



Informatica® PowerCenter
10.4.0

詳細ワークフローガイド

本ソフトウェアおよびマニュアルは、使用および開示の制限を定めた個別の使用許諾契約のもとでのみ提供されています。本マニュアルのいかなる部分も、いかなる手段（電子的複製、写真複製、録音など）によっても、Informatica LLC の事前の承諾なしに複製または転載することは禁じられています。

Informatica、Informatica ロゴ、PowerCenter、および PowerExchange は、米国およびその他の国における Informatica LLC の商標または登録商標です。Informatica の商標の最新リストは、Web (<https://www.informatica.com/trademarks.html>) にあります。その他の企業名および製品名は、それぞれの企業の商標または登録商標です。

米政府の権利プログラム、ソフトウェア、データベース、および関連文書や技術データは、米国政府の顧客に配信され、「商用コンピュータソフトウェア」または「商業技術データ」は、該当する連邦政府の取得規制と代理店固有の補足規定に基づきます。このように、使用、複製、開示、変更、および適応は、適用される政府の契約に規定されている制限およびライセンス条項に従うものとし、政府契約の条項によって適当な範囲において、FAR 52.227-19、商用コンピュータソフトウェアライセンスの追加権利を規定します。

本ソフトウェアまたはドキュメントの一部は、次のサードパーティが有する著作権に従います（ただし、これらに限定されません）。Copyright DataDirect Technologies. All rights reserved. Copyright (C) Sun Microsystems. All rights reserved. Copyright (C) RSA Security Inc. All rights reserved. Copyright (C) Ordinal Technology Corp. All rights reserved. Copyright (C) Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright (C) Meta Integration Technology, Inc. All rights reserved. Copyright (C) Intalio. All rights reserved. Copyright (C) Oracle. All rights reserved. Copyright (C) Adobe Systems Incorporated. All rights reserved. Copyright (C) DataArt, Inc. All rights reserved. Copyright (C) ComponentSource. All rights reserved. Copyright (C) Microsoft Corporation. All rights reserved. Copyright (C) Rogue Wave Software, Inc. All rights reserved. Copyright (C) Teradata Corporation. All rights reserved. Copyright (C) Yahoo! Inc. All rights reserved. Copyright (C) Glyph & Cog, LLC. All rights reserved. Copyright (C) Thinkmap, Inc. All rights reserved. Copyright (C) Clearpace Software Limited. All rights reserved. Copyright (C) Information Builders, Inc. All rights reserved. Copyright (C) OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright (C) International Organization for Standardization 1986. All rights reserved. Copyright (C) ej-technologies GmbH. All rights reserved. Copyright (C) Jaspersoft Corporation. All rights reserved. Copyright (C) International Business Machines Corporation. All rights reserved. Copyright (C) yWorks GmbH. All rights reserved. Copyright (C) Lucent Technologies. All rights reserved. Copyright (C) University of Toronto. All rights reserved. Copyright (C) Daniel Veillard. All rights reserved. Copyright (C) Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright (C) MicroQuill Software Publishing, Inc. All rights reserved. Copyright (C) PassMark Software Pty Ltd. All rights reserved. Copyright (C) LogiXML, Inc. All rights reserved. Copyright (C) 2003-2010 Lorenzi Davide, All rights reserved. Copyright (C) Red Hat, Inc. All rights reserved. Copyright (C) The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright (C) EMC Corporation. All rights reserved. Copyright (C) Flexera Software. All rights reserved. Copyright (C) Jinfonet Software. All rights reserved. Copyright (C) Apple Inc. All rights reserved. Copyright (C) Telerik Inc. All rights reserved. Copyright (C) BEA Systems. All rights reserved. Copyright (C) PDFlib GmbH. All rights reserved. Copyright (C) Orientation in Objects GmbH. All rights reserved. Copyright (C) Tanuki Software, Ltd. All rights reserved. Copyright (C) Ricebridge. All rights reserved. Copyright (C) Sencha, Inc. All rights reserved. Copyright (C) Scalable Systems, Inc. All rights reserved. Copyright (C) jQWidgets. All rights reserved. Copyright (C) Tableau Software, Inc. All rights reserved. Copyright (C) MaxMind, Inc. All rights reserved. Copyright (C) TMatte Software s.r.o. All rights reserved. Copyright (C) MapR Technologies Inc. All rights reserved. Copyright (C) Amazon Corporate LLC. All rights reserved. Copyright (C) Highsoft. All rights reserved. Copyright (C) Python Software Foundation. All rights reserved. Copyright (C) BeOpen.com. All rights reserved. Copyright (C) CNRI. All rights reserved.

本製品には、Apache Software Foundation (<http://www.apache.org/>) によって開発されたソフトウェア、およびさまざまなバージョンの Apache License（まとめて「License」と呼んでいます）の下に許諾された他のソフトウェアが含まれます。これらのライセンスのコピーは、<http://www.apache.org/licenses/> で入手できます。適用法にて要求されないか書面に同意されない限り、ライセンスの下に配布されるソフトウェアは「現状のまま」で配布され、明示的あるいは黙示的かを問わず、いかなる種類の保証や条件も付帯することはありません。ライセンス下での許諾および制限を定める具体的文言については、ライセンスを参照してください。

本製品には、Mozilla (<http://www.mozilla.org/>) によって開発されたソフトウェア、ソフトウェア Copyright (c) The JBoss Group, LLC, all rights reserved、ソフトウェア Copyright (c) 1999-2006 by Bruno Lowagie and Paulo Soares および GNU Lesser General Public License Agreement のさまざまなバージョン (<http://www.gnu.org/licenses/lgpl.html> で参照できる場合がある) に基づいて許諾されたその他のソフトウェアが含まれています。資料は、Informatica が無料で提供しており、一切の保証を伴わない「現状渡し」で提供されるものとし、Informatica LLC は市場性および特定の目的の適合性の黙示の保証などを含めて、一切の明示的及び黙示的保証の責任を負いません。

製品には、ワシントン大学、カリフォルニア大学アーバイン校、およびバンダービルト大学の Douglas C. Schmidt および同氏のリサーチグループが著作権を持つ ACE (TM) および TAO (TM) ソフトウェアが含まれています。Copyright (C) 1993-2006, All rights reserved.

本製品には、OpenSSL Toolkit を使用するために OpenSSL Project が開発したソフトウェア (copyright The OpenSSL Project. All Rights Reserved) が含まれています。また、このソフトウェアの再配布は、<http://www.openssl.org> および <http://www.openssl.org/source/license.html> にある使用条件に従います。

本製品には、Curl ソフトウェア Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>が含まれます。All rights reserved. 本ソフトウェアに関する許諾および制限は、<http://curl.haxx.se/docs/copyright.html> にある使用条件に従います。すべてのコピーに上記の著作権情報とこの許諾情報が記載されている場合、目的に応じて、本ソフトウェアの使用、コピー、変更、ならびに配布が有償または無償で許可されます。

本製品には、MetaStuff, Ltd. のソフトウェアが含まれます。Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved. 本ソフトウェアに関する許諾および制限は、<http://www.dom4j.org/license.html> にある使用条件に従います。

本製品には、Per Bothner のソフトウェアが含まれます。Copyright (C) 1996-2006. All rights reserved. お客様がこのようなソフトウェアを使用するための権利は、ライセンスで規定されています。<http://www.gnu.org/software/kawa/Software-License.html> を参照してください。

本製品には、OSSP UUID ソフトウェアが含まれます。Copyright (C) 2002 Ralf S. Engelschall, Copyright (C) 2002 The OSSP Project Copyright (C) 2002 Cable & Wireless Deutschland. 本ソフトウェアに関する許諾および制限は、<http://www.opensource.org/licenses/mit-license.php> にある使用条件に従います。

本製品には、Boost (<http://www.boost.org/>) によって開発されたソフトウェア、または Boost ソフトウェアライセンスの下で開発されたソフトウェアが含まれます。本ソフトウェアに関する許諾および制限は、http://www.boost.org/LICENSE_1_0.txt にある使用条件に従います。

本製品には、University of Cambridge のが含まれます。Copyright (C) 1997-2007. 本ソフトウェアに関する許諾および制限は、<http://www.pcre.org/license.txt> にある使用条件に従います。

本製品には、The Eclipse Foundation のソフトウェアが含まれます。Copyright (C) 2007. All rights reserved. 本ソフトウェアに関する許諾および制限は、<http://www.eclipse.org/org/documents/epl-v10.php> および <http://www.eclipse.org/org/documents/edl-v10.php> にある使用条件に従います。

本製品には、<http://www.tcl.tk/software/tcltk/license.html>、<http://www.bosrup.com/web/overlib/?License>、<http://www.stlport.org/doc/license.html>、<http://www.asm.ow2.org/license.html>、<http://www.cryptix.org/LICENSE.TXT>、<http://hsqldb.org/web/hsqldbLicense.html>、<http://httpunit.sourceforge.net/doc/license.html>、<http://jung.sourceforge.net/license.txt>、http://www.gzip.org/zlib/zlib_license.html、<http://www.openldap.org/software/release/license.html>、<http://www.libssh2.org>、<http://slf4j.org/license.html>、<http://www.sente.ch/software/OpenSourceLicense.html>、<http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>、<http://antlr.org/license.html>、<http://aopalliance.sourceforge.net/>、<http://www.bouncycastle.org/license.html>、<http://www.jgraph.com/jgraphdownload.html>、<http://www.jcraft.com/jsch/LICENSE.txt>、http://jotm.objectweb.org/bsd_license.html に基づいて許諾されたソフトウェアが含まれています。<http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>、<http://www.slf4j.org/license.html>、<http://nanoxml.sourceforge.net/orig/copyright.html>、<http://www.json.org/license.html>、<http://forge.ow2.org/projects/javaxservice/>、<http://www.postgresql.org/about/licence.html>、<http://www.sqlite.org/copyright.html>、<http://www.tcl.tk/software/tcltk/license.html>、<http://www.jaxen.org/faq.html>、<http://www.jdom.org/docs/faq.html>、<http://www.slf4j.org/license.html>、<http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License>、<http://www.keplerproject.org/md5/license.html>、

<http://www.toedter.com/en/jcalendar/license.html>, <http://www.edankert.com/bounce/index.html>, <http://www.net-snmp.org/about/license.html>, <http://www.openmdx.org/#FAQ>, http://www.php.net/license/3_01.txt, <http://srp.stanford.edu/license.txt>, <http://www.schneier.com/blowfish.html>, <http://www.jmock.org/license.html>, <http://xsom.java.net>, <http://benalman.com/about/license/>, <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>, <http://www.h2database.com/html/license.html#summary>, <http://jsoncpp.sourceforge.net/LICENSE>, <http://jdbc.postgresql.org/license.html>, <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>, <https://github.com/rantav/hector/blob/master/LICENSE>, <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>, <http://jibx.sourceforge.net/jibx-license.html>, <https://github.com/lyokato/libgeohash/blob/master/LICENSE>, <https://github.com/hjiang/jsonxx/blob/master/LICENSE>, <https://code.google.com/p/lz4/>, <https://github.com/jedisct1/libsodium/blob/master/LICENSE>, <http://one-jar.sourceforge.net/index.php?page=documents&file=license>, <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>, <http://www.scala-lang.org/license.html>, <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>, <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>, <https://aws.amazon.com/asl/>, <https://github.com/twbs/bootstrap/blob/master/LICENSE>, および <https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt>.

本製品には、Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>)、Common Development and Distribution License (<http://www.opensource.org/licenses/cddl1.php>)、Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>)、Sun Binary Code License Agreement Supplemental License Terms、BSD License (<http://www.opensource.org/licenses/bsd-license.php>)、BSD License (<http://opensource.org/licenses/BSD-3-Clause>)、MIT License (<http://www.opensource.org/licenses/mit-license.php>)、Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>)、Initial Developer's Public License Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>) に基づいて許諾されたソフトウェアが含まれています。

本製品には、ソフトウェア copyright (C) 2003-2006 Joe Walnes, 2006-2007 XStream Committers が含まれています。All rights reserved. 本ソフトウェアに関する許諾および制限は、<http://j.org/license.html> にある使用条件に従います。本製品には、Indiana University Extreme! Lab によって開発されたソフトウェアが含まれています。詳細については、<http://www.extreme.indiana.edu/>を参照してください。

本製品には、ソフトウェア Copyright (C) 2013 Frank Balluffi and Markus Moeller が含まれています。All rights reserved. 本ソフトウェアに関する許諾および制限は、MIT ライセンスの使用条件に従います。

特許については、<https://www.informatica.com/legal/patents.html> を参照してください。

免責: 本文書は、一切の保証を伴わない「現状渡し」で提供されるものとし、Informatica LLC は他社の権利の非侵害、市場性および特定の目的への適合性の黙示の保証などを含めて、一切の明示的および黙示的保証の責任を負いません。Informatica LLC では、本ソフトウェアまたはドキュメントに誤りのないことを保証していません。本ソフトウェアまたはドキュメントに記載されている情報には、技術的に不正確な記述や誤植が含まれる場合があります。本ソフトウェアまたはドキュメントの情報は、予告なしに変更されることがあります。

NOTICES

この Informatica 製品（以下「ソフトウェア」）には、Progress Software Corporation（以下「DataDirect」）の事業子会社である DataDirect Technologies からの特定のドライバ（以下「DataDirect ドライバ」）が含まれています。DataDirect ドライバには、次の用語および条件が適用されます。

1. DataDirect ドライバは、特定物として現存するままの状態提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。
2. DataDirect または第三者は、予見の有無を問わず発生した ODBC ドライバの使用に関するいかなる直接的、間接的、偶発的、特別、あるいは結果的損害に対して責任を負わないものとします。本制限事項は、すべての訴訟原因に適用されます。訴訟原因には、契約違反、保証違反、過失、厳格責任、詐欺、その他の不法行為を含みますが、これらに限るものではありません。

本マニュアルの情報は、予告なしに変更されることがあります。このドキュメントで問題が見つかった場合は、infa_documentation@informatica.com までご報告ください。

Informatica 製品は、それらが提供される契約の条件に従って保証されます。Informatica は、商品性、特定目的への適合性、非侵害性の保証等を含めて、明示的または黙示的ないかなる種類の保証をせず、本マニュアルの情報を「現状のまま」提供するものとします。

発行日: 2020-02-04

目次

序文	16
Informatica のリソース.....	16
Informatica Network.....	16
Informatica ナレッジベース.....	16
Informatica マニュアル.....	16
Informatica 製品可用性マトリックス.....	17
Informatica Velocity.....	17
Informatica Marketplace.....	17
Informatica グローバルカスタマサポート.....	17
 第 1 章: パイプラインのパーティション化について	18
パイプラインのパーティション化の概要について.....	18
パーティション化属性.....	19
パーティションポイント.....	19
パーティションの数.....	20
パーティションタイプ.....	22
動的パーティション.....	23
動的パーティションの設定.....	23
動的パーティションに関するルールおよびガイドライン.....	24
パーティションタイプでの動的パーティションの使用.....	24
パーティションレベルの属性の設定.....	25
キャッシュのパーティション化.....	25
パーティション化したパイプラインにおけるマッピング変数.....	26
パーティション化のルール.....	27
オブジェクトを編集する場合のパーティションの制限.....	27
PowerExchange に対するパーティション化の制限.....	27
パーティション化の設定.....	28
パイプラインへのパーティションポイントの追加.....	29
パーティションポイントの設定.....	29
[パーティションポイント] ノード.....	31
[パーティションポイント以外] ノード.....	32
 第 2 章: パーティションポイント	33
パーティションポイントの概要.....	33
パーティションポイントの追加および削除.....	34
パーティションポイントの追加と削除に関するルールおよびガイドライン.....	35
リレーショナルソースのパーティション化.....	36
SQL クエリの入力.....	37
フィルタ条件の入力.....	37
ファイルソースのパーティション化.....	38

ファイルソースのパーティション化の規則およびガイドライン.	38
ファイルソースの読み込むための 1 つのスレッドの使用.	39
ファイルソースの読み込むための複数のスレッドの使用.	39
ファイルのパーティション化の設定.	39
リレーショナルターゲットのパーティション化.	43
データベースの互換性.	44
ファイルターゲットのパーティション化.	44
接続設定の設定.	44
ファイルプロパティの設定.	45
カスタムトランスフォーメーションのパーティション化.	47
複数のパーティションに関する作業.	48
パーティションポイントの作成.	48
スレッドに関する作業.	48
ジョイナトランスフォーメーションのパーティション化.	50
ソート済みジョイナトランスフォーメーションのパーティション化.	50
ソート済みフラットファイルの使用.	51
ソート済みリレーショナルデータの使用.	53
ソータトランスフォーメーションの使用.	55
パーティションを使用したソート済みジョイナトランスフォーメーションの最適化.	55
ルックアップトランスフォーメーションのパーティション化.	56
ルックアップトランスフォーメーションのキャッシュのパーティション化.	56
パイプラインルックアップトランスフォーメーションのキャッシュのパーティション化.	57
シーケンスジェネレータトランスフォーメーションのパーティション化.	58
ソータトランスフォーメーションのパーティション化.	58
ソータトランスフォーメーションの作業用ディレクトリ設定.	59
XML ジェネレータトランスフォーメーションのパーティション化.	59
トランスフォーメーションに関する制限.	59
数値関数に関する制限.	60
第 3 章: パーティションタイプ.	61
パーティションタイプの概要.	61
パイプラインでのパーティションタイプの設定.	62
パーティションタイプの設定.	63
データベースパーティション化のパーティションタイプ.	65
データベースソースのパーティション化.	65
ターゲットデータベースのパーティション化.	68
ハッシュ自動キーのパーティションタイプ.	69
ハッシュユーザーキーパーティションタイプ.	69
キー範囲パーティションタイプ.	70
パーティションキーの追加.	71
キー範囲の追加.	72
パススルーパーティションタイプ.	73
ラウンドロビンパーティションタイプ.	75

第4章：プッシュダウンの最適化	76
プッシュダウンの最適化の概要	76
プッシュダウンの最適化のタイプ	77
ソース側でのプッシュダウンの最適化セッションの実行	77
ターゲット側でのプッシュダウンの最適化セッションの実行	77
完全なプッシュダウンの最適化セッションの実行	78
動作可能なデータベースとアイドル状態のデータベース	79
データベースに関する作業	79
ODBC 接続を使用したデータベースへのプッシュダウンの最適化	80
Integration Service およびデータベースの出力の比較	80
IBM DB2 のルールおよびガイドライン	81
Netezza に関するルールおよびガイドライン	81
PostgreSQL のルールおよびガイドライン	81
Teradata のルールおよびガイドライン	82
Vertica のルールとガイドライン	82
Microsoft Azure SQL Data Warehouse のルールおよびガイドライン	83
プッシュダウン互換性	83
データベース接続に互換性がないユーザー	85
アイドル状態のデータベース内のテーブル名の修飾	85
日付に関する作業	86
式に関する作業	87
演算子	87
変数	88
関数	89
プッシュダウンの最適化の関数に関するルールおよびガイドライン	101
エラー処理、ロギング、およびリカバリ	104
エラー処理	104
ロギング	105
リカバリ	105
緩やかに変化する次元に関する作業	105
シーケンスおよびビューに関する作業	106
シーケンス	106
ビュー	107
孤立シーケンスおよびビューのトラブルシューティング	108
\$\$PushdownConfig マッピングパラメータの使用	110
プッシュダウンの最適化のためのセッションの設定	111
プッシュダウンオプション	111
パーティション化	112
ターゲットロードルール	114
プッシュダウングループの表示	115

第 5 章 : プッシュダウンの最適化およびトランスフォーメーション...	117
プッシュダウンの最適化およびトランスフォーメーションの概要.....	117
プッシュダウンに関する一般的な制限.....	118
アグリゲータトランスフォーメーション.....	119
式トランスフォーメーション.....	120
フィルタトランスフォーメーション.....	120
ジョイナトランスフォーメーション.....	121
ルックアップトランスフォーメーション.....	122
接続されていないルックアップトランスフォーメーション.....	124
SQL オーバライドを使用する Lookup トランスフォーメーション.....	124
ルータトランスフォーメーション.....	125
シーケンスジェネレータトランスフォーメーション.....	126
ソータトランスフォーメーション.....	127
ソース修飾子トランスフォーメーション.....	128
SQL オーバライドを使用したソース修飾子トランスフォーメーション.....	129
ターゲット.....	129
共有体トランスフォーメーション.....	131
アップデートストラテジトランスフォーメーション.....	132
 第 6 章 : リアルタイム処理.....	 134
リアルタイム処理の概要.....	134
リアルタイムデータについて.....	135
メッセージとメッセージキュー.....	135
Web サービスメッセージ.....	136
PowerExchange CDC ソースの変更データ.....	136
リアルタイムセッションの設定.....	138
終了条件.....	138
アイドル時間.....	138
メッセージカウント.....	139
Reader 制限時間.....	139
フラッシュ待ち時間.....	139
コミットタイプ.....	140
メッセージリカバリ.....	140
前提条件.....	141
メッセージのリカバリを有効にする手順.....	141
リカバリファイル.....	142
JMS および WebSphere MQ ソースのメッセージリカバリ.....	142
SAP IDoc、TIBCO、および webMethods ソースのメッセージリカバリ.....	143
メッセージリカバリ.....	144
セッションリカバリデータのフラッシュ.....	144
リカバリテーブル.....	145
PM_REC_STATE テーブル.....	145

メッセージの処理.	145
メッセージリカバリ.	146
リカバリキューおよびリカバリトピック.	146
メッセージの処理.	146
メッセージリカバリ.	146
リカバリ無視リスト.	147
リアルタイムセッションの停止.	147
リアルタイムセッションのリスタートおよびリカバリ.	148
リアルタイムセッションのリスタート.	148
リアルタイムセッションのリカバリ.	148
リスタートコマンドおよびリカバリコマンド.	148
リアルタイムセッションに関するルールおよびガイドライン.	149
メッセージリカバリに関するルールおよびガイドライン.	150
リアルタイム処理の例.	150
PowerCenter リアルタイム製品.	152
第 7 章: コミットポイント.	154
コミットポイントの概要.	154
ターゲットベースのコミット.	155
ソースベースのコミット.	155
コミットソースの決定.	156
ソースベースコミットからターゲットベースコミットへの切り替え.	157
ユーザー定義コミット.	159
トランザクションのロールバック.	160
トランザクション制御について.	162
トランスフォーメーション範囲.	163
トランザクション制御単位について.	166
トランザクション制御に関する作業のルールおよびガイドライン.	167
トランザクション別ターゲットファイルの作成.	168
コミットプロパティの設定.	168
第 8 章: 行エラーのロギング.	170
行エラーログの概要.	170
エラーログのコードページ.	171
エラーログテーブルについて.	171
PMERR_DATA.	172
PMERR_MSG.	174
PMERR_SESS.	175
PMERR_TRANS.	176
エラーログファイルについて.	177
エラーログオプションの設定.	180

第 9 章 : ワークフローリカバリ	182
ワークフローリカバリの概要	182
操作の状態	183
ワークフローの操作の状態	183
操作のセッション状態	183
ターゲットリカバリテーブル	184
リカバリオプション	186
ワークフローのサスペンド	188
サスペンド時のメールの設定	188
ワークフローリカバリの設定	189
停止、強制終了、および終了されたワークフローのリカバリ	189
一時停止されたワークフローのリカバリ	189
タスクリカバリの設定	190
タスクのリカバリ戦略	190
終了したタスクの自動的なリカバリ	192
セッションの再開	192
再現可能なデータに関する作業	194
ソースの再現性	194
トランスフォーメーションの再現性	194
リカバリ用のマッピングの設定	195
ワークフローとタスクのリカバリの手順	198
ワークフローのリカバリ	198
セッションのリカバリ	198
セッションからのワークフローのリカバリ	199
セッションリカバリに関するルールおよびガイドライン	199
最後のチェックポイントから再開するようにするためのリカバリの設定	200
リカバリ不可能なワークフローまたはタスク	200
第 10 章 : 停止と強制終了	201
停止と強制終了の概要	201
エラーのタイプ	202
しきい値のエラー	202
致命的なエラー	202
Integration Service によるセッションの失敗の処理	203
ワークフローの停止または強制終了	203
タスクの停止または強制終了	204
停止または強制終了の手順	204
第 11 章 : コンカレントワークフロー	205
コンカレントワークフローの概要	205
一意のワークフローインスタンスの設定	206
ワークフローインスタンスのインスタンス名によるリカバリ	206

同一インスタンス名のコンカレントインスタンスの実行に関するルールおよびガイドライン. . .	206
同一名のコンカレントワークフローの設定.	206
コンカレント Web サービスワークフローの実行.	207
同一名のワークフローインスタンスの設定.	207
同一名のワークフローインスタンスのリカバリ.	207
同一インスタンス名のコンカレントインスタンスの実行に関するルールおよびガイドライン. .	208
パラメータおよび変数の使用.	208
実行インスタンス名または実行 ID へのアクセス.	208
コンカレントワークフローの設定手順.	209
コンカレントワークフローの開始および停止.	209
Workflow Designer からのワークフローインスタンスの開始.	209
単一のコンカレントワークフローの開始.	210
コマンドラインからのコンカレントワークフローの開始.	210
コンカレントワークフローの停止または強制終了.	210
コンカレントワークフローの監視.	211
セッションログおよびワークフローログの表示.	211
一意のワークフローインスタンス用のログファイル.	212
同一名のワークフローインスタンス用のログファイル.	212
コンカレントワークフローに関するルールおよびガイドライン.	212
第 12 章: グリッド処理.	214
グリッド処理の概要.	214
グリッド上でのワークフローの実行.	215
グリッド上でのセッションの実行.	216
パーティショングループに関する作業.	216
リソース要件を使用しないパーティショングループの作成.	217
リソース要件を使用するパーティショングループの形成.	217
パーティショングループの作成に関するルールおよびガイドライン.	218
キャッシュに関する作業.	218
グリッドの接続とリカバリ.	219
グリッド上で実行するためのワークフローまたはセッションの設定.	219
グリッド上で実行するためのワークフローまたはセッションの設定に関するルールおよびガイドライン.	220
第 13 章: ロードバランサ.	222
ロードバランサの概要.	222
ワークフローへのサービスレベルの割り当て.	222
タスクへのリソースの割り当て.	223
第 14 章: ワークフロー変数.	226
ワークフロー変数の概要.	226
定義済みワークフロー変数.	227
式における定義済みワークフロー変数の使用.	230

ワークフロー内の条件の評価.	230
ワークフロー内のタスクステータスの評価.	230
ワークフロー内の前回実行のタスクステータスの評価.	231
ユーザー定義ワークフロー変数.	231
ワークフロー変数の初期値とカレント値.	232
データタイプのデフォルト値.	233
ユーザー定義ワークフロー変数の作成.	233
ワークレット変数の使用.	234
永続ワークレット変数.	234
初期値のオーバーライド.	235
ワークレット変数の使用に関するルールおよびガイドライン.	235
ワークレットでの変数値の割り当て.	235
ワークレット間での変数値の受け渡し.	236
変数割り当ての設定.	236
第 15 章 : セッションのパラメータおよび変数.	238
セッションパラメータに関する作業.	238
セッションログ名の変更.	242
ターゲットファイルおよびディレクトリの変更.	242
ファイルでのソースパラメータの変更.	242
接続パラメータの変更.	242
ランタイム情報の取得.	243
ファイルパラメータとデータベース接続パラメータの作成に関するルールおよびガイドライン.	243
セッションのマッピングパラメータおよび変数.	244
セッションでのパラメータ値および変数値の割り当て.	245
セッション間でのパラメータ値および変数値の受け渡し.	245
変数割り当ての設定.	246
第 16 章 : パラメータファイル.	247
パラメータファイルの概要.	247
パラメータおよび変数のタイプ.	248
パラメータおよび変数を使用する場所.	249
パラメータファイル内の接続属性のオーバーライド.	257
パラメータファイル構造体.	258
パラメータファイルのセクション.	259
コメント.	260
NULL 値.	260
サンプルのパラメータファイル.	260
パラメータファイル名および場所の設定.	261
ワークフローまたはセッションでのパラメータファイルの使用.	261
pmcmd でのパラメータファイルの使用.	263
パラメータファイルの例.	263
パラメータファイルの作成に関するガイドライン.	264

パラメータおよびパラメータファイルのトラブルシューティング.....	265
パラメータおよびパラメータファイルに関するヒント.....	266
第 17 章 : FastExport.....	268
FastExport の使用の概要.....	268
手順 1。FastExport 接続の作成.....	269
コードページのマッピングファイルの確認.....	270
手順 2 reader の変更.....	271
手順 3。ソース接続の変更.....	271
手順 4。制御ファイルのオーバーライド（オプション）.....	272
FastExport の使用に関するルールおよびガイドライン.....	273
第 18 章 : 外部データのロード.....	274
外部データのロードの概要.....	274
はじめに.....	274
外部ローダーの動作.....	275
名前付きパイプへのデータのロード.....	275
フラットファイルへのデータのステージング.....	275
外部ローダーを使用したセッションのパーティション化.....	276
IBM DB2 へのロード.....	276
IBM DB2 EE 外部ローダー.....	277
IBM DB2 EEE 外部ローダー.....	277
IBM DB2 EEE 外部ローダーに関するルールおよびガイドライン.....	278
操作モードの設定.....	278
オーソリティ、特権、権限の設定.....	278
IBM DB2 EE 外部ローダーの属性の設定.....	279
IBM DB2 EEE 外部ローダーの属性の設定.....	281
Oracle へのロード.....	283
Oracle 外部ローダーに関するルールおよびガイドライン.....	283
Oracle へのマルチバイトデータのロード.....	283
Oracle 外部ローダーの属性の設定.....	284
Sybase IQ へのロード.....	284
Sybase IQ 外部ローダーに関するルールおよびガイドライン.....	285
Sybase IQ へのマルチバイトデータのロード.....	285
Sybase IQ 外部ローダーの属性の設定.....	285
Teradata へのロード.....	287
Teradata 外部ローダーに関するルールおよびガイドライン.....	287
制御ファイルのオーバーライド.....	287
制御ファイルでのユーザー変数の作成.....	288
Teradata MultiLoad 外部ローダーの属性の設定.....	289
Teradata TPump 外部ローダーの属性の設定.....	291
Teradata FastLoad 外部ローダーの属性の設定.....	294
セッション内での外部データのロードの設定.....	296

ファイルに書き込むようにセッションを設定.	296
ファイルプロパティの設定.	296
外部ローダ接続の選択.	297
外部データのロードのトラブルシューティング.	298
第 19 章: FTP.	299
FTP の概要.	299
FTP の使用に関するルールおよびガイドライン.	299
SFTP.	300
統合サービスの動作.	300
ソースファイルを対象とした FTP の使用.	301
ターゲットファイルでの FTP の使用.	301
セッションの FTP の設定.	302
セッションの SFTP の設定.	302
FTP 接続の選択.	302
ソースファイルプロパティの設定.	303
ターゲットファイルプロパティの設定.	304
第 20 章: セッションのキャッシュ.	307
セッションのキャッシュの概要.	307
キャッシュメモリ.	308
キャッシュファイル.	309
キャッシュファイルの命名規則.	310
キャッシュファイルディレクトリ.	311
キャッシュサイズの設定.	312
キャッシュサイズの計算.	312
自動キャッシュサイズ.	313
数値のキャッシュサイズの設定.	314
キャッシュサイズを設定する手順.	314
キャッシュのパーティション化.	315
キャッシュのパーティション化用のキャッシュサイズの設定.	315
アグリゲータキャッシュ.	316
差分集計.	316
アグリゲータトランスフォーメーションのキャッシュサイズの設定.	317
アグリゲータキャッシュのトラブルシューティング.	317
ジョイナキャッシュ.	317
1:n のパーティション化.	318
n:n のパーティション化.	319
ジョイナトランスフォーメーションのキャッシュサイズの設定.	319
ジョイナキャッシュのトラブルシューティング.	320
ルックアップキャッシュ.	321
キャッシュの共有.	322
ルックアップトランスフォーメーションのキャッシュサイズの設定.	322

ランクキャッシュ.....	322
ランクトランスフォーメーションのキャッシュサイズの設定.....	323
ソータキャッシュ.....	324
ソータトランスフォーメーションのキャッシュサイズの設定.....	324
XML ターゲットキャッシュ.....	324
XML ターゲットのキャッシュサイズの設定.....	325
キャッシュサイズの最適化.....	325

第 21 章: 差分集計..... 327

差分集計の概要.....	327
差分集計のための Integration Service の処理.....	328
集計ファイルの再初期化.....	328
集計ファイルの移動と削除.....	329
インデックスファイルとデータファイルの検索.....	329
差分集計を使用したパーティション化のガイドライン.....	330
差分集計のための準備.....	330
マッピングの設定.....	330
セッションの設定.....	331

第 22 章: セッションログインタフェース..... 332

セッションログインタフェースの概要.....	332
セッションログインタフェースの実装.....	332
Integration Service とセッションログインタフェース.....	332
セッションログインタフェースの実装に関するルールおよびガイドライン.....	333
セッションログインタフェースの関数.....	333
INFA_InitSessionLog.....	334
INFA_OutputSessionLogMsg.....	335
INFA_OutputSessionLogFatalMsg.....	336
INFA_EndSessionLog.....	336
INFA_AbnormalSessionTermination.....	337
セッションログインタフェースの例.....	337
外部セッションログライブラリの構築.....	338
外部セッションログライブラリの使用.....	338

第 23 章: バッファメモリについて..... 339

バッファメモリの概要について.....	339
自動バッファメモリ設定.....	340
セッション設定オブジェクトを使用したメモリの設定.....	340
バッファメモリの設定.....	340
セッションキャッシュメモリの設定.....	341
セッションキャッシュの制限.....	341
セッションキャッシュの自動メモリ設定.....	342

第 24 章 : 高精度データ	343
高精度データの概要.....	343
Bigint.....	343
Decimal.....	344
索引	345

序文

*PowerCenter(R)詳細ワークフローガイド*は、プッシュダウンの最適化、パイプラインのパーティション化、およびグリッド処理などの詳細ワークフローの概念の詳細を知るために使用します。また、セッションを作成し、セッションログで情報を表示することもできます。トランスフォーメーションロジックを追加し、パイプラインをパーティション化してパフォーマンスを最適化することができます。

Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

Informatica Network

Informatica Network は、Informatica ナレッジベースや Informatica グローバルカスタマサポートなど、多くのリソースへの入口です。Informatica Network を利用するには、<https://network.informatica.com> にアクセスしてください。

Informatica Network メンバーは、次のオプションを利用できます。

- ナレッジベースで製品リソースを検索できます。
- 製品の提供情報を表示できます。
- サポートケースを作成して確認できます。
- 最寄りの Informatica ユーザーグループネットワークを検索して、他のユーザーと共同作業を行えます。

Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、<https://search.informatica.com> にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム (KB_Feedback@informatica.com) です。

Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム (infa_documentation@informatica.com) までご連絡ください。

Informatica 製品可用性マトリックス

製品可用性マトリックス (PAM) には、製品リリースでサポートされるオペレーティングシステム、データベースなどのデータソースおよびターゲットが示されています。Informatica PAM は、<https://network.informatica.com/community/informatica-network/product-availability-matrices> で参照できます。

Informatica Velocity

Informatica Velocity は、Informatica プロフェッショナルサービスが開発したヒントとベストプラクティスのコレクションで、多数のデータ管理プロジェクトから得た実体験に基づいています。Informatica Velocity には、世界中の組織と連携してデータ管理ソリューションを計画、開発、デプロイ、管理する Informatica コンサルタントによる集合知を表しています。

Informatica Velocity リソースには、<http://velocity.informatica.com> からアクセスしてください。Informatica Velocity についての質問、コメント、またはアイデアがある場合は、ips@informatica.com から Informatica プロフェッショナルサービスにお問い合わせください。

Informatica Marketplace

Informatica Marketplace は、お使いの Informatica 製品を拡張したり強化したりするソリューションを検索できるフォーラムです。Marketplace で、Informatica デベロッパーやパートナーからの多数のソリューションを活用すれば、生産性を向上したり、プロジェクトでの実装時間を短縮したりできます。Informatica Marketplace は、<https://marketplace.informatica.com> からアクセスしてください。

Informatica グローバルカスタマサポート

電話または Informatica Network からグローバルサポートセンターに連絡できます。

各地域の Informatica グローバルカスタマサポートの電話番号は、Informatica Web サイト (<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>) を参照してください。

Informatica Network でオンラインサポートリソースを見つけるには、<https://network.informatica.com> にアクセスし、eSupport オプションを選択します。

第 1 章

パイプラインのパーティション化について

この章では、以下の項目について説明します。

- [パイプラインのパーティション化の概要について, 18 ページ](#)
- [パーティション化属性, 19 ページ](#)
- [動的パーティション, 23 ページ](#)
- [キャッシュのパーティション化, 25 ページ](#)
- [パーティション化したパイプラインにおけるマッピング変数, 26 ページ](#)
- [パーティション化のルール, 27 ページ](#)
- [パーティション化の設定, 28 ページ](#)

パイプラインのパーティション化の概要について

Integration Service で実行するセッションをマッピングごとに作成します。各マッピングには、1 つ以上のパイプラインが含まれています。パイプラインは、ソース修飾子と、そのソース修飾子からデータを受け取るすべてのトランスフォーメーションおよびターゲットから構成されます。Integration Service はセッションを実行すると、パイプラインをパーティション化してそれぞれのパーティションデータに対して、抽出、トランスフォーメーション、ロードを並列に実行することで、より高いパフォーマンスを実現することができます。

パーティションとは、1 つの reader スレッド、トランスフォーメーションスレッド、または writer スレッドで実行されるパイプラインステージを指します。任意のパイプラインステージにおけるパーティションの数は、そのステージにおけるスレッドの数と一致します。デフォルトでは、Integration Service はすべてのパイプラインステージにパーティションを 1 つ作成します。

パーティション化オプションを使用すると、1 つのパイプラインステージに対して複数のパーティションを設定することができます。パイプラインに対してマスタースレッドが作成する、reader スレッド、トランスフォーメーションスレッド、および writer スレッドの数を制御するパーティション化情報を設定できます。

