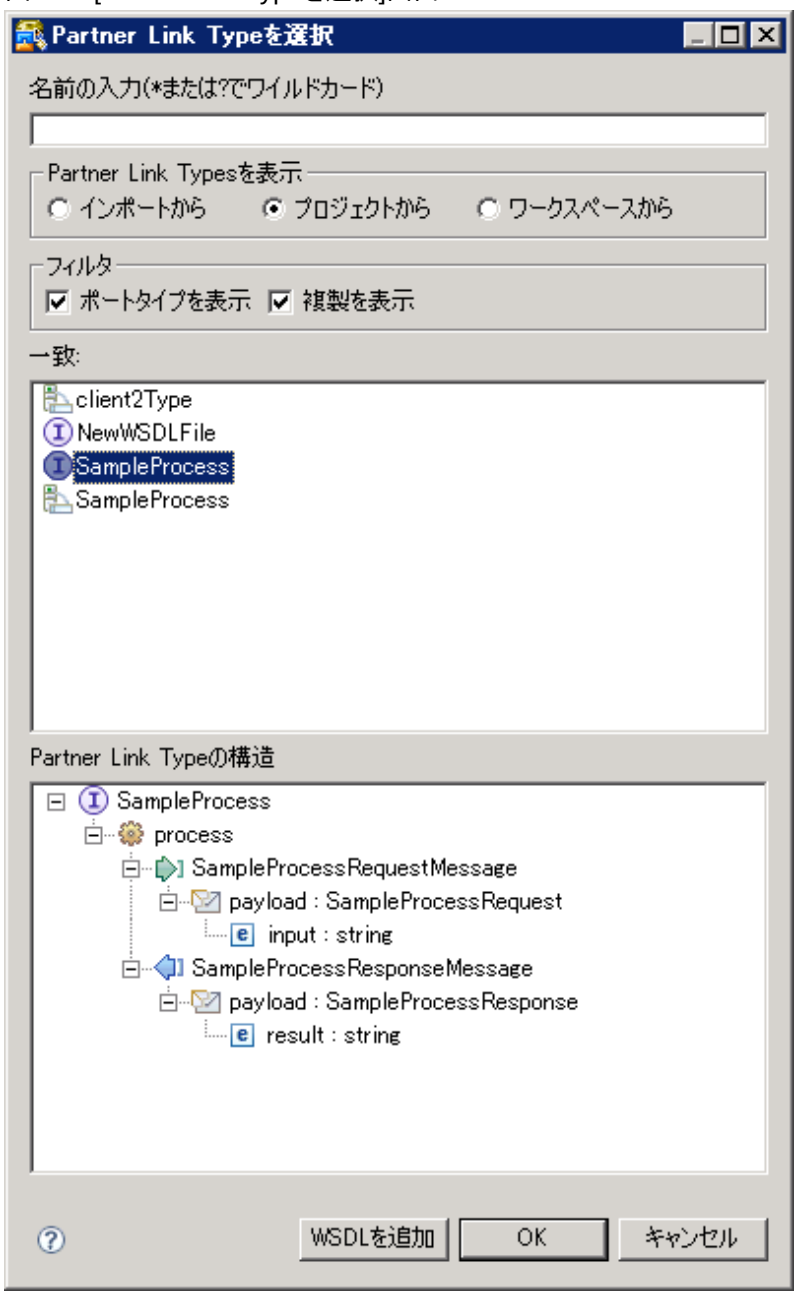
4. [Partner Link typeを選択]の設定
以下の設定を行います。

表E.31 [Partner Link typeを選択]の設定内容

設定項目	設定内容
名前	[一致]に表示されるPartner Link type、またはポートタイプを絞り込むためのキーワードを入力します。
Partner Link Typesを表示	[一致]に表示するPartner Link type、またはポートタイプの範囲を指定します。以下のどれかを選択します。 ・ [インポートから] BPELプロセスでインポートした定義 ・ [プロジェクトから] プロジェクト内の定義 ・ [ワークスペースから] ワークスペース内の定義
フィルタ	[一致]に表示するPartner Link type、またはポートタイプの種別を指定します。表示したい種別をチェックしてください。必ず必要なもの1つだけチェックしてください。
一致	追加するPartnerLinkのPartner Link type、またはポートタイプを選択してください。ポートタイプを選択して[OK]ボタンをクリックした場合は、[Partner Link Typeの作成]ウィザードが起動されます。[Partner Link Typeの作成]ウィザードについては以下の“4.[Partner Link Typeの作成]ウィザード”を参照してください。
Partner Link Typeの構造	[一致]で選択したPartner Link Type、またはポートタイプの構造を表示します。
[WSDLを追加]ボタン	Partner Link Typeを追加したい場合にクリックしてください。

設定項目	設定内容
[OK]ボタン	[一致]で選択したPartner Link Type、またはポートタイプの選択を確定します。
[キャンセル]ボタン	Partner Link Typeの選択を取り消します。

図E.35 [Partner Link typeを選択]画面



5. [Partner Link Typeの作成]ウィザード
   以下の手順でPartner Linkを作成します。

   1. [Partner Link Typeの名前]に作成するPartnerLinkの名前を入力し、[次へ]ボタンをクリックします。
   2. [Role名]にRoleの名前を入力し、[ポートタイプ]でRoleを作成するポートタイプを選択します。

3. Roleを作成するポートタイプが他にある場合は、[次へ]をクリックし、手順2)の操作を行います。
4. Roleを作成するポートタイプが他にない場合は、[完了]をクリックします。

- 削除

  BPELエディタの[設計]タブで、[PartnerLinks]下の削除するPartnerLinkをクリックし、[PartnerLinks]横の[×]をクリックします。

  なお、PartnerLinkを削除する場合は、事前に削除するPartnerLinkを使用している定義（Invoke/Receive/InvokeのPartner Linkなど）を編集して削除するPartnerLinkを使用しないようにしてください。

- 編集

  プロパティを変更する場合は、BPELエディタの[設計]タブで、[PartnerLinks]下の変更するPartnerLinkをクリックし、プロパティビューで変更する設定値を編集します。

  なお、PartnerLinkの名前の変更は行わないでください。名前の変更を行う場合は、該当PartnerLinkをいったん削除し、新たな名前のPartnerLinkを新規に作成してください。