Integration Service がソースからデータを読み込む方法、各トランスフォーメーションにデータ行を分散させる方法、およびターゲットにデータを書き込む方法を設定できます。使用するソース接続およびターゲット接続の数を設定できます。

セッションに対してパーティションを設定するには、次の作業を実行します。

- パーティションポイント、パーティション数、パーティションタイプなど、パーティション属性を設定します。

- Integration Service によりランタイム時にパーティション化を設定できます。動的パーティションを有効にする際、Integration Service により、ソースデータベースパーティションやグリッド内のノード数などの要素に基づいて、セッションパーティションの数が計算されます。
- パーティション化のためにセッションを設定した後、メモリ要件とキャッシュディレクトリをトランスフォーメーションごとに設定できます。
- Integration Service により、ターゲットロード順グループ内のパーティションごとにマッピング変数が評価されます。マッピングで変数関数を使用して、変数値を設定できます。
- パイプラインに複数のパーティションを作成した場合、Workflow Manager により、Integration Service がパーティションを使用してセッションでデータの一貫性を保持できることが確認されます。セッションでオブジェクトプロパティを編集した場合、パーティション化に影響を与え、セッションが失敗することがあります。
- セッションプロパティでパーティションポイントを追加または編集します。パーティションポイントを変更する場合は、パーティションタイプを定義し、パーティションを追加または削除することができます。

パーティション化属性

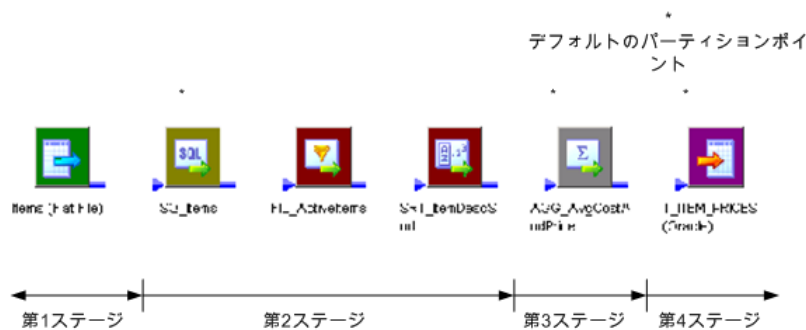
以下の属性を設定して、パイプラインをパーティション化できます。

- **パーティションポイント**。パーティションポイントによりスレッド境界がマークされ、パイプラインがステージに分割されます。Integration Service によりパーティションポイントでデータ行が再分散されます。
- **パーティションの数**。パーティションは、単一のスレッドで実行されるパイプラインステージです。Partitioning を別途購入した場合には、任意のパーティションポイントでパーティションの数を設定できます。パーティションを追加すると、処理スレッドの数が増え、セッションのパフォーマンスを向上できます。
- **パーティションタイプ**。Integration Service では、各パーティションポイントでデフォルトのパーティションタイプが作成されます。パーティション化オプションを使用すると、パーティションタイプを変更できます。パーティションタイプによって、パーティションポイントでパーティションにデータを配分する方法を制御します。

パーティションポイント

デフォルトでは、パーティションポイントは Integration Service によってパイプライン上のさまざまなトランスフォーメーションに設定されます。パーティションポイントはスレッド境界を示し、パイプラインをステージに分割します。ステージとは、2 つのパーティションポイントに挟まれたパイプライン上の区間です。トランスフォーメーションにパーティションポイントを設定すると、そのトランスフォーメーションは新規のパイプラインステージに含まれます。

以下の図に、1 つのパイプラインを持つマッピングに関する、デフォルトのパーティションポイントとパイプラインステージを示します。



パーティションポイントを1つ追加すると、パイプラインステージの数が1つ増加します。同様に、パーティションポイントを1つ削除すると、パイプラインステージの数が1つ減少します。パーティションポイントにより、Integration Service がパーティション全体でデータを再分散できるパイプライン上の点がマークされます。

例えば、パーティションポイントをフィルタトランスフォーメーションに配置して、複数のパーティションを定義する場合、フィルタトランスフォーメーションがデータを処理する前に、Integration Service によりデータ行がパーティション間で再分散されます。パーティションポイントで設定するパーティションタイプによって、Integration Service で各パーティションにデータ行を渡す方法が制御できます。

パーティションの数

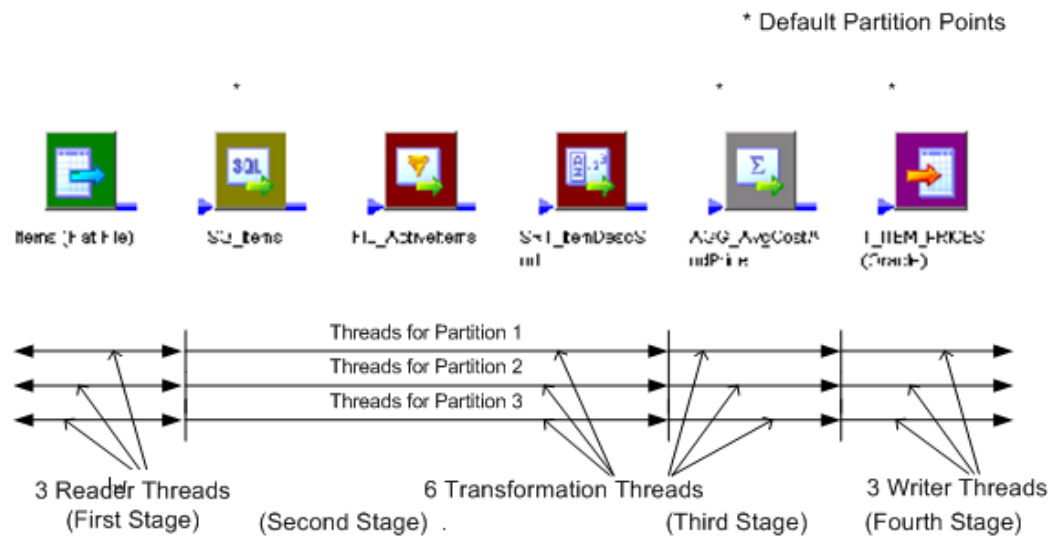
各パイプラインステージで処理するスレッドの数は、パーティションの数に依存します。パーティションとは、1つの読み取りスレッド、トランスフォーメーションスレッド、または書き込みスレッドで実行されるパイプラインステージを指します。任意のパイプラインステージにおけるパーティションの数は、当該ステージにおけるスレッドの数と一致します。

パイプライン上のパーティションポイントには最大 64 のパーティションを定義できます。任意のパーティションポイントでパーティションの数を増減すると、Workflow Manager はパイプライン上のすべてのパーティションポイントでパーティションの数を増減します。パーティションの数は、パイプライン全体を通して一定です。任意のパーティションポイントで3つのパーティションを定義すると、パイプライン上にある他のすべてのパーティションポイントで3つのパーティションが作成されます。特定の状況では、パイプラインのパーティションの数を1つに設定する必要があります。

パーティションまたはパーティションポイントの数が増えることで、スレッドの数が増えます。したがって、パーティションまたはパーティションポイントの数を増やすと、ノードの負荷も大きくなります。ノードに十分な CPU バンド幅がある場合は、セッションで並列にデータ行を処理することで、セッションのパフォーマンスを向上させることができます。ただし、大量のデータを処理するセッションでたくさんのパーティションまたはパーティションポイントを作成すると、システムの負荷が大きくなりすぎる場合もあります。

作成するパーティションの数は、ソースまたはターゲットへの接続の数と等しくなります。パイプラインにリレーショナルソースまたはリレーショナルターゲットが含まれる場合には、ソース修飾子またはターゲットインスタンスのパーティションの数はデータベースへの接続の数と等しくなります。パイプラインにファイルソースが含まれる場合は、1つまたは複数のスレッドでソースを読み込むようにセッションを設定できます。

次の図に、3つのパーティションのマッピングのスレッドを示します。



例えば、マッピングで3つのパーティションを定義する場合、マスタスレッドによりパイプラインステージごとに3つのスレッドが作成され、合計で12のスレッドが作成されます。

Integration Service により、パーティションスレッドが同時に実行されます。複数のパーティションを持つセッションを実行した場合、スレッドは次のように実行されます。

1. 読み取りスレッドが並列に実行されてソースからデータを抽出します。
2. トランスフォーメーションスレッドがそれぞれのトランスフォーメーションステージで並列に実行されてデータを処理します。Integration Service により、各パーティションポイントにおいてパーティションの間でデータが再配分されます。
3. 書き込みスレッドが同時に実行され、データがターゲットに書き込まれます。

複数の入力グループトランスフォーメーションのパーティション化

マスタスレッドにより、ターゲットロード順グループの各パイプラインに対して reader スレッドおよびトランスフォーメーションスレッドが作成されます。ターゲットロード順グループに、複数の入力グループを持つトランスフォーメーションが含まれている場合、そのターゲットロード順グループには複数のパイプラインがあります。

複数のパイプラインを複数の入力グループトランスフォーメーションに接続した場合、Integration Service により、複数の入力グループトランスフォーメーションがパーティションポイントであるかどうかに応じて、ト

ランスフォーメーションスレッドが保持されるか、または新しいランスフォーメーションスレッドが作成されます。

- **パーティションポイントは、複数の入力グループトランスフォーメーションには存在しません。**パーティションポイントが複数の入力グループトランスフォーメーションに存在しない場合、Integration Service により、複数の入力グループトランスフォーメーションとステージ内のすべてのダウストリームトランスフォーメーションについて一度に 1 つのスレッドが処理されます。
- **パーティションポイントは、複数の入力グループトランスフォーメーションに存在します。**パーティションポイントが複数の入力グループトランスフォーメーションに存在する場合、Integration Service により、新しいパイプラインステージが作成され、そのステージが各パーティションにつき 1 つのスレッドで処理されます。Integration Service により、ランスフォーメーションに含まれる出力グループの数に関係なく、パーティションごとに 1 つのランスフォーメーションスレッドが作成されます。

パーティションタイプ

パイプラインにパーティション化情報を設定する場合、パイプライン上の各パーティションポイントにパーティションタイプを定義する必要があります。パーティションタイプにより、PowerCenter Integration Service がパーティションポイント間にデータを再配分する方法が決定します。

PowerCenter Integration Service では、各パーティションポイントでデフォルトのパーティションタイプが作成されます。パーティション化オプションを使用すると、パーティションタイプを変更できます。パーティションタイプは、パーティションポイントでパーティションにデータを配分する方法を制御します。パイプライン上の様々なポイントに、様々なパーティションタイプを作成できます。

Workflow Manager では、以下のパーティションタイプを定義できます。

- **データベースパーティション化。**IBM DB2 または Oracle データベースシステムに対してクエリが実行され、テーブルのパーティション情報が取得されます。データベースの対応するノードからパーティション化データが読み込まれます。複数ノードのテーブル領域上の Oracle または IBM DB2 ソースインスタンスでは、データベースパーティション化を使用できます。DB2 ターゲットでは、データベースパーティション化を使用できます。
- **ハッシュ自動キー。**ハッシュ関数を使用され、データ行がパーティション間でグループ化されます。データはパーティションキーに基づいてグループ化されます。グループ化されたポートまたはソート済みのポートがすべて複合パーティションキーとして使用されます。ランクトランスフォーメーション、ソータランスフォーメーション、未ソートのアグリゲータランスフォーメーションで、ハッシュ自動キーパーティション化を使用する必要がある場合があります。
- **ハッシュユーザーキー。**PowerCenter Integration Service によりハッシュ関数を使用され、データ行がパーティション間でグループ化されます。パーティションキーを生成するポート数を定義します。
- **キー範囲。**キー範囲パーティション化を使用した場合、パーティションキーとして定義したポートまたはポートのセットに基づいて、データ行が配分されます。各ポートについて、値の範囲を定義します。PowerCenter Integration Service は、キーと範囲を使用して適切なパーティションに行を送ります。パイプライン上のソースまたはターゲットがキー範囲によってパーティション化される場合に、キー範囲パーティション化を使用します。
- **パススルー。**パススルーパーティション化では、データはパーティション間で再配分されことなく処理されます。ある 1 つのパーティションに置かれた行はすべて、パススルーパーティションポイントを通過した後も同じパーティションにとどまります。パフォーマンスを向上させるために新たなパイプラインステージを作成したいが、パーティション間のデータの配分を変更したくない場合に、パススルーパーティション化を選択します。
- **ラウンドロビン。**1 つ以上のパーティションにデータのブロックが配分されます。ラウンドロビンパーティション化は、各パーティションでブロックの数やサイズに基づいて行を処理する場合に使用します。

動的パーティション

データ量が増加した場合や CPU を追加した場合は、セッションの実行時間が増加しないようにパーティション化の調整が必要な場合があります。動的パーティションを使用した場合、Integration Service によりランタイム時に作成するパーティション数が決定されるようにパーティション情報を設定できます。

Integration Service は、ソースデータベースパーティションやグリッド内のノード数などを考慮して、実行時にセッションパーティションの数を計算します。

ステージ内の任意のトランスフォーメーションがパーティション化をサポートしていない場合や、パーティション設定が動的パーティション化をサポートしていない場合、Integration Service はパイプライン内のパーティションを計算しません。データは 1 つのパーティションを通過します。

動的パーティション化を使用してセッションパーティションを計算するには、次の作業を実行します。

- **パーティション化の設定。** Integration Service により、選択したパーティション化の方法に基づいて、パーティションの数が増加します。
- **動的パーティションのセッション属性の設定。** ソースおよびターゲットのファイル名とディレクトリを特定するセッション属性を設定できます。セッションでは、セッション属性が使用され、ランタイム時に作成するパーティションごとにパーティションレベルの属性が作成されます。
- **パーティションタイプの設定。** セッションプロパティの [マッピング] タブのパーティションビューを使用して、パーティションポイントとパーティションタイプを編集できます。

注: 手動パーティションを含むセッションに対して動的パーティション化を設定しないでください。動的パーティション化を [無効] 以外の値に設定し、セッションを手動でパーティション化すると、セッションは無効になります。

動的パーティションの設定

セッションプロパティの [設定オブジェクト] タブで動的パーティション化を設定します。次のいずれかの方法で、動的パーティション化を設定します。

- **無効化。** 動的パーティションを使用しません。 [マッピング] タブでパーティション数を定義します。
- **パーティションの数を基準。** パーティションの数属性で定義した数にパーティションを設定します。 `$DynamicPartitionCount` セッションパラメータを使用するか、1 より大きい数を入力します。
- **グリッドのノード数を基準。** セッションを実行しているグリッドのノード数にパーティションを設定します。 グリッド上で実行していないセッションにこのオプションを設定する場合、セッションは 1 つのパーティションで実行され、そのセッションログにメッセージが記録されます。
- **ソースパーティション化を基準。** データベースパーティション情報を使用してパーティション数を決定します。 パーティション数は、ソースのパーティションの最大数です。複合パーティションを使用する Oracle の場合、パーティション数はソースのサブパーティションの最大数です。
- **CPU の数を基準。** パーティション数を、セッションを準備するノード上の CPU の数と同じになるように設定します。 セッションがグリッド上で実行されるように設定されている場合、パーティションの数は、動的パーティションによって、グリッド内のノードの数を乗算した、セッションを準備するノード上の CPU の数と同じになるように設定されます。

関連項目：

- [「データベースパーティション化のパーティションタイプ」 \(ページ 65\)](#)

動的パーティションに関するルールおよびガイドライン

動的パーティション化を使用する場合は、次のルールおよびガイドラインに従ってください。

- 動的パーティション化は各パーティションに同じ接続を使用します。
- XML ソースおよびターゲットで動的パーティション化を使用することはできません。
- デバッガで動的パーティション化を使用することはできません。
- 動的パーティション化を有効にすると、SFTP を使用するセッションは失敗します。
- 動的パーティション化を [無効] 以外の値に設定し、[マッピング] タブでセッションを手動でパーティション化すると、セッションは無効になります。
- \$DynamicPartitionCount 以外のパラメータを使用してパーティションの数を設定した場合、セッションは失敗します。
- 次の動的パーティション化設定を使用すると、1 つのパーティションでセッションが実行されます。
 - アグリゲータ、ジョイナ、ルックアップ、またはランクトランスフォーメーションのデフォルトのキャッシュディレクトリを上書きします。デフォルトが\$PMCacheDir である場合に、Integration Service はトランスフォーメーションキャッシュディレクトリをパーティション化します。
 - ソータトランスフォーメーションのデフォルトのワークディレクトリを上書きします。デフォルトが\$PMTempDir の場合は、Integration Service によってソータトランスフォーメーションのワークディレクトリがパーティション化されます。
 - キー範囲パーティションタイプで無制限の数値または日付キーの範囲を使用する。
 - キー範囲パーティション化のキーとして数値または日付以外のデータ型を使用する。
 - キー範囲リレーショナルターゲットパーティション化を使用する。
 - ユーザー定義 SQL 文またはユーザー定義ソースフィルタを作成する。
 - 動的パーティション化をグリッド内のノードの数の数に設定するが、セッションがグリッド上で実行されない。
 - パススルーリレーショナルソースパーティション化を使用する。
 - アプリケーションソース修飾子で動的パーティション化を使用する。
 - 動的パーティション化で SDK または PowerConnect のソースおよびターゲットを使用する。

パーティションタイプでの動的パーティションの使用

異なるパーティションタイプで動的パーティションを使用するには、以下のルールおよびガイドラインが適用されます。

- **パススルーパーティション化。**パーティションポイントでパーティション数を変更した場合、各パイプラインステージ内のパーティション数を変更されます。リレーショナルソースでパススルーパーティション化を使用する場合、セッションはステージ内の 1 つのパーティションで実行されます。
- **キー範囲パーティション化。**動的パーティションを使用するには、制限された数値または日付キーの範囲を定義する必要があります。キーは、数値データタイプまたは日付データタイプである必要があります。動的パーティション化は、リレーショナルターゲット上でキー範囲パーティション化を使用する場合、パーティションを計算しません。
- **データベースパーティション化。**データベースパーティション化を使用する場合、Integration Service によりソースデータベースパーティションに基づいてセッションパーティションが作成されます。Oracle および IBM DB2 ソースでは、データベースパーティション化を使用します。

- **ハッシュ自動キー、ハッシュユーザーキー、ラウンドロビン。** 動的パーティション化で行を振り分けるには、ユーザー定義ハッシュキー、自動ハッシュキー、およびラウンドロビンパーティションタイプを使用します。行をグループごとにパーティションに振り分ける場合に、ユーザー定義ハッシュキーおよび自動ハッシュキーパーティション化を使用します。1つ以上のパーティションにデータのブロックを振り分ける場合は、ラウンドロビンパーティション化を使用します。

パーティションレベルの属性の設定

動的パーティションを使用する場合、Integration Service によりランタイム時に作成するパーティションごとにパーティションレベルの属性が定義されます。セッションプロパティで定義したセッションレベル属性名に基づいてファイル属性およびディレクトリ属性に名前を付けられます。

たとえば、セッション拒否ファイル名を `accting_detail.bad` として定義したとします。Integration Service は、実行時にパーティションを作成すると、パーティションごとに、`accting_detail1.bad`、`accting_detail2.bad`、`accting_detail3.bad` などの拒否ファイルを作成します。

キャッシュのパーティション化

複数のパーティションを持つセッションを作成した場合、Integration Service ではアグリゲータ、ジョイナ、ルックアップ、ランク、ソータの各トランスフォーメーションについて、キャッシュのパーティション化が使用される場合があります。キャッシュをパーティション化する場合、Integration Service ではパーティションごとに個別のキャッシュが作成され、各パーティションに設定済みのキャッシュサイズが割り当てられます。Integration Service では各キャッシュに異なるデータが格納されます。この場合、各キャッシュには、そのパーティションが必要とする行のみが格納されます。したがって、Integration Service には各パーティションのキャッシュメモリ全体の一部が必要です。

パーティション化を使用するようにセッションを設定した後、セッションプロパティの [マッピング] タブから [トランスフォーメーション] ビューを表示して、メモリ要件とキャッシュディレクトリをトランスフォーメーション別に設定できます。各メモリ要件を設定するには、トランスフォーメーションに必要な合計サイズを計算し、それをパーティション数で割ります。パーティションごとに別々のディレクトリを設定することによって、パフォーマンスを向上させることができます。

以下の表に、適用可能なそれぞれのトランスフォーメーションに対して Integration Service が使用するキャッシュのパーティション化を示します。

トランスフォーメーション	説明
Aggregator トランスフォーメーション	Aggregator トランスフォーメーションを含むセッションに複数のパーティションを作成します。Aggregator トランスフォーメーションにパーティションポイントを設定する必要はありません。
Joiner トランスフォーメーション	Joiner トランスフォーメーションにパーティションポイントを作成します。
Lookup トランスフォーメーション	Lookup トランスフォーメーションに自動ハッシュキーパーティションポイントを作成します。

トランスフォーメーション	説明
Rank トランスフォーメーション	Rank トランスフォーメーションを含むセッションに複数のパーティションを作成できます。Rank トランスフォーメーションにパーティションポイントを設定する必要はありません。
Sorter トランスフォーメーション	Sorter トランスフォーメーションを含むセッションに複数のパーティションを作成できます。Sorter トランスフォーメーションにパーティションポイントを設定する必要はありません。

パーティション化したパイプラインにおけるマッピング変数

マッピング変数を使用するターゲットロード順グループで複数のパーティションを指定する場合、Integration Service により各パーティションのマッピング変数の値が別々に評価されます。Integration Service では、以下のプロセスが使用され、変数値が評価されます。

1. Integration Service により、マッピングで使用される変数関数に応じて、各パーティションの変数のカレント値が別々に更新されます。
2. ターゲットロード順グループのターゲットをすべてロードした後で、Integration Service により、変数の集計タイプに基づいて各パーティションのカレント値が単一の最終値に組み合わせられます。
3. セッションに複数のターゲットロード順グループがある場合は、1 つのターゲットロード順グループのマッピング変数の最終カレント値が、次のターゲットロード順グループのカレント値になります。
4. Integration Service が最後のターゲットロード順グループのロードを終了した場合、変数の最終カレント値がリポジトリに保存されます。

マッピングで以下の変数関数の 1 つを使用して、変数値を設定します。

- SetCountVariable
- SetMaxVariable
- SetMinVariable

以下の表に、Integration Service がパーティション全体の変数値を計算する方法を示します。

変数関数	パーティション全体での変数値の計算
SetCountVariable	Integration Service では、すべてのパーティションから最終カレント値が計算されます。
SetMaxVariable	Integration Service では、各パーティションの最終変数値が比較され、最も大きい値が保存されます。
SetMinVariable	Integration Service では、各パーティションの最終変数値が比較され、最も小さい値が保存されます。

注: パイプラインの各マッピング変数に対して 1 回のみ変数関数を使用します。Integration Service では、マッピングで変数関数が検出された場合、その変数関数が処理されます。Integration Service がマッピングで変数関数が検出される順序は、セッションを実行するたびに異なります。これにより、マッピングで同じ変数関数を複数回使用した場合、結果が矛盾することがあります。

パーティション化のルール

Integration Service により、パーティション化されたデータが処理される際に、データの一貫性が保持される場合、パイプラインに複数のパーティションを作成することができます。セッションを作成した場合、Workflow Manager により各パイプラインがパーティション化について検証されます。

オブジェクトを編集する場合のパーティションの制限

オブジェクトプロパティを編集した場合、セッションでの複数パーティションの作成や、複数のパーティションを持つ既存のセッションの実行のための機能に影響を与える場合があります。

セッションを作成する前に

セッションを作成すると、Workflow Manager はマッピングプロパティをチェックします。マッピングは動的にショートカットへの変更を取得しますが、再利用可能なトランスフォーメーションやマプレットなどの再利用可能なオブジェクトに対してはそれを行いません。したがって、マッピングを保存した後でセッションを作成する前に再利用可能なオブジェクトを Designer で編集した場合には、マッピングを開いて保存し直すことによって、オブジェクトへの変更を Workflow Manager が認識できるようにする必要があります。

複数のパーティションを持つセッションを作成した後に

複数のパーティションを持つセッションを作成した後にマッピングを編集した場合、パーティション化のルールに違反する変更が行われた場合でも、Workflow Manager ではセッションは無効になりません。パーティション化のルールに違反しないようにセッションを編集しない場合、次回セッションを実行したときに Integration Service はセッションに失敗します。

マッピングに次の変更を加えると、セッションの失敗を招く場合があります。

- パーティションポイントに当たるトランスフォーメーションを削除する。
- デフォルトのパーティションポイントに当たるトランスフォーメーションを追加する。
- パーティションポイントに当たるトランスフォーメーションを別のパイプラインに移動する。
- パーティションポイントに当たるトランスフォーメーションを変更し、次のいずれかの状態にする。
 - 既存のパーティションタイプが無効。
 - トランスフォーメーションが複数のパーティションをサポートできない。
 - トランスフォーメーションが有効なパーティションポイントではない。
- 複数のパーティションを持つパイプラインを作成した後に、パーティション化を無効にするか、トランスフォーメーションで単一ノードとグリッド間のパーティション化を変更する。
- 複数のパーティションを持つパイプラインを作成した後で、ジョイナトランスフォーメーションのマスターおよび明細ソースを切り替える。

PowerExchange に対するパーティション化の制限

PowerExchange®および PowerExchange Client for PowerCenter に対しては、複数のパーティションを指定できます。ただし、これには制約が伴います。これらの製品の詳細については、各製品のマニュアルを参照してください。

パーティション化の設定

セッションを作成または編集する際に、マッピング中の各パイプラインのパーティション化を変更できます。マッピングに複数のパイプラインが含まれる場合には、あるパイプラインには複数のパーティションを指定し、別のパイプラインには1つのパーティションを指定することもできます。セッションプロパティの「マッピング」タブからパーティションビューを使用して、パーティション化情報を更新できます。パーティションは、再利用不可能なセッションの場合には Workflow Designer で、再利用可能なセッションの場合には Task Developer で設定できます。

セッションプロパティの「パーティション」ビューでパーティションポイントを追加、削除、または編集します。キー範囲パーティションポイントを追加した場合は、各範囲のキーを定義できます。

以下の表に、「マッピング」タブのパーティションビューの設定オプションを一覧表示します。

パーティションビューのオプション	説明
パーティションポイントの追加	新しいパーティションポイントを追加するときにクリックします。パーティションポイントを追加するとき、該当するトランスフォーメーション名が「パーティションポイント」ノードに表示されます。
パーティションポイントの削除	選択したパーティションポイントを削除するときにクリックします。削除できないパーティションポイントもあります。
パーティションポイントの編集	選択したパーティションを編集するときにクリックします。これにより「パーティションポイントの編集」ダイアログボックスが開きます。
キー範囲	パーティションタイプに応じて、パーティションポイントのキーとキー範囲を表示します。 キー範囲パーティション化の場合には、キー範囲を指定します。 ユーザー定義ハッシュキーパーティション化の場合、このフィールドにはパーティションキーが表示されます。 他のパーティションタイプでは、この領域は表示されません。
キーの編集	キー範囲パーティション化またはユーザー定義ハッシュキーパーティション化に用いるパーティションキーを追加または削除するときにクリックします。自動ハッシュキーパーティション化、ラウンドロビンパーティション化、パススルーパーティション化の場合は、パーティションキーは作成できません。

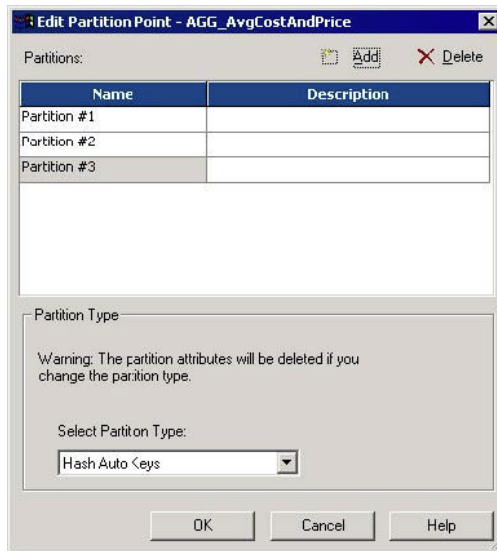
パイプラインへのパーティションポイントの追加

セッションプロパティの [マッピング] タブからパーティションポイントを追加します。

1. [マッピング] タブの [パーティション] ビューで、まだパーティションポイントになっていないトランスフォーメーションを選択し、パーティションポイントの [追加] をクリックします。

ヒント: トランスフォーメーションは [パーティションポイント以外] ノードから選択できます。

次の図に、パイプラインにパーティションポイントを追加するために使用できる [パーティションポイントの編集] ダイアログボックスを示します。



セッションプロパティの [マッピング] タブから [パーティション] ビューを開くと、このトランスフォーメーションが [パーティションポイント] ノードに表示されています。

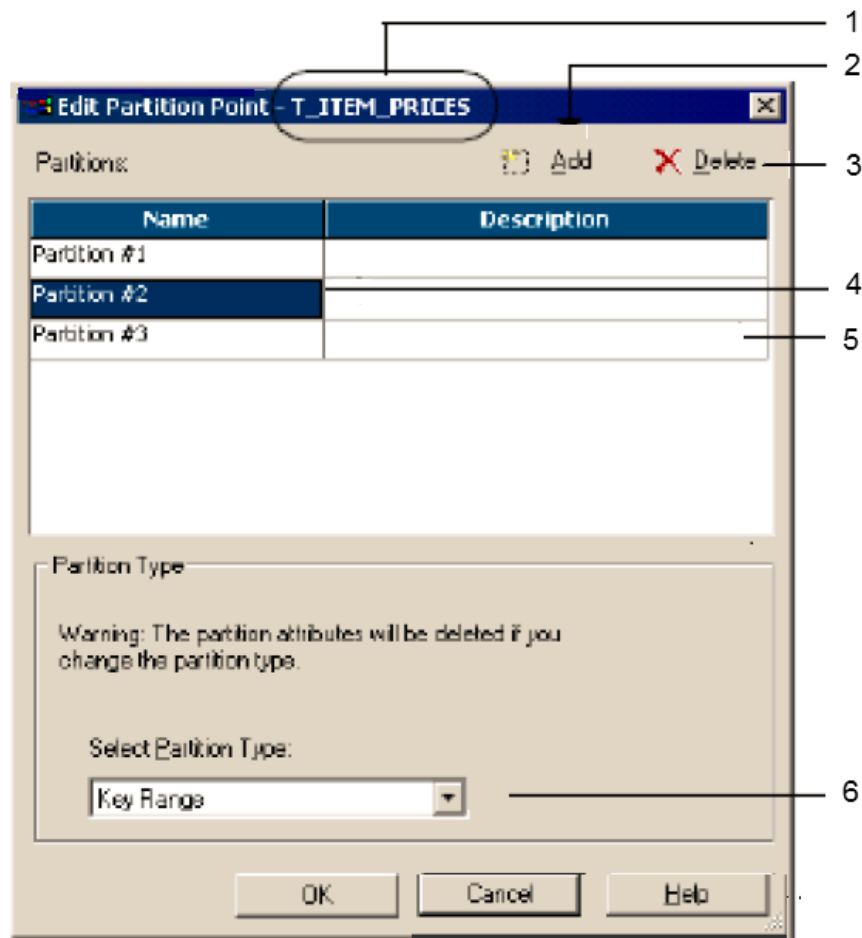
2. パーティションポイントのパーティションタイプを選択するか、デフォルトの値を適用します。
3. [OK] をクリックします。

パーティションポイントの設定

パーティションポイントの編集または追加時に、以下のタスクを実行できます。

- パーティションポイントにおけるパーティションタイプの指定
- パーティションの追加と削除
- 各パーティションの説明の入力

次の図に、パーティションポイントの編集に使用できる設定オプションを示します。



1. 選択されたパーティションポイント
2. パーティションを追加します
3. パーティションを削除します
4. パーティションを選択します
5. パーティションの説明を入力します
6. パーティションタイプを指定します。

以下の表に、パーティションポイントの設定オプションを示します。

パーティションオプション	説明
パーティションタイプの編集	パーティションタイプを変更します。
パーティション名	編集するパーティションを選択します。

パーティションオプション	説明
パーティションの追加	パーティションを追加します。任意のパーティションポイントに、最大 64 のパーティションを追加できます。パーティションの数はパイプラインを通じて同数である必要があります。このため、たとえば 1 つのパーティションポイントで 3 つのパーティションを定義すると、パイプライン上の全パーティションポイントで 3 つのパーティションが定義されます。
パーティションの削除	選択したパーティションを削除します。それぞれのパーティションポイントには、最低 1 つのパーティションが必要です。
説明	現在のパーティションの説明をオプションで入力します。

作成する各パーティションに、説明を入力できます。説明を入力するには、[パーティションポイントの編集] ダイアログボックスでパーティションを選択した後、[説明] フィールドに説明を入力します。

[パーティションポイント] ノード

[パーティションポイント] ノードでは、マッピングがトランスフォーメーションのアイコンで表示されます。パーティションポイントは、ツリー形式で表示されます。属性を設定するパーティションポイントを選択してください。

[パーティションポイント] ノードでは、マッピング内の各パイプラインに対して以下のオプションを設定できます。

- パーティションポイントの追加と削除
- 各パーティションポイントのパーティションタイプの指定
- パーティションの追加と削除
- 各パーティションの説明の入力
- 特定のパーティションタイプへのキーとキー範囲の追加

以下の表に、[パーティションポイント] ノードを示します。

[パーティションポイント] ノード	説明
パーティションポイントの追加	[トランスフォーメーション] リストに新しいパーティションポイントを追加するときにクリックします。
パーティションポイントの削除	現在のパーティションポイントを削除するときにクリックします。削除できないパーティションポイントもあります。
パーティションポイントの編集	現在のパーティションポイントを編集するときにクリックします。
キーの編集	キー範囲パーティション化またはユーザー定義ハッシュキーパーティション化に用いるキーを追加、削除、編集するときにクリックします。このボタンは、自動ハッシュキーパーティション化、ラウンドロビンパーティション化、パススルーパーティション化の場合には使用できません。

パーティションポイントの編集

〔パーティションポイントの編集〕ダイアログボックスでは、パーティションの追加と削除のほか、パーティションタイプを選択することもできます。

以下の表に、〔パーティションポイントの編集〕ダイアログボックスのオプションを示します。

〔パーティションポイントの編集〕のオプション	説明
〔追加〕 ボタン	パーティションを追加するときにクリックします。最大で 64 のパーティションを追加できます。
〔削除〕 ボタン	選択したパーティションを削除するときにクリックします。
名前	パーティション番号です。
説明	カレントパーティションの説明を入力します。
パーティションタイプの編集	リストからパーティションタイプを選択します。

パーティションキーの編集

パーティションポイントでキー範囲パーティション化またはユーザー定義ハッシュキーパーティション化を指定した場合には、1 つまたは複数のポートをパーティションキーとして指定する必要があります。〔キーの編集〕をクリックすると、〔パーティションキーの編集〕ダイアログボックスが表示されます。

パーティションキーとしてポートを 1 つまたは複数指定できます。キーを構成するポートの順序を変更するには、〔選択されたポート〕リストでポートを選択して上下の矢印ボタンをクリックします。

〔パーティションポイント以外〕 ノード

〔パーティションポイント以外〕 ノードには、マッピングの各オブジェクトがアイコン表示されます。非パーティションポイントが、ツリー形式で表示されます。非パーティションポイントを選択し、必要に応じてパーティションを追加できます。

第 2 章

パーティションポイント

この章では、以下の項目について説明します。

- [パーティションポイントの概要, 33 ページ](#)
- [パーティションポイントの追加および削除, 34 ページ](#)
- [リレーショナルソースのパーティション化, 36 ページ](#)
- [ファイルソースのパーティション化, 38 ページ](#)
- [リレーショナルターゲットのパーティション化, 43 ページ](#)
- [ファイルターゲットのパーティション化, 44 ページ](#)
- [カスタムトランスフォーメーションのパーティション化, 47 ページ](#)
- [ジョイナトランスフォーメーションのパーティション化, 50 ページ](#)
- [ルックアップトランスフォーメーションのパーティション化, 56 ページ](#)
- [シーケンスジェネレータトランスフォーメーションのパーティション化, 58 ページ](#)
- [ソータトランスフォーメーションのパーティション化, 58 ページ](#)
- [XML ジェネレータトランスフォーメーションのパーティション化, 59 ページ](#)
- [トランスフォーメーションに関する制限, 59 ページ](#)

パーティションポイントの概要

パーティションポイントは、パイプライン内のスレッド間の境界を表す印となります。Integration Service によりパーティションポイントでデータ行が再分散されます。パーティションポイントを追加して、トランスフォーメーションスレッドの数を増やし、セッションのパフォーマンスを向上させることができます。

ソースデータベースを読み込むようにセッションを設定した場合、Integration Service によってパーティションごとにそのソースデータベースへの別々の接続と SQL クエリが作成されます。SQL クエリをカスタマイズまたはオーバーライドすることができます。

リレーショナルターゲットにデータをロードするようにセッションを設定した場合、Integration Service によってターゲットインスタンスの各パーティションについてターゲットデータベースへ別々の接続が作成されます。ターゲットの拒否ファイル名とディレクトリを設定します。Integration Service によって、ターゲットパーティションごとに拒否ファイルが 1 つ作成されます。

1 つまたは複数のスレッドでソースファイルを読み込むようにセッションを設定できます。ファイルを読み込むすべてのパーティションに対して同じ接続タイプを選択する必要があります。

セッションにファイルターゲットへの書き込みを設定した場合、各パーティションの別々のファイルまたはすべてのパーティションのターゲット出力を含む統合ファイルにターゲット出力を書き込むことができます。それぞれのターゲットパーティションに、接続設定およびファイルプロパティを設定できます。

トランスフォーメーションにパーティションポイントを作成すると、Workflow Manager により、デフォルトのパーティションタイプが設定されます。トランスフォーメーションタイプに応じて、パーティションタイプを変更できます。

パーティションポイントの追加および削除

パーティションポイントがパイプラインのスレッド境界をマークし、パイプラインをステージに分割します。

パーティションポイントを追加すると、トランスフォーメーションスレッドの数が増え、セッションのパフォーマンスを向上できます。Integration Service ではパーティションポイントでデータ行を再分散できるため、これもまたセッションのパフォーマンスを向上することができます。セッションを作成した場合、Workflow Manager によりパイプライン上の各トランスフォーメーションで、それぞれ1つのパーティションポイントが作成されます。

パイプラインにある、以下のトランスフォーメーションまたはターゲットインスタンスにもとづいて、パーティションポイントを保持または削除できます。

ソース修飾子トランスフォーメーション

ソースからデータを抽出してソース修飾子に送る方法を制御します。このパーティションポイントは削除できません。

ノーマライザトランスフォーメーション

ソースからデータを抽出してソース修飾子に送る方法を制御します。このパーティションポイントは削除できません。

ランクトランスフォーメーション

行がトランスフォーメーションに送られる前に、行を適正にグループ化するようにします。パイプラインにパーティションが1つだけ含まれる場合、あるいは1つのグループに属するすべての行が、トランスフォーメーションに送られる前に1つのパーティションに送られる場合に、このパーティションポイントを削除できます。

未ソートアグリゲータトランスフォーメーション

行がトランスフォーメーションに送られる前に、行を適正にグループ化するようにします。パイプラインにパーティションが1つだけ含まれる場合、あるいは1つのグループに属するすべての行が、トランスフォーメーションに送られる前に1つのパーティションに送られる場合に、このパーティションポイントを削除できます。

ターゲットインスタンス

writer がターゲットにデータを渡す方法を制御します。このパーティションポイントは削除できません。

複数入力グループトランスフォーメーション

1つのスレッドで各パーティションを処理するように複数入力グループトランスフォーメーションが設定されている場合や、1つのスレッドで各パーティションを処理するようにダウストリームの複数入力グループカスタムトランスフォーメーションが設定されている場合、Workflow Manager は複数入力グループトランスフォーメーションにパーティションポイントを作成します。

例えば、Workflow Manager は、ソート済みジョイナトランスフォーメーションにパーティションポイントを作成できます。Workflow Manager は、ジョイナトランスフォーメーションが、パーティションごとに1つのスレッドを使用するように設定されたダウストリームのカスタムトランスフォーメーションに接続されると、パーティションポイントを作成します。

これにより、Integration Service は 1 つのスレッドを使用して、パーティションごとに 1 つのスレッドを必要とする Custom トランスフォーメーションの各パーティションを処理するようになります。このパーティションポイントは削除できません。

パーティションポイントの追加と削除に関するルールおよびガイドライン

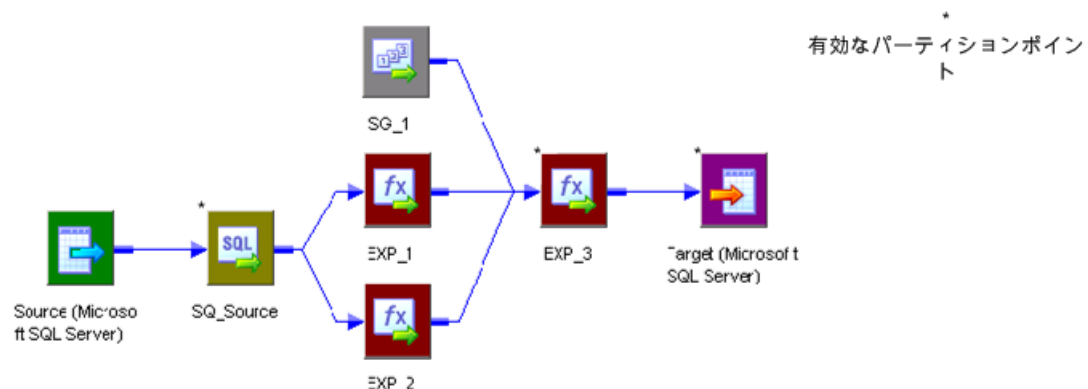
パーティションポイントを追加および削除する際は、以下のルールおよびガイドラインが適用されます。

- ソースインスタンスにはパーティションポイントを作成できません。
- シーケンスジェネレータトランスフォーメーションやコネクタされていないトランスフォーメーションには、パーティションポイントを作成できません。
- 2 つ以上のパイプラインステージから入力を受け入れるパーティションポイントが存在しなければ、他の任意のトランスフォーメーションにパーティションポイントを追加できます。
- ソース修飾子トランスフォーメーション、COBOL ソースのノーマライズトランスフォーメーション、ターゲットインスタンスでは、パーティションポイントを削除できません。
- パーティションごとに 1 つのスレッドを使用するように設定されている複数の入力グループカスタムトランスフォーメーションでは、パーティションポイントを削除できません。
- パーティションごとに 1 つのスレッドを使用するように設定されている複数の入力グループカスタムトランスフォーメーションからのアップストリームである複数の入力グループトランスフォーメーションでは、パーティションポイントを削除できません。
- 次のパーティションタイプには、動的パーティションに関する制限があります。
 - パススルー。動的パーティション化を使用すると、パーティションポイントでパーティションの数を変更した場合に、各パイプラインステージ内のパーティションの数を変更されます。
 - キー範囲。動的パーティション化でキー範囲を使用するには、制限された数値および日付キーの範囲を定義する必要があります。無制限の範囲を使用する場合、セッションは 1 つのパーティションで実行されます。

パイプライン上のその他のトランスフォーメーションにおけるパーティションポイントの削除と追加は、次の規則に従って行います。

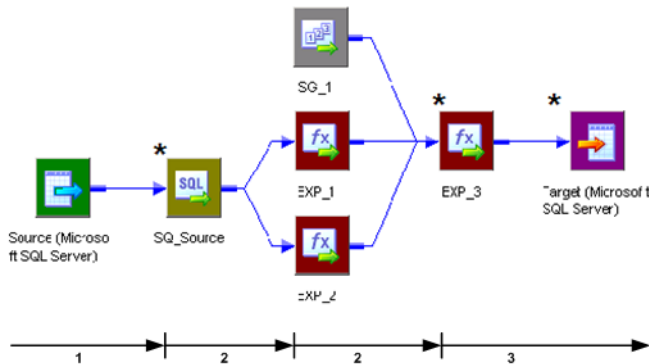
- ソースインスタンスにはパーティションポイントを作成できません。
- シーケンスジェネレータトランスフォーメーションまたは接続されていないトランスフォーメーションには、パーティションポイントを作成できません。
- 複数のパイプラインステージから入力を受け入れるパーティションポイントが存在しなければ、他の任意のトランスフォーメーションにパーティションポイントを追加することができます。

以下の図に、マッピング内の有効パーティションポイントを示します。



このマッピングでは、Workflow Manager により、デフォルトでソース修飾子とターゲットインスタンスにパーティションポイントが作成されます。式トランスフォーメーション EXP_3 に、新しいパーティションポイントを追加できます。

EXP_3 にパーティションポイントを置いてパーティションを 1 つ定義すると、マスタースレッドにより次のスレッドが作成されます。



1. reader スレッド。
2. トランスフォーメーションスレッド。
3. writer スレッド。

この場合、各パーティションポイントが 1 つのパイプラインステージからしかデータを受け取らないため、EXP__3 は有効なパーティションポイントになります。

次のトランスフォーメーションは、有効なパーティションポイントではありません。

トランスフォーメーション	理由
ソース	ソースインスタンス。
SG_1	シーケンスジェネレータトランスフォーメーション。
EXP_1 および EXP_2	EXP_1 または EXP_2 にパーティションポイントを置く場合、EXP_1 または EXP_2 にソース修飾子からのデータを処理する新しいパイプラインステージを追加します。この場合、EXP_3 は 2 つのパイプラインステージからデータを受け取ることになるため、この操作は許可されません。

リレーショナルソースのパーティション化

リレーショナルまたはアプリケーションソースをパーティション化するセッションを実行した場合、Integration Service によって各パーティションについてソースデータベースへ別々の接続が作成されます。その後、各パーティションに SQL クエリが作成されます。[マッピング] タブの [トランスフォーメーション] ビューにフィルタ条件を入力することで、各ソースパーティションのクエリをカスタマイズできます。また、[マッピング] タブの [トランスフォーメーション] ビューを使用して各ソースパーティションの SQL クエリを上書きすることもできます。

注: データベーステーブルを読み込むカスタム SQL クエリを作成し、データベースパーティション化を設定した場合、Integration Service はパススルーパーティション化に戻り、セッションログにメッセージが出力されます。

SQL クエリの入力

SQL クエリーの SELECT 文をカスタマイズしたい場合は、SQL オーバーライドを入力します。[マッピング] タブの [トランスフォーメーション] ビューに入力した SQL 文により、ソース修飾子トランスフォーメーションを設定する際に Designer で設定したカスタマイズ SQL クエリーが上書きされます。

SQL クエリにより、ソースパーティションに対して入力したキー範囲およびフィルタ条件がすべてオーバーライドされます。したがって、ユーザーがキー範囲およびソースフィルタも入力した場合、Integration Service によって SQL クエリオーバーライドが使用され、ソースデータが抽出されます。

NULL 値を含むキーを作成した場合、別のパーティションを作成して NULL 値を抽出する SQL クエリーまたはフィルタを入力することによって、NULL 値を抽出することができます。

各パーティションに SQL クエリーを入力するには、まず [Sql Query] フィールドの右端のボタンをクリックします。[SQL エディタ] ダイアログボックスにクエリーを入力し、[OK] をクリックします。

ソース修飾子トランスフォーメーションを設定したときに Designer で SQL クエリーを入力した場合、各パーティションの [Sql Query] フィールドにそのクエリーが表示されます。このクエリーを上書きするには、[Sql クエリ] フィールドの右端のボタンをクリックし、[SQL エディタ] ダイアログボックスでクエリーを変更したあと、[OK] をクリックします。

フィルタ条件の入力

リレーショナルソース修飾子にキー範囲パーティション化を指定する場合、追加のフィルタ条件を入力することができます。この操作を行った場合、セッションプロパティで入力したフィルタ条件を含む WHERE 句が Integration Service によって生成されます。

[マッピング] タブの [トランスフォーメーション] ビューに入力したフィルタ条件により、ソース修飾子トランスフォーメーションを設定する際に Designer で設定したフィルタ条件が上書きされます。

キー範囲パーティション化を使用する場合、フィルタ条件はキー範囲と関連しながら機能します。たとえば、顧客 ID に基づいてデータを選択したいが、USA 以外の顧客の情報は抽出したくないとします。次のキー範囲を定義します。

CUSTOMER_ID	Start Range	End Range
Partition #1		135000
Partition #2	135000	

USA 以外の顧客の ID が特定のパーティションの範囲内にあると分かっている場合は、そのパーティションにフィルタを入力してそれらを排除することができます。したがって、2 番目のパーティションに次のフィルタ条件を入力します。

```
CUSTOMERS.COUNTRY = 'USA'
```

このセッション実行すると、次のような 2 つのパーティション用クエリーがセッションログに記録されます。

```
READER_1.1.1> RR_4010 SQ instance [SQ_CUSTOMERS] SQL Query [SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY, CUSTOMERS.LAST_NAME FROM CUSTOMERS WHERE CUSTOMER.CUSTOMER ID < 135000]
```

[...]

```
READER_1.1.2> RR_4010 SQ instance [SQ_CUSTOMERS] SQL Query [SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY, CUSTOMERS.LAST_NAME FROM CUSTOMERS WHERE CUSTOMERS.COUNTRY = 'USA' AND 135000 <= CUSTOMERS.CUSTOMER_ID]
```

フィルタ条件を入力するには、まず [ソースフィルタ] フィールドで [参照] ボタンをクリックします。[SQL エディタ] ダイアログボックスにフィルタ条件を入力し、[OK] をクリックします。

ソースフィルタトランスフォーメーションを設定したときに Designer でフィルタ条件を入力している場合は、各パーティションの [ソースフィルタ] フィールドにそのクエリーが表示されます。このフィルタを上書きす

るには、[ソースフィルタ] フィールドの右端のボタンをクリックし、[SQL エディタ] ダイアログボックスでフィルタ条件を変更したあと、[OK] をクリックします。

ファイルソースのパーティション化

セッションによってファイルソースが使用される場合は、1 つまたは複数のスレッドでソースを読み込むようにセッションを設定できます。1 つのスレッドで読み込むようにセッションを設定した場合、Integration Service によってファイルソースへの接続が 1 つ作成されます。複数のスレッドで読み込むようにセッションを設定した場合は、ファイルソースへの複数の同時接続が作成されます。

次のタイプのパーティション化されたファイルソースを使用します。

- **フラットファイル。**フラットファイル、XML ファイル、または COBOL ソースファイルを読み込むようにセッションを設定できます。
- **コマンド。**オペレーティングシステムコマンドを使用してソースデータ行を生成するか、ファイルリストを生成するようにセッションを設定できます。

ファイルソースに接続する際には、全パーティションに同じ接続タイプを選択する必要があります。すべての接続オブジェクトが同じタイプであれば、異なる接続オブジェクトを選択できます。

そのフラットファイルソースに対してシングルスレッドまたはマルチスレッド読み込みを指定するには、パーティション 2-*n* のソースファイル名プロパティを設定します。1 つのスレッドによる読み込み（シングルスレッド読み込み）を設定するには、パーティション 2-*n* に空データを渡します。複数のスレッドによる読み込み（マルチスレッド読み込み）を設定するには、パーティション 2-*n* のソースファイル名を空欄のままにしておきます。

ファイルソースのパーティション化の規則およびガイドライン

複数のパーティションを持つファイルソースセッションを設定する場合は、以下の規則およびガイドラインに従ってください。

- ソース修飾子でパススルーパーティション化を使用します。
- フラットファイルまたは COBOL ソースで、シングルスレッドまたはマルチスレッド読み込みを使用します。
- XML ソースでは、シングルスレッド読み込みを使用します。
- ソースファイルがディスク以外のファイル（FTP ファイルや WebSphere MQ ソースなど）の場合は、マルチスレッド読み込みは使用できません。
- マルチスレッド読み込みを使用してよいのは、シフト依存コードのページを使用していて、次の条件に該当する場合です。
 - 固定長ファイルである。
 - ラインシーケンシャルファイルではない。
 - ソース定義中のユーザー定義シフト状態を有効にしていない。
- 3 つのフラットファイルから同時にデータを読み込むには、ソース修飾子に 3 つのパーティションを指定する必要があります。デフォルトのパーティションタイプであるパススルーパーティション化を適用します。
- セッションにマルチスレッド読み込みを設定している場合で、Integration Service でファイルソースへの複数のスレッドが作成できない場合は、セッションログにメッセージが書き込まれ、1 つのスレッドでソースが読み込まれます。

- Integration Service によって複数のスレッドが使用され、1 つのソースファイルが読み込まれる場合は、ファイルの行が順次読み込まれない場合があります。ソート順が重要な場合は、1 つのスレッドでファイルを読み込むようにセッションを設定してください。たとえば、マッピング中にソート済みジョイントランスフォーマーメーションがあり、ファイルソースがソートの基点である場合は、ソート順が重要な場合があります。
- 負荷を平均化するために、直接ファイルと間接ファイルを組み合わせることもできます。
- マルチスレッド読み込みのセッションパフォーマンスは、大きなソースファイルの場合に最適になります。入力データの量が少ない場合は、負荷が分散されない可能性があります。
- コマンドがソースデータを生成する場合で、セッションがグリッド上で実行されるように設定されているか、最後のチェックポイントからの再開のリカバリ戦略を使用するように設定されている場合は、ファイルソースにこのコマンドを使用することはできません。

ファイルソースの読み込むための 1 つのスレッドの使用

Integration Service によって 1 つのスレッドが使用され、ファイルソースが読み込まれる場合、そのソースへの接続が 1 つ作成されます。Integration Service によって、ファイル中またはファイルリスト中の行が順次読み込まれます。セッション中の直接または間接ファイルソースに対してシングルスレッド読み込みを設定できます。

- **直接ファイルの読み込み。**1 つ以上の直接ファイルから読み込むように Integration Service を設定することができます。複数の直接ファイルがあるセッションを設定する場合は、Integration Service によってファイルごとに 1 つの同時接続が作成されます。1 つのファイルに対して複数の接続は作成しません。
- **間接ファイルの読み込み。**Integration Service によって間接ファイルが読み込まれる場合は、ファイルリストが読み込まれ、次にリスト中のファイルが順次読み込まれます。セッションに複数のファイルリストがある場合は、Integration Service によってファイルリストが同時に読み込まれ、次にリスト内のファイルが順次読み込まれます。

ファイルソースの読み込むための複数のスレッドの使用

Integration Service によって複数のスレッドでソースファイルが読み込まれる場合、そのソースに対して複数の同時接続が作成されます。Integration Service では、ファイル中の行を順次読み込む場合も読み込まない場合もあります。

セッション内の直接または間接ファイルに対してマルチスレッド読み込みを設定することができます。

- **直接ファイルの読み込み。**Integration Service によって 1 つの直接ファイルが読み込まれる場合に、複数の reader スレッドが作成され、同時に読み込まれます。1 つ以上の直接ファイルから読み込むように Integration Service を設定することができます。例えば、セッションによって 2 つのファイルから読み込みが行われ、ユーザーが 5 つのパーティションを作成している場合は、Integration Service によって 1 つのファイルを 2 つのパーティションに、もう 1 つのファイルを 3 つのパーティションに分けることができます。
- **間接ファイルの読み込み。**Integration Service によって 1 つの間接ファイルが読み込まれる場合に、複数のスレッドが作成され、同時に読み込まれます。リスト中のファイルを同時に読み込むためにも、複数のスレッドが作成されます。Integration Service によって、複数のスレッドが使用され単一のファイルが読み込まれる場合があります。

ファイルのパーティション化の設定

パーティションポイントを作成してパーティション情報を設定した後、[マッピング] の [トランスフォーマーメーション] ビューでソース接続に関する設定とファイルプロパティの設定ができます。ソースノードの下にある設定対象のソースインスタンス名をクリックします。ファイルソースのソースインスタンス名をクリックすると、セッションプロパティに接続およびファイルのプロパティが表示されます。

ソースファイル名やディレクトリを、ソースパーティションごとに設定できます。Workflow Manager は、パーティションごとにファイル名と格納場所を生成します。

以下の表に、マッピングにおけるファイルソースのファイルプロパティ設定を示します。

属性	説明
入力タイプ	<p>ソース入力のタイプ。次のソース入力のタイプを選択できます。</p> <ul style="list-style-type: none"> - ファイル。フラットファイルの場合は、COBOL または XML ソース。 - コマンド。コマンドによって生成されたソースデータまたはファイルリスト。 <p>コマンドを使用して XML ソースデータを生成することはできません。</p>
同時読み込みのパーティション化	<p>複数のパーティションがソースファイルから入力行を読み込む順序。以下のオプションを選択することができます。</p> <ul style="list-style-type: none"> - スループットを最適化します。Integration Service は入力行の順序を保持しません。 - 相対入力行の順序を保持します。Integration Service は、各パーティションによって読み込まれる行について、入力行の順序を保持します。 - 絶対入力行の順序を保持します。Integration Service は、すべてのパーティションによって読み込まれるすべての行について、入力行の順序を保持します。
Source File Directory	<p>フラットファイルソースのディレクトリ名。デフォルトで、Integration Service は、サービスのプロセス変数ディレクトリ \$PMSourceFileDir でファイルソースを検索します。</p> <p>[Source Filename] フィールドにディレクトリとファイル名の両方を指定する場合は、このフィールドをクリアします。Integration Service はセッションの実行時に、このフィールドと [ソースファイル名] フィールドを連結します。</p> <p>また、セッションパラメータ \$InputFileName を使ってファイルの場所を指定することもできます。</p>
ソースファイル名	<p>フラットファイルソースのファイル名およびパス。必要に応じて、このファイル名にセッションパラメータ \$InputFileName を指定することもできます。</p> <p>Integration Service はセッションの実行時に、このフィールドと [ソースファイルのディレクトリ] フィールドを連結します。たとえば [Source File Directory] フィールドに 「C:\data\」 と入力されている場合は、[Source Filename] フィールドに 「filename.dat」 と入力します。Integration Service はセッションを開始するときに 「C:\data\filename.dat」 を検索します。</p> <p>デフォルトでは、Workflow Manager はソース定義に設定されているファイル名を入力します。</p>
ソースファイルタイプ	<p>次のソースファイルのタイプを選択できます。</p> <ul style="list-style-type: none"> - Direct。ソースデータを含むソースファイル。 - Indirect。ファイルのリストを含むソースファイル。[Indirect] を選択した場合、セッションを開始すると Integration Service はファイルリストを検索し、リスト内の各ファイルを読み込みます。
コマンドのタイプ	<p>コマンドが生成するソースデータのタイプ。次のコマンドのタイプを選択できます。</p> <ul style="list-style-type: none"> - ソースデータ入力行を生成するコマンドのデータを生成するコマンド。 - ファイルリストを生成するコマンドのファイルリストを生成するコマンド。
コマンド	<p>ソースファイルデータを生成するために使用するコマンド。</p>
文字列の NULL の切り捨て	<p>最初の NULL キャラクタおよび最初の NULL キャラクタの後のすべての文字を文字列値から削除します。</p> <p>文字列に NULL キャラクタを含む区切りフラットファイルに対して、このオプションを有効にします。このオプションを選択しない場合、PowerCenter Integration Service は文字列に NULL 文字を含む行に対して行エラーを生成します。</p> <p>デフォルトでは無効になっています。</p>

1つのスレッドを使用するセッションの設定

1つのスレッドでファイルを読み込むようにセッションを設定するには、パーティション 2- n に空データを渡します。空データを渡すには、データのないファイル（empty.txt など）を作成し、それをソースファイルディレクトリに置きます。次に、empty.txt をソースファイル名として使用します。

注: コマンドを使用してソースデータを生成するパーティション化されたソースに対してシングルスレッド読み込みを設定することはできません。

次の表に、Integration Service で 1つのスレッドを作成して ProductsA.txt を読み込む場合のソースファイル名と値を示します。ファイル内の行は順番に読み込まれます。ファイルを読み込んだ後、トランスフォーマーセッションパイプラインの 3つのパーティションにデータが渡されます。

ソースファイル名	値
Partition #1	ProductsA.txt
Partition #2	empty.txt
Partition #3	empty.txt

次の表に、Integration Service で 2つのスレッドを作成する場合のソースファイル名と値を示します。ProductsA.txt を読み込むスレッドと ProductsB.txt を読み込むスレッドがそれぞれ作成されます。2つのファイルが同時に読み込まれ、各ファイル内の行は順番に読み込まれます。

ソースファイル名	値
Partition #1	ProductsA.txt
Partition #2	empty.txt
Partition #3	ProductsB.txt

FTP を使用してソースファイルにアクセスする場合は、直接ファイルごとに異なる接続を選択できます。

複数のスレッドを使用するセッションの設定

複数のスレッドでファイルを読み込むようにセッションを設定するには、パーティション 2- n のソースファイル名を空欄のままにしておきます。Integration Service によって、パーティション 2- n が使用され、前回のパーティションファイルまたはファイルリストの一部が読み込まれます。Integration Service によってそのパーティションのディレクトリフィールドは無視されます。

セッションを複数のスレッドでコマンドから読み込むように設定するには、各パーティションごとにコマンドを入力するか、パーティション 2- n のコマンドプロパティを空白のままにします。パーティションごとにコマンドを入力した場合、Integration Service によって、各コマンドによって生成されたデータを読み込むスレッドが作成されます。それ以外の場合は、Integration Service によって、パーティション 2- n が使用され、最初のパーティションのコマンドによって生成されたデータの一部が読み込まれます。

次の表に、Integration Service で 3 つのスレッドを作成して ProductsA.txt を同時に読み込む場合の属性と値を示します。

属性	値
Partition #1	ProductsA.txt
Partition #2	<空欄>
Partition #3	<空欄>

次の表に、Integration Service で 3 つのスレッドを作成して ProductsA.txt と ProductsB.txt を同時に読み込む場合の属性と値を示します。ProductsA.txt を 2 つのスレッドで読み込み、ProductsB.txt を 1 つのスレッドで読み込みます。

属性	値
Partition #1	ProductsA.txt
Partition #2	<空欄>
Partition #3	ProductsB.txt

次の表に、Integration Service で 3 つのスレッドを作成して、コマンドからパイプで渡されるデータを同時に読み込む場合の属性と値を示します。

属性	値
Partition #1	CommandA
Partition #2	<空欄>
Partition #3	<空欄>

次の表に、Integration Service で 3 つのスレッドを作成して、CommandA と CommandB からパイプで渡されるデータを同時に読み込む場合の属性と値を示します。CommandA からパイプで渡されるデータを 2 つのスレッドで読み込み、CommandB からパイプで渡されるデータを 1 つのスレッドで読み込みます。

属性	値
Partition #1	CommandA
Partition #2	<空欄>
Partition #3	CommandB

同時読み込みのパーティション化の設定

デフォルトでは、複数のパーティションが1つのファイルソースから読み込む場合、Integration Service は行の順序を保持しません。複数のパーティションが1つのファイルソースから読み込む場合に行の順序を保持するには、同時読み取りのパーティション化を設定します。以下のオプションを設定することができます。

- **スループットを最適化します。** 複数のパーティションによって単一のファイルソースから読み込まれる場合、Integration Service では行の順序は保持されません。複数のパーティションによってファイルソースから読み込まれる順序が重要でない場合にこのオプションを使用します。
 - **相対入力行の順序を維持します。** 各パーティションによって読み込まれる入力行のソート順を維持します。各パーティションによって読み込まれる入力行のソート順を維持する場合にこのオプションを使用します。
- 以下の表に、2つのパーティションごとに10行を含むファイルソースのソート順の例を示します。

パーティション	読み込まれる行
Partition #1	1,3,5,8,9
Partition #2	2,4,6,7,10

- **絶対入力行の順序を維持します。** すべてのパーティションによって読み込まれるすべての入力行のソート順を維持します。セッションが実行されるたびに入力行のソート順を保持する場合にこのオプションを使用します。パッシブトランスフォーメーションを含むパススルーマッピングでは、ターゲットに書き込まれる行の順序は、入力行と同じ順序になります。

以下の表に、2つのパーティションごとに10行を含むファイルソースのソート順の例を示します。

パーティション	読み込まれる行
Partition #1	1,2,3,4,5
Partition #2	6,7,8,9,10

注: デフォルトでは、Integration Service は、最後のチェックポイントからの再開のリカバリ戦略が設定されたセッションで、[絶対入力行の順序の保持] オプションを使用します。

リレーショナルターゲットのパーティション化

リレーショナルターゲットにデータをロードするようにパイプラインを設定した場合、Integration Service によってターゲットインスタンスの各パーティションについてターゲットデータベースへ別々の接続が作成されます。各パーティションのデータが、同時にターゲットデータベースにロードされます。

セッションプロパティの [マッピング] タブでパイプラインにおけるターゲットのパーティション属性を設定します。リレーショナルターゲットの場合、拒否ファイル名とディレクトリを設定します。Integration Service によって、ターゲットパーティションごとに拒否ファイルが1つ作成されます。

以下の表に、パイプライン中のリレーショナルターゲットのパーティション化属性を示します。

属性	説明
拒否ファイルディレクトリ	ターゲット拒否ファイルの場所。デフォルトは、\$PMBadFileDir です。
拒否ファイル名	拒否ファイルの名前。デフォルトは「ターゲット名パーティション番号 <i>bad</i> 」です。また、セッションパラメータファイルで定義したように、セッションパラメータ「\$BadFileName」を使用することもできます。

データベースの互換性

ターゲットインスタンスに複数のパーティションがあるセッションを設定した場合、Integration Service によってターゲットへの接続が各パーティションに 1 つずつ作成されます。データベースにロードするセッションや、テーブルへの複数の同時接続をサポートしない ODBC ターゲットにロードするセッションに複数のターゲットパーティションを設定すると、セッションは失敗します。

Informix データベースにデータをロードするセッションに複数のターゲットパーティションを作成する場合、行レベルのロックを指定したターゲットテーブルを作成する必要があります。複数のパーティションを作成したセッションから、ページレベルのロックが設定された Informix ターゲットにデータを挿入すると、セッションは失敗し、次のメッセージが返されます。

WRT_8206 Error: The target table has been created with page level locking. The session can only run with multi partitions when the target table is created with row level locking.

Sybase IQ では、テーブルへの複数の同時接続は利用できません。Sybase IQ にロードするセッションに複数のターゲットパーティションを作成した場合、Integration Service によって 1 つのパーティションにすべてのデータがロードされます。

ファイルターゲットのパーティション化

セッションにファイルターゲットへの書き込みを設定した場合、各パーティションの別々のファイルまたはすべてのパーティションのターゲット出力を含む統合ファイルにターゲット出力を書き込むことができます。セッションを実行する際、Integration Service によって個別の出力ファイルまたは統合ファイルに同時に書き込まれます。1 つのパーティションまたはすべてのターゲットパーティションのデータをオペレーティングシステムコマンドに送信することもできます。

それぞれのターゲットパーティションに、接続設定およびロパティを設定できます。これらの設定は、[マッピング] タブの [トランスフォーメーション] ビューで設定します。パーティション化された FTP ファイルターゲットを使用するようにセッションを設定することもできます。

接続設定の設定

すべてのターゲットパーティションの接続タイプを設定するには、[マッピング] タブの [トランスフォーメーション] ビューの [接続] 設定を使用します。各パーティションに別々の接続オブジェクトを指定できますが、すべて同じタイプである必要があります。

次の接続タイプのいずれかをターゲットファイルで使用します。

- **なし**。パーティション化されたターゲットファイルをローカルマシンに書き込みます。
- **FTP**。パーティション化されたターゲットファイルを別のマシンに転送します。Integration Service が接続できるすべてのマシンにファイルを転送することができます。

- **ローダー。**複数の出力ファイルからロードできる外部ローダーを使用します。パイプラインによりデータがリレーショナルターゲットにロードされ、[マッピング] タブの [Writers] 設定でファイル writer が選択されている場合に、このオプションが表示されます。複数の出力ファイルからロードできないローダーを選択した場合、Integration Service ではセッションが失敗します。
- **メッセージキュー。**パーティション化されたターゲットファイルを WebSphere MQ メッセージキューに転送します。

注: すべてのターゲットパーティションに対してローカルまたは FTP 接続タイプを選択した場合、ターゲットファイルを統合することができます。外部ローダーまたは WebSphere MQ メッセージキューをターゲット接続タイプとして使用する場合、複数のパーティションを持つセッションからの出力ファイルは統合できません。

以下の表に、マッピングにおけるファイルターゲットの接続オプションを示します。

属性	説明
接続タイプ	FTP、外部ローダ、メッセージキューのいずれかの接続を選択します。ローカル接続を行う場合は、[なし] を選択します。 接続タイプはすべてのパーティションで同じになります。
値	FTP、外部ローダ、メッセージキューのいずれかの接続の場合は、このフィールドの [開く] をクリックして接続オブジェクトを選択します。 各パーティションに別々の接続オブジェクトを指定できます。

ファイルプロパティの設定

フラットファイルソースのファイルプロパティを設定するには、[マッピング] タブの [トランスフォーメーション] ビューの [プロパティ] 設定を使用します。

以下の表に、マッピングにおけるファイルターゲットのファイルプロパティを示します。

属性	説明
Merge Type	Integration Service によって実行される、パーティション化されたターゲットのデータの統合のタイプ。 ターゲットファイルのマージ時に、Integration Service はすべてのパーティションの出力をセッションの実行中にマージファイルまたはコマンドに書き込みます。 セッションで外部ローダ、メッセージキューのいずれかを使用する場合には、ファイルはマージできません。
マージファイルディレクトリ	マージファイルの場所。デフォルトは、\$PMTARGETFILEDIR です。
マージファイル名	結合ファイルの名前。デフォルトでは「ターゲット名.out」です。
Append if Exists	各パーティションのターゲットファイルおよびリジェクトファイルに出力データを追加します。ターゲットファイルをマージする場合、出力データをマージファイルに追加します。ターゲットファイルがディスク以外のファイル（FTP ターゲットファイルなど）の場合は、このオプションは使用できません。 このオプションを選択しないと、出力データがターゲットファイルに書き込まれる前に、各ターゲットファイルが切り詰められます。ファイルが存在しない場合は、作成されます。

属性	説明
Output Type	セッションのターゲットのタイプ。ターゲットデータをファイルターゲットに書き込むためのファイルを選択します。ターゲットデータをコマンドに送信するには、[コマンド]を選択します。FTP またはキューターゲット接続に対して [コマンド] を選択することはできません。
ヘッダーオプション	ファイルターゲットにヘッダー行を作成します。
ヘッダーコマンド	ファイルターゲットにヘッダー行を生成するために使用するコマンド。
Footer Command	ファイルターゲットにフッタ行を生成するために使用するコマンド。
Merge Command	マージされたターゲットデータを処理するために使用するコマンド。
Output File Directory	ターゲットファイルの場所。デフォルトは、\$PMTargetFileDir です。
Output File Name	ターゲットファイルの名前。デフォルトは「ターゲット名パーティション番号.out」です。また、パラメータファイルで定義したように、セッションパラメータ「\$OutputFileName」を使用することもできます。
Rejet File Directory	ターゲットリジェクトファイルの場所。デフォルトは、\$PMBadFileDir です。
Reject File Name	リジェクトファイルの名前。デフォルトは「 <i>target name partition number.bad</i> 」です。 また、パラメータファイルで定義したセッションパラメータ「\$BadFileName」を使用することもできます。
Command	1 つのパーティションのターゲット出力データを処理するために使用するコマンド。

パーティション化されたファイルターゲットのコマンドの設定

コマンドを使用して、1 つのパーティションのターゲットデータを処理するか、セッション内のすべてのターゲットパーティションのマージデータを処理します。UNIX では、有効な任意の UNIX コマンドまたはシェルスクリプトを使用します。Windows では、有効な任意の DOS またはバッチファイルを使用します。Integration Service は、フラットファイルターゲットまたはマージファイルではなく、コマンドにデータを送信します。

コマンドを使用して次のタイプのターゲットデータを処理します。

- 単一のパーティションのターゲットデータ。** ターゲットパーティションごとにコマンドを入力できます。Integration Service によって、セッションの実行中にターゲットデータがコマンドに送信されます。
 1 つのパーティションのターゲットデータをコマンドに送信するには、[出力タイプ] に対して [コマンド] を選択します。セッションプロパティでパーティションの [コマンド] プロパティにコマンドを入力します。
- すべてのターゲットパーティションの統合データ。** コマンドを入力してすべてのパーティションの統合データを処理できます。Integration Service によって、セッション実行中にすべてのパーティションのターゲットデータがこのコマンドに同時に送信されます。このコマンドはターゲットデータの順序を維持できません。
 すべてのパーティションのマージデータをコマンドに送信するには、[出力タイプ] として [コマンド] を選択し、セッションプロパティで [Merge Command Line] プロパティにコマンドを入力します。

統合オプションの設定

セッション内のパーティションのターゲットデータを統合できます。ターゲットデータを統合する際、Integration Service によってすべてのターゲットパーティションに対して統合ファイルが作成されます。

次の統合ファイルオプションを設定できます。

- **シーケンシャル統合。** Integration Service によってすべてのパーティションに対して出力ファイルが作成され、続いてセッションの最後に出力ファイルが単一の統合ファイルに統合されます。 Integration Service によって各パーティションの出力データが統合ファイルに順次追加されます。 Integration Service によって、パーティションの [出力ファイル名] および [出力ファイルディレクトリ] の値を使用して、個々のターゲットファイルが作成されます。
- **ファイルリスト。** Integration Service によって、すべてのパーティションに対してターゲットファイルが作成され、個々のファイルのパスを含むファイルリストが作成されます。 Integration Service によって、パーティションの [出力ファイル名] および [出力ファイルディレクトリ] の値を使用して、個々のターゲットファイルが作成されます。 ターゲットファイルをマージディレクトリまたはマージディレクトリの下ディレクトリに書き込んだ場合、ファイルリストには相対パスが含まれます。それ以外の場合は、リストファイルには絶対パスが含まれます。別のマッピングでソースファイルとしてターゲットファイルを使用する場合は、このリストファイルをソースファイルとして使用します。
- **同時統合** Integration Service により、すべてのターゲットパーティションのデータが統合ファイルに同時に書き込まれます。パーティションごとに中間ファイルは作成されません。 Integration Service によってすべてのパーティションに対し統合ファイルに同時に書き込まれるため、統合ファイル内のデータのソート順は連続していない可能性があります。

カスタムトランスフォーメーションのパーティション化

マッピングに、カスタムトランスフォーメーション、Java トランスフォーメーション、SQL トランスフォーメーション、または HTTP トランスフォーメーションが含まれる場合は、以下のパーティション化情報を編集することができます。

- **複数のパーティションの追加。** カスタムトランスフォーメーションで複数のパーティションの設定ができる場合、複数のパーティションを作成することができます。
- **パーティションポイントの作成** トランスフォーメーションで複数のパーティションの設定ができない場合でも、カスタムトランスフォーメーションにパーティションポイントを作成することはできます。

Java、SQL、および HTTP の各トランスフォーメーションは、カスタムトランスフォーメーションを使用して構築され、同じ機能を持っています。カスタムトランスフォーメーションを使用して作成されたすべてのトランスフォーメーションが、カスタムトランスフォーメーションと同じパーティション化機能を持つとは限りません。

各パーティションを 1 つのスレッドで処理するようにカスタムトランスフォーメーションを設定した場合、Workflow Manager によってマッピングの設定に基づいてパーティションポイントが追加されます。

複数のパーティションに関する作業

マッピングで複数のパーティションが許可されるように、カスタムトランスフォーメーションを設定することができます。トランスフォーメーションの [Is Partitionable] プロパティを設定した場合、パイプラインにパーティションを追加できます。[Is Partitionable] オプションに対して次の値を選択できます。

- **いいえ。** トランスフォーメーションはパーティション化できません。同一パイプライン内のこのトランスフォーメーションおよびその他のトランスフォーメーションは、1つのパーティションに含まれる必要があります。データクレンジングなど、トランスフォーメーションによりすべての入力データが一度に処理される場合は、[いいえ] を選択する場合があります。
- **ローカルで。** トランスフォーメーションをパーティション化することはできますが、Integration Service により同じノード上のパイプラインですべてのパーティションが実行される必要があります。トランスフォーメーションの異なるパーティションがメモリ内のオブジェクトを共有する必要がある場合に [ローカル] を選択します。
- **グリッドをまたがる。** トランスフォーメーションをパーティション化することができ、Integration Service により各パーティションは異なるノードに分散されます。