注意

- PartnerLinkの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

FaultHandlers

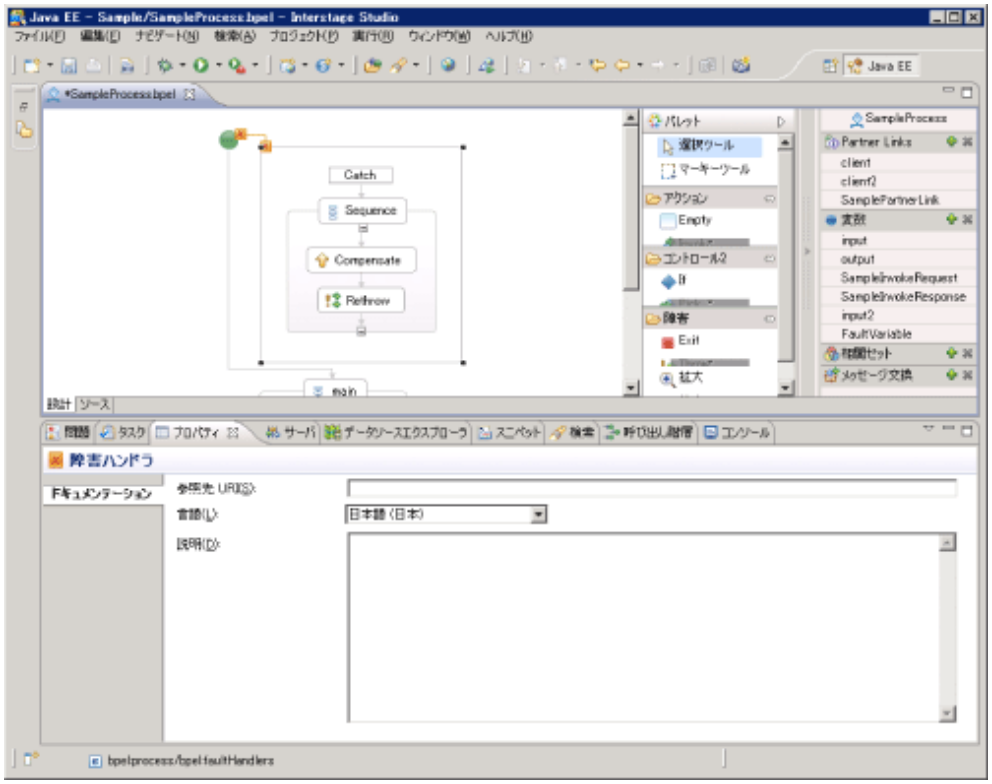
障害ハンドラの追加、削除、および編集は、以下のとおり行います。

- 追加

  1. 障害ハンドラの追加
     BPEL定義全体に障害ハンドラを追加する場合は、BPELエディタの[設計]タブのBPELプロセスの始点（先頭の緑色の丸印）を右クリックし、ポップアップメニューの[障害ハンドラを追加]をクリックします。

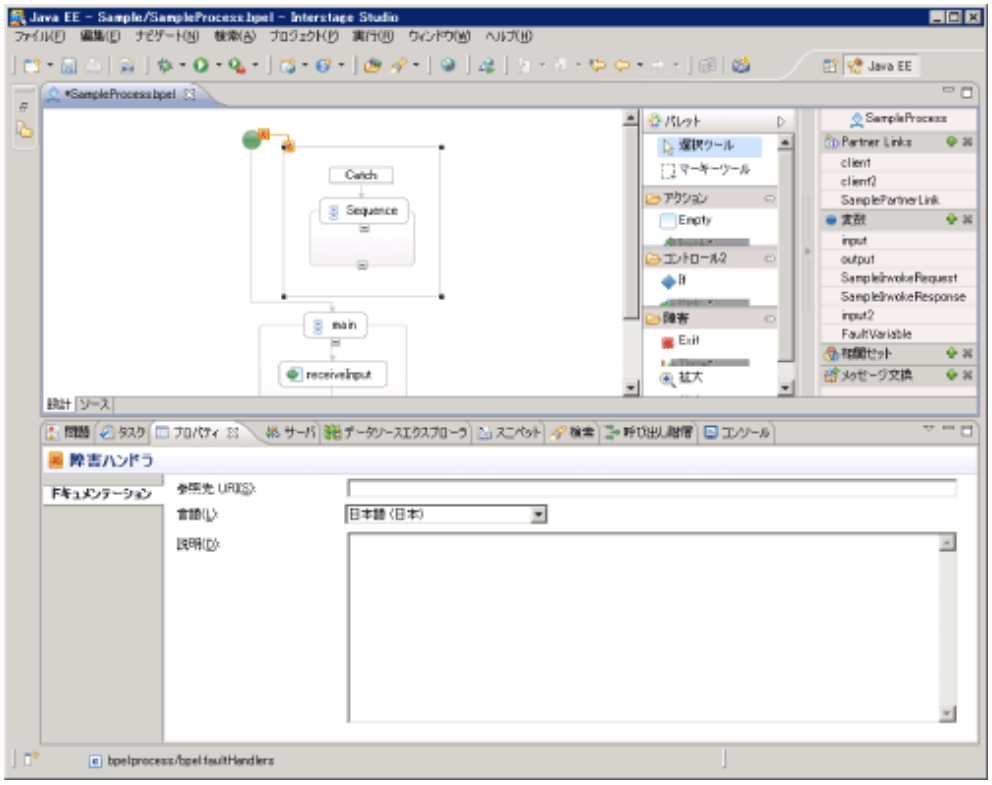
また、Invokeに障害ハンドラを追加する場合は、BPELエディタの[設計]タブの障害ハンドラを追加するInvokeを右クリックし、ポップアップメニューの[障害ハンドラを追加]をクリックします。

図E.36 障害ハンドラの作成画面



2. 不要なアクティビティの削除
   BPELエディタの[設計]タブで、作成した障害ハンドラの枠内の[Conpensate]、および[Rethrow]を削除します。削除は、それぞれのアクティビティを選択のうえ、キーボードの[Delete]キーで削除してください。

図E.37 障害ハンドラの作成画面（不要なアクティビティを削除）



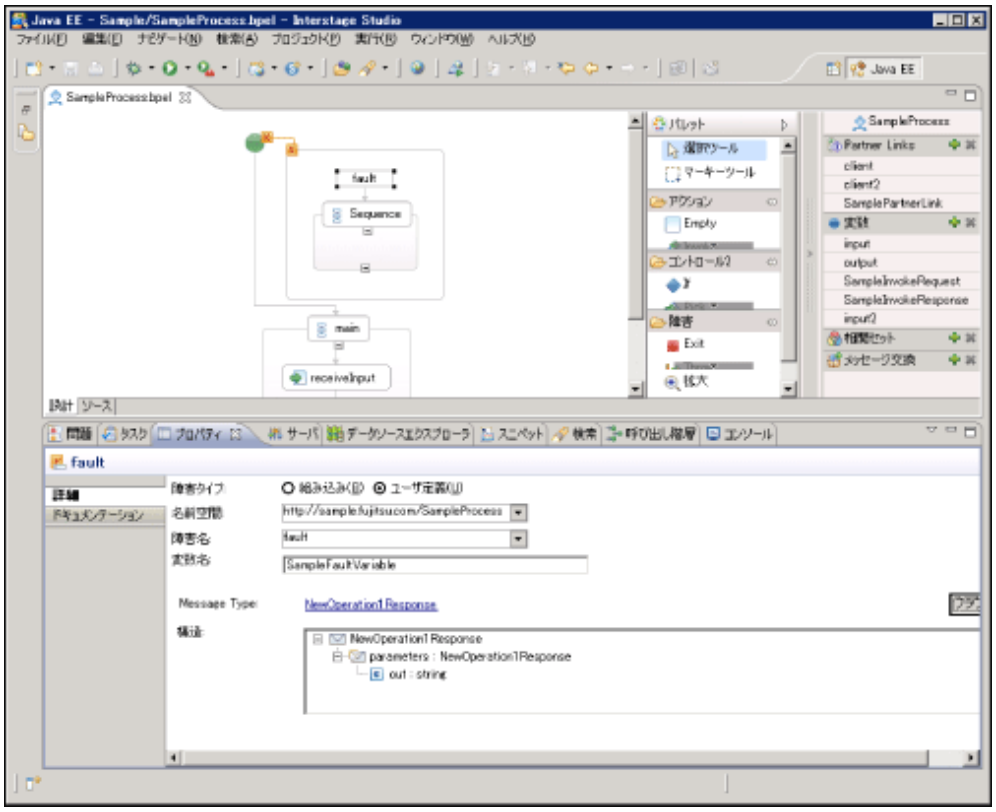
3. プロパティの設定
   BPELエディタの[設計]タブで、追加した障害ハンドラ内のCatchをクリックし、プロパティビューの[詳細]で以下の設定を行います。