注: 複数入力または出力グループカスタムトランスフォーメーションを含むマッピングに複数のパーティションを追加する場合は、すべてのグループに対して同じ数のパーティションを定義します。

パーティションポイントの作成

トランスフォーメーションで複数のパーティションの設定ができない場合でも、カスタムトランスフォーメーションにパーティションポイントを作成することはできます。カスタムトランスフォーメーションにパーティションポイントを作成する際は、以下のルールおよびガイドラインを使用します。

- トランスフォーメーション内の各入力グループについてパーティションタイプを定義することができます。出力グループについてパーティションタイプを定義することはできません。
- 有効なパーティションタイプは、パススルー、ラウンドロビン、キー範囲、およびユーザー定義ハッシュキーです。

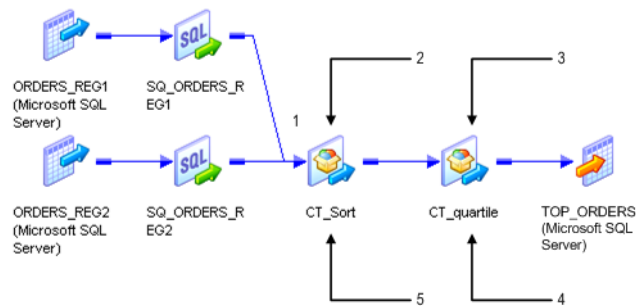
スレッドに関する作業

Integration Service がパーティションごとに1つのスレッドを使用してトランスフォーメーションを処理するようにカスタムトランスフォーメーションを設定するには、[パーティションごとに1つのスレッドを要求します] カスタムトランスフォーメーションプロパティを有効にします。Workflow Manager によって、マッピング内の入力グループの数およびカスタムトランスフォーメーションの場所に基づいて、パススルーパーティションポイントが作成されます。

1つの入力グループ

単一の入力グループのカスタムトランスフォーメーションが、パーティションポイントのない複数の入力グループのカスタムトランスフォーメーションからのダウンストリームである場合、Workflow Manager により、最も近いアップストリームの複数の入力グループトランスフォーメーションにパススルーパーティションポイントが配置されます。

例えば、以下のマッピングを検討します。



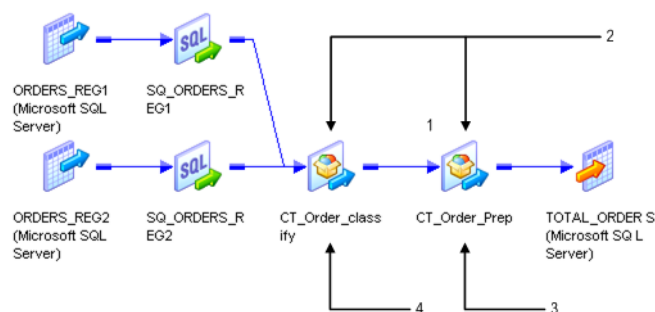
1. パーティションポイント。
2. 複数の入力グループ。
3. 単一の入力グループ。
4. パーティションごとに 1 つのスレッドが必要です。
5. パーティションごとに 1 つのスレッドを必要としません。

CT_quartile は 1 つの入力グループを含んでおり、複数の入力グループを含むトランスフォーメーションである CT_sort からのダウンストリームです。CT_quartile によりパーティションごとに 1 つのスレッドが必要とされますが、アップストリームのカスタムトランスフォーメーションである CT_sort では必要とされません。Workflow Manager により、最も近いアップストリームの複数の入力グループトランスフォーメーション、CT_Sort にパーティションポイントが作成されます。

複数の入力グループ

Workflow Manager により、パーティションごとに単一のスレッドを必要とする複数の入力グループのカスタムトランスフォーメーションにパーティションポイントが配置されます。

例えば、以下のマッピングを検討します。



1. パーティションポイント
2. 複数の入力グループ。
3. パーティションごとに 1 つのスレッドが必要です。
4. パーティションごとに 1 つのスレッドを必要としません。

CT_Order_class および CT_Order_Prep には複数の入力グループがありますが、CT_Order_Prep だけがパーティションごとに 1 つのスレッドを必要とします。Workflow Manager は CT_Order_Prep にパーティションポイントを作成します。

ジョイナトランスフォーメーションのパーティション化

トランスフォーメーション範囲が「すべての入力」の場合、ジョイナトランスフォーメーションにパーティションポイントを作成すると、Workflow Manager はパーティションタイプを自動ハッシュキーに設定します。トランスフォーメーション範囲が「トランスフォーメーション」の場合は、パーティションタイプがパススルーに設定されます。

マスターソースと明細ソースで同数のパーティションを作成する必要があります。ソート済み入力を使用するようにジョイナトランスフォーメーションを設定した場合は、パーティションタイプをパススルーに変更できません。パイプラインにジョイナトランスフォーメーションのマスターソースが含まれる場合、指定できるパーティションは1つだけで、このジョイナトランスフォーメーションにパーティションポイントを追加することはできません。

ジョイナトランスフォーメーションにパーティションポイントを作成した場合、Integration Service によってキャッシュのパーティション化が使用されます。ジョイナトランスフォーメーションでパーティション化を使用する場合、ジョイナトランスフォーメーションのマスターソースと明細ソースに対し、複数のパーティションを作成することができます。

ジョイナトランスフォーメーションにパーティションポイントを作成しない場合、明細ソースについては n 個のパーティションを、マスターソースについては1つのパーティションを作成できます (1: n)。

注: 行トランスフォーメーションスコープを使用するようにジョイナトランスフォーメーションを設定する場合は、ジョイナトランスフォーメーションにパーティションポイントを追加することはできません。

ソート済みジョイナトランスフォーメーションのパーティション化

ソート済み入力を使用するジョイナトランスフォーメーションを追加した場合、ジョイナトランスフォーメーションがソート済みのデータを受け取っていることをチェックする必要があります。ソースに大量のデータが含まれる場合、パーティション化を設定してパフォーマンスを高めることができます。しかし、行を再配分するパーティションではソート済みデータの順序が変わる可能性があるため、ソート済みデータが正しく維持されるようにパーティションを設定することが重要です。

例えば、ハッシュ自動キーパーティションポイントを使用した場合、統合サービスによってハッシュ関数が使用され、複数のパーティション間でデータを分散する最良の方法が決定されます。ただし、この場合、ソート順が統合サービスによって維持されないため、このタイプのパーティションポイントを使用するにあたってはパーティション化のガイドラインに従う必要があります。

データを結合する場合、マスターソースおよび明細ソースのパーティション数が同じになるように設定して、マスターパイプラインおよび明細パイプラインのデータをパーティション化することができます。Integration Service は複数のパーティションを同時に処理します。

ジョイナトランスフォーメーションで使用するパーティションタイプに基づいてソート順が維持されるようにパーティションを設定しなければならない場合があります。ジョイナトランスフォーメーションが 1: n のパーティション化を使用する場合、マスタパイプラインと明細パイプラインはソート済みポートで連結され、セッションは予期せず終了します。

パーティション化に関する次のガイドラインを考慮してください。

- **ソート済みのフラットファイルまたはソート済みのリレーショナルデータを使用します。** マスターパイプラインおよび明細パイプラインに大きなフラットファイルが1つある場合は、最初のパーティション内のすべてのソート済みデータを渡し、さらに他のパーティション内の空のファイルデータを渡すように設定します。
- **ソータトランスフォーメーションを使用します。** ジョイナトランスフォーメーションに自動ハッシュキーパーティションを使用する場合は、各ソータトランスフォーメーションでも自動ハッシュキーパーティションポイントが使用されるように設定する必要があります。

ソートの基点とジョイナトランスフォーメーションの間には、パススルーパーティションポイントだけを追加するようにしてください。

ソート済みフラットファイルの使用

マスタパイプラインに1つのフラットファイルが、詳細パイプラインに複数のフラットファイルが存在する場合は、 $1:n$ のパーティションを使用します。 $1:n$ のパーティションを使用した場合、Integration Service によってデータがパーティション間で再分散されないため、ソート順は維持されます。マスタパイプラインと明細パイプラインに大きなフラットファイルが1つずつ存在する場合は、 $n:n$ のパーティションを使用し、ジョイナトランスフォーメーションにパススルーパーティションまたは自動ハッシュキーパーティションを追加します。自動ハッシュキーパーティションポイントを追加する場合は、ソート順を維持するため、ソート済みのすべてのデータを最初のパーティションで渡すようパーティションを設定する必要があります。

1:n パーティションの使用

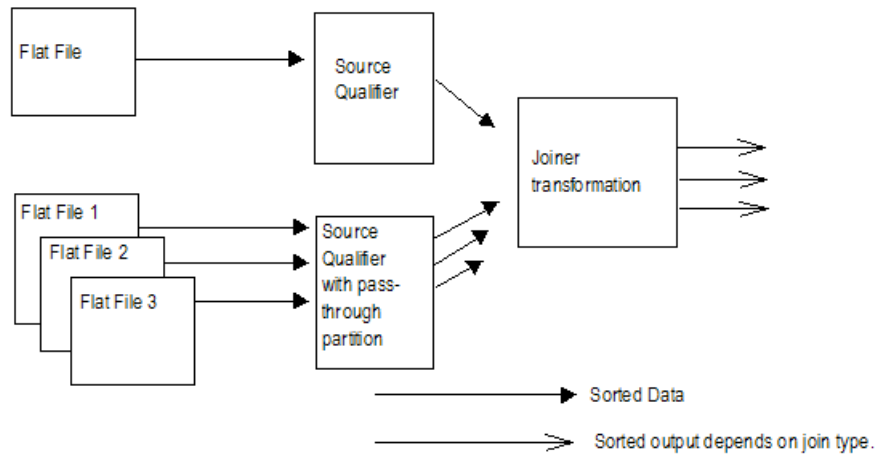
マスタパイプラインに1つのフラットファイルを、明細パイプラインに複数のフラットファイルを使用するセッションでは、マスタソースに1つのパーティションを使用し、明細ファイルソースに n 個のパーティションを使用します ($1:n$)。パススルーパーティションポイントは、明細ソース修飾子トランスフォーメーションに追加してください。ジョイナトランスフォーメーションにはパーティションポイントを追加しません。マスタソースに対して1つのパーティションを作成した場合、ソート済みのデータが統合サービスによってパーティション間で再分散されないため、ソート順は維持されます。

同じ構造を持つ明細パイプラインに複数のファイルがある場合は、以下のガイドラインを使用してファイルをジョイナトランスフォーメーションに渡します。

- 各パイプラインに1つのソースおよび1つのソース修飾子トランスフォーメーションを含むマッピングを設定します。
- セッションプロパティの「マッピング」タブから「トランスフォーメーション」ビューを表示し、「プロパティ」設定で各フラットファイルのパスおよびファイル名を指定します。
- すべてのファイルについて、ソース定義で設定したものと同一ファイルプロパティを使用する必要があります。
- フラットファイル間でソート済みデータの範囲が重複していても構いません。ファイルごとに一意のデータ範囲を使用する必要はありません。

$1:n$ パーティション化を使用してファイルデータをソートした場合、ジョイナトランスフォーメーションによって結合タイプに基づいて未ソートデータが出力される可能性があります。完全外部結合、または詳細な外部結合を使用した場合、未ソートデータの原因となる一致しないマスタ行は統合サービスによって最後に処理されます。

次の図に、1:n のパーティション化で結合されたソート済みファイルデータを示します。



n:n パーティションの使用

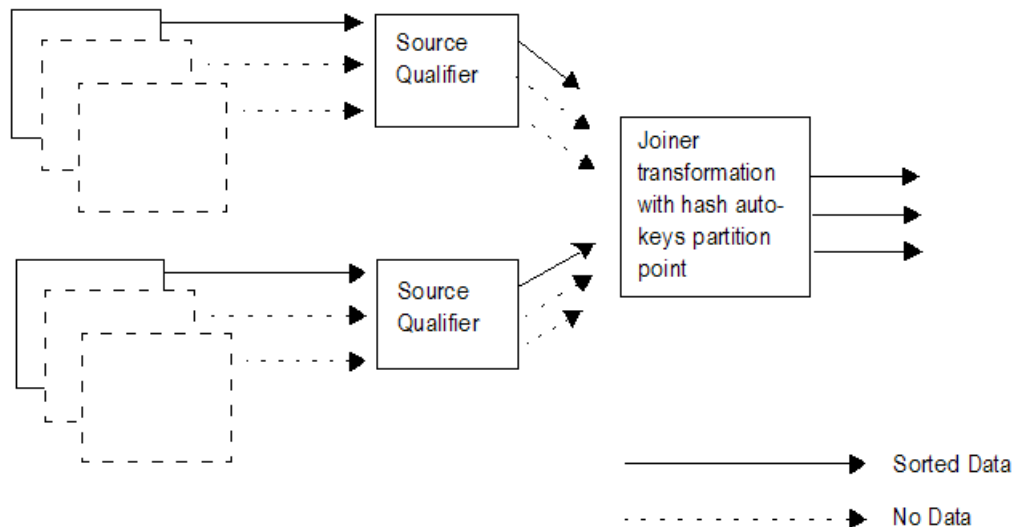
ソート済みのフラットファイルデータを使用するセッションでは、マスターパイプラインと明細パイプラインに対し、 $m:n$ のパーティションを使用します。ジョイナトランスフォーメーションでは、パススルーパーティションまたはハッシュ自動キーパーティションを追加できます。

ジョイナトランスフォーメーションでパススルーパーティションを追加する場合は、マッピングでソート順を維持します。ジョイナトランスフォーメーションでハッシュ自動キーパーティションポイントを追加する場合は、単一のパーティションで、ソート済みのすべてのデータをジョイナトランスフォーメーションに渡すことにより、ソート順を維持することができます。ソート済みのデータを単一のパーティションで渡した場合、統合サービスによりハッシュ関数を使用してデータを再分散する際にソート順が維持されます。

ソート済みのすべてのデータを統合サービスによって1つのパーティションで渡すためには、最初のパーティションでソート済みのファイルを、残りのパーティションでは空のファイルを使用するようにセッションを設定します。

統合サービスにより、複数のパーティション間で行が再分散され、ソート済みデータが結合されます。

次の図では、ソート順を維持するために単一のパーティションを介して渡されるソート済みのファイルデータを示します。



ソート済みリレーショナルデータの使用

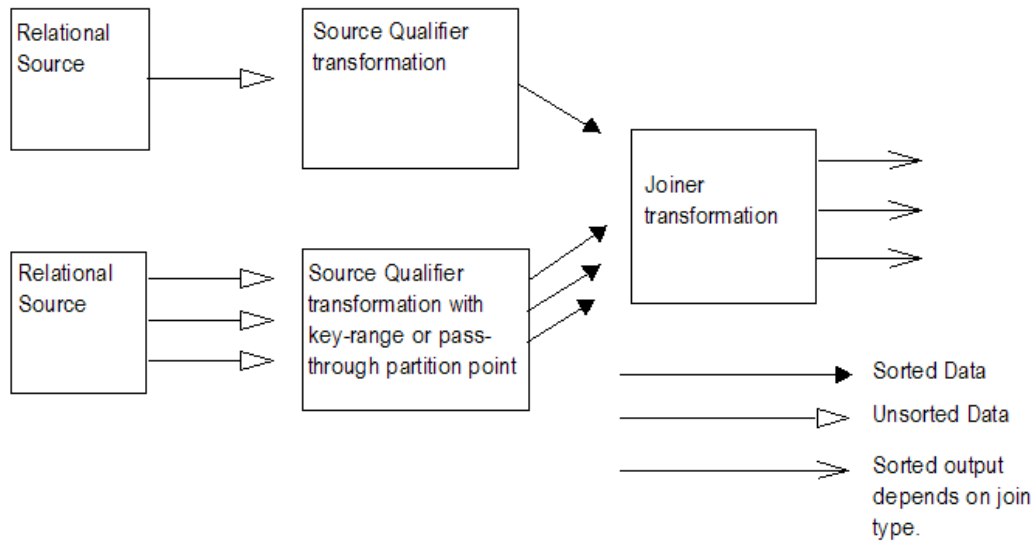
リレーショナルデータを結合する場合、マスターパイプラインと明細パイプラインに $1:n$ のパーティションを使用します。 $1:n$ のパーティションを使用する場合は、ジョイナトランスフォーメーションにパーティションポイントを追加することはできません。 $n:n$ のパーティションを使用した場合、ジョイナトランスフォーメーションにパススルーパーティションまたは自動ハッシュキーパーティションを追加できます。自動ハッシュキーパーティションポイントを使用する場合は、ソート順を維持するため、ソート済みのすべてのデータを最初のパーティションで渡すようパーティションを設定する必要があります。

1:n パーティションの使用

ソート済みのリレーショナルデータを使用するセッションでは、マスターソースには 1 つのパーティションを、明細ソースには n 個のパーティションを使用します ($1:n$)。ソース修飾子トランスフォーメーションにキー範囲またはパススルーパーティションポイントを追加します。ジョイナトランスフォーメーションにはパーティションポイントを追加しません。マスターソースに対して 1 つのパーティションを作成した場合、データが統合サービスによってパーティション間で再分散されないため、ソート順は維持されます。

$1:n$ パーティション化を使用してリレーショナルデータをソートした場合、ジョイナトランスフォーメーションによって結合タイプに基づく未ソートデータが出力される可能性があります。完全外部結合、または詳細な外部結合を使用した場合、未ソートデータの原因となる一致しないマスタ行は統合サービスによって最後に処理されます。

次の図に、1:n でパーティション化されたソート済みのリレーショナルデータを示します。

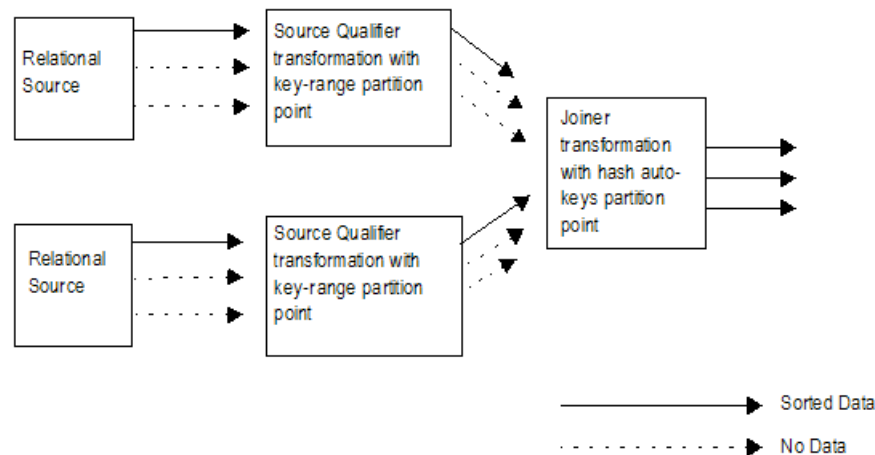


n:n パーティションの使用

セッションがソート済みのリレーショナルデータを使用する場合、マスターパイプラインと明細パイプラインに対して $n:n$ のパーティションを使用し、ジョイナトランスフォーメーションでパススルーまたはハッシュ自動キーパーティションポイントを追加します。

ジョイナトランスフォーメーションでパススルーパーティションを使用する場合は、マッピングでソート済みデータを維持します。自動ハッシュキーパーティションポイントを使用すると、ソート済みのすべてのデータを単一のパーティションを介して Joiner トランスフォーメーションに渡すことにより、ソート順を維持することができます。キー範囲パーティションポイントは、最初のパーティションにすべてのソースデータが保持される Source Qualifier トランスフォーメーションに追加します。ソート済みのデータを単一のパーティションで渡した場合、統合サービスによりハッシュ関数を使用して複数のパーティション間でデータが再分散され、ソート済みのデータが結合されます。

次の図に、ソート順を維持するために単一のパーティションを介して渡されるソート済みのリレーショナルデータを示します。

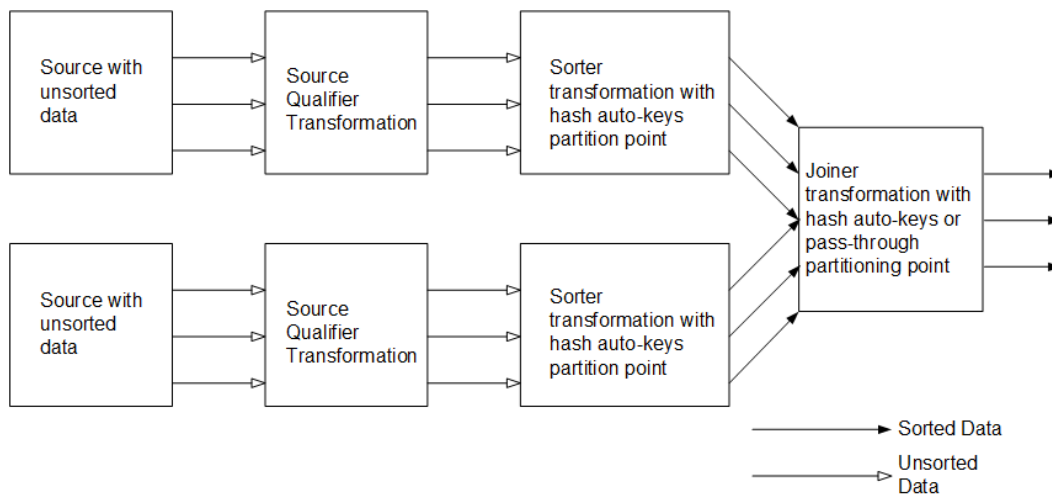


ソータトランスフォーメーションの使用

ソータトランスフォーメーションを使用してデータをソートするセッションでは、マスタパイプラインと明細パイプラインに対し、 $n:n$ のパーティションを使用します。ソータトランスフォーメーションに自動ハッシュキーパーティションポイントを使用してデータをグループ化します。ジョイナトランスフォーメーションには、パススルーパーティションまたは自動ハッシュキーパーティションを追加できます。

データは、統合サービスによって同じハッシュ値を持つパーティションへとグループ化され、ソータトランスフォーメーションによってソートされてから、ジョイナトランスフォーメーションに渡されます。統合サービスがハッシュ自動キーパーティションを使用して設定されたジョイナトランスフォーメーションを処理する際、各ソータトランスフォーメーションからのデータをルーティングするときに使用したパーティションと同じパーティションを使用してソート済みのデータを処理することによって、ソート順が保持されます。

次の図では、ソータトランスフォーメーションと自動ハッシュキーパーティションを使ってソート順を維持しています。



注: 最適なパフォーマンスを得るには、ソート済みのフラットファイルまたはソート済みのリレーショナルデータを使用します。必要に応じて、マッピングにソータトランスフォーメーションを追加した場合の処理オーバーヘッドを計算してください。

パーティションを使用したソート済みジョイナトランスフォーメーションの最適化

ソート済みジョイナトランスフォーメーションにパーティションを使用する場合、データのグループ化や、 $n:n$ パーティションの使用によって、パフォーマンスを最適化できる場合があります。

ソート基点の上流への自動ハッシュキーパーティションの追加

ソート済みジョイナトランスフォーメーションをパーティション化するとき、適切な結果と最適なパフォーマンスを両立するには、データをグループ化しソートしておく必要があります。データをグループ化するには、同じキー値を持つ行を確実に同じパーティションにルーティングする必要があります。複数のパーティションに対してデータを均等にグループ化し配分するための最も効果的な方法は、自動ハッシュキーまたはキー範囲のパーティションポイントを、ソート基点より前に追加することです。このパーティションポイントを、データのソート前に追加することによって、各グループ内でデータを確実にグループ化しソートすることができます。

n:n パーティションの使用

ソート済みジョイナトランスフォーメーションでのパフォーマンスを向上させるには、 $n:n$ のパーティションを使用します。 $n:n$ のパーティションが使用されている場合、ジョイナトランスフォーメーションはマスター行と明細行を同時に読み込むため、すべてのマスターデータをキャッシュする必要はありません。これにより、メモリの使用量が減り、高速な処理が可能になります。 $1:n$ のパーティションが使用されている場合、ジョイナトランスフォーメーションは、マスターパイプラインからのすべてのデータをキャッシュに格納し、メモリのキャッシュが容量を越えた場合、キャッシュをディスクに書き込みます。ジョイナトランスフォーメーションは、明細パイプラインからデータを受け取った後、マスターパイプラインと明細パイプラインとを比較するためにディスクからデータを読み込む必要があります。

ルックアップトランスフォーメーションのパーティション化

ルックアップトランスフォーメーションに対してキャッシュのパーティション化を設定できます。静的ルックアップキャッシュおよび動的ルックアップキャッシュ用に複数のパーティションを作成できます。

パイプラインルックアップトランスフォーメーションのキャッシュは、ルックアップトランスフォーメーションを含むパイプラインとは独立したパイプラインに構築されます。両方のパイプラインに複数のパーティションを作成できます。

ルックアップトランスフォーメーションのキャッシュのパーティション化

キャッシュのパーティション化は、静的キャッシュ、動的キャッシュ、名前付きキャッシュ、名前なしキャッシュで使用します。接続されたルックアップトランスフォーメーションにパーティションポイントを作成するとき、以下に示す条件でキャッシュのパーティション化を使用します。

- ルックアップトランスフォーメーションに自動ハッシュキーパーティションタイプを使用する。
- ルックアップ条件に等価演算子だけが含まれる。
- データベースが大文字と小文字を比較するように設定されている。

例えば、ルックアップ条件に文字列ポートが含まれ、データベースが大文字と小文字を比較しないように設定されている場合、Integration Service によって、キャッシュのパーティション化は実行されずに、次のメッセージがセッションログに書き込まれます。

CMN_1799 Cache partitioning requires case sensitive string comparisons. Lookup will not use partitioned cache as the database is configured for case insensitive string comparisons.

ルックアップトランスフォーメーションにハッシュ自動キーパーティションポイントを作成する場合、Integration Service によりキャッシュのパーティション化が使用されます。

Integration Service によりキャッシュパーティションが作成された場合で、任意のパーティションの最初の行がルックアップトランスフォーメーションに到達した場合、ルックアップトランスフォーメーションのキャッシュの作成が開始されます。コンカレントキャッシュ用にルックアップトランスフォーメーションを設定した場合、Integration Service によってパーティションのすべてのキャッシュが同時に構築されます。

パーティション化キャッシュの共有

パーティション化されたルックアップキャッシュを共有する場合は、次のガイドラインに従います。

- ルックアップトランスフォーメーションでは、次の条件が満たされている場合、パーティション化されたキャッシュを共有できます。
 - キャッシュの構造が同一である。最初の共有トランスフォーメーションのルックアップ/出力ポートが、それ以降のトランスフォーメーションのルックアップ/出力ポートと一致する必要があります。
 - 各トランスフォーメーションに同じルックアップ条件が存在し、ルックアップ条件カラムが同じ順序で配置されている。
- 非パーティション化キャッシュとの間で、パーティション化されたキャッシュを共有することはできません。
- ルックアップキャッシュをターゲットロード順グループ間で共有するには、同じパーティション数のターゲットロード順グループを設定する必要があります。
- 名前なしキャッシュを共有するルックアップトランスフォーメーション間で Integration Service によって不一致が検出された場合、キャッシュファイルが再構築されます。
- 名前付きキャッシュを共有するルックアップトランスフォーメーション間で Integration Service によって不一致が検出された場合、そのセッションは失敗します。

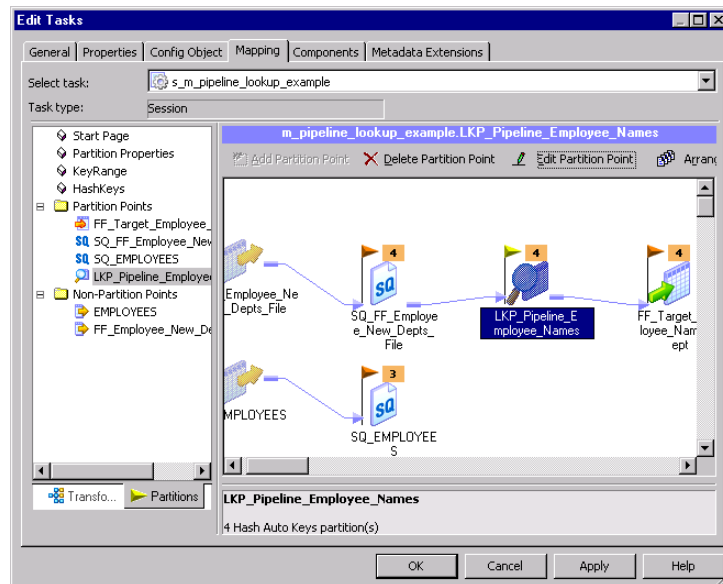
パイプラインルックアップトランスフォーメーションのキャッシュのパーティション化

パイプラインルックアップトランスフォーメーションは、キャッシュに対してデフォルトで有効になっています。Integration Service でルックアップキャッシュを構築するとき、ルックアップソースをパーティション化すると、パフォーマンスが向上します。ルックアップトランスフォーメーションは、ルックアップソースがキャッシュされたときに行の処理を開始します。

パイプラインルックアップトランスフォーメーションを設定すると、ルックアップソースおよびソース修飾子はルックアップトランスフォーメーションとは別のパイプラインに含められます。このパイプラインは部分パイプラインです（ターゲットが含まれないため）。Integration Service では、この部分パイプライン内のソースデータを読み込みます。パイプラインに複数のパーティションを作成すると、処理のパフォーマンスが向上します。

Integration Service でキャッシュを構築するとき、部分パイプラインのソースデータが他のパイプラインに渡されます。部分パイプライン内のパーティションの数がルックアップトランスフォーメーションのパーティションの数と異なる場合、Integration Service によってパーティションポイントが作成されます。ルックアップトランスフォーメーションに自働ハッシュキーパーティションポイントが含まれる場合、キャッシュにはルックアップトランスフォーメーションと同じ数のパーティションが作成されます。それ以外の場合は、キャッシュのパーティションは 1 つだけです。

以下の図に、パイプラインルックアップトランスフォーメーションおよびソース修飾子ルックアップソースを含むセッションのパーティションを示します。



Integration Service では、3つのパーティション内の Employee 行を処理します。ルックアップトランスフォーメーションを含むパイプラインには、4つのパーティションがあります。ルックアップトランスフォーメーションには自動ハッシュキーパーティションポイントが含まれるため、キャッシュは4つのパーティションにパーティション化されます。

シーケンスジェネレータトランスフォーメーションのパーティション化

未キャッシュのシーケンスジェネレータトランスフォーメーションを使用するグリッド上のセッションに複数のパーティションを設定する場合、Integration Service によって各パーティションに生成されるシーケンス番号は連続しません。

ソートトランスフォーメーションのパーティション化

ソートトランスフォーメーションを使用したセッションで、複数のパーティションを設定した場合、Integration Service によって各パーティション内のデータが別々にソートされます。ソートトランスフォーメーションでパーティションポイントを追加する場合、Workflow Manager を使って、自動ハッシュキー、キー範囲パーティション、またはパススルーパーティション機能を選択できます。

自動ハッシュキーは、ソート済み入力を使用するように設定されたアグリゲータトランスフォーメーションの前にソートトランスフォーメーションを置く場合に使用します。自動ハッシュキーは、同じ値を持つ行を、パーティションキーに基づいて同じパーティションの中でグループ化します。行をグループ化した後、Integration Service によってソートトランスフォーメーションへ行が渡されます。Integration Service によ

って各パーティション内のデータが別々に処理されますが、ハッシュ自動キーパーティション化によってすべてのソースデータは正しくソートされます。それは一致する値を持つ行は同じパーティション内で処理されるからです。アグリゲータトランスフォーメーションにあるデフォルトのパーティションポイントは削除できません。

パーティション化されたセッション内の行すべてを複数のパーティションから1つのソート用パーティションへ送りたい場合は、キー範囲パーティションを使用します。すべての行を単一のソート用パーティションに統合した場合、Integration Service ではデータをすべて一緒に処理できます。

パイプライン内で既にハッシュパーティション機能を使用している場合は、パススルーパーティション機能を使用します。この機能により、ソータトランスフォーメーションに渡されるデータは複数のパーティションにおいて正しくグループ化されます。パススルーパーティション機能は、パイプライン内のパーティションの数を増やさずにセッションのパフォーマンスを向上させることができます。

ソータトランスフォーメーションの作業用ディレクトリ設定

パイプライン内の各ソータトランスフォーメーションについて、Integration Service によって一時ファイルが作成されます。ソートの実行中は、このファイルにデータが読み込みおよび書き込みされます。一時ファイルは、Integration Service によってソータトランスフォーメーションの作業ディレクトリに格納されます。

デフォルトでは、ソータトランスフォーメーションの全パーティションの作業ディレクトリが\$PMTempDir に設定されます。セッションのプロパティで各パーティションに別々の作業ディレクトリを指定できます。

XML ジェネレータトランスフォーメーションのパーティション化

XML を複数のパーティションで作成した場合、各パーティションに対して常に個別の文書を生成します。これは、コミット時のフラグの値に関係なく発生します。XML ジェネレータトランスフォーメーションでキー範囲パーティション化を設定した場合、トランスフォーメーションが孤立した行を受信し、セッションが失敗する可能性があります。これは、XML ジェネレータトランスフォーメーションによって行の間のプライマリキーと外部キーの関係が作成するために発生することがあります。キー範囲パーティション化によって、親の行と子の行を区切ることができます。

トランスフォーメーションに関する制限

パーティション数に関する一部の制約は、パイプラインにおけるトランスフォーメーションのタイプによって決まります。この制約は、再利用可能なトランスフォーメーション、マッピングやマプレットで作成されるトランスフォーメーション、およびショートカットから参照するトランスフォーメーションやマプレット、マッピングなど、あらゆるトランスフォーメーションに適用されます。

以下の表に、トランスフォーメーションに関するパーティションの数の制限を示します。

トランスフォーメーション	制限
カスタムトランスフォーメーション	デフォルトでは、パイプラインにカスタムトランスフォーメーションが含まれる場合に指定できるパーティションの数は1つだけです。 ただし、このトランスフォーメーションでは、複数のパーティションを許可するオプションが「プロパティ」タブにあります。このオプションを有効にすると、このトランスフォーメーションに複数のパーティションを指定できます。カスタムトランスフォーメーションプロシージャがデータクレンジングなどの手続きをすべての入力データの総合に基づいて実行する場合には、「パーティション化可能」を選択しません。
エクスターナルプロシージャトランスフォーメーション	デフォルトでは、パイプラインにエクスターナルプロシージャトランスフォーメーションが含まれる場合に指定できるパーティションの数は1つだけです。 このトランスフォーメーションでは、複数のパーティションを許可するオプションが「プロパティ」タブにあります。このオプションを有効にすると、このトランスフォーメーションに複数のパーティションを指定できます。
ジョイナトランスフォーメーション	パイプラインにジョイナトランスフォーメーションのマスタースソースが含まれる場合、指定できるパーティションは1つだけで、このジョイナトランスフォーメーションにパーティションポイントを追加することはできません。
XML ターゲットインスタンス	パイプラインに XML ターゲットが含まれる場合、指定できるパーティションは1つだけです。

ノーマライザおよびシーケンスジェネレータトランスフォーメーションが生成するシーケンス番号は、パーティション化されたソースの場合はシーケンシャルでないことがあります、一意になっています。

数値関数に関する制限

数値関数の CUME、MOVINGSUM、および MOVINGAVG は、行単位で合計および平均を計算します。パイプラインをパーティション化する方法に応じて、これらの関数のいずれかを含むトランスフォーメーションをデータ行が通過する順序が変化する場合があります。したがって、CUME、MOVINGSUM、または MOVINGAVG 関数を使用する複数のパーティションを持つセッションは、常に同じ計算結果を返すわけではありません。

第 3 章

パーティションタイプ

この章では、以下の項目について説明します。

- [パーティションタイプの概要, 61 ページ](#)
- [パーティションタイプの設定, 63 ページ](#)
- [データベースパーティション化のパーティションタイプ, 65 ページ](#)
- [ハッシュ自動キーのパーティションタイプ, 69 ページ](#)
- [ハッシュユーザーキーパーティションタイプ, 69 ページ](#)
- [キー範囲パーティションタイプ, 70 ページ](#)
- [パススルーパーティションタイプ, 73 ページ](#)
- [ラウンドロビンパーティションタイプ, 75 ページ](#)

パーティションタイプの概要

PowerCenter Integration Service では、各パーティションポイントでデフォルトのパーティションタイプが作成されます。パーティション化オプションを使用すると、パーティションタイプを変更できます。パーティションタイプは、パーティションポイントでパーティションにデータを配分する方法を制御します。

パイプラインにパーティション化情報を設定する場合、パイプライン上の各パーティションポイントにパーティションタイプを定義する必要があります。パーティションタイプにより、PowerCenter Integration Service がパーティションポイント間にデータを再配分する方法が決定します。

Workflow Manager では、以下のパーティションタイプを定義できます。

- **データベースパーティション化。** IBM DB2 または Oracle システムに対してクエリが実行され、テーブルのパーティション情報が取得されます。データベースの対応するノードからパーティション化データが読み込まれます。複数ノードのテーブル領域に格納された Oracle ソースインスタンスまたは IBM DB2 ソースインスタンスでは、データベースパーティション化を使用します。DB2 ターゲットでは、データベースパーティション化を使用します。
- **ハッシュパーティション化。** 行をグループ別にパーティションに配分するには、ハッシュパーティション化を使用します。例えば品目 ID で品目をソートする必要があるものの、特定の ID 番号を持つ品目の数が判らない場合などです。

使用できるハッシュパーティション化には、次の種類があります。

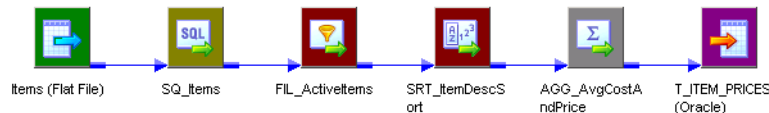
- **ハッシュ自動キー。** グループ化されたポートまたはソート済みのポートがすべて複合パーティションキーとして使用されます。ランクトランスフォーメーション、ソータランスフォーメーション、未ソートのアグリゲータランスフォーメーションで、ハッシュ自動キーパーティション化を使用する必要がある場合があります。

- **ハッシュユーザーキー**。PowerCenter Integration Service によりハッシュ関数が使用され、データ行がパーティション間でグループ化されます。パーティションキーを生成するポート数を定義します。
- **キー範囲**。複合パーティションキーを形成するポートを1つ以上指定します。各ポートに対して指定した範囲に基づいて各パーティションにデータが渡されます。パイプライン上のソースまたはターゲットがキー範囲によってパーティション化される場合に、キー範囲パーティション化を使用します。
- **パススルー**。1つのパーティションポイントのすべての行が、再配分されずに次のパーティションポイントに渡されます。パフォーマンスを向上させるために新しいパイプラインステージを作成して、パーティション間のデータの分散を変更しない場合に、パススルーパーティション化を選択します。
- **ラウンドロビン**。1つ以上のパーティションにデータのブロックが配分されます。ラウンドロビンパーティション化は、各パーティションでブロックの数やサイズに基づいて行を処理する場合に使用します。

パイプラインでのパーティションタイプの設定

パイプライン上の様々なポイントに、様々なパーティションタイプを作成できます。

以下の図に、セッションのパフォーマンスを向上させるために異なったパーティションタイプを作成できるマッピングを示します。



このマッピングでは、品目に関するデータが読み出され、卸売原価と価格の平均値が算出されます。このマッピングではサイズの異なる3つのフラットファイルから品目情報を読み込み、製造中止品目をフィルタで除外する必要があります。説明を使ってアクティブな品目をソートし、平均価格と平均卸売原価を算出して、結果をリレーショナルデータベースに書き込みます。

ソータトランスフォーメーションにおける自動ハッシュキーパーティション化により、同じ説明を持つ品目を含む行がすべて同じパーティションに送られるため、アグリゲータトランスフォーメーションにあるデフォルトのパーティションポイントを削除できます。したがって、アグリゲータトランスフォーメーションでは1つのパーティションで同じ説明を持つすべての品目に関するデータが受信され、この品目の平均原価と平均価格が正しく算出されます。

セッションでこのマッピングを使用する場合、パイプライン上の以下のようなパーティションポイントで異なるパーティションタイプを定義することで、セッションのパフォーマンスを向上させることができます。

- **ソース修飾子**。3つのフラットファイルから同時にデータを読み込むには、ソース修飾子に対して3つのパーティションを指定する必要があります。デフォルトのパーティションタイプであるパススルーパーティション化を受け入れます。
- **フィルタトランスフォーメーション**。ソースファイルのサイズが変化するため、各パーティションで処理するデータの量が異なります。フィルタトランスフォーメーションに1つのパーティションポイントを設定し、フィルタトランスフォーメーションに送られる負荷を分散するために、ラウンドロビンパーティション化を選択します。
- **ソータトランスフォーメーション**。ソータトランスフォーメーションとアグリゲータトランスフォーメーションで重複するグループを無くすために、ソータトランスフォーメーションでハッシュ自動キーパーティション化を使用します。これにより、Integration Service はソータトランスフォーメーションとアグリゲータトランスフォーメーションで行が処理される前に、同じ説明を持つすべての項目が同じパーティションにグループ化されます。アグリゲータトランスフォーメーションにあるデフォルトのパーティションポイントは削除できます。
- **ターゲット**。ターゲットテーブルはキー範囲によりパーティション化されるため、ターゲットにキー範囲パーティション化を指定してターゲットへのデータの書き込みを最適化します。

パーティションタイプの設定

Workflow Manager により、パイプライン上の各パーティションポイントにデフォルトのパーティションタイプが設定されます。トランスフォーメーションのトランスフォーメーション範囲がすべての入力ではない限り、Workflow Manager によって、すべてのパーティションポイントのデフォルトのパーティションタイプとしてパススルーが指定されます。デフォルトのタイプは変更できます。

例えば、ソース修飾子とターゲットインスタンスに対しては、Workflow Manager によってパススルーパーティション化が指定されます。トランスフォーメーション範囲がすべての入力の場合、ランクトランスフォーメーションと未ソートアグリゲータトランスフォーメーションに対して、Workflow Manager によってハッシュ自動キーパーティション化が指定されます。

トランザクションジェネレータ（コミットを生成するアクティブソース）のダウストリームにあり、ターゲット（トランスフォーメーション範囲がトランザクションのトランスフォーメーション）のアップストリームにあるすべてのトランスフォーメーションには、パススルーパーティションタイプを指定する必要があります。また、制約に基づくロードを使用するようセッションを設定した場合、最後のアクティブソースの下流にあるすべてのトランスフォーメーションに対し、パススルーパーティションタイプを指定する必要があります。

ワークフローリカバリが有効化されている場合、Workflow Manager は、パーティションポイントがアグリゲータトランスフォーメーションまたはランクトランスフォーメーションである場合を除き、パーティションタイプをパススルーに設定します。

以下のトランスフォーメーションに対してはパーティションポイントを作成することができません。

- ソース定義
- シーケンスジェネレータ
- XML パーサー
- XML ターゲット
- 接続されていないトランスフォーメーション

以下の表に、パイプライン上の異なるパーティションポイントについて、有効なパーティションタイプおよびデフォルトのパーティションタイプを示します。

トランスフォーメーション (パーティションポイント)	ラウンド ロビン	ハッシュ 自動 キー	ハッシュ ユー ザー キー	キー範 囲	パススル ー	データベース パーティショ ン化
ソース修飾子 (リレーショナルソース)	×	×	×	○	○	○ (Oracle、 DB2)
ソース修飾子 (フラットファイルソース)	×	×	×	×	○	×
Web Service ソース修飾子	×	×	×	×	○	×
XML ソース修飾子	×	×	×	×	○	×
ノーマライザ (COBOL ソース)	×	×	×	×	○	×
ノーマライザ (リレーショナル)	○	×	○	○	○	×

トランスフォーメーション (パーティションポイント)	ラウンド ロビン	ハッシュ 自動 キー	ハッシュ ユーザー キー	キー範 囲	パススル ー	データベース パーティショ ン化
アグリゲータ (ソート済み)	×	×	×	×	○	×
アグリゲータ (未ソート)	×	○	×	×	○	×
カスタム	○	×	○	○	○	×
データマスク	○	×	○	○	○	×
式	○	×	○	○	○	×
エクスターナルプロシージャ	○	×	○	○	○	×
フィルタ	○	×	○	○	○	×
HTTP	×	×	×	×	○	×
Java	○	×	○	○	○	×
ジョイナ	×	○	×	×	○	×
ルックアップ	○	○	○	○	○	×
ランク	×	○	×	×	○	×
ルータ	○	×	○	○	○	×
ソータ	×	○	×	○	○	×
ストアドプロシージャ	○	×	○	○	○	×
トランザクション制御	○	×	○	○	○	×
共有体	○	×	○	○	○	×
構造化されていないデータ	○	×	○	○	○	×
アップデートストラテジ	○	×	○	○	○	×
Web サービスコンシューマ	×	×	×	×	○	×
XML ジェネレータ	×	×	×	×	○	×
XML パーサー	×	×	×	×	○	×
リレーショナルターゲット 定義	○	×	○	○	○	○ (DB2)

トランスフォーメーション (パーティションポイント)	ラウンド ロビン	ハッシュ 自動 キー	ハッシュ ユーザー キー	キー範 囲	パススル ー	データベース パーティショ ン化
フラットファイルターゲット定義	○	×	○	○	○	×
Web サービスターゲット	×	×	×	×	○	×

以下のトランスフォーメーションの場合、トランスフォーメーション範囲がトランザクションの場合はパススルーがデフォルトのパーティションタイプで、トランスフォーメーション範囲がすべての入力の場合はハッシュ自動キーがデフォルトのパーティションタイプです。

- アグリゲータ（未ソート）
- ジョイナ
- ランク
- ソータ

データベースパーティション化のパーティションタイプ

ソースおよびターゲットデータベースに対してデータベースパーティション化用のパーティションタイプを使用することにより、セッションのパフォーマンスを最適化できます。ソースデータベースのパーティション化を使用する場合、Integration Service は、データベースシステムにテーブルパーティション情報を要求し、データをセッションパーティションに取り出します。ターゲットデータベースのパーティション化を使用する場合、Integration Service は対応するデータベースパーティションノードにデータをロードします。

Oracle ソース、IBM DB2 ソース、および IBM DB2 ターゲットに対してデータベースパーティション化を使用します。パイプラインのパーティション数とデータベースのパーティション数は任意です。ただし、パイプラインのパーティション数とデータベースのパーティション数が同じである場合、より高いパフォーマンスを得ることができます。

範囲パーティション化を使用している IBM DB2 ソースおよびターゲットでは、データベースパーティション化によってパフォーマンスを向上させることができます。

複合パーティション化を使用する Oracle ソースの場合、パイプラインパーティションの数をデータセットのサブパーティションの数に一致させるとパフォーマンスを向上させることができます。たとえば、Oracle ソースが3つのパーティションを含み、そのパーティションごとに2つのサブパーティションが含まれる場合、ソースのパイプラインパーティションの数を6に設定します。

データベースソースのパーティション化

ソースデータベースのパーティション化を使用する場合、Integration Service により、パーティション情報のデータベースシステムカタログに対しクエリが行われます。セッションパーティション間のデータベースパーティションからのデータが分散されます。

セッションにデータベースよりも多くのパーティションがある場合、Integration Service によってデータベースパーティションごとに SQL が生成され、次のパーティションポイントでデータがセッションパーティションに再分散されます。

1つのソースでのデータベースのパーティション化

1つのソースを持つソース修飾子でデータベースパーティション化を使用する場合、Integration Service によりデータベースパーティションごとに SQL クエリが生成され、データベースパーティションからのデータがセッションパーティション間に均等に分散されます。

例えば、セッションに3つのパーティションがあり、データベースに5つのパーティションがある場合、Integration Service により、そのセッションパーティション内で SQL クエリが、そのデータベースパーティションに対して実行されるとします。最初のセッションパーティションと2番目のセッションパーティションは、2つのデータベースパーティションからデータを受け取ります。3番目のセッションパーティションは、1つのデータベースパーティションからデータを受け取ります。

Oracle データベースを使用する場合は、Integration Service により、以下の文と同様の SQL 文がパーティション1に対して生成されます。

```
SELECT <column list> FROM <table name> PARTITION <database_partition1 name> UNION ALL
```

```
SELECT <column list> FROM <table name> PARTITION <database_partition4 name> UNION ALL
```

IBM DB2 データベースを使用する場合は、Integration Service により、以下と同様の SQL 文がパーティション1に対して作成されます。

```
SELECT <column list> FROM <table name>  
WHERE (nodenumber(<column 1>)=0 OR nodenumber(<column 1>) = 3)
```

Oracle ソースに5つのパーティション (1-5) があり、各パーティション内に2つのサブパーティション (*a* と *b*) があり、1つのセッションに3つのパーティションがある場合、Integration Service により、データベースのサブパーティションに対してセッションパーティション内で SQL クエリが実行されます。最初のセッションパーティションと2番目のセッションパーティションは、4つのデータベースサブパーティションからデータを受け取ります。3番目のセッションパーティションは、2つのデータベースサブパーティションからデータを受け取ります。

Integration Service により、以下の文と同様の SQL 文がパーティション1に対して生成されます。

```
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition1_a name> UNION ALL  
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition1_b name> UNION ALL  
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition4_a name> UNION ALL  
SELECT <column list> FROM <table name> SUBPARTITION <database_subpartition4_b name> UNION ALL
```

複数のソースでのソース修飾子のパーティション化

リレーショナルソース修飾子では、複数のソーステーブルからのデータを受け取ることができます。

Integration Service では、データベースパーティションの SQL クエリは、ほとんどのパーティションを持つデータベーステーブル内のパーティションの数に基づいて作成されます。データベースパーティションからデータを取得するための SQL 結合条件を作成します。

たとえば、ソース修飾子は2つのソーステーブルからデータを受け取ります。各ソーステーブルには2つのパーティションがあります。セッションに3つのパーティションがあり、データベーステーブルに2つのパーティションがある場合、セッションパーティションの1つはデータを受け取りません。

Oracle の場合、Integration Service では以下の SQL 文が生成されます。

Session Partition 1:

```
SELECT <column list> FROM t1 PARTITION (p1), t2 WHERE <join clause>
```

Session Partition 2:

```
SELECT <column list> FROM t1 PARTITION (p2), t2 WHERE <join clause>
```

Session Partition 3:

No SQL query.

IBM DB2 の場合、Integration Service では以下の SQL 文が生成されます。

Session Partition 1:

```
SELECT <column list> FROM t1,t2 WHERE ((nodenumber(t1 column1)=0) AND <join clause>
```

Session Partition 2:

```
SELECT <column list> FROM t1,t2 WHERE ((nodenumber(t1 column1)=1) AND <join clause>
```

Session Partition 3:

No SQL query.

ソースデータベースのパーティション化での Integration Service の処理

Integration Service では、データベースパーティション化について以下のルールが使用されます。

- Oracle と IBM DB2 以外のデータベースにデータベースパーティション化を指定する場合、Integration Service により 1 つのパーティションでデータが読み込まれ、メッセージがセッションログに書き込まれます。
- セッションのパーティション数がデータベースのテーブルのパーティション数よりも大きい場合、超過しているパーティションはデータを受け取りません。セッションログでは、どのパーティションがデータを受け取らないかが記述されます。
- セッションのパーティション数がデータベースのテーブルのパーティション数よりも小さい場合、データはセッションパーティションに均等に配分されます。一部のセッションパーティションは、複数のデータベースパーティションからデータを受け取ります。
- データベースパーティション化を動的パーティションと共に使用する場合、Integration Service により、セッションの開始時にセッションのパーティション数が決定されます。
- パーティション化を行ったセッションのパフォーマンスは、データベースパーティション内のデータ分散に応じて異なります。Integration Service により、データベースパーティションに対して SQL クエリが生成されます。この SQL クエリは、共有体コマンドまたは join コマンドを実行します。これにより、パフォーマンスに影響を与える大きなクエリ文になる場合があります。

ソースデータベースのパーティション化に関するルールおよびガイドライン

リレーショナルソースでデータベースパーティション化パーティションタイプを使用する場合は、次の規則およびガイドラインに従ってください。

- ソースベースのコミット、ユーザー定義のコミット、制約ベースのロード、ワークフローリカバリのいずれかを使用するようにセッションを設定した場合、データベースパーティション化を使用することはできません。
- データベースパーティション化のためにソース修飾子を設定した場合、以下の状況では、Integration Service によりパススルーパーティション化に戻されます。
 - データベーステーブルが 1 つのデータベースパーティションに格納されている。
 - デバッグモードでセッションを実行する。
 - データベースパーティション化を、1 つのパーティションを持つセッションに対して指定する。
 - プッシュダウンの最適化を使用する。プッシュダウンの最適化は、他のパーティションタイプで使用できません。
- データベーステーブルを読み込む SQL オーバーライドを作成し、データベースパーティション化を設定した場合、Integration Service はパススルーパーティション化に戻り、メッセージをセッションログに書き込みます。
- ユーザー定義結合を作成した場合、Integration Service によって、各パーティションに対して生成する SQL 文に、この結合が追加されます。

- ソースフィルタを作成した場合、Integration Service によって、各パーティションに対して SQL クエリの WHERE 句に、このソースフィルタが追加されます。

ターゲットデータベースのパーティション化

ターゲットデータベースのパーティション化は、IBM DB2 データベースに対してのみ使用できます。複数ノードのテーブル領域に格納された IBM DB2 テーブルにデータをロードすると、データベースパーティション化パーティションタイプを使用することにより、セッションのパフォーマンスを最適化できます。データベースパーティション化を使用した場合、Integration Service によって、テーブルのパーティション情報のために DB2 システムに対してクエリが行われ、パーティション化データがターゲットデータベースで対応するノードにロードされます。

DB2 以外のターゲットに対してデータベースパーティション化を使用した場合、デフォルトでは、Integration Service はセッションに失敗します。ただし、DB2 以外のリレーショナルターゲットに対してデータベースパーティション化を使用した場合、デフォルトでパススルーパーティション化が使用されるように Integration Service を設定することができます。Administrator ツールで Integration Service プロパティ、TreatDBPartitionAsPassThrough をはいに設定します。

パイプラインのパーティション数やデータベースのノード数に関係なく、データベースパーティション化をターゲットパーティションタイプとして指定できます。ただし、パイプラインのパーティション数とデータベースのノード数が同じである場合、より高いロードパフォーマンスを得ることができます。

ターゲットデータベースのパーティション化に関するルールおよびガイドライン

データベースターゲットでデータベースパーティション化を使用する場合は、次の規則およびガイドラインに従ってください。

- ソースベースのコミット、ユーザー定義のコミット、制約ベースのロード、セッションリカバリのいずれかを使用するようにセッションを設定した場合、データベースパーティション化を使用することはできません。
- ターゲットテーブルが範囲によってパーティション化されている場合、データベースパーティション化を使用することはできません。ターゲットテーブルが範囲によってパーティション化されている場合は、パススルーパーティション化またはキー範囲パーティション化を使用します。
- ターゲットテーブルに、パーティションキーが含まれている必要があります。また、ターゲットインスタンスに含まれるすべての非 NULL パーティションキーカラムを、マッピング内のトランスフォーメーションにリンクさせる必要があります。
- IBM DB2 ターゲットテーブルのパーティションキーが小数のカラムの場合、セッションに対して高精度を有効にします。パーティションキーが小数のカラムであり、かつセッションに対して高精度を有効にしない場合は、統合サービスでセッションが失敗することがあります。
- DB2 バルクロードに複数のパーティションを作成する場合は、ターゲットパーティションタイプにデータベースパーティション化を使用します。他のパーティションタイプを選択した場合、統合サービスが通常のロードに戻り、セッションログに以下のメッセージが書き込まれます。
ODL_26097 Only database partitioning is support for DB2 bulk load. Changing target load type variable to Normal.
- データベースパーティション化を使用するようにセッションを設定した場合、以下の状況では、統合サービスによってパススルーパーティション化に戻されます。
 - DB2 ターゲットテーブルが 1 つのノードに格納されている。
 - デバッガを使用したデバッグモードでセッションを実行した。
 - データベースパーティション化のパーティションタイプをパススルーパーティション化として扱うように統合サービスを設定して、DB2 以外のリレーショナルターゲットに対してデータベースパーティション化を使用する。

ハッシュ自動キーのパーティションタイプ

ハッシュ自動キーパーティション化を、ランクトランスフォーメーション、ソータランスフォーメーション、ジョイナトランスフォーメーション、および未ソートアグリゲータトランスフォーメーションで（またはそれより前で）使用します。これにより、行がこれらのトランスフォーメーションに入力される前に適切にグループ化されるようにします。

以下の図に、ハッシュ自動キーパーティション化を含むマッピングを示します。行は、Integration Service によって、ソータおよびアグリゲータトランスフォーメーションに入力される前にグループに応じて各パーティションに分散されます。



このマッピングでは、ソータランスフォーメーションによって品目が品目の説明別にソートされます。2つ以上のソースファイルに同じ説明を持つ品目が存在すれば、各パーティションに同じ説明を持つ品目が含まれます。自動ハッシュキーパーティション化を使用しなければ、アグリゲータトランスフォーメーションで各品目の平均原価と平均価格が誤って算出される場合があります。

原価と価格の計算が間違わないようにするためには、ソータランスフォーメーションにパーティションポイントを設定し、パーティションタイプを自動ハッシュキーに設定します。これを行う場合、Integration Service により、同じ説明を持つすべての項目が1つのパーティション内でソータおよびアグリゲータトランスフォーメーションに送られるようにデータが再分散されます。

ハッシュユーザーキーパーティションタイプ

ハッシュユーザーキーパーティション化では、Integration Service でハッシュ関数を使用し、ユーザー定義のパーティションキーに基づいてデータの行を異なるパーティションごとにグループ化します。パーティションキーを定義するポートを選択します。



上に示すマッピングで自動ハッシュキーパーティション化を指定した場合、ITEM_DESC などのソートキーによりグループ化されたデータの行がソータランスフォーメーションに受け取られます。品目の説明が長く、各品目に固有の ID 番号が与えられていることが判っていれば、ソータランスフォーメーションにユーザー定義ハッシュキーパーティション化を指定し、ハッシュキーに ITEM_ID を選択できます。通常、ハッシュ関数では文字列データよりも数値データの方が速く処理できるため、この操作によってセッションのパフォーマンスが向上することがあります。

パーティションポイントでハッシュユーザーキーパーティション化を選択した場合、ハッシュキーを指定する必要があります。Integration Service では、ハッシュキーを使用して、グループごとに適切なパーティションに行が分散されます。

例えば、ソース修飾子トランスフォーメーションにキー範囲パーティション化を指定した場合、ソースからデータを選択する際に、Integration Service によってキーと範囲が使用されて WHERE 句が作成されます。したがって、Integration Service を使用して、あるパーティションに 135000 未満の顧客 ID を含むすべての行を渡し、別のパーティションに 135000 以上の顧客 ID を含むすべての行を渡すことができます。

トランスフォーメーションにハッシュユーザーキーパーティション化を指定する場合、Integration Service によってこのキーが使用されて、キーとして選択したポートに基づいてデータがグループ化されます。例えば、ハッシュキーとして ITEM_DESC を指定すると、同じ説明を持つ項目を含むすべての行が同じパーティションに送られるように、Integration Service によってデータが分散されます。

ハッシュキーを指定するには、[マッピング] タブの [パーティション] ビューでパーティションポイントを選択し、[キーの編集] をクリックします。[パーティションキーの編集] ダイアログボックスが表示されます。[利用可能なポート] リストには、トランスフォーメーションのコネクトされた入力ポートと出力ポートが表示されます。ハッシュキーを追加するには、このリストで 1 つ以上のポートを選択してから [追加] をクリックします。

キーを定義するポートの順序を変更するには、[選択されたポート] リストでポートを選択して上下の矢印ボタンをクリックします。

キー範囲パーティションタイプ

キー範囲パーティション化を使用すると、パーティションキーとして定義したポートまたはポートのセットに基づいて、Integration Service によりデータの行が分散されます。各ポートについて、値の範囲を定義します。Integration Service は、キーと範囲を使用して適切なパーティションに行を送ります。

例えば、ソース修飾子トランスフォーメーションにキー範囲パーティション化を指定した場合、ソースからデータを選択する際に、Integration Service によってキーと範囲が使用されて WHERE 句が作成されます。したがって、Integration Service を使用して、あるパーティションに 135000 未満の顧客 ID を含むすべての行を渡し、別のパーティションに 135000 以上の顧客 ID を含むすべての行を渡すことができます。

トランスフォーメーションにハッシュユーザーキーパーティション化を指定する場合、Integration Service によってこのキーが使用されて、キーとして選択したポートに基づいてデータがグループ化されます。例えば、ハッシュキーとして ITEM_DESC を指定すると、同じ説明を持つ項目を含むすべての行が同じパーティションに送られるように、Integration Service によってデータが分散されます。

ソーステーブルとターゲットテーブルがキー範囲によってマッピングでパーティション化される場合に、キー範囲パーティション化を使用します。

以下の図に、キー範囲パーティション化によってターゲットテーブルへの書き込みが最適化できるマッピングを示します。



データベース内のターゲットテーブルは、ITEM_ID によって次のようにパーティション化されます。

- パーティション 1：0001-2999
- パーティション 2：3000-5999
- パーティション 3：6000-9999

ターゲットテーブルへの書き込みを最適化するには、次の操作を実行します。

1. ターゲットインスタンスのパーティションタイプをキー範囲に設定します。
2. 3 つのパーティションを作成します。
3. パーティションキーとして ITEM_ID を選択します。

Integration Service によってこのキーが使用され、適切なパーティションにデータが渡されます。

4. キー範囲を次のように設定します。

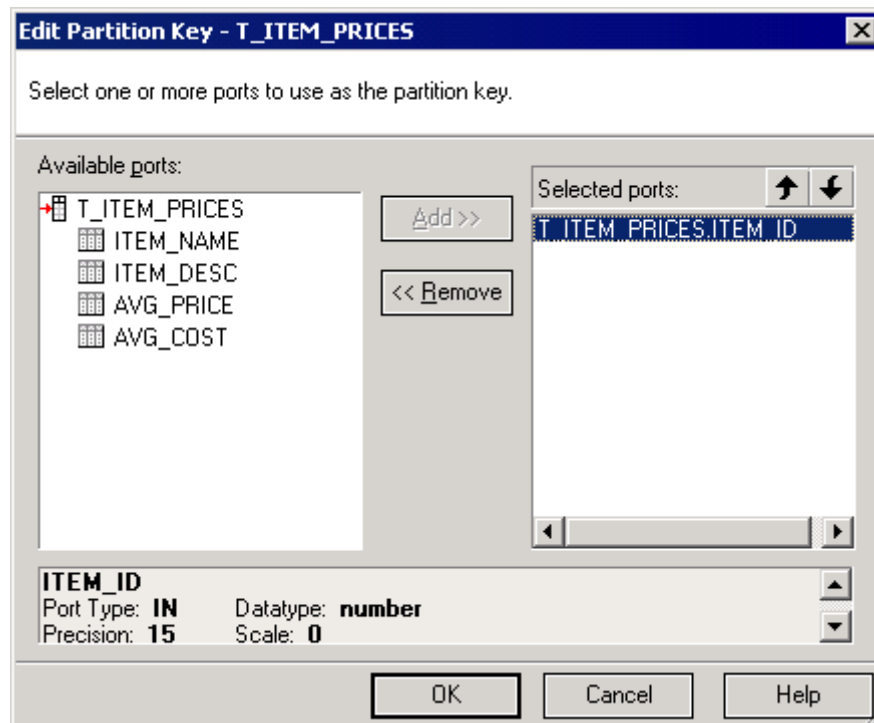
ITEM_ID	Start Range	End Range
Partition #1	-	3000
Partition #2	3000	6000
Partition #3	6000	-

このキー範囲を設定すると、ID が 3000 未満のすべての項目が Integration Service によって最初のパーティションに送られます。ID が 3000 - 5999 のすべての項目が、パーティション 2 に送られます。ID が 6000 以上のすべての品目が、パーティション 3 に送られます。

パーティションキーの追加

キー範囲パーティション化のパーティションキーを指定するには、[マッピング] タブの [パーティション] ビューでパーティションポイントを選択し、[キーの編集] をクリックします。[パーティションキーの編集] ダイアログボックスが表示されます。[利用可能なポート] リストには、トランスフォーメーションのコネクトされた入力ポートと入出力ポートが表示されます。パーティションキーを追加するには、このリストで 1 つ以上のポートを選択してから [追加] をクリックします。

次の図に、ターゲットテーブル T_ITEM_PRICES に対して 1 つのポートをパーティションキーとして選択した [パーティションキーの編集] ダイアログボックスを示します。



パーティションキーを定義するポートの順序を変更するには、[選択したポート] リストでポートを選択して上下の矢印ボタンをクリックします。

キー範囲パーティション化では、ポートの順序は Integration Service によるパーティション間の行の再分散には影響しませんが、セッションのパフォーマンスに影響する場合があります。例えば、次の複合パーティションキーを設定するとします。

Selected Ports

ITEMS.DESCRPTION

ITEMS.DISCONTINUED_FLAG

通常、論理比較は文字列比較よりも速く処理されるため、次のような順序でポートを並べるとセッションの処理速度が向上する場合があります。

Selected Ports

ITEMS.DISCONTINUED_FLAG

ITEMS.DESCRPTION

キー範囲の追加

パーティションキーを構成するポートを特定した後に、[マッピング] タブのパーティションビューで各ポートの範囲を入力する必要があります。

パーティションの開始レンジまたは終了レンジを空白にしておくことができます。範囲の開始を空白にした場合、Integration Service は最小データ値を範囲の開始として使用します。範囲の終了を空白にした場合、Integration Service は最大データ値を範囲の終了として使用します。

例えば、2 つのパーティションを含むパイプラインで、CUSTOMER_ID に基づくキーに次の範囲を追加できます。

CUSTOMER_ID	Start Range	End Range
Partition #1		135000
Partition #2	135000	

顧客テーブルが Integration Service によって読み込まれるとき、顧客 ID が 135000 未満のすべての行が最初のパーティションに送られ、顧客 ID が 135000 以上のすべての行が次のパーティションに送られます。Integration Service は、NULL 値またはキー範囲に該当しない値を含む行を除外します。

データをリレーショナルターゲットにロードするようにパイプラインを設定している場合で、パーティションキーを定義するいずれかのカラムの NULL 値が行に含まれているか、行にどのキー範囲にも当てはまらない値が含まれている場合、その行は Integration Service によって最初のパーティションに送られます。

リレーショナルソースからデータを読み込むようにパイプラインを設定していると、Integration Service によってキー範囲に該当する行が読み込まれます。NULL 値のパーティションキーカラムがある行は読み取りません。

パーティションキーが NULL 値の行を読み込むときは、パススルーパーティション化を使用して SQL オーバーライドを作成します。

フィルタ条件の追加

リレーショナルソースにキー範囲パーティション化を指定する場合、オプションでフィルタ条件を指定したり、SQL クエリをオーバーライドしたりすることができます。

キー範囲の作成に関するルールおよびガイドライン

キー範囲を作成する場合は、以下の規則およびガイドラインに従ってください。

- それぞれのパーティションキーには、最低 1 つのポートが必要です。
- いずれかのパーティションポイントでキー範囲パーティション化を指定した場合には、パーティションキーの各ポートに範囲を指定する必要があります。
- PowerCenter の標準の日付形式を使用して、キー範囲の日付を入力します。
- Workflow Manager では、文字列範囲や数値範囲の重複は検査されません。
- Workflow Manager では、空白または欠落した範囲は検査されません。
- キー範囲パーティション化を指定し、すべてのポートに日付の範囲を入力する必要がある場合には、PowerCenter の標準日付形式を使用します。
- ソース修飾子トランスフォーメーションにキー範囲パーティション化を定義した場合、ソース修飾子トランスフォーメーションで SQL 文を変更すると、デフォルトでパススルーパーティション化が使用されます。
- Workflow Manager では、文字列範囲の重複、数値範囲の重複、空白または欠落した範囲については検査されません。
- パーティションキーを定義するいずれかのカラムの NULL 値が行に含まれているか、行にどのキー範囲にも当てはまらない値が含まれている場合、その行は Integration Service によって最初のパーティションに送られます。

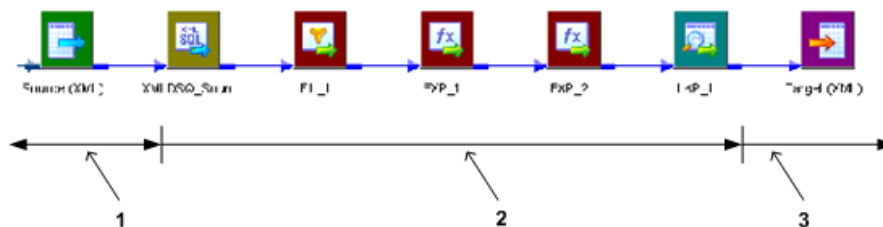
パススルーパーティションタイプ

パススルーパーティション化では、Integration Service によりデータはパーティション間で再分散されことなく処理されます。したがって、ある 1 つのパーティションに置かれた行はすべて、パススルーパーティションポイントを通過した後も同じパーティションにとどまります。

パイプラインにパーティションポイントを追加すると、マスタースレッドにより新しいパイプラインステージが追加されます。データスループットを向上させたいが、パーティションの数を増やしたくない場合に、パススルーパーティション化を使用します。

パススルーパーティション化はパイプライン上の有効な任意のパーティションポイントで指定できます。

以下の図に、パススルーパーティション化を使用してデータスループットを向上できるマッピングを示します。



1. reader スレッド (第 1 ステージ)。
2. トランスフォーメーションスレッド (第 2 ステージ)。
3. writer スレッド (第 3 ステージ)。

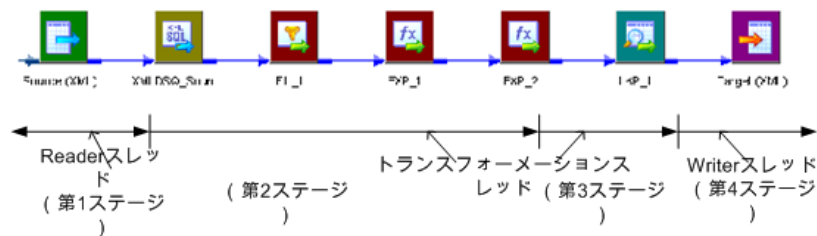
デフォルトでは、このマッピングにはソース修飾子とターゲットインスタンスにパーティションポイントが置かれます。このマッピングには XML ターゲットが含まれるため、パーティションポイントに設定できるパーティションは 1 つだけになります。

この場合マスタースレッドにより作成されるのは、ソースからデータを読み込む reader スレッドが1つ、データを処理するトランスフォーメーションスレッドが1つ、ターゲットにデータを書き込む writer スレッドが1つです。それぞれのパイプラインステージで、次のように行が処理されます。

ソース修飾子 (第1ステージ)	トランスフォーメーション (第2ステージ)	ターゲットインスタンス (第3ステージ)
行セット1	-	-
行セット2	行セット1	-
行セット3	行セット2	行セット1
行セット4	行セット3	行セット2
...
行セット n	行セット (n-1)	行セット (n-2)

パイプラインには3つのステージが含まれるため、Integration Service では行のセットを3つ同時に処理することができます。

式トランスフォーメーションが非常に複雑な場合には、第2（トランスフォーメーション）ステージの処理に時間がかかり、データスループットが低下することがあります。パフォーマンスを向上させるためには、式トランスフォーメーション EXP_2 にパーティションポイントを設定し、パーティションタイプをパススルーパーティション化に設定します。この操作により新しいパイプラインステージが作成されます。新しいトランスフォーメーションスレッドがマスタースレッドにより追加されます。



1. reader スレッド（第1ステージ）。
2. トランスフォーメーションスレッド（第2ステージ）。
3. トランスフォーメーションスレッド（第3ステージ）。
4. writer スレッド（第4ステージ）。

これで、Integration Service は行の4つのセットを次のように同時に処理できます。

ソース修飾子 (第1ステージ)	FI_1 & EXP_1 トランスフォーメーション (第2ステージ)	EXP_2 & LKP_1 トランスフォーメーション (第3ステージ)	ターゲットインスタンス (第4ステージ)
行セット1	-	-	-
行セット2	行セット1	-	-
行セット3	行セット2	行セット1	-
行セット4	行セット3	行セット2	行セット1
...
行セット n	行セット (n-1)	行セット (n-2)	行セット (n-3)

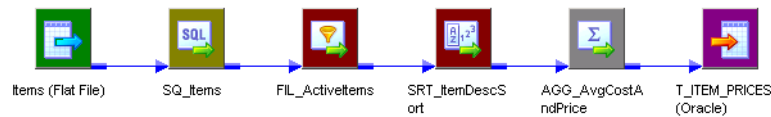
式トランスフォーメーション「EXP_2」にパーティションポイントを追加すると、実行時間の長い1つのトランスフォーメーションステージが、実行時間の短い2つのトランスフォーメーションステージに置き換えられます。データスループットは実行時間の最も長いステージにより決定されます。したがってこの場合は、データスループットが向上します。

ラウンドロビンパーティションタイプ

ラウンドロビンパーティション化では、1つ以上のパーティションにデータのブロックが配分されます。各パーティションでブロックの数やサイズに基づいて行が処理されます。

ラウンドロビンパーティション化は、パーティション間でデータをグループ化する必要がない場合に使用します。サイズの異なるファイルソースからデータを読み込むパイプラインでは、ラウンドロビンパーティション化を使用して、各パーティションに行のブロックを配分します。

以下の図に、行がフィルタトランスフォーメーションに入力される前に、ラウンドロビンパーティション化を使用して行を分散するマッピングを示します。



このマッピングに基づくセッションでは、サイズの異なる3つのフラットファイルから品目情報が読み込まれます。

- ソースファイル1：80,000 行
- ソースファイル2：5,000 行
- ソースファイル3：15,000 行

ソースデータが PowerCenter Integration Service に読み込まれると、第1パーティションでデータの80%、第2パーティションで5%、第3パーティションで15%の処理が開始されます。

負荷をさらに均等に分散するためには、フィルタトランスフォーメーションにパーティションポイントを設定し、パーティションタイプをラウンドロビンに設定します。この結果、各パーティションでデータの約3分の1が処理されるようにデータが振り分けられます。

第 4 章

プッシュダウンの最適化

この章では、以下の項目について説明します。

- [プッシュダウンの最適化の概要, 76 ページ](#)
- [プッシュダウンの最適化のタイプ, 77 ページ](#)
- [動作可能なデータベースとアイドル状態のデータベース, 79 ページ](#)
- [データベースに関する作業, 79 ページ](#)
- [プッシュダウン互換性, 83 ページ](#)
- [日付に関する作業, 86 ページ](#)
- [式に関する作業, 87 ページ](#)
- [エラー処理、ロギング、およびリカバリ, 104 ページ](#)
- [緩やかに変化する次元に関する作業, 105 ページ](#)
- [シーケンスおよびビューに関する作業, 106 ページ](#)
- [\\$\\$PushdownConfig マッピングパラメータの使用, 110 ページ](#)
- [プッシュダウンの最適化のためのセッションの設定, 111 ページ](#)

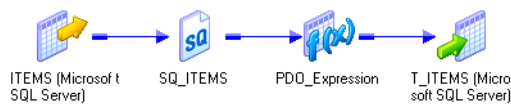
プッシュダウンの最適化の概要

プッシュダウンの最適化を使用して、ソースまたはターゲットデータベースにトランスフォーメーションロジックをプッシュできます。プッシュダウンの最適化用に設定されたセッションを実行した場合、Integration Service でトランスフォーメーションロジックが SQL クエリに変換され、その SQL クエリがデータベースに送信されます。ソースデータベースまたはターゲットデータベースで SQL クエリが実行され、トランスフォーメーションが処理されます。