表E.32 障害ハンドラの設定内容

設定項目	設定内容
障害タイプ	以下のどちらかを選択します。 ・ [組み込み] BPEL標準の障害をCatchする場合。 ・ [ユーザ定義] ユーザ定義の障害をCatchする場合。
名前空間	[ユーザ定義]を選択した場合にだけ設定できます。 入力フィールド右にある▼をクリックして表示された一覧から、障害名に設定した障害が属する名前空間をダブルクリックで選択します。
障害名	Catchする障害名を設定します。 入力フィールド右にある▼をクリックして表示された一覧から、Catchする障害をダブルクリックで選択します。 障害名を設定すると、BPELエディタの[設計]タブに表示されている[Catch]が設定した障害名に変わります。

設定項目	設定内容
変数名	障害名とともに通知された情報を格納する変数名を設定します。ここで設定した変数名は該当するCatch内でだけ使用できます。
データ型	変数名で設定した変数のデータ型を設定します。 [ブラウズ]ボタンをクリックし、[変数の型を選択]ウインドウでデータ型を指定します。ここではメッセージ型を指定します。 [変数の型を選択]ウインドウについては、以下の“4.[変数の型を選択]の設定”を参照してください。
構造	[データ型]で設定したデータ型の構造を表示します。

図E.38 障害ハンドラの設定画面



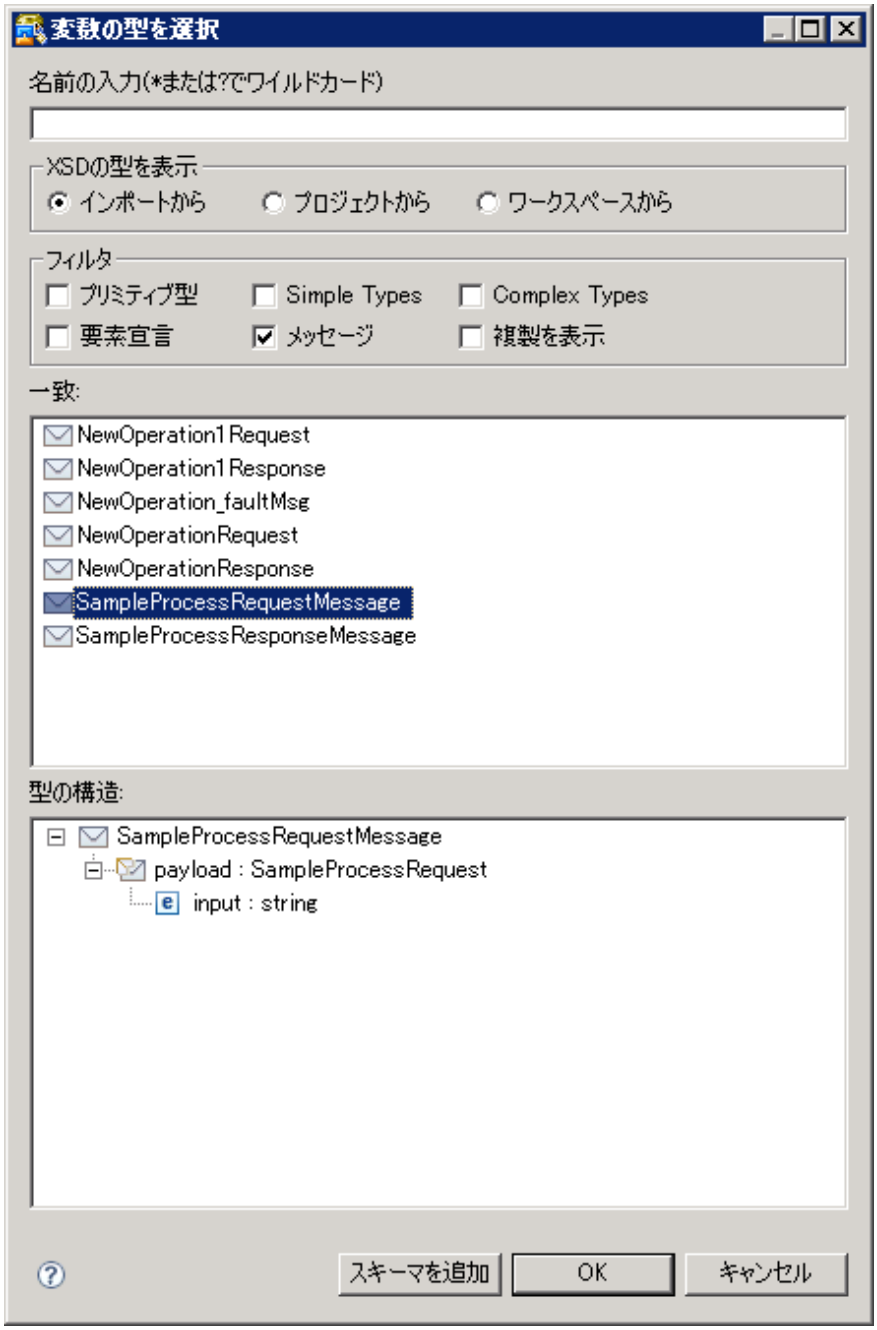
4. [変数の型を選択]の設定
以下の設定を行います。

表E.33 [変数の型を選択]の設定内容

設定項目	設定内容
名前の入力	[一致]に表示されるデータ型を絞り込むためのキーワードを入力します。
XSDの型を表示	[一致]に表示するデータ型の範囲を指定します。以下のどれかを選択します。 ・ [インポートから] BPELプロセスでインポートした定義 ・ [プロジェクトから] プロジェクト内の定義 ・ [ワークスペースから] ワークスペース内の定義

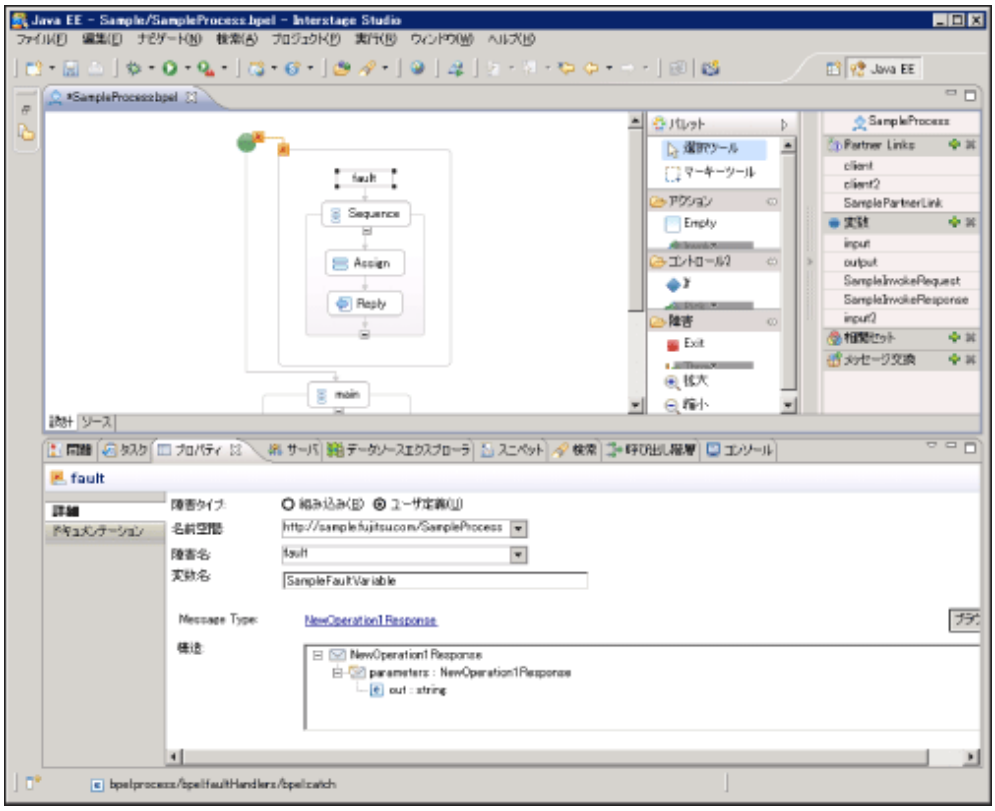
設定項目	設定内容
フィルタ	[一致]に表示するデータ型の種別を指定します。表示したい種別をチェックしてください。必ず必要なもの1つだけチェックしてください。
一致	追加する変数のデータ型を選択してください。 BPEL機能で使用できるデータ型については、以下の“注意事項”を参照してください。
型の構造	[一致]で選択したデータ型の構造を表示します。
[スキーマを追加]ボタン	データ型を追加したい場合にクリックしてください。
[OK]ボタン	[一致]で選択したデータ型の選択を確定します。
[キャンセル]ボタン	データ型の選択を取り消します。