データベースにプッシュできるトランスフォーメーションロジックの量は、データベース、トランスフォーメーションロジック、マッピング設定、およびセッション設定によって異なります。Integration Service では、データベースにプッシュできないすべてのトランスフォーメーションロジックが処理されます。

Integration Service がソースまたはターゲットデータベースに渡すことのできる SQL 文およびマッピングロジックをプレビューするには、プッシュダウンの最適化ビューを使用します。プッシュダウンの最適化ビューを使用して、プッシュダウンの最適化に関連するメッセージを表示することもできます。

以下の図に、ソースデータベースにプッシュできるトランスフォーメーションロジックを含むマッピングを示します。



このマッピングには、店舗番号 5419 に基づいた商品 ID、およびソースからの商品 ID を作成する式トランスフォーメーションが含まれます。 トランスフォーメーションロジックをデータベースにプッシュするため、Integration Service により以下の SQL 文が生成されます。

```
INSERT INTO T_ITEMS(ITEM_ID, ITEM_NAME, ITEM_DESC) SELECT CAST((CASE WHEN 5419 IS NULL THEN '' ELSE 5419 END) + '-' + (CASE WHEN ITEMS.ITEM_ID IS NULL THEN '' ELSE ITEMS.ITEM_ID END) AS INTEGER), ITEMS.ITEM_NAME, ITEMS.ITEM_DESC FROM ITEMS2 ITEMS
```

Integration Service により、INSERT SELECT 文が生成され、ID、名前、説明の値がソーステーブルから取得され、新しい商品 ID が作成されて、これらの値がターゲットテーブル内の ITEM_ID、ITEM_NAME、および ITEM_DESC のカラムに挿入されます。 Integration Service では、店舗番号 5419、アンダースコア、元の ITEM ID が連結され、新しい商品 ID が取得されます。

プッシュダウンの最適化のタイプ

以下のタイプのプッシュダウンの最適化を設定できます。

- **ソース側でのプッシュダウンの最適化。** Integration Service によりトランスフォーメーションロジックが可能な限りソースデータベースへプッシュされます。
- **ターゲット側でのプッシュダウンの最適化。** Integration Service によりトランスフォーメーションロジックが可能な限りターゲットデータベースへプッシュされます。
- **完全なプッシュダウンの最適化。** Integration Service によりすべてのトランスフォーメーションロジックのターゲットデータベースへのプッシュが試みられます。 Integration Service によりすべてのトランスフォーメーションロジックがデータベースへプッシュできない場合、ソース側とターゲット側の両方でプッシュダウンの最適化が実行されます。

ソース側でのプッシュダウンの最適化セッションの実行

ソース側でのプッシュダウンの最適化が設定されたセッションを実行した場合、Integration Service によりソースからターゲットまで、またはソースデータベースにプッシュできないダウンストリームトランスフォーメーションに到達するまで、マッピングが分析されます。

Integration Service では、データベースにプッシュできるトランスフォーメーションごとのトランスフォーメーションロジックに基づいて、SELECT 文が生成、実行されます。次に、Integration Service ではこの SQL クエリの結果が読み込まれ、残りのトランスフォーメーションが処理されます。

ターゲット側でのプッシュダウンの最適化セッションの実行

ターゲット側でのプッシュダウンの最適化が設定されたセッションを実行した場合、Integration Service によりターゲットからソースまで、またはターゲットデータベースにプッシュできないアップストリームトランスフォーメーションに到達するまで、マッピングが分析されます。 Integration Service では、ターゲットデータベースにプッシュできるトランスフォーメーションごとのトランスフォーメーションロジックに基づいて、INSERT 文、DELETE 文、または UPDATE 文が生成されます。 Integration Service では、データベースにトランスフォーメーションロジックをプッシュできるポイントまで、トランスフォーメーションロジックが処理されます。次に、Integration Service では生成された SQL がターゲットデータベース上で実行されます。

完全なプッシュダウンの最適化セッションの実行

完全なプッシュダウンの最適化を使用するには、ソースデータベースとターゲットデータベースが同じリレーショナルデータベース管理システム内にある必要があります。ソースとターゲットの接続が同じ場合は、完全なプッシュダウンの最適化を構成できます。完全なプッシュダウンの最適化が設定されたセッションを実行した場合、Integration Service によりソースからターゲットまで、またはターゲットデータベースにプッシュできないダウストリームトランスフォーメーションに到達するまで、マッピングが分析されます。Integration Service では、データベースにプッシュできるトランスフォーメーションロジックに基づいて、ソースまたはターゲットに対して SQL 文が生成、実行されます。

大量のデータおよび完全なプッシュダウンの最適化を使用するセッションを実行した場合、データベースサーバーは長いトランザクションを実行しなければなりません。長いトランザクションを生成する場合は、データベースのパフォーマンスに関する以下の問題を検討します。

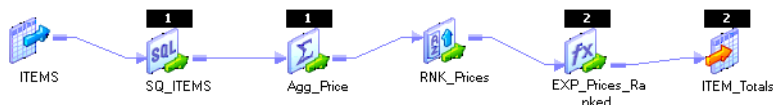
- トランザクションが長くなると、使用されるデータベースリソースが増加します。
- トランザクションが長いと、データベースのロック状態が長引きます。この状態では、データベースの並行性が低下し、デッドロックが起りやすくなります。
- トランザクションが長いと、予期しないイベントが発生しやすくなります。

長いトランザクションに関するデータベースのパフォーマンス問題を最小限にするため、ソース側またはターゲット側でのプッシュダウンの最適化を使用することを検討します。

Integration Service の完全な最適化での動作

完全な最適化のためにセッションを設定した場合、Integration Service によりソースからターゲットまで、またはターゲットデータベースにプッシュできないダウストリームトランスフォーメーションに到達するまで、マッピングが分析されます。Integration Service によりすべてのトランスフォーメーションロジックがターゲットデータベースにプッシュできない場合、すべてのトランスフォーメーションロジックの、ソースデータベースへのプッシュが試行されます。Integration Service により、すべてのトランスフォーメーションロジックがソースまたはターゲットにプッシュできない場合、可能な限り多くのトランスフォーメーションロジックがソースデータベースにプッシュされ、いずれのデータベースにもプッシュできない中間トランスフォーメーションが処理されて、残りのトランスフォーメーションロジックがターゲットデータベースにプッシュされます。Integration Service では、トランスフォーメーションロジックをプッシュする各データベースに対し、INSERT SELECT 文、DELETE 文、または UPDATE 文が生成、実行されます。

例えば、マッピングに以下のトランスフォーメーションが含まれているとします。



ランクトランスフォーメーションをソースデータベースまたはターゲットデータベースにプッシュすることはできません。完全なプッシュダウンの最適化のためにセッションを設定した場合、Integration Service によりソース修飾子トランスフォーメーションおよびアグリゲータトランスフォーメーションがソースにプッシュされ、ランクトランスフォーメーションが処理されて、式トランスフォーメーションおよびターゲットがターゲットデータベースにプッシュされます。Integration Service は、一部のトランスフォーメーションロジックしかデータベースにプッシュできない場合にも、セッションに失敗しません。

動作可能なデータベースとアイドル状態のデータベース

プッシュダウンの最適化中に、Integration Service によりトランスフォーメーションロジックが1つのデータベースにプッシュされます。このデータベースは動作可能なデータベースと呼ばれます。トランスフォーメーションロジックを処理しないデータベースは、アイドル状態のデータベースと呼ばれます。例えば、あるマッピングにジョイナトランスフォーメーションにより結合された2つのソースが含まれているとします。セッションがソース側でのプッシュダウンの最適化に設定されている場合、Integration Service ではジョイナトランスフォーメーションロジックが詳細パイプライン内のソース、つまり動作可能なデータベースへプッシュされます。マスタパイプラインのソースは、トランスフォーメーションロジックを処理しないため、アイドル状態のデータベースです。

Integration Service では以下の条件が使用され、動作可能なデータベースとアイドル状態のデータベースが決定されます。

- 完全なプッシュダウンの最適化を使用した場合、ターゲットデータベースが動作可能なデータベースとなり、ソースデータベースがアイドル状態のデータベースになる。
- ルックアップトランスフォーメーションが含まれるセッションでは、ソースまたはターゲットデータベースが動作可能なデータベースとなり、ルックアップデータベースがアイドル状態のデータベースになる。
- ジョイナトランスフォーメーションが含まれるセッションでは、詳細パイプラインのソースが動作可能なデータベースとなり、マスタパイプラインのソースがアイドル状態のデータベースになる。
- 共有体トランスフォーメーションが含まれるセッションでは、最初の入力グループのソースが動作可能なデータベースとなる。その他の入力グループのソースは、アイドル状態のデータベースになる。

トランスフォーメーションロジックを動作可能なデータベースにプッシュするには、動作可能なデータベースのデータベースユーザーアカウントが、アイドル状態のデータベースから読み込まれる必要があります。

データベースに関する作業

以下のデータベース用にプッシュダウンの最適化を設定することができます。

- Amazon Redshift
- Greenplum
- Google BigQuery
- IBM DB2
- Microsoft Azure SQL Data Warehouse
- Microsoft SQL Server
- Netezza
- Oracle
- PostgreSQL
- SAP HANA
- Snowflake
- Sybase ASE
- Teradata
- Vertica

データベースにトランスフォーメーションロジックをプッシュする場合、データベースが統合サービスとは異なる出力を生成することがあります。

ODBC 接続を使用したデータベースへのプッシュダウンの最適化

ODBC 接続を使用して、プッシュダウンの最適化を設定した場合、PowerCenter 統合サービスはデータベース固有の ODBC ドライバを使用するデータベースに、トランスフォーメーションロジックをプッシュできます。

トランスフォーメーションロジックをデータベースにプッシュするための ODBC 接続の ODBC サブタイプを選択します。ODBC サブタイプは、ODBC 接続オブジェクト定義で指定できます。

以下の ODBC 接続タイプに対して、特定の ODBC サブタイプを設定できます。

- AWS Redshift
- Azure DW
- Greenplum
- Google Big Query
- PostgreSQL
- Snowflake
- SAP HANA
- なし

デフォルトは「なし」です。ODBC サブタイプとして「なし」を選択すると、PowerCenter 統合サービスは、トランスフォーメーションロジックをデータベースにプッシュできません。

Integration Service およびデータベースの出力の比較

Integration Service とデータベースでは、同じトランスフォーメーションロジックを処理した場合でも、異なる結果になる場合があります。データを読み込む際、Integration Service によって、データが別の形式に変換されることがあります。Integration Service とデータベースは、NULL 値、大文字と小文字の区別、ソート順を異なる方法で処理することがあります。

以下の設定と変換が異なる場合、データベースと Integration Service とでは異なる出力が生成されます。

- **最高値または最低値として取り扱われる NULL。** NULL 値の取り扱い方は、Integration Service とデータベースとで異なる場合があります。たとえば、ソートトランスフォーメーションを Oracle データベースにプッシュしたいとします。セッションで、NULL をソート順の最低値として扱われるように設定します。Oracle の場合、NULL 値がソート順の最高値として扱われます。
- **ソート順。** Integration Service とデータベースとで、使用されるソート順が異なる場合があります。例えば、セッション内のトランスフォーメーションを、大文字小文字を区別しないソート順を使用するように設定された Microsoft SQL Server データベースにプッシュしたいとします。大文字小文字を区別するバイナリソート順を使用するようにセッションプロパティを設定します。返される結果は、トランスフォーメーションロジックが Integration Service または Microsoft SQL Server データベースのどちらで処理されるかに応じて異なる可能性があります。
- **大文字と小文字の区別。** Integration Service とデータベースは、異なる方法で大文字と小文字の区別を扱うことがあります。例えば、Integration Service では大文字小文字が使用されるのに対し、データベースでは使用されません。フィルタトランスフォーメーションで使用されるフィルタ条件を IIF (col_varchar2 = 'CA', TRUE, FALSE) とします。'CA' に一致する行を返すデータベースが必要です。ただし、大文字と小文字を区別しない Microsoft SQL Server データベースにこのトランスフォーメーションロジックをプッシュする場合は、値「Ca」、「ca」、「cA」、および「CA」に一致する行が返されます。

- **文字値に変換される数値。** 同じ数値を文字値に変換するにしても、Integration Service とデータベースとで変換形式が異なる可能性があります。データベースでは、数値を許容できない文字形式に変換されてしまう可能性があります。たとえば、テーブルに数値 1234567890 が含まれているとします。Integration Service でこの数を文字値に変換した場合、文字「1234567890」が挿入されます。ただし、この数はデータベースで、「1.2E9」に変換されます。これら文字列の 2 つのセットは同じ値を示しています。ただし、文字に形式「1234567890」を用いる必要がある場合は、プッシュダウンの最適化を無効にすることができます。
- **精度。** Integration Service とデータベースとで、特定のデータタイプの精度が異なる場合があります。トランスフォーメーションデータ型によって、ネイティブデータ型とは異なる可能性があるデフォルトの数値精度が使用されます。例えば、トランスフォーメーション Decimal データ型の精度は 1~28 です。対応する Teradata の Decimal データ型の精度は 1~18 です。データベースで使用されている精度が Integration Service とは異なる場合、結果が変わる可能性があります。

IBM DB2 のルールおよびガイドライン

特定のルールとガイドラインが、IBM DB2 データベースのプッシュダウンの最適化に適用されます。

IBM DB2 データベースにプッシュダウンの最適化を行う際には、以下のルールおよびガイドラインに従います。

- IBM DB2 データベースにプッシュダウンの最適化を適用する場合、タイプキャストの必要なセッションに失敗することがあります。浮動小数点または倍精度浮動小数点から文字列にキャストが行われる場合、または IBM DB2 データベースが許可しないタイプのキャストが必要な場合、セッションが失敗します。
- TO_DATE()および TO_CHAR()関数を IBM DB2 データベースにプッシュするには、次の形式を使用する必要があります。
 - YYYY-MM-DD HH24:MI:SS
 - YYYY-MM-DD-HH24.MI.SS.US
 - YYYY-MM-DD-HH24.MI.SS.MS
 - YYYY-MM-DD-HH24.MI.SS

Netezza に関するルールおよびガイドライン

Netezza データベースへのプッシュダウンの最適化には、以下のルールおよびガイドラインを使用します。

- Netezza データベーステーブルに日付、時刻、またはタイムスタンプのカラムが含まれる場合、[85 以前のタイムスタンプの互換性] セッションプロパティによって、Netezza 上でターゲット側のプッシュダウンの最適化を実行できるようにする必要があります。このオプションを無効にした場合、Integration Service によりターゲット操作が処理されます。
- Netezza には、接続されているまたは接続されていないパッシブなルックアップトランスフォーメーションのトランスフォーメーションロジックはプッシュできません。

PostgreSQL のルールおよびガイドライン

PostgreSQL データベースにプッシュダウンの最適化を行う際には、以下のルールおよびガイドラインに従います。

- TRUNC(DATE)関数を PostgreSQL データベースにプッシュするには、日付引数と形式引数を定義する必要があります。
- TO_DATE()関数と TO_CHAR()関数に対して文字列引数のみ定義し、形式引数を省略した場合、PowerCenter 統合サービスは、セッションプロパティで指定されているデフォルトの日付形式 MM/DD/YYYY HH24:MI:SS に基づいて、文字列を返します。

- SYSTIMESTAMP()関数を PostgreSQL データベースにプッシュする際には、形式引数を指定しないでください。SYSTIMESTAMP に対して形式を指定した場合、データベースはその形式を無視し、完全なタイムスタンプを返します。
- TO_BIGINT 関数または TO_INTEGER 関数を PostgreSQL データベースにプッシュすると、PowerCenter 統合サービスはフラグ引数を無視します。
- IN()関数を PostgreSQL データベースにプッシュすると、PowerCenter 統合サービスは CaseFlag 引数を無視します。
- NS 形式文字列を使って ADD_TO_DATE()関数に対してナノ秒を設定した場合、PowerCenter 統合サービスは ADD_TO_DATE()関数を PostgreSQL にプッシュしません。
- 次の形式を使用した場合、PowerCenter 統合サービスは TO_CHAR()関数と TO_DATE()関数を PostgreSQL にプッシュできません。

NS

JQW

SSSSS

RR

- TRUNC(DATE)関数を PostgreSQL データベースにプッシュする場合には、次の形式を使用できます。

D

HH24

MI

MM

MS

SS

US

YYYY

Teradata のルールおよびガイドライン

特定のルールとガイドラインが、Teradata データベースのプッシュダウンの最適化に適用されます。

Teradata データベースにプッシュダウンの最適化を行う際には、以下のルールおよびガイドラインに従います。

- Decimal または Double データ型を String データ型に変換すると、プッシュダウンの最適化セッションに失敗します。
- Date データ型を String データ型に変換する場合、ターゲット側のプッシュダウンの最適化セッションに失敗します。

Vertica のルールとガイドライン

特定のルールとガイドラインが、Vertica データベースに対するプッシュダウンの最適化に適用されます。

CHAR カラムを引数として持つ関数にプッシュダウンの最適化を適用すると、Vertica データベースはパディング空間を CHAR カラム値から削除します。

Microsoft Azure SQL Data Warehouse のルールおよびガイドライン

次のルールとガイドラインを使用して、完全なプッシュダウンの最適化を使用し、Microsoft Azure SQL Data Warehouse に対するデータの読み書きを行います:

1. Windows および Linux オペレーティングシステムに Microsoft ODBC ドライバをインストールします。
ドライバをダウンロードするには、次を参照してください:
<https://docs.microsoft.com/en-us/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server#microsoft-odbc-driver-131-for-sql-server>.
2. Linux オペレーティングシステムに対しては、ドライバの ODBCINI 環境変数および LD_LIBRARY_PATH 環境変数を設定し、DSN エントリを作成する必要があります。ODBCINI ファイルの値を odbc.ini ファイルの場所に設定します。例えば、`setenv ODBCINI "/data/home/adputf_9/cloud_td/ODBCINI/odbc.ini"`とします。
3. LD_LIBRARY_PATH 環境変数を設定するには、次の形式を使用します:
`setenv LD_LIBRARY_PATH "/opt/microsoft/msodbcsql/lib64/libmsodbcsql-11.0.so.2270.0"`
4. odbc.ini ファイルに Microsoft Azure SQL Data Warehouse データソースのエントリを追加します。次のセクションは odbc.ini ファイルのエントリの例を示しています: [Sample Azure DW ODBC DSN]
`[SD_Azure_DW]Driver=/opt/microsoft/msodbcsql/lib64/libmsodbcsql-11.0.so.2270.0Description=Microsoft ODBC Driver 11 for SQL
ServerServer=dghgdx2ad3.database.windows.netDatabase=INFASQLDW_DEVLoginID=infadwadminPassword=QuotedId=Yes
AnsiNPW=YesEncryptionMethod=1SeedBeforeConnect=1EnableQuotedIdentifiers=1ValidateServerCertificate=0DriverUnicodeType=`
5. PowerCenter 統合サービスを再起動します。

注: ODBC 接続で、サブタイプを **AzureDW** と指定して完全なプッシュダウンの最適化を使用します。

プッシュダウン互換性

データベースへの複数の接続を持つトランスフォーメーションをプッシュするには、その接続がプッシュダウン互換である必要があります。接続が同じデータベース管理システムのデータベースに接続されており、統合サービスで接続がアクセスするデータベーステーブルが特定できる場合、これらの接続はプッシュダウン互換です。

以下のトランスフォーメーションは、複数の接続を持つことができます。

- **ジョイナ**。ジョイナトランスフォーメーションは、複数のソース接続からデータを結合できます。
- **共有体**。共有体トランスフォーメーションは、複数のソース接続からデータを統合できます。
- **ルックアップ**。ルックアップトランスフォーメーションの接続は、ソース接続とは異なる場合があります。
- **ターゲット**。ターゲット接続は、ソース接続とは異なる場合があります。

各接続オブジェクトは、オブジェクト自身に対してプッシュダウン互換です。ソースおよびターゲット接続に対して、同じ接続オブジェクトを使用するようにセッションを設定した場合、統合サービスによりトランスフォーメーションロジックをソースまたはターゲットデータベースにプッシュできます。

リレーショナル接続が、同じデータベースタイプであり、同じデータベースユーザー名、パスワード、特定の同一プロパティを持っている場合、リレーショナル接続の一部はプッシュダウン互換です。

以下の表に、各データベースタイプで同一でなければならない接続プロパティを一覧表示します。

データベースタイプ	同一でなければならない接続プロパティ
IBM DB2	接続文字列 コードページ 接続環境 SQL トランザクション環境 SQL
Greenplum	コードページ 接続文字列 接続環境 SQL トランザクション環境 SQL
Microsoft SQL Server	コードページ サーバー名 ドメイン名 信頼される接続を用いる 接続環境 SQL トランザクション環境 SQL
Oracle	接続文字列 コードページ 接続環境 SQL トランザクション環境 SQL
Sybase ASE	コードページ サーバー名 接続環境 SQL トランザクション環境 SQL
Teradata	コードページ データソース名 接続環境 SQL トランザクション環境 SQL
Vertica	コードページ 接続文字列 接続環境 SQL トランザクション環境 SQL
Microsoft Azure SQL Data Warehouse	コードページ サーバー名 ドメイン名 信頼される接続を用いる 接続環境 SQL トランザクション環境 SQL

注: 統合サービスにより、大文字と小文字を区別した文字列の比較が実行され、接続プロパティが同一であることが確認されます。

同じリレーショナルデータベース管理システム内の Netezza データベースは、プッシュダウン互換です。Netezza データベースは、Netezza データベース自身に対してのみプッシュダウン互換です。

同じデータベースタイプの接続に対し、[「プッシュダウン互換性」 \(ページ 83\)](#)の接続プロパティが同一であるが、データベースユーザー名とパスワードが異なる場合、接続をプッシュダウン互換にできる場合もあります。

データベース接続に互換性がないユーザー

互換性のない接続のデータベースユーザー名とパスワードが一致しない場合は、追加の情報を提供して、接続に互換性を持たせる必要があります。

接続をプッシュダウン互換にするには、以下のアクションを実行します。

1. 動作可能なデータベースのデータベースユーザーが、すべてのアイドル状態のデータベースに対して読み取り権限があることを確認します。
2. [ユーザー互換のない接続に対するプッシュダウンを許可] セッションプロパティを有効にします。
3. Microsoft SQL Server および Sybase への各アイドル状態の接続に対しても、接続プロパティのデータベース名および、すべてのルックアップおよびソースのためのテーブルオーナーを指定する必要があります。

アイドル状態のデータベース内のテーブル名の修飾

Integration Service により SQL が生成され動作可能なデータベースへトランスフォーメーションがプッシュされる場合、生成された SQL ではアイドル状態のデータベース内のテーブルが最低でも 1 つ参照されます。

Integration Service がすべてのテーブルを特定できることを確認するには、以下の場合に対して、アイドル状態のデータベース内のテーブル名を修飾する必要があります。

- アクティブな接続とアイドル状態の接続が同じ接続プロパティを持ち、データタイプが同じで、データベースユーザー名とパスワードが異なる場合。
- ソース修飾子トランスフォーメーションにソースフィルタまたはユーザー定義結合が含まれる場合。

注: その他の場合、Integration Service は、アイドル状態のデータベース内のテーブル名を修飾します。

ソース修飾子トランスフォーメーションの [オーナー名] セッションプロパティ内のソーステーブル名を修飾します。ルックアップトランスフォーメーションの [ルックアップテーブル名] セッションプロパティ内のルックアップテーブル名を修飾します。

以下の構文を使用してテーブル名を修飾します。

データベースタイプ	構文
IBM DB2	<テーブル所有者>.<テーブル名>
Microsoft SQL Server	<データベース名>.<テーブル所有者>.<テーブル名>
Netezza	サポートされていません。
Oracle	<テーブル所有者>.<テーブル名>
Sybase ASE	<データベース名>.<テーブル所有者>.<テーブル名>
Teradata	<データベース名>.<テーブル名>

データベースタイプ	構文
Vertica	<データベース名>.<スキーマ名>.<テーブル名>
Microsoft Azure SQL Data Warehouse	<データベース名>.<テーブル所有者>.<テーブル名>

日付に関する作業

Integration Service とデータベースは、異なる方法で日付を処理することがあります。日付変換をデータベースにプッシュするようにセッションを設定した場合、予期しない結果を受け取るかセッションが失敗する可能性があります。

次の日付設定と変換が異なっている場合、データベースでは Integration Service と異なる出力が生成される可能性があります。

- 文字値に変換される日付値。** Integration Service では、トランスフォーメーション Date/Time データ型が、データベース内のサブ秒精度をサポートしているネイティブデータ型に変換されます。セッション内の日時形式をデータベースでサポートされていない形式に設定した場合、セッションが失敗します。例えば、Integration Service が日付に対して ROUND 関数を実行する際、MM/DD/YYYY HH:MI:SS.US の形式が使用され、文字カラムに日付値が格納されます。しかし、この関数をデータベースで実行すると、そのデータベースのデフォルト日付形式で日付が格納されます。データベースが Oracle の場合、日付はデフォルトの DD-MON-YY 形式で格納されます。日付を MM/DD/YYYY HH:MI:SS.US の形式にする必要のある場合は、プッシュダウンの最適化を無効にします。

- TO_CHAR および TO_DATE 関数の日付形式。** Integration Service によって関数がデータベースにプッシュされる際、Integration Service によって TO_CHAR または TO_DATE 関数内の日付形式が使用されます。データベースは各日付文字列を、そのデータベースでサポートされている日時値に変換します。

例として、Integration Service でデータベースにプッシュされる式を次に挙げます。

```
TO_DATE( DATE_PROMISED, 'MM/DD/YY' )
```

データベースは指定された日付形式文字列 MM/DD/YY にもとづいて DATE_PROMISED ポートを解釈します。データベースは、各日付文字列（例えば 01/22/98）を、サポートされている日付値（例えば Jan 22 1998 00:00:00）に変換します。

Integration Service は、IBM DB2、Microsoft SQL Server、または Sybase データベースにプッシュされた日付形式がそのデータベースでサポートされていなかった場合、プッシュダウンの最適化を停止して、トランスフォーメーションを処理します。

Integration Service は、すべての日付を変換してから、トランスフォーメーションを Oracle データベースまたは Teradata データベースにプッシュします。データベースが日付変換後に日付形式をサポートしない場合、セッションが失敗します。

- HH24 日付形式。** HH24 日付形式は、Teradata 用の日付形式文字列には使用できません。Integration Service で Teradata 用 SQL が生成される際は、HH 形式文字列が代わりに使用されます。
- 日付形式文字列内の空白スペース。** Teradata 内の日付形式文字列には、空のスペースは使用できません。Integration Service で Teradata 用 SQL が生成される際は、スペースが「B」で置き換えられます。
- ルックアップトランスフォーメーションでのサブ秒精度の処理** ルックアップトランスフォーメーションでサブ秒精度を有効にした場合、データベースおよび Integration Service でサブ秒精度を使用したルックアップ比較が実行されます。ただし、異なる結果が返されます。Integration Service とは異なり、データベースではルックアップ結果をサブ秒精度に基づいて切り詰めません。例えば、サブ秒精度をミリ秒まで示すようにルックアップトランスフォーメーションを設定したと想定します。ルックアップ結果が

8:20:35.123456 の場合、データベースからは 8:20:35.123456 が返されますが、Integration Service から
は 8:20:35.123 が返されます。

- **SYSDATE 組み込み変数。** SYSDATE 組み込み変数を使用した場合、Integration Service によってサービス
プロセスを実行しているノードの現在の日付と時刻が返されます。しかし、トランスフォーメーションロ
ジックをデータベースに渡した場合、SYSDATE 変数は、データベースのホストマシンの現在の日付と時刻
を返します。データベースをホストしているマシンのタイムゾーンが、Integration Service を実行してい
るマシンのタイムゾーンと同じでない場合、結果に相違が生じることがあります。

式に関する作業

プッシュダウンの最適化を使用する場合、Integration Service により、データベース内の同等の演算子、変
数、および関数を決定することによって、トランスフォーメーションまたはワークフローリンク内の式が変換
されます。等価な演算子、変数、または関数が存在しない場合、Integration Service はトランスフォーメーシ
ョンロジックを処理します。例えば、Integration Service は集計関数 STDDEV() を Teradataue 上では
STDDEV_SAMP() に変換し、Microsoft SQL Server 上では STDEV() に変換します。FIRST() 集計関数はどのデー
タベースでもサポートされていないため、この関数を使用するトランスフォーメーションは、Integration
Service によって処理されます。

注: Integration Service は式をデータベースにプッシュできない場合、ワークフローログおよびプッシュダウ
ンの最適化ビューアにメッセージを記録します。メッセージを使用して、式をデータベースにプッシュできな
い原因を特定することができます。

この節の表に、データベース内での PowerCenter の演算子、変数、および関数の可用性を要約して示します。

演算子

以下の表に、リレーショナルデータベース内での PowerCenter 演算子の可用性を要約して示します。X マーク
の付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、演算子をデ
ータベースにプッシュできることを示しています。S マークの付いたカラムは、ソース側のプッシュダウン最
適化を使用して、演算子をデータベースにプッシュできることを示しています。

演算子	IBM DB2	Microsoft SQL Server	Oracle	PostgreSQL	SAP HANA	Sybase ASE
+ - * /	X	X	X	X	X	X
%	X	X	X	X	X	X
	S	S	X	X	X	S
= > < >= <= <>	X	X	X	X	X	X
!=	X	X	X	X	X	X
^=	X	X	X	X	X	X
not and or	X	X	X	X	X	X

以下の表に、データウェアハウス内での PowerCenter 演算子の可用性を要約して示します。X マークの付いた
カラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、演算子をデータベー

スにプッシュできることを示しています。S マークの付いたカラムは、ソース側のプッシュダウン最適化を使用して、演算子をデータベースにプッシュできることを示しています。

演算子	Greenplum	Netezza	Teradata	Vertica
+ - * /	X	X	X	X
%	-	X	X	X
	X	X	S	X
= > < >= <= <>	X	X	X	X
!=	X	X	X	X
^=	X	X	X	X
not and or	X	X	X	X

以下の表に、クラウドアプリケーションデータベース内での PowerCenter 演算子の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、演算子をデータベースにプッシュできることを示しています。S マークの付いたカラムは、ソース側のプッシュダウン最適化を使用して、演算子をデータベースにプッシュできることを示しています。

演算子	Amazon Redshift	Azure DW	Snowflake
+ - * /	X ¹	X	X
%	X	X	X
	X	S	X
= > < >= <= <>	S	X	X
!=	S	X	X
^=	S	X	X
not and or	S	X	X

¹. Amazon Redshift は * 演算子をサポートしていません。

変数

以下の表に、リレーショナルデータベース内での PowerCenter 変数の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、変数をデータ

ベースにプッシュできることを示しています。ダッシュ (-) 記号の付いたカラムは、変数をデータベースにプッシュできないことを示しています。

変数	IBM DB2	Microsoft SQL Server	Oracle	PostgreSQL	SAP HANA	Sybase ASE
SESSSTARTTIME	X	X	X	X	X	X
SYSDATE	X	X	X	X	X	X
WORKFLOWSTARTTIME	-	-	-	-	-	-

以下の表に、データウェアハウス内での PowerCenter 変数の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、変数をデータベースにプッシュできることを示しています。ダッシュ (-) 記号の付いたカラムは、変数をデータベースにプッシュできないことを示しています。

変数	Greenplum	Netezza	Teradata	Vertica
SESSSTARTTIME	X	X	X	X
SYSDATE	X	X	X	X
WORKFLOWSTARTTIME	-	-	-	-

以下の表に、クラウドアプリケーションデータベース内での PowerCenter 変数の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、変数をデータベースにプッシュできることを示しています。ダッシュ (-) 記号の付いたカラムは、変数をデータベースにプッシュできないことを示しています。

変数	MS Azure SQL DW
SESSSTARTTIME	X
SYSDATE	X
WORKFLOWSTARTTIME	-

関数

以下の表に、リレーショナルデータベース内での PowerCenter 関数の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、関数をデータベースにプッシュできることを示しています。S マークの付いたカラムは、ソース側のプッシュダウン最適化を使用して、関数をデータベースにプッシュできることを示しています。SF マークの付いたカラムは、ソース

側および完全なプッシュダウンの最適化を使用して、関数をデータベースにプッシュできることを示しています。ダッシュ（-）記号の付いたカラムは、関数をデータベースにプッシュできないことを示しています。

機能	IBM DB2	Microsoft SQL サーバ	Oracle	PostgreSQL	SAP HANA	Sybase ASE
ABORT()	-	-	-	-	-	-
ABS()	X	X	X	X	X	X
ADD_TO_DATE()	X	S	X	X	X	S
AES_DECRYPT()	-	-	-	-	-	-
AES_ENCRYPT()	-	-	-	-	-	-
ASCII()	X	X	X	X	X	X
AVG()	X	X	X	X	SF	X
CEIL()	X	X	X	X	X	X
CHOOSE()	-	-	-	-	-	-
CHR()	X	X	X	X	X	X
CHRCODE()	-	-	-	-	-	-
COMPRESS()	-	-	-	-	-	-
CONCAT()	S	S	X	X	X	S
COS()	X	X	X	X	X	X
COSH()	X	S	X	-	X	S
COUNT()	X	X	X	X	SF	X
CRC32()	-	-	-	-	-	-
CUME()	-	-	-	-	-	-
DATE_COMPARE()	S	S	S	-	SF	S
DATE_DIFF()	-	-	-	X	X	-
DECODE()	X	X	X	X	X	X
DECODE_BASE64()	-	-	-	-	-	-
DECOMPRESS()	-	-	-	-	-	-
ENCODE_BASE64()	-	-	-	-	-	-
EXP()	X	X	X	X	X	X

機能	IBM DB2	Microsoft SQL サーバ	Oracle	PostgreSQL	SAP HANA	Sybase ASE
FIRST()	-	-	-	-	-	-
FLOOR()	X	X	X	X	X	X
FV()	-	-	-	-	-	-
GET_DATE_PART()	X	X	X	-	X	X
GREATEST()	-	-	X	-	-	-
IIF()	X	X	X	X	X	X
IN()	X	X	X	SF	SF	X
INDEXOF()	-	-	-	-	-	-
INITCAP()	-	-	X	X	X	-
INSTR()	S	X	X	-	-	S
IS_DATE()	-	-	-	-	-	-
IS_NUMBER()	-	-	-	-	-	-
IS_SPACES()	-	-	-	-	-	-
ISNULL()	X	X	X	X	SF	X
LAST()	-	-	-	-	-	-
LAST_DAY()	-	-	X	X	X	-
LEAST()	-	-	X	-	-	-
LENGTH()	X	X	X	X	X	X
LN()	-	-	-	X	X	-
LOG()	X	S	X	X	X	S
LOOKUP	X	X	X	X	-	X
LOWER()	X	X	X	X	X	X
LPAD()	-	-	X	X	X	-
LTRIM()	X	X	X	X	X	X
MAKE_DATE_TIME()	-	-	-	-	-	-
MAX()	X	X	X	X	SF	X

機能	IBM DB2	Microsoft SQL サーバ	Oracle	PostgreSQL	SAP HANA	Sybase ASE
MD5()	-	-	-	-	-	-
MEDIAN()	-	-	-	-	-	-
METAPHONE()	-	-	-	-	-	-
MIN()	X	X	X	X	SF	X
MOD()	X	X	X	X	SF	X
MOVINGAVG()	-	-	-	-	-	-
MOVINGSUM()	-	-	-	-	-	-
NPER()	-	-	-	-	-	-
PERCENTILE()	-	-	-	-	-	-
PMT()	-	-	-	-	-	-
POWER()	X	X	X	X	X	X
PV()	-	-	-	-	-	-
RAND()	-	-	-	-	-	-
RATE()	-	-	-	-	-	-
REG_EXTRACT()	-	-	-	-	-	-
REG_MATCH()	-	-	-	-	-	-
REG_REPLACE	-	-	-	-	-	-
REPLACECHR()	-	-	-	-	-	-
REPLACESTR()	-	-	-	-	-	-
REVERSE()	-	-	-	-	-	-
ROUND(DATE)	-	-	X	-	-	-
ROUND(NUMBER)	X	X	X	X	X	X
RPAD()	-	-	X	X	X	-
RTRIM()	X	X	X	X	X	X
SET_DATE_PART()	-	-	-	-	-	-
SIGN()	X	X	X	X	X	X

機能	IBM DB2	Microsoft SQL サーバ	Oracle	PostgreSQL	SAP HANA	Sybase ASE
SIN()	X	X	X	X	X	X
SINH()	X	S	X	-	X	S
SOUNDEX()	X	X	X	-	-	X
SQRT()	X	X	X	X	X	X
STDDEV()	X	X	X	X	-	-
SUBSTR()	S	S	X	X	X	S
SUM()	X	X	X	X	SF	X
SYSDATE()	X	X	X	X	X	X
SYSTIMESTAMP()	X	X	X	X	X	X
TAN()	X	X	X	X	X	X
TANH()	X	S	X	-	X	S
TO_BIGINT	X	X	X	X	X	X
TO_CHAR(DATE)	X	X	X	X	SF	X
TO_CHAR(NUMBER)	X	X	X	X	X	X
TO_DATE()	X	X	X	X	X	X
TO_DECIMAL()	X	X	X	X	X	X
TO_FLOAT()	X	X	X	X	X	X
TO_INTEGER()	X	S	X	X	X	X
TRUNC(DATE)	-	-	X	-	-	-
TRUNC(NUMBER)	X	X	X	X	-	S
UPPER()	X	X	X	X	X	X
VARIANCE()	X	X	X	X	-	-

以下の表に、データウェアハウス内での PowerCenter 関数の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、関数をデータベースにプッシュできることを示しています。S マークの付いたカラムは、ソース側のプッシュダウン最適化を使用して、関数をデータベースにプッシュできることを示しています。SF マークの付いたカラムは、ソース側および

完全なプッシュダウンの最適化を使用して、関数をデータベースにプッシュできることを示しています。ダッシュ（-）記号の付いたカラムは、関数をデータベースにプッシュできないことを示しています。