図E.39 [変数の型を選択]画面



5. Catch内にアクティビティを配置
   BPELエディタの[設計]タブで、Catch後に実行したいアクティビティをCatchの枠の中に追加します。

図E.40 Catchの設定画面

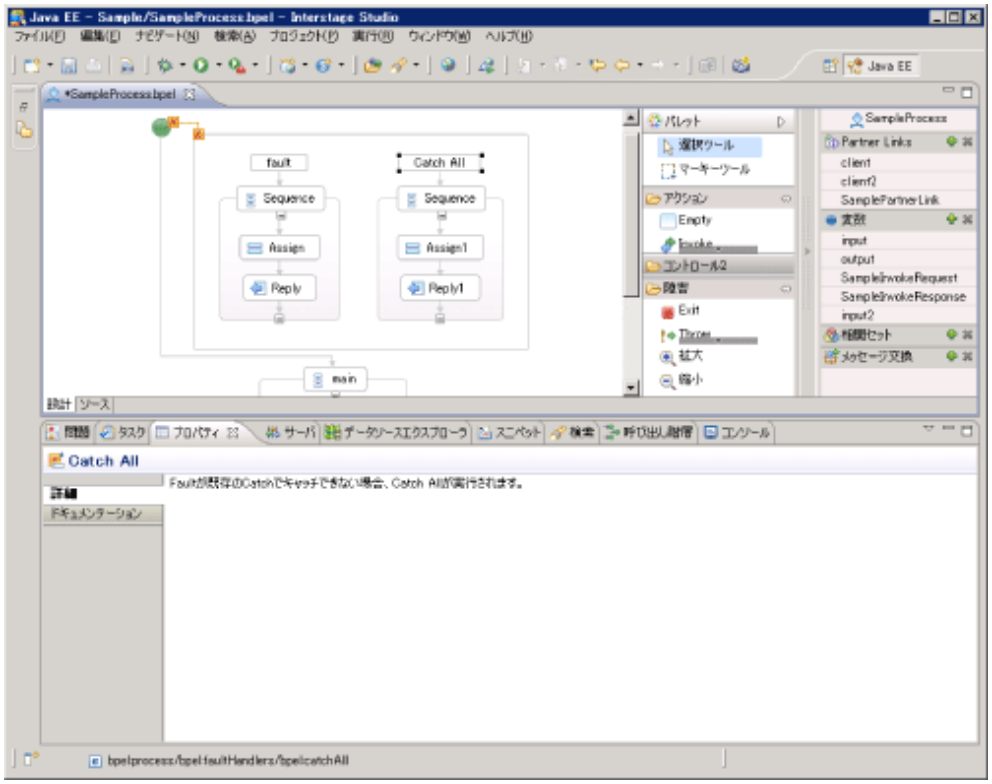


6. Catchの追加
   “3.プロパティの設定”で設定したCatchで受信する障害名以外の障害名のCatch処理を追加する場合は、BPELエディタの[設計]タブで、Catchを追加する障害ハンドラをクリックし、ポップアップメニューの[Catchを追加]をクリックします。追加したCatchのプロパティやアクティビティの設定は“3.プロパティの設定”～“5.Catch内にアクティビティを配置”と同じ手順で設定してください。

7. Catch Allの追加
   “3.プロパティの設定”で設定したCatchで受信する障害名以外の障害名すべてをCatchする場合は、BPELエディタの[設計]タブで、Catchを追加する障害ハンドラをクリックし、ポップアップメニューの[Catch Allを追加]をクリックします。Catch Allは、

プロパティを設定する必要はありません。アクティビティの配置は、“5.Catch内にアクティビティを配置”と同じ手順で設定してください。

図E.41 Catch Allの設定画面



- 削除

  BPELエディタの[設計]タブで、削除する障害ハンドラ、Catch、またはCatchAllをクリックし、キーボードの[Delete]キーを押下します。なお、障害ハンドラ内にCatchが1つだけ存在する場合、Catchは削除しないでください。

- 編集

  プロパティを変更する場合は、BPELエディタの[設計]タブで、変更する障害ハンドラ、またはCatchをクリックし、プロパティビューで変更する設定値を編集します。BPELエディタの[設計]タブに障害ハンドラが表示されていない場合は、BPELプロセスの始点（先頭の緑色の丸印）右上の[×]をクリックすることで障害ハンドラが表示されます。

注意

- 変数名で定義した変数で使用できるデータ型は、Interstage Application Serverに準拠していますので、その他のデータ型は使用しないでください。使用できるデータ型については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発”の“アプリケーションの作成方法”の“Webサービスの通信で利用できるデータ型”を参照してください。
  また、Complex Type、QName、およびAttributeは使用しないでください。
  要素宣言については、WSDLで定義したものだけ使用してください。

- ユーザ定義の障害は、WSDLのSOAPFaultの定義に対応しています。WSDLにSOAPFaultの定義が無い場合は障害ハンドラでSOAPFaultを障害としてcatchすることはできません。

CorrelationSets

相関セットの追加、削除、および編集は、以下のとおり行います。

・ 追加

1. 相関セットの追加
   BPELエディタの[設計]タブの[相関セット]横の[＋]をクリックします。
2. プロパティの設定
   BPELエディタの[設計]タブで、追加された相関セットをクリックし、プロパティビューの[説明]で[名前]に相関セットの名前を設定します。
   次にプロパティビューの[詳細]で以下の設定を行います。

表E.34 相関セットの設定内容

設定項目	設定内容
プロパティ	相関セットのプロパティの追加、編集、削除を行います。 ・ [追加]ボタン プロパティの追加。 [メッセージプロパティの作成]ダイアログボックスが起動されます。詳細は以下の“3.[プロパティを選択]の設定”を参照してください。 ・ [編集]ボタン [編集]ボタンは使用できません。 プロパティを編集する場合は、[削除]ボタンで該当するプロパティいったん削除し、[追加]ボタンで新たな内容のプロパティを追加してください。 ・ [削除]ボタン 選択したプロパティの削除。
プロパティ	追加済のプロパティの一覧を表示します。