関数	Greenplum	Netezza	Teradata	Vertica
ABORT()	-	-	-	-
ABS()	X	X	X	X
ADD_TO_DATE()	X	X	X	X
AES_DECRYPT()	-	-	-	-
AES_ENCRYPT()	-	-	-	-
ASCII()	X	X	-	X
AVG()	S	X	X	X
CEIL()	X	X	S	S
CHOOSE()	-	-	-	-
CHR()	X	X	-	X
CHRCODE()	-	-	-	-
COMPRESS()	-	-	-	-
CONCAT()	X	-	S	X
COS()	X	X	X	X
COSH()	X	X	X	X
COUNT()	S	X	X	X
CRC32()	-	-	-	-
CUME()	-	-	-	-
DATE_COMPARE()	S	X	S	X
DATE_DIFF()	-	-	-	X
DECODE()	X	X	X	X
DECODE_BASE64()	-	-	-	-
DECOMPRESS()	-	-	-	-
ENCODE_BASE64()	-	-	-	-
EXP()	X	X	X	X
FIRST()	-	-	-	-

関数	Greenplum	Netezza	Teradata	Vertica
FLOOR()	X	X	S	X
FV()	-	-	-	-
GET_DATE_PART()	S	X	X	X
GREATEST()	-	-	-	-
IIF()	X	X	X	X
IN()	S	-	X	-
INDEXOF()	-	-	-	-
INITCAP()	X	-	-	X
INSTR()	-	-	S	X
IS_DATE()	-	-	-	-
IS_NUMBER()	-	-	-	-
IS_SPACES()	-	-	-	-
ISNULL()	S	X	X	X
LAST()	-	-	-	-
LAST_DAY()	-	-	-	X
LEAST()	-	-	-	-
LENGTH()	X	X	X	X
LN()	X	X	-	X
LOG()	X	X	S	S
LOOKUP	-	X	X	X
LOWER()	X	X	X	X
LPAD()	X	X	-	X
LTRIM()	X	X	X	X
MAKE_DATE_TIME()	-	-	-	-
MAX()	S	X	X	X
MD5()	-	-	-	-
MEDIAN()	-	-	-	-

関数	Greenplum	Netezza	Teradata	Vertica
METAPHONE()	-	-	-	-
MIN()	S	X	X	X
MOD()	S	X	X	X
MOVINGAVG()	-	-	-	-
MOVINGSUM()	-	-	-	-
NPER()	-	-	-	-
PERCENTILE()	-	-	-	-
PMT()	-	-	-	-
POWER()	X	X	X	X
PV()	-	-	-	-
RAND()	-	-	-	-
RATE()	-	-	-	-
REG_EXTRACT()	-	-	-	-
REG_MATCH()	-	-	-	-
REG_REPLACE	-	-	-	-
REPLACECHR()	-	-	-	-
REPLACESTR()	-	-	-	-
REVERSE()	-	-	-	-
ROUND(DATE)	-	-	-	S
ROUND(NUMBER)	X	X	S	S
RPAD()	X	X	-	X
RTRIM()	X	X	X	X
SET_DATE_PART()	-	-	-	-
SIGN()	X	X	S	S
SIN()	X	X	X	X
SINH()	X	X	X	X
SOUNDEX()	-	-	-	X

関数	Greenplum	Netezza	Teradata	Vertica
SQRT()	X	X	X	X
STDDEV()	S	-	X	X
SUBSTR()	X	X	S	X
SUM()	S	X	X	X
SYSDATE()	X	X	X	X
SYSTIMESTAMP()	X	X	X	X
TAN()	X	X	X	X
TANH()	X	X	X	X
TO_BIGINT	X	X	X	X
TO_CHAR(DATE)	X	X	S	X
TO_CHAR(NUMBER)	X	X	X	X
TO_DATE()	X	X	X	X
TO_DECIMAL()	X	X	X	X
TO_FLOAT()	X	X	X	X
TO_INTEGER()	X	X	X	X
TRUNC(DATE)	X	X	-	S
TRUNC(NUMBER)	X	X	S	S
UPPER()	X	X	X	X
VARIANCE()	S	-	X	X

以下の表に、クラウドアプリケーションデータベース内での PowerCenter 関数の可用性を要約して示します。X マークの付いたカラムは、ソース側、ターゲット側、または完全なプッシュダウンの最適化を使用して、関数をデータベースにプッシュできることを示しています。S マークの付いたカラムは、ソース側のプッシュダウン最適化を使用して、関数をデータベースにプッシュできることを示しています。SF マークの付いたカラムは、ソース側および完全なプッシュダウンの最適化を使用して、関数をデータベースにプッシュできることを示しています。ダッシュ (-) 記号の付いたカラムは、関数をデータベースにプッシュできないことを示しています。

関数	Amazon Redshift	Snowflake
ABORT()	-	-
ABS()	X	X

関数	Amazon Redshift	Snowflake
ADD_TO_DATE()	X	-
AES_DECRYPT()	-	-
AES_ENCRYPT()	-	-
ASCII()	-	X
AVG()	S	X
CEIL()	X	X
CHOOSE()	-	-
CHR()	X	X
CHRCODE()	-	-
COMPRESS()	-	-
CONCAT()	X	X
COS()	X	X
COSH()	-	X
COUNT()	S	X
CRC32()	-	-
CUME()	-	-
DATE_COMPARE()	X	X
DATE_DIFF()	X	X
DECODE()	X	X
DECODE_BASE64()	-	-
DECOMPRESS()	-	-
ENCODE_BASE64()	-	-
EXP()	X	X
FIRST()	-	-
FLOOR()	X	X
FV()	-	-
GET_DATE_PART()	X	X

関数	Amazon Redshift	Snowflake
GREATEST()	-	-
IIF()	X	X
IN()	S	X
INDEXOF()	-	-
INITCAP()	X	X
INSTR()	X	X
IS_DATE()	-	-
IS_NUMBER()	-	-
IS_SPACES()	-	-
ISNULL()	S	X
LAST()	-	-
LAST_DAY()	X	X
LEAST()	-	-
LENGTH()	X	X
LN()	X	X
LOG()	-	X
LOOKUP	-	-
LOWER()	X	X
LPAD()	X	X
LTRIM()	X	X
MAKE_DATE_TIME()	-	-
MAX()	S	X
MD5()	-	-
MEDIAN()	-	X
METAPHONE()	-	-
MIN()	S	X
MOD()	S	X

関数	Amazon Redshift	Snowflake
MOVINGAVG()	-	-
MOVINGSUM()	-	-
NPER()	-	-
PERCENTILE()	-	-
PMT()	-	-
POWER()	X	X
PV()	-	-
RAND()	-	-
RATE()	-	-
REG_EXTRACT()	-	-
REG_MATCH()	-	-
REG_REPLACE	-	-
REPLACECHR()	-	X
REPLACESTR()	-	X
REVERSE()	-	-
ROUND(DATE)	-	-
ROUND(NUMBER)	X	X
RPAD()	X	X
RTRIM()	X	X
SET_DATE_PART()	-	-
SIGN()	X	X
SIN()	X	X
SINH()	-	X
SOUNDEX()	-	-
SQRT()	X	X
STDDEV()	S	X
SUBSTR()	X	X

関数	Amazon Redshift	Snowflake
SUM()	S	X
SYSDATE()	S	X
SYSTIMESTAMP()	S	X
TAN()	X	X
TANH()	-	X
TO_BIGINT	X	X
TO_CHAR(DATE)	S	X
TO_CHAR(NUMBER)	X	X
TO_DATE()	X	X
TO_DECIMAL()	X	X
TO_FLOAT()	X	X
TO_INTEGER()	X	X
TRUNC(DATE)	S	-
TRUNC(NUMBER)	S	X
UPPER()	X	X
VARIANCE()	S	X

プッシュダウンの最適化の関数に関するルールおよびガイドライン

データベースに関数をプッシュする際には、以下のルールおよびガイドラインを使用します。

リレーショナルデータベース

- トランスフォーメーションロジック内で ADD_TO_DATE を使用して日、時間、分、または秒を変更した場合、この関数を Teradata データベースにプッシュできません。
- LAST_DAY() を Oracle にプッシュした場合、秒まで示された日付が Oracle によって返されます。入力日付にサブ秒が含まれている場合、Oracle では日付が秒にトリミングします。
- LTRIM、RTRIM、または SOUNDEX をデータベースにプッシュした場合、引数（' '）はそのデータベースでは NULL として扱われますが、PowerCenter 統合サービスではスペースとして扱われます。
- IBM DB2 データベースと PowerCenter 統合サービスでは、STDDEV および VARIANCE で結果が異なります。IBM DB2 では、STDDEV および VARIANCE の計算に、データベース以外の異なったアルゴリズムが使用されます。
- SYSDATE または SYSTIMESTAMP をデータベースにプッシュした場合、データベースサーバーによりタイムスタンプが、PowerCenter 統合サービスでなく、データベースサーバーのタイムゾーンで返されます。
- SYSTIMESTAMP を IBM DB2 または Sybase データベースにプッシュし、SYSTIMESTAMP の形式を指定した場合、データベースではその形式は無視され、完全なタイムスタンプが返されます。

- TO_DATE 関数を IBM DB2 データベースにプッシュするには、日付形式を「YYYYMMDD」のように指定します。

PowerExchange for Amazon Redshift

- TRUNC(DATE)を Amazon Redshift にプッシュするには、日付および形式引数を定義する必要があります。定義しないと、PowerCenter 統合サービスはこの関数を Amazon Redshift にプッシュしません。
- Amazon Redshift の集計関数が受け取る引数は 1 つだけで、それは集計関数のフィールドセットです。フィルタ条件引数は認められません。また、出力にマップされているすべてのポートが、GROUP BY 句に指定されていることを確認します。
- Amazon Redshift の場合、TO_DATE()および TO_CHAR()に文字列引数のみを定義すると、PowerCenter 統合サービスでは、デフォルトの日付形式はセッションプロパティにあると解釈されます。セッションプロパティのデフォルトの日付形式は MM/DD/YYYY HH24:MI:SS.US です。
- SYSTIMESTAMP()に形式を指定して、SYSTIMESTAMP を Amazon Redshift にプッシュしないでください。Amazon Redshift データベースは完全なタイムスタンプを返します。
- INSTR()を Amazon Redshift にプッシュするには、文字列、search_value、および開始引数のみを定義します。Amazon Redshift では、オカレンスおよび comparison_type 引数はサポートされていません。
- TO_BIGINT および TO_INTEGER を Amazon Redshift にプッシュすると、フラグ引数は無視されます。
- IN()を Amazon Redshift にプッシュすると、CaseFlag 引数は無視されます。
- ADD_TO_DATE()関数の一部として NS 形式を使用する場合、PowerCenter 統合サービスではこの関数が Amazon Redshift にプッシュされません。
- 次の形式のいずれかを TO_CHAR()および TO_DATE()関数の一部として使用する場合、PowerCenter 統合サービスではこれらの関数が Amazon Redshift にプッシュされません。
 - NS
 - SSSS
 - SSSSS
 - RR
- TRUNC(DATE)および DATE_DIFF()を Amazon Redshift にプッシュするには、次の形式を使用する必要があります。
 - D
 - HH24
 - MI
 - MM
 - MS
 - SS
 - US
 - YYYY
- GET_DATE_PART()を Amazon Redshift にプッシュするには、次の形式を使用する必要があります。
 - D
 - DDD
 - HH24
 - MI
 - MM

- MS
- SS
- US
- YYYY

PowerExchange for Netezza

- SYSTIMESTAMP('SS')は Netezza データベースにプッシュできますが、SYSTIMESTAMP('MS')と SYSTIMESTAMP('US')はプッシュできません。
- TO_CHAR(DATE)または TO_DATE()を Netezza にプッシュする場合、サブ秒精度の日付は YYYY-MM-DD HH24:MI:SS.US 形式である必要があります。形式が異なる場合、PowerCenter 統合サービスはこの関数を Netezza にプッシュしません。

PowerExchange for Vertica

- DATE_DIFF()関数の一部として以下の形式のいずれかを使用する場合、PowerCenter 統合サービスは関数を Vertica にプッシュしません。
 - YYYY
 - MON
 - MONTH
 - HH12
 - HH24
- DATE_DIFF 関数を Vertica にプッシュすると、Vertica は日付の差違値を最も近い整数に丸めます。ただし、PowerCenter 統合サービスは、浮動小数点値を返します。例えば、最初のデータが 2000-08-15 で、2 番目のデータが 1997-08-16 の場合、Vertica は、日付の差違値を 3 に丸めますが、PowerCenter 統合サービスは 2.99731182795699 を返します。日付の差違値を Vertica データベースで浮動小数点値として扱う場合は、プッシュダウンの最適化を無効にすることができます。
- 形式に Y を指定し、DATE_DIFF 関数を Vertica にプッシュした場合、Vertica は日付の差違を日数で計算します。ただし、PowerCenter 統合サービスは、差違を年数で計算します。差違値を年数で扱う場合は、プッシュダウンの最適化を無効にすることができます。

PowerExchange for Greenplum

- TRUNC(DATE)を Greenplum にプッシュするには、次の形式を使用する必要があります。
 - YYYY
 - DD
 - DOY
 - HH
 - US
 - MS
 - MI
 - MM
 - SS

PowerExchange for Snowflake

- TRUNC(DATE)関数または TO_CHAR()関数を Snowflake データベースにプッシュするには、日付と形式の引数を定義する必要があります。

- Snowflake 集計関数は、引数を 1 つだけとります（集計関数のフィールドセット）。PowerCenter 統合サービスは、引数に定義されたフィルタ条件を無視します。ターゲットにマップされているすべてのフィールドが GROUP BY 句に指定されているようにします。
- SYSTIMESTAMP()関数または SYSDATE()関数を Snowflake データベースにプッシュするときには、どの形式も指定しないでください。Snowflake データベースは完全なタイムスタンプを返します。
- 複数の引数を指定した TO_BIGINT()関数または TO_INTEGER()関数を Snowflake データベースにプッシュすることはできません。
- REPLACECHR()関数または REPLACESTR()関数を Snowflake データベースにプッシュすると、PowerCenter 統合サービスは caseFlag 引数を無視します。
例えば、REPLACECHR(false, in_F_CHAR, 'a', 'b')引数と REPLACECHR(true, in_F_CHAR, 'a', 'b')引数のどちらも同じ値を返します。
- 関数を Snowflake データベースにプッシュするときには、ミリ秒値とマイクロ秒値を使用できません。
- ADD_TO_DATE()関数と TRUNC(DATE)関数では、ナノ秒値を使用できます。
- TRUNC(DATE)、GET_DATE_PART()、DATE_DIFF()の各関数を Snowflake データベースにプッシュするには、以下の時刻形式を引数として使用する必要があります。
 - D
 - DDD
 - HH
 - MI
 - MM
 - SS
 - YYYY

例えば、TRUNC(<datefieldname>, 'dd')。

日付と時刻に関する関数についての詳細は、次のウェブサイトを参照してください:

<https://docs.snowflake.net/manuals/sql-reference/functions-date-time.html#label-supported-date-time-parts>

エラー処理、ロギング、およびリカバリ

Integration Service とデータベースでは、エラー処理、ロギング、およびリカバリが異なる方法で処理されません。

エラー処理

Integration Service により、トランスフォーメーションロジックがデータベースにプッシュされる場合、そのデータベースで発生するエラーは追跡できません。結果として、Integration Service は、セッションのトランスフォーメーションを処理する場合とは別の方法でエラーを処理します。Integration Service により完全なプッシュダウンの最適化に設定されたセッションが実行されている際にエラーが発生した場合、そのエラーはデータベースで処理されます。エラーがデータベースで処理された場合、Integration Service では拒否された行は拒否ファイルに書き込まれません。

ロギング

Integration Service により、トランスフォーメーションロジックがデータベースにプッシュされる場合、データベースサーバー内で発生するすべてのイベントは追跡することができません。Integration Service が追跡できる統計は、プッシュダウンの最適化のタイプによって異なります。トランスフォーメーションロジックをデータベースにプッシュした場合、Integration Service で生成されるログのそれぞれに次のような差異が生じます。

- セッションログには、データベースで処理されたトランスフォーメーションの詳細情報が含まれない。
- セッションが完全なプッシュダウンの最適化に設定されている場合、セッションログにスレッドビジー率が含まれない。
- Integration Service により完全なプッシュダウンの最適化が使用されており、すべてのトランスフォーメーションロジックがデータベースにプッシュされる場合、ソースから読み込まれた行数がセッションログに含まれない。
- Integration Service によりソース側でのプッシュダウンの最適化が使用されている場合、最適化されたソースから読み込まれた行数がセッションログに含まれる。

リカバリ

セッションが完全なプッシュダウンの最適化に設定されていて、セッションが失敗した場合、トランスフォーメーションがデータベースによって処理されるため、Integration Service では増分リカバリ実行できません。代わりに、データベースによってトランザクションがロールバックされます。データベースサーバーに障害が発生すると、データベースサーバーはリスタート時にトランザクションをロールバックします。Integration Service に障害が発生すると、データベースサーバーはトランザクションをロールバックします。

Integration Service で、一時シーケンスオブジェクトまたはビューをデータベース内に作成しているとき、行をいっさい処理しないうちに障害が発生した場合、生成された SQL がデータベースに対して再度実行されません。

すべての行の処理が未完了のうちに障害が発生した場合、次のタスクが Integration Service で実行されます。

1. 適用可能な場合、Integration Service は一時ビューまたはシーケンスオブジェクトをデータベースから削除またはデータベース内に再作成して、重複値が生成されないようにします。
2. Integration Service は、生成された SQL をデータベースに対して再度実行します。

Integration Service で一時ビューまたはシーケンスオブジェクトをデータベースから削除しているとき、すべての行の処理後に障害が発生した場合、一時オブジェクトの削除が再度試みられます。

緩やかに変化する次元に関する作業

タイプ 1 およびタイプ 3 の緩やかに変化する次元ロジックをデータベースにプッシュできます。マッピング内の緩やかに変化する次元ロジックは、複数のトランスフォーメーションから構成できます。各トランスフォーメーションのルールおよびガイドラインに応じて、緩やかに変化する次元ロジックをどれだけデータベースにプッシュできるかが決定されます。

緩やかに変化する次元のトランスフォーメーションロジックをデータベースにプッシュするように Integration Service を設定する際には、次のルールおよびガイドラインを使用してください。

- タイプ 1 およびタイプ 3 の緩やかに変化する次元マッピングに含まれているトランスフォーメーションは、Oracle データベースまたは IBM DB 2 データベースにプッシュできます。
- ソースデータに重複行があってはなりません。複数の更新を同じ行に対して行った場合、データベースがデッドロック状態になる可能性があります。

- 緩やかに変化する次元マッピングは、バージョン 8.5 以降の緩やかに変化する次元のウィザードを使用して作成する必要があります。緩やかに変化する次元ロジックは、以前のバージョンの緩やかに変化する次元ウィザードで作成されたものである場合、データベースにプッシュできません。

シーケンスおよびビューに関する作業

トランスフォーメーションロジックをデータベースにプッシュするために、Integration Service では一時シーケンスまたは一時ビューがデータベース内に作成されます。データベーストランザクションが完了した後は、Integration Service でプッシュダウンの最適化用に作成されたシーケンスオブジェクトおよびビューオブジェクトが削除されます。

シーケンス

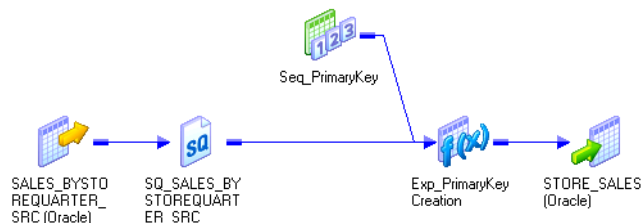
シーケンスジェネレータートランスフォーメーションロジックをデータベースにプッシュするには、セッションをシーケンスでプッシュダウンの最適化に設定する必要があります。

シーケンスジェネレータートランスフォーメーションロジックをデータベースにプッシュするようにセッションを設定した場合、Integration Service では以下のタスクが実行されます。

1. **データベース内へのシーケンスオブジェクトの作成。** Integration Service により、シーケンスジェネレータートランスフォーメーションロジックに基づいてシーケンスオブジェクトがデータベース内に作成されます。Integration Service ではシーケンスオブジェクトごとに一意の名前が作成されます。一意のシーケンスオブジェクト名を作成するために、ハッシュ関数によって生成された値に接頭語 PM_S が追加されます。
2. **SQL クエリの生成とデータベースに対する SQL クエリの実行。** Integration Service により、シーケンスジェネレータートランスフォーメーションロジックをデータベースにプッシュする SQL クエリが生成され、実行されます。
3. **データベースからのシーケンスオブジェクトの削除。** トランザクションが完了したときに、Integration Service によってデータベース内に作成されたシーケンスオブジェクトが削除されます。

シーケンスの作成例

シーケンスジェネレータートランスフォーメーションを使用してリレーショナルターゲット用プライマリキーを生成する、次のようなマッピングを作成します。



Integration Service が、トランスフォーメーションロジックをデータベースにプッシュする際、次の SQL 文を実行してソースデータベース内にシーケンスオブジェクトを作成します。

```
CREATE SEQUENCE PM_S6UHW420GXTY7NICHYIOSRMC5XQ START WITH 1 INCREMENT BY 1 MINVALUE 0 MAXVALUE 9223372036854775807 NOCYCLE CACHE 9223372036854775807
```

Integration Service でシーケンスオブジェクトが作成された後、マッピング内に含まれるトランスフォーメーションロジックを処理する SQL クエリが Integration Service によって実行されます。

```
INSERT INTO STORE_SALES(PRIMARYKEY, QUARTER, SALES, STORE_ID) SELECT
CAST(PM_S6UHW420GXTY7NICHYIOSRMC5XQ.NEXTVAL AS FLOAT), CAST(CAST(SALES_BYSTOREQUARTER_SRC.QUARTER AS FLOAT) AS
VARCHAR2(10)), CAST(CAST(SALES_BYSTOREQUARTER_SRC.SALES AS NUMBER(10, 2)) AS NUMBER(25, 2)),
CAST(SALES_BYSTOREQUARTER_SRC.STORE_ID AS NUMBER(0, 0)) FROM SALES_BYSTOREQUARTER_SRC
```

セッション完了後、Integration Service によってデータベースからシーケンスオブジェクトが削除されます。セッションが失敗した場合、Integration Service はシーケンスオブジェクトをいったん削除して再作成した後、リカバリタスクを実行します。

ビュー

Integration Service がデータベース内にビューオブジェクトを作成できるように、セッションをビューとともにプッシュダウンの最適化に設定する必要があります。

Integration Service では、次の条件下でビューオブジェクトが作成されます。

- SQL オーバーライドで設定されたソース修飾子またはルックアップトランスフォーメーション用にプッシュダウンの最適化を設定します。
- フィルタで設定されたルックアップトランスフォーメーション用にプッシュダウンの最適化を設定します。
- コネクトされていないルックアップトランスフォーメーション用にプッシュダウンの最適化を設定します。

Integration Service がソース修飾子またはルックアップトランスフォーメーションをデータベースにプッシュする際、トランスフォーメーション定義に基づいてビューが作成されます。たとえば、Integration Service がフィルタを使用するルックアップトランスフォーメーションに基づいてビューを作成する場合、フィルタされていない行のみを含むビューが作成されます。Integration Service が SQL オーバーライドを使用するルックアップトランスフォーメーションをデータベースにプッシュすると、投影されたルックアップポートだけではなく、すべてのルックアップポートに基づいてビューが作成されます。

Integration Service では、SQL オーバーライドはパースまたは検証されません。SQL オーバーライドを使用するソース修飾子トランスフォーメーションまたはルックアップトランスフォーメーションをデータベースにプッシュするようにセッションを設定する場合は、SQL オーバーライドをそのデータベースに対してテストしてから、セッションを実行してください。

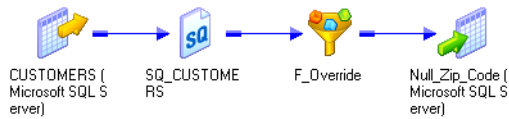
ソース修飾子トランスフォーメーションのロジックを一時ビューとともに Teradata にプッシュする場合、Teradata のデータディクショナリにより SQL 文が失敗することがあります。多数のプッシュダウン最適化セッションを使用する環境でビューの動的な作成および削除を行うと、SQL 文は失敗します。ソース修飾子トランスフォーメーションにソースフィルタ、ユーザー定義の結合、SQL オーバーライドのいずれかが含まれている場合は、Teradata に対するプッシュダウンの最適化の一時ビューの作成は無効にできます。Integration Service が、ビューの代わりに派生テーブルを作成します。

セッションをビューとともにプッシュダウンの最適化に設定した場合、Integration Service では以下のタスクが実行されます。

1. **データベース内でのビューの作成。** Integration Service によって、ルックアップフィルタ、接続されていないルックアップ、あるいはソース修飾子トランスフォーメーションまたはルックアップトランスフォーメーションでの SQL オーバーライドに基づいて、データベース内にビューが作成されます。一意のビュー名を作成するために、Integration Service によって、ハッシュ関数から生成された値にプレフィックス PM_V が追加されます。
2. **ビューに対する SQL クエリの実行。** Integration Service によってビューオブジェクトが作成された後、データベース内に作成されたビューに対して、トランスフォーメーションロジックをソースにプッシュする SQL クエリが実行されます。
3. **データベースからのビューの削除。** トランザクションが完了した時に、Integration Service によって、作成したビューが削除されます。

ビューの作成例

カスタマデータベース内の 94117 郵便番号を検索する、次のマッピングを作成します。



Johnson という名前のバリエーション（Johnsen、Jonssen、Jonson など）に一致する名前の顧客を返す検索を行います。名前の照合を実行するには、ソース修飾子トランスフォーメーション用の SQL オーバーライドを入力します。

```
SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY, CUSTOMERS.FIRST_NAME, CUSTOMERS.LAST_NAME,
CUSTOMERS.ADDRESS1, CUSTOMERS.ADDRESS2, CUSTOMERS.CITY, CUSTOMERS.STATE, CUSTOMERS.POSTAL_CODE,
CUSTOMERS.PHONE, CUSTOMERS.EMAIL FROM CUSTOMERS WHERE CUSTOMERS.LAST_NAME LIKE 'John%' OR CUSTOMERS.LAST_NAME
LIKE 'Jon%'
```

Integration Service は、このセッションのトランスフォーメーションロジックをデータベースにプッシュする場合に、次の SQL 文を実行してソースデータベースにビューを作成します。

```
CREATE VIEW PM_V4RZRW5GWCKUEWH35RKDMDPRNXI (CUSTOMER_ID, COMPANY, FIRST_NAME, LAST_NAME, ADDRESS1, ADDRESS2,
CITY, STATE, POSTAL_CODE, PHONE, EMAIL) AS SELECT CUSTOMERS.CUSTOMER_ID, CUSTOMERS.COMPANY,
CUSTOMERS.FIRST_NAME, CUSTOMERS.LAST_NAME, CUSTOMERS.ADDRESS1, CUSTOMERS.ADDRESS2, CUSTOMERS.CITY,
CUSTOMERS.STATE, CUSTOMERS.POSTAL_CODE, CUSTOMERS.PHONE, CUSTOMERS.EMAIL FROM CUSTOMERS WHERE
CUSTOMERS.LAST_NAME LIKE 'John%' OR CUSTOMERS.LAST_NAME LIKE 'Jon%'
```

Integration Service でビューが作成された後、マッピング内のトランスフォーメーションロジックを実行する SQL クエリーが実行されます。

```
SELECT PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.CUSTOMER_ID, PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.COMPANY,
PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.FIRST_NAME, PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.LAST_NAME,
PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.ADDRESS1, PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.ADDRESS2,
PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.CITY, PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.STATE,
PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.POSTAL_CODE, PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.PHONE,
PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.EMAIL FROM PM_V4RZRW5GWCKUEWH35RKDMDPRNXI WHERE
(PM_V4RZRW5GWCKUEWH35RKDMDPRNXI.POSTAL_CODE = 94117)
```

セッション完了後、Integration Service によってデータベースからビューが削除されます。セッションが失敗した場合、Integration Service はビューをいったん削除して再作成した後、リカバリタスクを実行します。

孤立シーケンスおよびビューのトラブルシューティング

Integration Service、セッション、または接続に障害が発生した場合、Integration Service によってデータベースからシーケンスオブジェクトまたはビューオブジェクトは削除されません。この場合、これらのオブジェクトをデータベースから手動で削除してください。

注: データベース内の孤立シーケンスオブジェクトおよびビューオブジェクトは、パフォーマンスに影響を及ぼしません。

孤立シーケンスオブジェクトまたはビューオブジェクトをデータベースから削除するには、次の作業を完了します。

1. **データベース内の孤立オブジェクトの特定。** データベースのセッションログまたはクエリに基づいて、孤立オブジェクトを特定できます。セッション実行から孤立オブジェクトを特定する場合は、セッションログを解析します。特定時点でデータベース内の孤立オブジェクトをすべて判別する場合は、データベースクエリーを実行します。
2. **データベースからの孤立オブジェクトの削除。** 特定した孤立オブジェクトは、SQL 文を実行して削除できます。

セッションログを使用した孤立したオブジェクトの特定

Integration Service によって、ビューオブジェクトまたはシーケンスオブジェクトの作成および削除時に、イベントログが書き込まれます。セッション実行中に、Integration Service、セッション、または接続性に障害が発生した場合、セッションログを調べて、セッション中に削除されたシーケンスオブジェクトまたはビューオブジェクトを指定することができます。

例えば、Integration Service でビュー PM_V4RZRW が削除された場合、次のようなメッセージがセッションログに表示されます。

```
MAPPING> TM_6356 SQL のプッシュダウンクリーンアップ（ソース：CUSTOMERS）を開始します。：(Tue Feb 14 13:23:46 2006)
```

```
MAPPING> TM_6358 SQL のプッシュダウンクリーンアップ（ソース：DROP VIEW PM_V4RZRW）を実行しています。
```

```
MAPPING> TM_6360 SQL のプッシュダウンクリーンアップ（ソース：CUSTOMERS）が正常に完了しました。：(Tue Feb 14 13:23:46 2006)]
```

SQL クエリを使用した孤立オブジェクトの特定

Integration Service でシーケンスオブジェクトまたはビューオブジェクトが削除されない場合、データベースに対して SQL クエリを実行し、Integration Service で作成された孤立シーケンスオブジェクトまたはビューオブジェクトをすべて特定できます。Integration Service で複数のセッションが実行された場合、または複数の Integration Service で同じデータベースアカウントに書き込まれた場合、実行され、シーケンスオブジェクトまたはビューオブジェクトを削除しなかった各セッションからすべての孤立オブジェクトが SQL クエリによって返されます。

Integration Service でシーケンスオブジェクトまたはビューオブジェクトがデータベース内に作成された場合、シーケンスオブジェクトの名前にプレフィックス PM_S、ビューオブジェクトの名前に PM_V が追加されます。これらのオブジェクトは、そのオブジェクトを特定するプレフィックスに基づいて検索できます。

以下のクエリに、Integration Service で作成されたシーケンスオブジェクトを検索する構文を示します。

IBM DB2：

```
SELECT SEQNAME FROM SYSCAT.SEQUENCES
WHERE SEQSCHEMA = CURRENT SCHEMA
AND SEQNAME LIKE 'PM\_S%' ESCAPE '\u2019
```

Oracle：

```
SELECT SEQUENCE_NAME FROM USER_SEQUENCES
WHERE SEQUENCE_NAME LIKE 'PM\_S%' ESCAPE '\u2019
```

以下のクエリに、Integration Service で作成されたビューオブジェクトを検索する構文を示します。

IBM DB2：

```
SELECT VIEWNAME FROM SYSCAT.VIEWS
WHERE VIEWSCHEMA = CURRENT SCHEMA
AND VIEW_NAME LIKE 'PM\_V%' ESCAPE '\u2019
```

Oracle：

```
SELECT VIEW_NAME FROM USER_VIEWS
WHERE VIEW_NAME LIKE 'PM\_V%' ESCAPE '\u2019
```

Microsoft SQL Server または Sybase ASE：

```
SELECT NAME FROM SYSOBJECTS
WHERE TYPE='V' AND NAME LIKE 'PM\_V%' ESCAPE '\u2019
```

Teradata：

```
SELECT TableName FROM DBC.Tables
WHERE CreatorName = USER
```

```
AND TableKind ='V'  
AND TableName LIKE 'PM\_V%' ESCAPE '\\u2019
```

孤立したオブジェクトの削除

Integration Service で作成されたシーケンスオブジェクトまたはビューオブジェクトのリストを取得した後、SQL DROP 文を実行して、シーケンスオブジェクトまたはビューオブジェクトをデータベースから削除することができます。

以下のクエリは、任意のデータベース上の Integration Service で作成されたシーケンスオブジェクトを削除する構文を示しています。

```
DROP SEQUENCE <sequence name>
```

以下のクエリは、任意のデータベース上の Integration Service で作成されたビューオブジェクトを削除する構文を示しています。

```
DROP VIEW <view name>
```

\$\$PushdownConfig マッピングパラメータの使用

データベース負荷に応じて、ソース側、ターゲット側、または完全なプッシュダウンの最適化を時と場合によって使い分けることができます。例えば、1 日のうちピーク時間ではソース側またはターゲット側のプッシュダウン最適化を使用し、データベースアクティビティが低下する深夜 0 時から午前 2 時までは完全なプッシュダウンの最適化を使用します。

プッシュダウンの最適化を時と場合によって使い分けるには、\$\$PushdownConfig マッピングパラメータを使用します。このパラメータを使用して、セッションを様々なタイプのプッシュダウンの最適化で実行することができます。\$\$PushdownConfig パラメータの設定によって、セッションプロパティの最適化設定がオーバーライドされます。

次の手順に従って、マッピングパラメータを設定します。

1. Mapping Designer で\$\$PushdownConfig を作成します。
2. Mapping Designer で\$\$PushdownConfig マッピングパラメータを追加する場合は、以下の値を使います。

フィールド	値
名前	\$\$PushdownConfig
タイプ	パラメータ
データ型	文字列
精度または位取り	20
集計	なし
初期値	なし
説明	オプション

3. セッションを設定するとき、プッシュダウンの最適化属性として\$\$PushdownConfig を選択します。

4. パラメータファイルのパラメータを定義します。
5. パラメータファイルの\$\$PushdownConfigとして以下の値のいずれかを入力します。

値	説明
なし	Integration Service によりセッションのすべてのトランスフォーメーションロジックが処理されます。
Source [Seq View Conn]	Integration Service によりトランスフォーメーションロジックが可能な限りソースデータベースへプッシュされます。
Target [Seq View Conn]	Integration Service によりトランスフォーメーションロジックが可能な限りターゲットデータベースへプッシュされます。
Full [Seq View Conn]	Integration Service によりトランスフォーメーションロジックが可能な限りソースデータベースとターゲットデータベースへプッシュされます。Integration Service は、データベースにプッシュできなかったトランスフォーメーションロジックを処理します。

オプションとして、以下のオプションを 1 つ以上指定します。

- **Seq。** Integration Service でデータベース内にシーケンスオブジェクトを作成できるようにします。
- **View。** Integration Service でデータベース内にビューオブジェクトを作成できるようにします。
- **Conn。** 動作可能なデータベースのデータベースユーザーが、動作可能なデータベースにトランスフォーメーションロジックをプッシュするのに必要な、アイドル状態のデータベースに対して読み取り権限を持っていることを示します。

例えば、「Full View Conn」と入力して完全なプッシュダウンの最適化を使用した場合、動作可能なデータベース内にビューオブジェクトの作成が有効になり、動作可能なデータベースにアイドル状態のデータベースに対する読み取り権限があることを示します。

プッシュダウンの最適化のためのセッションの設定

セッションプロパティでセッションをプッシュダウンの最適化に設定します。ただし、より多くのトランスフォーメーションロジックをデータベースに渡すために、トランスフォーメーション、マッピング、またはセッションの設定を編集することが必要になる場合があります。データベースに渡すことのできるトランスフォーメーションを、プッシュダウンの最適化ビューアを使用して調べます。

プッシュダウンオプション

セッションプロパティで設定できるプッシュダウン最適化オプションは、次のとおりです。

- **プッシュダウンの最適化。** プッシュダウンの最適化のタイプ。\$\$PushdownConfig マッピングパラメータを使用する場合は、マッピングパラメータを設定したこと、およびその値をパラメータファイル内に定義したことを確認します。
- **プッシュダウンに対する一時ビューの許可。** PowerCenter Integration Service によりセッションがデータベースにプッシュされる場合に、PowerCenter Integration Service でデータベースに一時ビューオブジェクトを作成できます。セッションに、ソース修飾子トランスフォーメーションまたはルックアップトランスフォーメーションの SQL オーバーライド、フィルタリングされたルックアップ、あるいは接続されてい

ないルックアップが含まれている場合、PowerCenter Integration Service によりデータベースにビューが作成されます。

Teradata ソースを使用し、ソースフィルタ、ユーザー定義の結合、SQL オーバーライドのいずれかがソース修飾子トランスフォーメーションに含まれている場合、一時プッシュダウンビューを許可する必要はありません。ソース修飾子トランスフォーメーションのロジックを一時ビューとともに Teradata にプッシュする場合、Teradata のデータディクショナリにより SQL 文が失敗することがあります。多数のプッシュダウン最適化セッションを使用する環境でビューの動的な作成および削除を行うと、SQL 文は失敗します。

- **プッシュダウンに対する一時シーケンスの許可。**PowerCenter Integration Service でデータベースに一時シーケンスオブジェクトを作成できるようにします。セッションにシーケンスジェネレータトランスフォーメーションが含まれている場合、PowerCenter Integration Service がデータベースにシーケンスオブジェクトを作成する必要があります。
- **ユーザー互換のない接続に対するプッシュダウンの許可。**動作可能なデータベースのデータベースユーザーが、アイドル状態のデータベースに対して読み取り権限があることを示します。動作可能なデータベースのデータベースユーザーが、アイドル状態のデータベースに対して読み取り権限があることを示しているが、実際にはない場合、セッションは失敗します。アクティブ状態のデータベースのデータベースユーザーにアイドル状態のデータベースの読み取り権限があることを示さない場合、トランスフォーメーションロジックはデータベースにプッシュされません。