3. [プロパティを選択]の設定
   以下の設定を行います。

表E.35 [プロパティを選択]の設定

設定項目	設定内容
プロパティ名	選択したプロパティの名前が表示されます。 新たにプロパティを作成する場合は、[新規]ボタンをクリックします。[メッセージプロパティの作成]ダイアログボックスが起動されます。詳細は以下の“4.[メッセージプロパティの作成]/[メッセージプロパティの編集]の設定”を参照してください。
一致	作成済のプロパティを選択します。
プロパティタイプ	[一致]で選択したプロパティのプロパティタイプを表示します。

図E.42 [プロパティを選択]画面



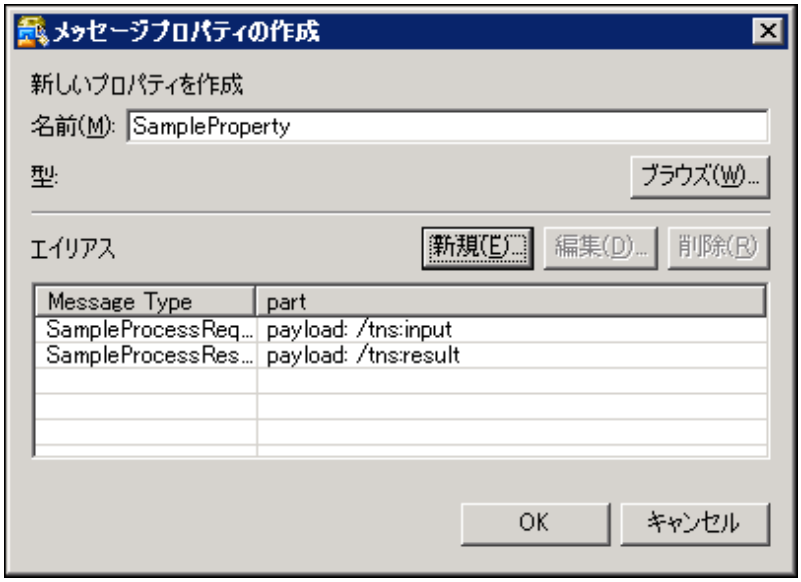
4. [メッセージプロパティの作成]/[メッセージプロパティの編集]の設定
以下の設定を行います。

表E.36 [メッセージプロパティの作成]/[メッセージプロパティの編集]の設定内容

設定項目	設定内容
名前	プロパティの名前を設定します。
型	プロパティの型を設定します。 [ブラウズ]ボタンをクリックして起動される[変数の型を選択]ダイアログボックスで型を選択します。詳細は、以下の“5.[変数の型を選択]の設定”を参照してください。
エイリアス	ビジネス・プロセスのインスタンスとメッセージを関連付けるための識別子を格納するメッセージ内の位置を定義します。相関セットで同じインスタンスで動作させるメッセージすべてについて定義します(Invokeで使用するメッセージ、Receive/Replyで使用するメッセージ)。 ・ [新規]ボタン エイリアスの追加。 [プロパティエイリアスの作成]ダイアログボックスが起動されます。詳細は“6.[プロパティエイリアスの作成]/[プロパティエイリアスを編集]の設定”を参照してください。 ・ [編集]ボタン 選択したエイリアスの編集。

設定項目	設定内容
	[プロパティエイリアスを編集]ダイアログボックスが起動されます。詳細は“6.[プロパティエイリアスの作成]/[プロパティエイリアスを編集]の設定”を参照してください。 ・ [削除]ボタン 選択したAliasの削除。

図E.43 [メッセージプロパティの作成]画面

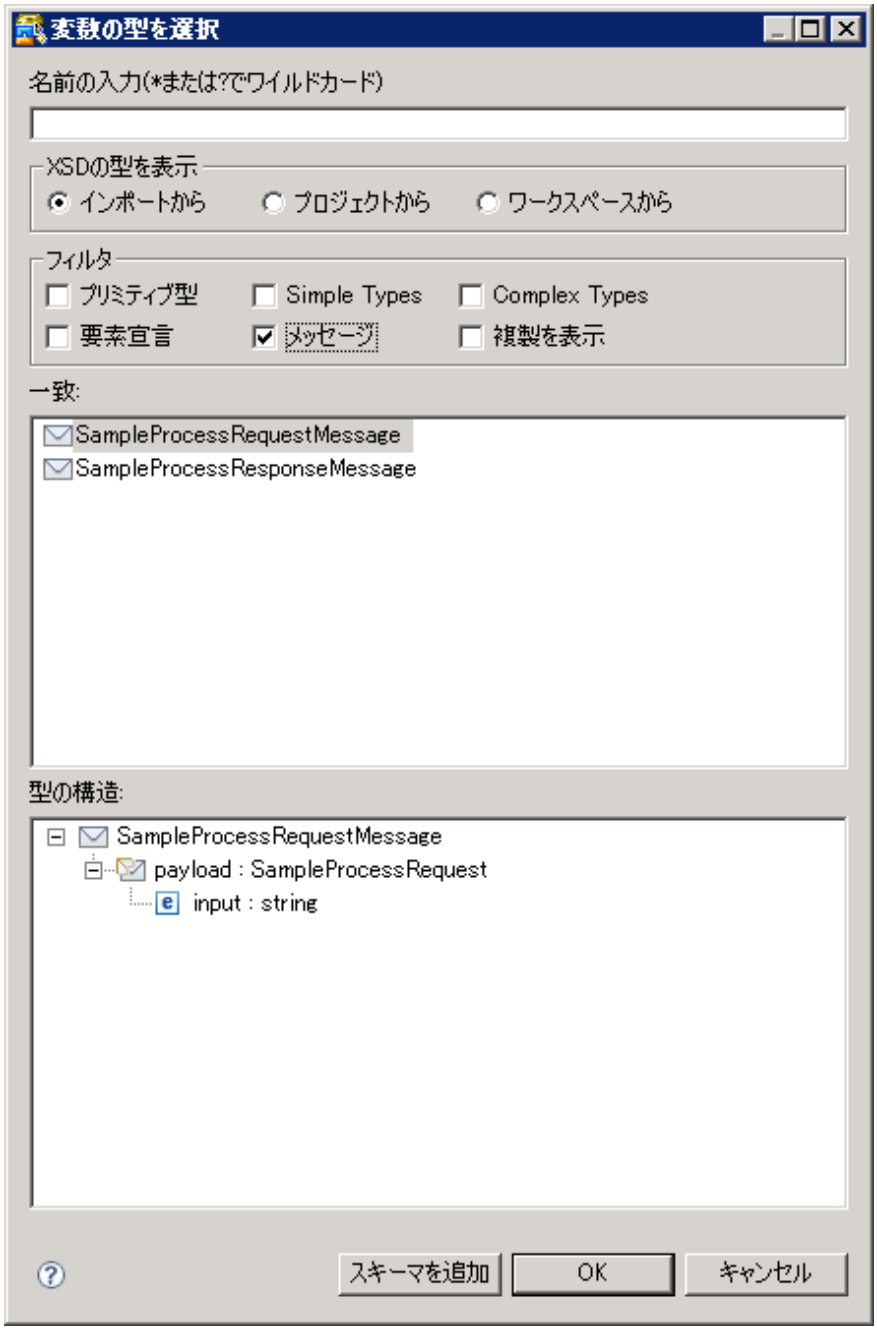


5. [変数の型を選択]の設定
以下の設定を行います。

表E.37 [変数の型を選択]の設定内容

設定項目	設定内容
名前の入力	[一致]に表示される型を絞り込むためのキーワードを入力します。
XSDの型を表示	[一致]に表示する型の範囲を指定します。 必ず[インポートから]を選択してください。
フィルタ	[一致]に表示する型の種別を指定します。表示したい種別をチェックしてください。必ず必要なもの1つだけチェックしてください。
一致	型を選択してください。 BPEL機能で使用できる型については、以下の“注意事項”を参照してください。
型の構造	[一致]で選択した型の構造を表示します。
[スキーマを追加]ボタン	型を追加したい場合にクリックしてください。
[OK]ボタン	[一致]で選択した型の選択を確定します。
[キャンセル]ボタン	型の選択を取り消します。

図E.44 [変数の型を選択]画面



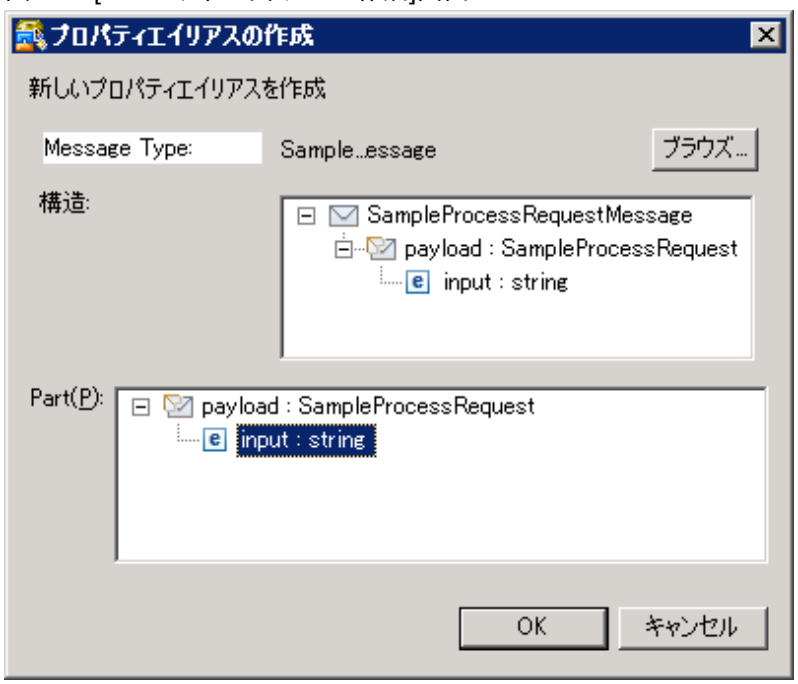
6. [プロパティエイリアスの作成]/[プロパティエイリアスを編集]の設定
以下の設定を行います。

表E.38 [プロパティエイリアスの作成]/[プロパティエイリアスを編集]の設定内容

設定項目	設定内容
データ型	エイリアスの型を設定します。ビジネス・プロセスのインスタンスとメッセージを関連付けるための識別子を格納するメッセージのMessage Typeを設定します。

設定項目	設定内容
	[ブラウズ]ボタンをクリックして起動される[変数の型を選択]ダイアログボックスでデータ型を選択します。詳細は、上記の“4.[変数の型を選択]の設定”を参照してください。
構造	[データ型]で設定したデータ型の構造を表示します。
Part	ビジネス・プロセスのインスタンスとメッセージを関連付けるための識別子を格納するメッセージのPartを選択します。

図E.45 [プロパティエイリアスの作成]画面



7. Invoke/Receive/Replyの設定
ビジネス・プロセスのインスタンスとメッセージを関連付けるための識別子をインスタンス内に記憶し、該当するインスタンスにメッセージをルーティングするための設定を行います。
設定は、Invoke/Receive/Replyのプロパティで行います。各アクティビティをクリックし、プロパティビューの[相関]で以下の設定を行います。

表E.39 Invoke/Receive/Replyの設定内容

設定項目	設定内容
相関セット	相関セットの追加、削除を行います。 ・ [追加]ボタン 相関セットの追加。 追加後、[方向]、[開始]、および[相関セット] を設定してください。 ・ [削除]ボタン 選択した相関セットの削除。
方向	[相関セット]で定義されたメッセージの識別子をインスタンス内に記憶するタイミングを設定します。 ・ [Send] メッセージ送信時に記憶。 Invoke、およびReplyの場合だけ選択可。

設定項目	設定内容
	・ [Receive] メッセージ受信時に記憶。 Invoke、およびReceiveの場合だけ選択可。 ・ [Both] メッセージ送信時、受信時の双方で記憶。 Invokeの場合だけ選択可。
開始	[相関セット]で定義されたメッセージの識別子をインスタンス内に記憶する方法を設定します。 ・ [する] 識別子を記憶します。 ・ [Join] すでに識別子が記憶されていない場合だけ識別子を記憶します。 ・ [しない] 識別子は記憶しません。
相関セット	使用する相関セットを設定します。

・ 削除

BPELエディタの[設計]タブで、[相関セット]下の削除する相関セットをクリックし、[相関セット]横の[×]をクリックします。

・ 編集

プロパティを変更する場合は、BPELエディタの[設計]タブで、[相関セット]下の変更する相関セットをクリックし、プロパティビューで変更する設定値を編集します。

なお、相関セットの名前の変更は行わないでください。名前の変更を行う場合は、該当相関セットをいったん削除し、新たな名前の相関セットを新規に作成してください。

注意

・ プロパティで使用できる型は、Interstage Application Serverに準拠していますので、その他のデータ型は使用しないでください。使用できるデータ型については、Interstage Application Serverの“Java EE運用ガイド”の“Java EEアプリケーションの開発”の“アプリケーションの作成方法”の“Webサービスの通信で利用できるデータ型”を参照してください。
また、Complex Typeは使用しないでください。
要素宣言については、WSDLで定義したものだけ使用してください。

・ 相関セットの名前は、同じBPEL定義内の他の要素と重複しない名前を、プロパティビューの[説明]の[名前]に必ず設定してください。

・ [メッセージプロパティの作成]、または[メッセージプロパティの編集]で大量の要素を持つメッセージを持つプロパティエイリアスを[エイリアス]で選択して[編集]ボタンをクリックすると、[プロパティエイリアスを編集]ダイアログの画面が異常となり、プロパティエイリアスの編集ができなくなる場合があります。このような場合は、[キャンセル]ボタンで[プロパティエイリアスを編集]ダイアログを閉じ、該当するプロパティエイリアスをいったん削除し、新たなプロパティエイリアスを作成することでプロパティエイリアスの内容を変更してください。

・ 相関セットのプロパティを設定しない状態でBPELエディタやInterstage Studioを終了しないでください。BPELエディタやInterstage Studioを終了する場合は、相関セットのプロパティを設定したあとで終了してください。

E.6.7 Webアプリケーション環境定義ファイル（deployment descriptor）の作成

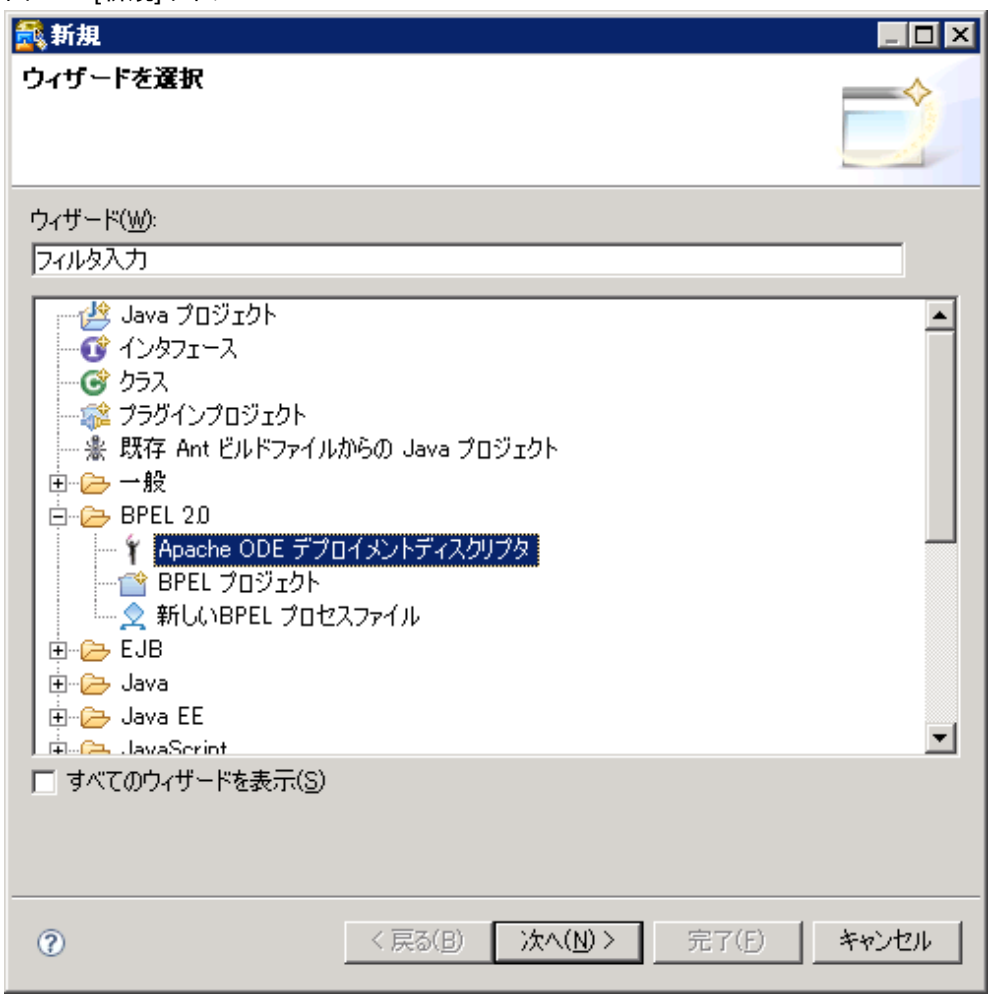
BPEL定義のWebアプリケーション環境定義ファイル（deployment descriptor）を作成します。

なお、Webアプリケーション環境定義ファイル(deployment descriptor)を作成したあとで、BPEL定義、またはWSDL定義を変更した場合は、Webアプリケーション環境定義ファイル(deployment descriptor)をいったん削除し、再作成してください。

1. [Apache ODE デプロイメントディスクリプタ]エディタの起動

   メニューバーから[ファイル]-[新規]-[その他]を選択すると、[新規]ウィザードが表示されます。[Apache ODE デプロイメントディスクリプタ]を選択し、[次へ]をクリックします。

   図E.46 [新規]ウィザード

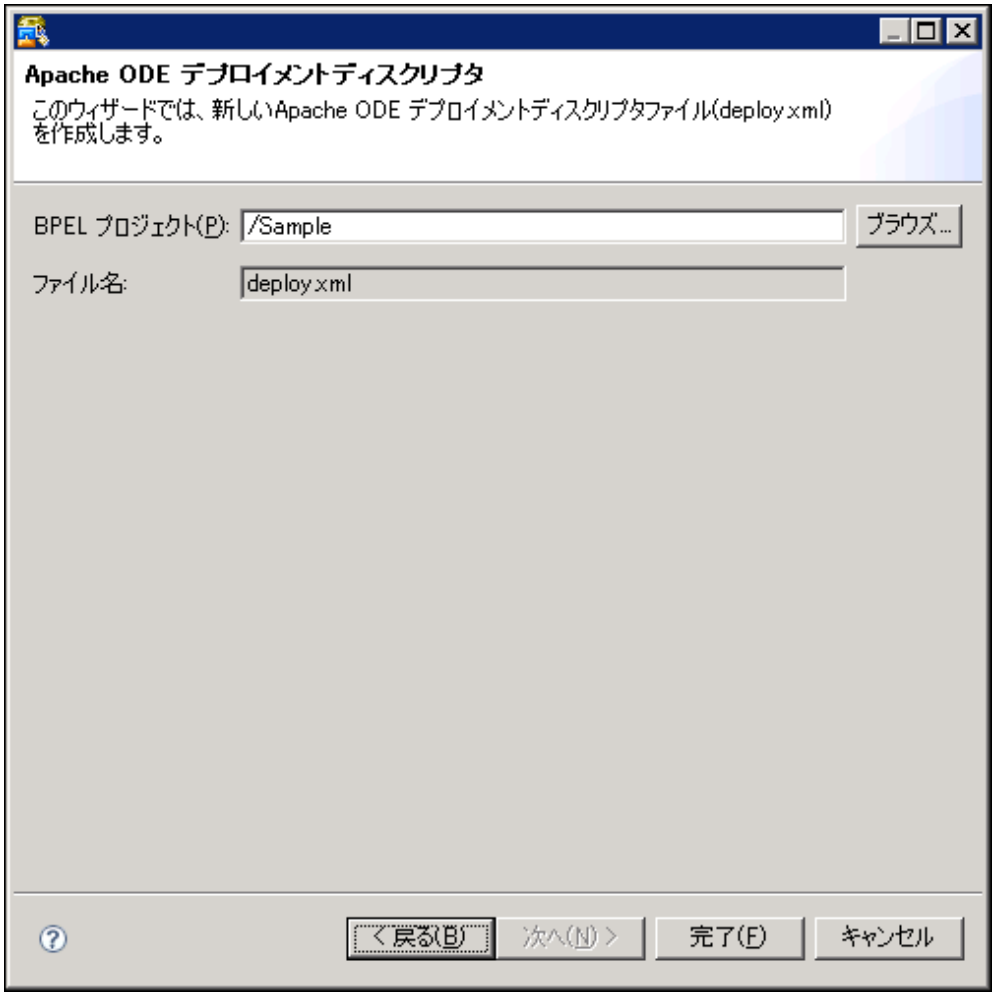


2. 以下の設定項目を確認、入力し、[完了]をクリックしてください。

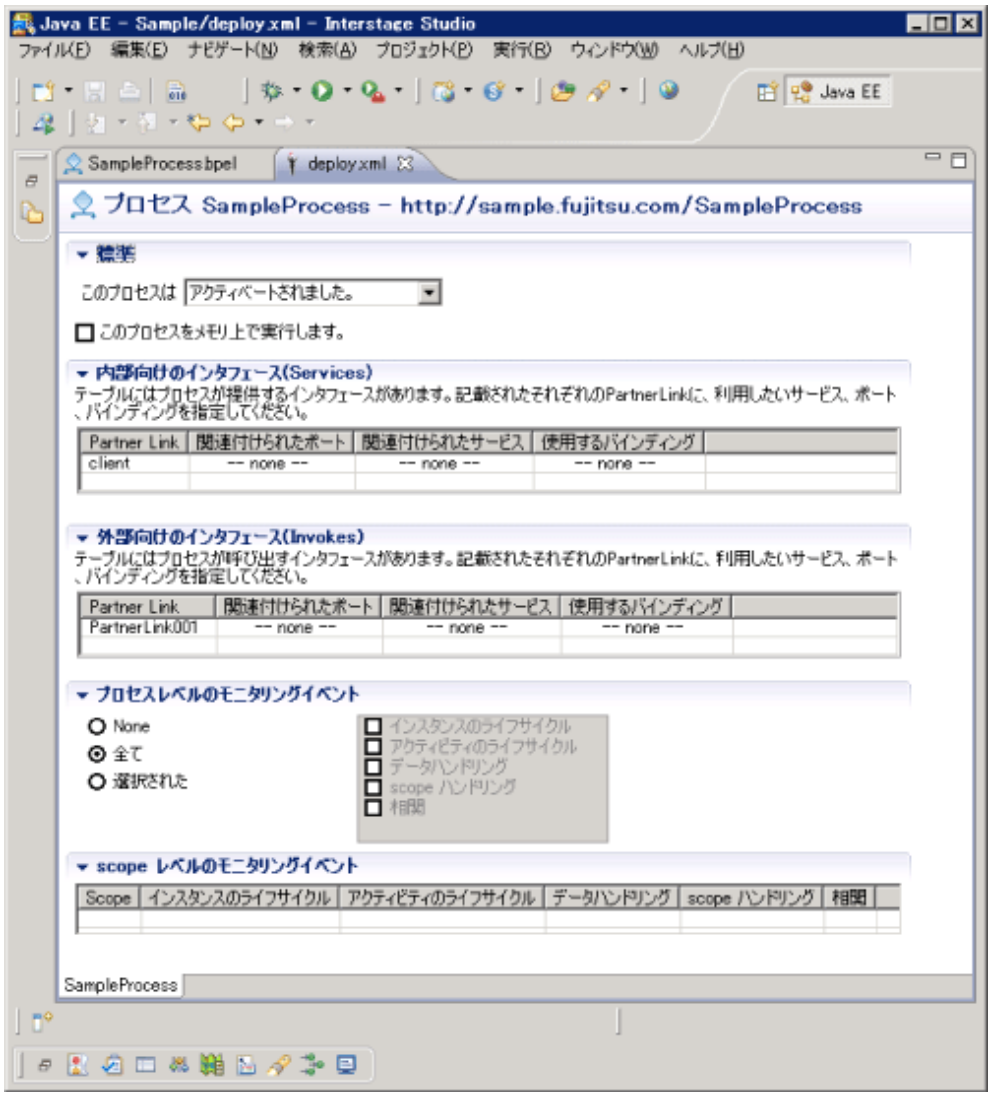
   [Apache ODE デプロイメントディスクリプタ]の設定項目

設定項目	設定内容
BPELプロジェクト	Apache ODE デプロイメントディスクリプタを作成するプロジェクトを選択します。

図E.47 [Apache ODE デプロイメントディスクリプタ]画面



次の画面が表示されます。



[標準]の設定項目

ここで設定することはありません。初期状態のまま操作しないでください。

[内部向けのインタフェース(Services)]の設定項目

ここでは、BPEL定義のReceive/Reply で使用するすべてのPartnerLinkの情報を設定します。

設定項目	設定内容
PartnerLink	Receive/Replyで使用するすべてのPartnerLinkが表示されます。
関連付けられたポート	入力フィールドをクリックすると入力フィールド右に▼が表示されます。▼をクリックして表示された一覧から、PartnerLinkに対応するポートを選択します。選択後PartnerLinkをクリックしてください。
関連付けられたサービス	[関連付けられたポート]を選択するとPartnerLinkにバインディングされたサービス名が表示されます。
使用するバインディング	[関連付けられたポート]を選択すると、ポートに対応するバインディングが表示されます。

[関連付けられたポート]を操作中の画面を以下に示します。

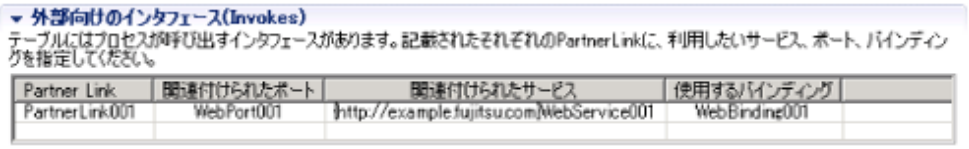


[外部向けのインタフェース(invokes)]の設定項目

invokeで使用するすべてのPartnerLinkの情報を設定します。

設定項目	設定内容
PartnerLink	Invokeで使用するすべてのPartnerLinkが表示されます。
関連付けられたポート	入力フィールドをクリックすると入力フィールド右に▼が表示されます。▼をクリックして表示された一覧から、PartnerLinkに対応するポートを選択します。選択後PartnerLinkをクリックしてください。
関連付けられたサービス	[関連付けられたポート]を選択すると、PartnerLinkにバインディングされたサービス名が表示されます。
使用するバインディング	[関連付けられたポート]を選択すると、ポートに対応するバインディングが表示されます。

[関連付けられたポート]を操作中の画面を以下に示します。



[プロセス・レベルのモニタリングイベント]の設定項目

ここで設定することはありません。初期状態のまま操作しないでください。

[Scopeレベルのモニタリングイベント]の設定項目

ここで設定することはありません。初期状態のまま操作しないでください。

E.6.8 BPEL起動アプリケーション開発者へのWSDL定義の提供

- ビジネス・プロセス実行環境へBPEL定義を配備する前にBPEL起動アプリケーションの開発を行う場合

  BPEL起動アプリケーション開発者に、BPELエディタで開発したBPEL定義のBPEL起動アプリケーション開発用WSDL定義を提供します。BPEL起動アプリケーション開発用WSDL定義は、isibpelwsdlexportコマンドを使用して作成します。BPEL定義のWSDL定義から外部ファイルを参照している場合は、参照している外部ファイルも提供してください。

- ビジネス・プロセス実行環境へBPEL定義を配備した後にBPEL起動アプリケーションの開発を行う場合

  BPEL起動アプリケーション開発者に、BPEL定義の公開用WSDL定義を提供します。公開用WSDL定義は、isibpelwsdlexportコマンドを使用して作成します。

isibpelwsdlexportの詳細は、“ISIリファレンス”の“BPEL機能”の“セットアップコマンド”の“isibpelwsdlexport”を参照してください。

索　引

[A]

Assign.....353

[B]

BPEL起動アプリケーション.....324
BPEL機能のアプリケーション開発.....308
BPEL定義.....309,327,334,344,345
BPELプロセスファイル.....330

[C]

CORBAアダプタ利用時の定義.....55
CorrelationSets.....382

[F]

FaultHandlers.....374
Flow.....361

[I]

If.....365
Invoke.....346

[P]

PartnerLinks.....371

[R]

Receive.....349
Reply.....350

[S]

Sequence.....359

[T]

Throw.....357

[V]

Variables.....369

[W]

Webアプリケーション環境定義ファイル.....387
Webアプリケーションの折り返し定義.....325
Webサービス化.....309
Webサービスの作成.....308
While.....367
WSDL取得.....309,324
WSDL定義.....335,391

[あ]

アプリケーション開発の概要.....1

[か]

グループ定義の詳細.....121

[さ]

シミュレーション.....324
スタブを利用したサービス利用側アプリケーションの開発...268

[た]

タイムアウト.....312,313,314
チューニング.....312
定義の作成と登録.....83
定義の種類.....76
データベースアダプタ利用時の定義.....58
データベース連携の定義.....166

[は]

バイナリデータの使用.....63
配備.....309,310,326
配備解除.....322
フォーマット変換定義.....40
変換定義の開発.....39

[ら]

ログ.....317,318,319,322