プッシュダウンの最適化ビューアを使用して、マッピング、トランスフォーメーション、またはセッションの設定を編集して、より多くのトランスフォーメーションロジックをデータベースにプッシュする必要があるかどうかを決定します。プッシュダウンの最適化ビューアには、ソース側、ターゲット側または完全なプッシュダウンの最適化を使用してトランスフォーメーションロジックをデータベースにプッシュできるかどうかが表示されます。トランスフォーメーションロジックをデータベースにプッシュできる場合は、データベースにプッシュできるトランスフォーメーションの全一覧がプッシュダウンの最適化ビューアに表示されます。

プッシュダウンの最適化ビューアでプッシュダウンオプションまたはプッシュダウングループを選択し、指定の選択範囲に対して生成された対応する SQL 文を表示することもできます。

注: プッシュダウンオプションまたはプッシュダウングループを選択した場合は、プッシュダウンの設定を変更しません。設定を変更するには、セッションプロパティでプッシュダウンオプションを更新する必要があります。

パーティション化

パーティションタイプがパススルーパーティション化またはキー範囲パーティション化の場合、複数のパーティションを持つセッションをデータベースにプッシュできます。

パススルーパーティション化用のプッシュダウン最適化

パススルーパーティション化を使用するセッションをプッシュダウン最適化の設定にした場合、データベースはパーティション間に行を再配分することなしにデータを処理します。ある 1 つのパーティションに置かれた行はすべて、パススルーパーティションポイントを通過した後も同じパーティションにとどまります。

トランスフォーメーションロジックをデータベースにプッシュするには、すべてのパーティションポイントをパススルーパーティション化の設定にする必要があります。たとえば、セッションに 4 つのパーティションポイントがあるとします。最初の 3 つのパーティションポイントをパススルーパーティション化の設定にし、最後のパーティションポイントを自動ハッシュキーパーティション化の設定にします。トランスフォーメーションロジック（最後のパーティションポイント以降のトランスフォーメーション以外すべて）は、Integration Service によってデータベースにプッシュされます。最後のパーティションポイント以降のトランスフォーメーションは、Integration Service によって処理されます。

キー範囲パーティション化用のプッシュダウン最適化

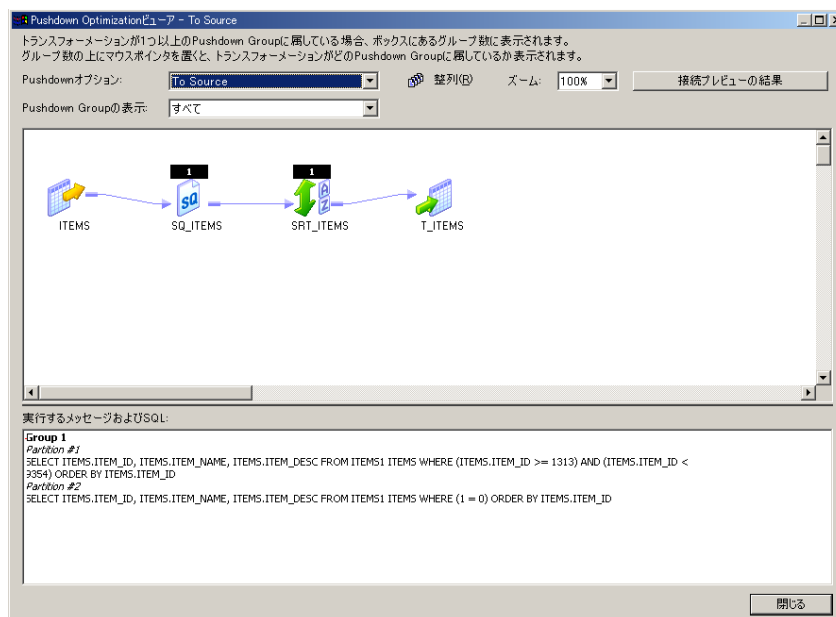
ソース修飾子トランスフォーメーションでキー範囲パーティション化を使用するセッションをプッシュダウン最適化の設定にした場合、Integration Service はすべての行を最初のパーティションにマージし、後続の各パーティションに対しては空のデータを渡します。Integration Service では各パーティション用に SQL 文が作成されます。Integration Service により、一部のトランスフォーメーションロジックのみがデータベースにプッシュされる場合、セッションの実行時にはパーティション間に行が再分散されません。

Integration Service ですべてのトランスフォーメーションロジックをデータベースにプッシュできるようにするには、セッションが以下の基準を満たしている必要があります。

- すべての行を最初のパーティションにマージするには、各パーティションの終了キー範囲が次のパーティションの開始範囲と等しくなければならない。終了キー範囲が、次のパーティションと重なり合うことはできない。たとえば、1 つめのパーティションの終了レンジが 3386 の場合、2 つめのパーティションの開始レンジは 3386 である必要があります。
- ソース修飾子トランスフォーメーションでパーティションポイントをキー範囲パーティション化が使用されるように設定し、後続のすべてのパーティションポイントをハッシュ自動キーパーティション化またはパススルーパーティション化のどちらかが使用されるように設定する必要があります。

複数のパーティションを持つセッションのプッシュダウンの最適化の例

以下の図に、ソータトランスフォーメーションとハッシュ自動キーパーティション化を持つマッピングを示します。



1 つめのキー範囲は 1313 - 9340 で、2 つめのキー範囲は 9340 - 9354 です。次の SQL 文は、すべてのデータを 1 つめのパーティションに統合します。

```
SELECT ITEMS.ITEM_ID, ITEMS.ITEM_NAME, ITEMS.ITEM_DESC FROM ITEMS1 ITEMS WHERE (ITEMS.ITEM_ID >= 1313) AND (ITEMS.ITEM_ID < 9354) ORDER BY ITEMS.ITEM_ID
```

この SQL 文は品目 1313 - 9354 を選択します。これにはキー範囲内のすべての値が含まれています。この SQL 文は、両方のパーティションからのデータを 1 つめのパーティションに統合します。

2 つめのパーティションの SQL 文は空データを渡します。

```
SELECT ITEMS.ITEM_ID, ITEMS.ITEM_NAME, ITEMS.ITEM_DESC FROM ITEMS1 ITEMS WHERE (1 = 0) ORDER BY ITEMS.ITEM_ID
```

複数のパーティションを持つセッションに関するルールおよびガイドライン

複数のパーティションを使用するセッションをデータベースにプッシュするように Integration Service を設定する際には、次のルールおよびガイドラインを使用してください。

Integration Service で複数のパーティションを使用するセッションをデータベースにプッシュできるのは、次の場合です。

- セッションにより、ソース修飾子トランスフォーメーションのパーティションポイントおよび後続のすべてのパーティションポイントでパススルーパーティション化が使用される場合、Integration Service ではソース側、ターゲット側、または完全なプッシュダウンの最適化が使用されトランスフォーメーションロジックがデータベースにプッシュされます。
- セッションにより、ソース修飾子トランスフォーメーションでキー範囲パーティション化が使用され、ダウストリームパーティションポイントにハッシュ自動キーパーティションまたはパススルーパーティションが含まれる場合、Integration Service ではソース側または完全なプッシュダウンの最適化が使用されトランスフォーメーションロジックがデータベースにプッシュされます。

プッシュダウンの最適化でトランスフォーメーションのパーティションからのデータが最初のパーティションにマージされ、ダウストリームトランスフォーメーションのトランスフォーメーションロジックが Integration Service によって処理された場合、ダウストリームトランスフォーメーション内のパーティション間に行が再配分されません。引き続き最初のパーティションに行が渡され、他のパーティションに空のデータが渡されます。

ターゲットロードルール

ターゲットロードルールは、セッションをデータベースにプッシュできるかどうかに影響を与える場合があります。

次の表に、各種のターゲットロードオプションに対応するプッシュダウンの最適化を示します。

ターゲットオプション	ソース	ターゲット	完全
挿入	X	X	X
削除	X	X	X
更新（更新として）	X	X	X
更新（挿入として）	X	X	X
更新（でなければ挿入）	X	X	はい/いいえ

ターゲットロードロジックをデータベースにプッシュするように Integration Service を設定する際には、以下のルールおよびガイドラインを使用します。

- 完全なプッシュダウンの最適化を使用してもパフォーマンス向上が達成されず、ソース行を削除または更新として扱う場合は、ソース側のプッシュダウン最適化を使用してください。
- セッションに共有体トランスフォーメーションが含まれていて、Integration Service によってトランスフォーメーションロジックが Sybase データベースに渡される場合、完全なプッシュダウンの最適化が使用できないため、ソース行を削除または更新として扱うことになります。

プッシュダウングループの表示

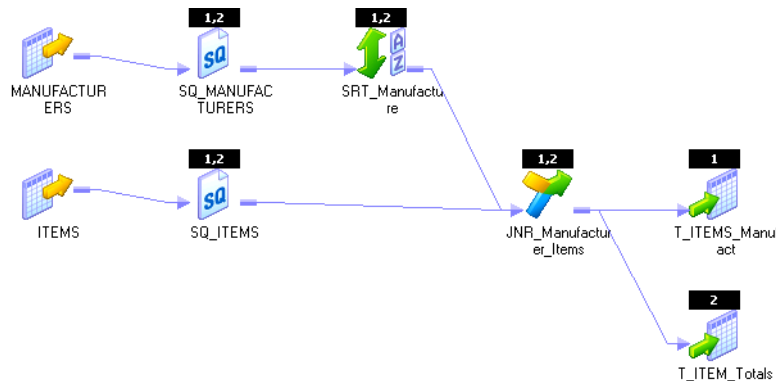
セッションをプッシュダウンの最適化に設定した場合、Integration Service によりトランスフォーメーションロジックに基づいて SQL 文が生成されます。1 つの SQL 文として処理できるトランスフォーメーションのグループは、プッシュダウングループと呼ばれます。

トランスフォーメーションロジックをデータベースにプッシュする場合、パイプライン、ソース、およびターゲットの数と使用するプッシュダウンの最適化のタイプに応じて、Integration Service により複数のプッシュダウングループが作成される場合があります。セッションに複数のパーティションがある場合、Integration Service によりグループ内のパーティションごとに SQL 文が実行されます。パイプラインを結合した場合、各パイプラインのトランスフォーメーションは 1 つのプッシュダウングループに統合されます。同じトランスフォーメーションが 2 つ以上のターゲットに渡されるトランスフォーメーションロジックの一部である場合、そのトランスフォーメーションは、各ターゲットのプッシュダウングループの一部です。

プッシュダウングループは、プッシュダウンの最適化ビューアを使用して表示できます。プッシュダウンの最適化ビューアでプッシュダウングループを表示した場合、データベースにプッシュできるトランスフォーメーションと、Integration Service で処理されるトランスフォーメーションを特定できます。プッシュダウンの最適化ビューアに表示されるメッセージを基に、より多くのトランスフォーメーションロジックがデータベースにプッシュされるようにトランスフォーメーションまたはマッピングを編集する方法を決めることもできます。マッピング変数を使用する場合、またはグリッド上で実行されるようにセッションを設定する場合、プッシュダウンの最適化ビューアは、セッション内で実行される SQL を表示できません。

生成された SQL を表示したときに、一時ビューオブジェクトおよびシーケンスオブジェクトの名前は、セッション中に生成されたビューオブジェクトおよびシーケンスオブジェクトの名前とは異なっています。Integration Service ではハッシュ関数を使用され、生成したシーケンスオブジェクトおよびビューオブジェクトごとに一意の名前が作成されます。

以下の図に、プッシュダウンの最適化ビューアに表示されるマッピングを示します。ソースデータベースとターゲットデータベースにプッシュできる 2 つのプッシュダウングループが含まれています。



パイプライン 1 とパイプライン 2 は異なるソースから発生しており、プッシュダウンの最適化に有効なトランスフォーメーションを含んでいます。Integration Service によりターゲットごとにプッシュダウングループが作成され、プッシュダウングループごとに SQL 文が生成されます。これら 2 つのパイプラインは結合されているため、ジョイナトランスフォーメーションに至るまでのトランスフォーメーション（ジョイナトランスフォーメーションを含む）は両方のパイプラインの一部であり、両方のプッシュダウングループに含まれます。

プッシュダウングループを表示するには、プッシュダウンの最適化ビューアを開きます。プッシュダウンの最適化ビューアでは、Integration Service でランタイム時に生成されるプッシュダウングループと SQL 文がレビューされます。

プッシュダウングループを表示する手順:

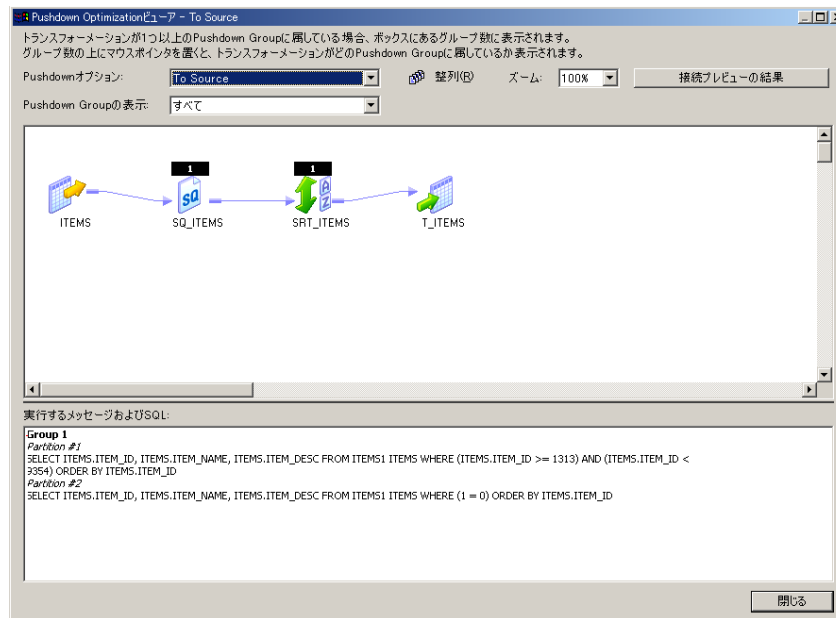
1. Workflow Manager で、プッシュダウンの最適化に設定されたセッションを開きます。

2. [マッピング] タブで、左ペインの [プッシュダウンの最適化]、または右ペインの [プッシュダウンの最適化の表示] を選択します。

プッシュダウンの最適化ビューアにプッシュダウングループと、各プッシュダウングループを構成しているトランスフォーメーションが表示されます。パイプラインで複数のパーティションを設定している場合、SQL 文はパーティションごとに表示されます。プッシュダウングループおよびプッシュダウンオプションごとに生成されたメッセージと SQL 文を表示できます。プッシュダウンオプションには、[None]、[To Source]、[To Target]、[Full]、および [\$\$PushdownConfig] があります。

以下の図に、ソースデータベースにプッシュできる 2 つのパーティションを持つ 1 つのパイプラインを含むマッピングを示します。

図 1. プッシュダウンの最適化ビューア



3. プッシュダウンの最適化ビューアで、プッシュダウンオプションを選択して、SQL 文をプレビューします。

ビューアのプッシュダウンオプションによって、実行時に発生する最適化は影響を受けません。セッションのプッシュダウンの最適化を変更するには、セッションプロパティを編集します。

4. 接続変数を使用するようにセッションを設定する場合は、[接続プレビューの結果] をクリックし、プレビューする接続値を選択します。

セッションが接続変数を使用する場合は、プッシュダウンの最適化ビューアを開くたびに接続値を選択する必要があります。選択した値は、Workflow Manager で保存されないため、Integration Service でランタイム時には使用されません。

SQL オーバーライドに\$\$\$SessStartTime 変数が含まれている場合、プッシュダウンの最適化をプレビューした際にプッシュダウンの最適化ビューアではこの変数は展開されません。

第 5 章

プッシュダウンの最適化およびトランスフォーメーション

この章では、以下の項目について説明します。

- [プッシュダウンの最適化およびトランスフォーメーションの概要, 117 ページ](#)
- [アグリゲータトランスフォーメーション, 119 ページ](#)
- [式トランスフォーメーション, 120 ページ](#)
- [フィルタトランスフォーメーション, 120 ページ](#)
- [ジョイナトランスフォーメーション, 121 ページ](#)
- [ルックアップトランスフォーメーション, 122 ページ](#)
- [ルータトランスフォーメーション, 125 ページ](#)
- [シーケンスジェネレータトランスフォーメーション, 126 ページ](#)
- [ソータトランスフォーメーション, 127 ページ](#)
- [ソース修飾子トランスフォーメーション, 128 ページ](#)
- [ターゲット, 129 ページ](#)
- [共有体トランスフォーメーション, 131 ページ](#)
- [アップデートストラテジトランスフォーメーション, 132 ページ](#)

プッシュダウンの最適化およびトランスフォーメーションの概要

Integration Service でプッシュダウンの最適化を設定すると、各トランスフォーメーションがデータベースにプッシュされます。Integration Service がトランスフォーメーションをデータベースにプッシュできるかどうかに影響する基準は、次のとおりです。

- トランスフォーメーションのタイプ
- マッピング内のトランスフォーメーションの位置
- トランスフォーメーションに対するマッピングおよびセッションの設定
- トランスフォーメーション内に格納されている式

この基準は、Integration Service で実行できるプッシュダウン最適化のタイプ、およびトランスフォーメーションをプッシュできるデータベースのタイプにも影響を及ぼす場合があります。

Integration Service では、次のトランスフォーメーションのロジックをデータベースにプッシュできます。

- アグリゲータ
- 式
- フィルタ
- ジョイナ
- ルックアップ
- ルータ
- シーケンスジェネレータ
- ソータ
- ソース修飾子
- ターゲット
- 共有体
- アップデートストラテジ

プッシュダウンに関する一般的な制限

データベースにプッシュできるトランスフォーメーションロジックの量は、データベース、トランスフォーメーションロジック、マッピング設定、およびセッション設定によって異なります。統合サービスでは、データベースにプッシュできないすべてのトランスフォーメーションロジックが処理されます。

次のトランスフォーメーションまたはマッピングのいずれかの条件が true である場合、Integration Service はデータベースにプッシュしないでロジックを処理します。

- トランスフォーメーションロジックがマッピング変数を更新し、それをリポジトリデータベースに保存する。
- トランスフォーメーションに変数ポートが含まれている。
- トランスフォーメーションが以下の条件のすべてを満たしている。
 - ソータートランスフォーメーション、共有体トランスフォーメーション、ターゲットのどれでもない。
 - Microsoft SQL Server、Sybase、または Teradata にプッシュされる。
 - ソータートランスフォーメーションからのダウンストリームである。これは、共有体トランスフォーメーションからのダウンストリームである、または識別ソートを含んでいる。
- 入力ポートまたは出力ポートのデフォルト値を上書きするようにセッションが設定されている。
- トランスフォーメーション内の式に使用されている等価演算子、変数または関数が、データベース内にならない。
- マッピングに含まれているブランチが多すぎる。パイプラインをブランチに分岐する場合、マッピングロジックを表すために必要な SQL 文は複雑になります。Integration Service は、64 個を超える 2 方向へのブランチ、43 個の 3 方向へのブランチ、または 32 個の 4 方向へのブランチが含まれるマッピングの SQL クエリを生成することはできません。ブランチ数がこれらの制限を超えている場合、統合サービスではダウンストリームトランスフォーメーションが処理されます。

次のセッションのプロパティのいずれかが true である場合、Integration Service はデータベースにプッシュしないでロジックを処理します。

- セッションがデバッグセッションである。
- セッションが行エラーをログ記録するように設定されている。

前述の条件すべてが false の場合、個別のトランスフォーメーションおよびデータベースのプッシュダウンルールを参照できます。

アグリゲータトランスフォーメーション

次の表に、アグリゲータトランスフォーメーションをプッシュできる各データベースのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
Vertica	ソース側、完全
Microsoft Azure SQL Data Warehouse	ソース側、完全

以下の条件のいずれかに当てはまる場合、統合サービスによってアグリゲータトランスフォーメーションが処理されます。

- セッションおよびマッピングが差分集計用に設定されている。
- トランスフォーメーションに、ネストされた集計関数が含まれている。
- トランスフォーメーションに、集計式の条件節が含まれている。
- トランスフォーメーションでポート式に FIRST()関数、LAST()関数、MEDIAN()関数、または PERCENTILE()関数が使用されている。
- 出力ポートは、集計ポートでないかまたは GroupBy ポートの一部ではない。
- トランスフォーメーションは、Microsoft SQL Server、Sybase、または Teradata にプッシュされ、ソータトランスフォーメーションからのダウンストリームである。

式トランスフォーメーション

次の表に、式トランスフォーメーションをプッシュできる各データベースのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、ターゲット側、完全
Greenplum	ソース側、ターゲット側、完全
IBM DB2	ソース側、ターゲット側、完全
Microsoft SQL Server	ソース側、ターゲット側、完全
Netezza	ソース側、ターゲット側、完全
Oracle	ソース側、ターゲット側、完全
PostgreSQL	ソース側、完全
Snowflake	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Sybase ASE	ソース側、ターゲット側、完全
Teradata	ソース側、ターゲット側、完全
Vertica	ソース側、ターゲット側、完全
Microsoft Azure SQL Data Warehouse	ソース側、ターゲット側、完全

接続されていないストアドプロシージャを呼び出す場合、統合サービスによって式トランスフォーメーションが処理されます。

フィルタトランスフォーメーション

PowerCenter 統合サービスで処理するデータの量を減らすには、データベースにフィルタトランスフォーメーションをプッシュします。フィルタ式をデータベースにプッシュできない場合、フィルタトランスフォーメーションは PowerCenter 統合サービスによって処理されます。例えば、データベースにプッシュできない演算子がフィルタ式に含まれている場合、統合サービスはそのフィルタ式をデータベースにプッシュしません。

次の表に、フィルタトランスフォーメーションをプッシュできるデータベースごとにプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全

データベース	プッシュダウンタイプ
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
Vertica	ソース側、完全
Microsoft Azure SQL Data Warehouse	ソース側、完全

ジョイナトランスフォーメーション

データベースのインデックスおよび統計の使用を最適化するには、データベースにジョイナトランスフォーメーションをプッシュします。

次の表に、ジョイナトランスフォーメーションをプッシュできるデータベースごとにプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全

データベース	プッシュダウンタイプ
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
Vertica	ソース側、完全
Microsoft Azure SQL Data Warehouse	ソース側、完全

以下の条件のいずれかに当てはまる場合、統合サービスによってジョイナトランスフォーメーションが処理されます。

- 統合サービスがジョイナトランスフォーメーションのマスタパイプラインおよび詳細パイプラインをデータベースにプッシュできない。
- 結合条件が、バイナリデータ型のカラムに基づいている。
- ジョイナトランスフォーメーションの入力グループが異なるリレーショナルデータベース管理システム上のデータベースから発生している。
- セッションがソース行をすべてアップデートとしてマークするように設定されており、Teradata へのプッシュダウンの最適化が設定されている。
- トランスフォーメーションに外部結合が設定されており、マスタソースまたは詳細ソースが複数テーブル結合である。複数テーブル結合に結合された外部結合を表す SQL を、統合サービスで生成できない。
- トランスフォーメーションに完全外部結合が設定されており、Sybase へのプッシュダウンの最適化が設定されている。
- マスタブランチ内のトランスフォーメーションに基づいて統合サービスによってビューまたはシーケンスが作成されており、マスタブランチと詳細ブランチが同じデータベースから生じていない。
- トランスフォーメーションは、Microsoft SQL Server、Sybase、または Teradata にプッシュされ、アグリゲータトランスフォーメーションからのダウンストリームであるソータトランスフォーメーションからのダウンストリームである。
- トランスフォーメーションはソータトランスフォーメーションからのダウンストリームであり、Microsoft SQL Server、Sybase、または Teradata にプッシュされ、マスタテーブルと詳細テーブルが、同じソース修飾子トランスフォーメーションインスタンスから生じている。

ルックアップトランスフォーメーション

ルックアップトランスフォーメーションにプッシュダウンの最適化を設定した場合、データベースは、データベースルックアップテーブルでルックアップを実行します。データベースにルックアップトランスフォーメーションをプッシュすると、行ごとに追加のサブクエリが必要になるため負荷が高くなります。ルックアップの回数が多いマッピングのパフォーマンスを向上させるには、プッシュダウンの最適化を使用する代わりに、PowerCenter でルックアップキャッシュを有効にします。

次の表に、ルックアップトランスフォーメーションをプッシュできるデータベースごとにプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全
IBM DB2	ソース側、ターゲット側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、ターゲット側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
ODBC	ソース側、完全

ルックアップトランスフォーメーションのロジックをデータベースにプッシュするように Integration Service を設定する際には、次のルールおよびガイドラインを使用してください。

- データベースはトランスフォーメーションロジックの処理時に PowerCenter キャッシュを使用しません。
- 複数のルックアップトランスフォーメーションが別個のパイプラインブランチに存在していて、ブランチがダウストリームで統合されている場合、Integration Service はパイプラインブランチの後のすべてのトランスフォーメーションを処理します。
- ターゲット側に対してプッシュダウン最適化を実行するように設定されたセッションがデータタイプ変換を必要とする場合、このセッションは失敗します。
- Integration Service と異なり、Netezza データベースは単一のルックアップに対して複数の行を返すことができます。
- ルックアップトランスフォーメーションに SQL オーバーライドが含まれるか、フィルタが含まれるか、あるいはルックアップトランスフォーメーションが接続されていないルックアップトランスフォーメーションである場合は、ビューを使用してプッシュダウンの最適化を設定します。
- マッピングに Netezza、Redshift、Snowflake の各テーブルのルックアップが含まれ、ルックアップ一致ポリシーが **【すべての値を使用】** 以外のポリシーオプションに設定されている場合、ルックアップトランスフォーメーション時、プッシュダウンの最適化は停止します。ルックアップ一致ポリシーが **【すべての値を使用】** または **【エラーを報告】** 以外のポリシーオプションに設定されている場合、その他すべてのデータベースにおいて、プッシュダウンの最適化は停止します。

以下の条件のいずれかに該当する場合、ルックアップトランスフォーメーションは Integration Service によって処理されます。

- トランスフォーメーションがパイプラインルックアップに設定されている。
- トランスフォーメーションに動的キャッシュが使用されている。

- 最初、最後、または任意のマッチング値を返すようにトランスフォーメーションが設定されている。プッシュダウンの最適化を使用するには、複数の一致についてエラーがレポートされるようにルックアップトランスフォーメーションを設定する必要があります。
- トランスフォーメーションでは、データベース内でビューが作成され、ルックアップ入力を提供するデータベースがビューの作成元データベースとは異なっていることが要求される。
- トランスフォーメーションは、Microsoft SQL Server、Sybase、または Teradata にプッシュされ、アグリゲータトランスフォーメーションからのダウンストリームであるソータトランスフォーメーションからのダウンストリームである。
- セッションがソース行をすべてアップデートとしてマークするように設定されており、Teradata へのプッシュダウンの最適化が設定されている。
- ソース側のプッシュダウンの最適化を実行するようにセッションが設定され、ルックアップテーブルとソーステーブルが別々のリレーショナルデータベース管理システム内に存在する。
- ターゲット側のプッシュダウンの最適化を実行するようにセッションが設定され、ルックアップテーブルとターゲットテーブルが別々のリレーショナルデータベース管理システム内に存在する。
- The Integration Service により、Netezza データベースターゲットに対してトランスフォーメーションのプッシュが試行される。

接続されていないルックアップトランスフォーメーション

接続されていないルックアップトランスフォーメーションをデータベースにプッシュするように Integration Service を設定する際には、次のルールおよびガイドラインを使用してください。

- コネクトされていないルックアップがセッションに複数含まれている場合は、データベースの実行速度が Integration Service よりも遅くなる可能性があります。Integration Service では、コネクトされていないルックアップを呼び出すたびにアウトジョインが作成されるため、生成された SQL が複雑になる可能性があります。セッションをプッシュダウン最適化ありまたはなしでテストして、パフォーマンスが良いのはどちらのセッションかを確認します。
- セッションをビューと共にプッシュダウンの最適化に設定します。

以下の条件のいずれかに当てはまる場合、Integration Service によって接続されていないルックアップトランスフォーメーションが処理されます。

- ルックアップ接続に、ソース接続とのプッシュダウン互換がない。
- ターゲット側でのプッシュダウンの最適化を設定する。
- トランスフォーメーションがアグリゲータトランスフォーメーションからのダウンストリームトランスフォーメーションである。
- トランスフォーメーションがアクティブであり、Netezza データベースからルックアップしています。

SQL オーバライドを使用する Lookup トランスフォーメーション

SQL オーバライドを使用する Lookup トランスフォーメーションのロジックをデータベースにプッシュするように Integration Service を設定する際には、次のルールおよびガイドラインを使用してください。

- ルックアップオーバーライドでは、SQL 文に ORDER BY 句を追加できません。ORDER BY 句を追加すると、セッションが失敗します。
- SQL オーバライドが、ルックアップトランスフォーメーション内のポートの順序と同じ順序でルックアップトランスフォーメーション内のすべてのポートを選択することを確認します。
- SQL オーバライド内の SELECT 文がデータベースシーケンスを参照する場合、セッションが失敗します。

SQL オーバーライドを使用するルックアップトランスフォーメーションは、SQL オーバーライドの Informatica アウタージョイン構文を含む場合、によって処理されます。SQL オーバーライドに ANSI アウタージョイン構文を使用して、トランスフォーメーションをデータベースにプッシュします。

ルータトランスフォーメーション

次の表に、ルータトランスフォーメーションをプッシュできるデータベースごとのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
ODBC	ソース側、完全
Microsoft Azure SQL Data Warehouse	ソース側、完全

ソースデータベースにプッシュできる単一のトランスフォーメーションに出力グループをすべて統合できる場合は、ソース側プッシュダウンを使用することができます。

ルータ式をデータベースにプッシュできない場合、ルータトランスフォーメーションは統合サービスによって処理されます。例えば、データベースにプッシュできない演算子が式の中に含まれている場合、統合サービスではその式はデータベースにプッシュされません。

シーケンスジェネレータトランスフォーメーション

次の表に、シーケンスジェネレータトランスフォーメーションをプッシュできるデータベースごとにプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Greenplum	サポートされていません。
IBM DB2	ソース側、ターゲット側、完全
Microsoft SQL Server	サポートされていません。
Netezza	サポートされていません。
Oracle	ソース側、ターゲット側、完全
PostgreSQL	サポートされていません。
Sybase	サポートされていません。
Teradata	サポートされていません。
ODBC	サポートされていません。
Microsoft Azure SQL Data Warehouse	サポートされていません。

以下の条件のいずれかに当てはまる場合、統合サービスによってシーケンスジェネレータトランスフォーメーションが処理されます。

- トランスフォーメーションが再利用可能である。
- トランスフォーメーションが複数のターゲットに接続されている。
- トランスフォーメーションが CURRVAL ポートに接続する。
- トランスフォーメーションは、別個の行を選択するように設定されたソース修飾子トランスフォーメーションからのトランスフォーメーションダウンストリームに対してシーケンス値を提供します。
- 統合サービスがシーケンスジェネレータトランスフォーメーションのロジックを必ずしもすべてデータベースにプッシュできるとは限らない。例えば、パイプラインの2つのブランチに提供されるシーケンス値をシーケンスジェネレータトランスフォーメーションによって作成するとします。プッシュダウンの最適化を設定した場合、データベースは1つのパイプラインブランチ用にしかシーケンス値を作成できません。統合サービスで必ずしもすべてのシーケンスジェネレータロジックをデータベースにプッシュできない場合、次のメッセージが表示されます。

Pushdown optimization stops at the transformation <transformation name> because the upstream Sequence Generator <Sequence Generator transformation name> cannot be pushed entirely to the database.

- パイプラインはシーケンスジェネレータトランスフォーメーションの前でいったん分岐してから、シーケンスジェネレータトランスフォーメーションの後で元どおり1つに結合します。
- パイプラインはシーケンスジェネレータトランスフォーメーションの後で分岐してからは、元どおり1つに結合しません。
- シーケンス値は、アグリゲータ、フィルタ、ジョイナ、ソータ、または共有体の各トランスフォーメーションをバススルーします。
- シーケンスオブジェクトを作成するデータベースは、動作可能なデータベースまたは動作可能なデータベースと同じデータベースタイプである必要があります。

シーケンスジェネレータトランスフォーメーションからのトランスフォーメーションダウンストリームが統合サービスによって処理されるのは、このトランスフォーメーションが CASE 式にシーケンスジェネレータトランスフォーメーションの NEXTVAL ポートを使用していて、しかも IBM DB2 に対してプッシュダウンの最適化を実行するように設定されている場合です。

ソータートランスフォーメーション

次の表に、ソータートランスフォーメーションをプッシュできる各データベースのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
Snowflake	ソース側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
Vertica	ソース側、完全
Microsoft Azure SQL Data Warehouse	サポートされていません。

Sorter トランスフォーメーションのロジックをデータベースにプッシュするように統合サービスを設定する際には、次のルールおよびガイドラインを使用してください。

- ソータートランスフォーメーションが異なるソート用に設定され、Microsoft SQL Server、Sybase、または Teradata データベースにプッシュされる場合、統合サービスは、ソータートランスフォーメーションをデータベースにプッシュし、ダウンストリームのトランスフォーメーションを処理します。
- マッピングに複数の連続するソータートランスフォーメーションが含まれている場合、少なくとも 1 つのソータートランスフォーメーションが異なるソート用に設定されている場合、次のような結果になります。
 - ソータートランスフォーメーションの 1 つがすべての出力ポートを投影しない場合、プッシュダウンの最適化は、異なるソートをチェーンの最後のソータートランスフォーメーションに適用します。
 - プッシュダウンの最適化は、すべての出力ポートを投影しない最初のソータートランスフォーメーションに異なるソートを適用します。

以下の条件のいずれかに該当する場合、ソータ変換は統合サービスによって処理されません。

- ソータ変換が Union 変換からのダウストリームであり、ソータ変換内でソートキーとして使用されているポートが Union 変換からソータ変換に投影されない場合
- ソータ変換がすべての出力ポートを投影せず、それがマッピング内の複数の連続するソータ変換である場合。
- ソータ変換がすべての出力ポートを投影せず、次のいずれかの文に当てはまる場合。
 - ソータ変換が異なるソート用に設定されている。
 - ソータ変換が 1 つ以上のソータ変換（いずれかが異なるソート用に設定されている）によって直ちに処理される。

ソース修飾子変換

次の表に、ソース修飾子変換をプッシュできるデータベースごとにプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
Vertica	ソース側、完全
Microsoft Azure SQL Data Warehouse	ソース側、完全

ソース修飾子変換のロジックをデータベースにプッシュするように PowerCenter 統合サービスを設定する際には、以下のルールおよびガイドラインを使用します。

- シーケンスジェネレータ変換が、ダウストリーム共有体変換、ジョイナ変換、またはターゲットで接続がアイドル状態になっている場合、および異なるデータベースタイプの接続時に、ソースフィルタまたはユーザー定義結合に入力するテーブル名を修飾します。この場合にテーブル名を修飾しない場合、PowerCenter 統合サービスではすべての変換はデータベースにプッシュされません。
- ソース修飾子変換でショートカットオブジェクトにユーザー定義結合を設定し、かつプッシュダウンの最適化を有効にすると、セッションが失敗します。

以下の条件のいずれかに当てはまる場合、PowerCenter 統合サービスによってソース修飾子トランスフォーメーションのロジックが処理されます。

- トランスフォーメーションには、SQL オーバーライドまたはユーザー定義結合の Informatica 外部結合構文が含まれている。SQL オーバーライドに ANSI 外部結合構文を使用して、PowerCenter 統合サービスでソース修飾子トランスフォーメーションをデータベースにプッシュできるようにします。
- ソースが、データベースパーティション化の設定になっている。
- ソースが、XMLType データ型を使用する Oracle ソースである。

関連項目：

- [「アイドル状態のデータベース内のテーブル名の修飾」 \(ページ 85\)](#)

SQL オーバーライドを使用したソース修飾子トランスフォーメーション

SQL オーバーライドを使用するソース修飾子トランスフォーメーションを含むセッション用にプッシュダウンの最適化を設定する際には、次のルールおよびガイドラインを使用してください。

- ポート名は、トランスフォーメーション内での出現順序どおりに、カスタム SQL クエリの SELECT 文中にリストする必要があります。ポートが正しい順序でない場合、セッションは失敗するか、または予期しない結果を出力する可能性があります。
- セッションをビューと共にプッシュダウンの最適化に設定します。
- SQL オーバーライド内の SELECT 文がデータベースシーケンスを参照する場合、セッションが失敗します。
- SQL オーバーライドに ORDER BY 句が含まれている場合、ソース修飾子トランスフォーメーションのロジックを IBM DB2、Microsoft SQL Server、Sybase ASE、または Teradata データベースにプッシュするときに、セッションが失敗します。
- ソース修飾子トランスフォーメーションが個別の値を選択するように設定されていて、SQL オーバーライドが含まれている場合は、Integration Service により、個別の設定が無視されます。
- セッションに複数のパーティションが含まれている場合は、すべてのパーティションに対して SQL オーバーライドを指定します。
- ソース修飾子トランスフォーメーションに SQL オーバーライドが含まれている場合は、Teradata のプッシュダウン最適化に対して一時ビューの作成を無効にする必要があります。PowerCenter Integration Service で、ビューの代わりに派生テーブルが作成されます。
- SQL オーバーライドの構文は PowerCenter では検証されないため、SQL オーバーライドクエリーを先にソースデータベースに対してテストしてから、データベースにプッシュしてください。SQL 構文にソースソースとの互換性がない場合、セッションが失敗します。

ターゲット

次の表に、ターゲットロジックをプッシュできるデータベースごとのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
IBM DB2	ターゲット側、完全
Microsoft SQL Server	ターゲット側、完全

データベース	プッシュダウンタイプ
Netezza	ターゲット側、完全
Oracle	ターゲット側、完全
SAP HANA	ターゲット側、完全
Sybase ASE	ターゲット側、完全
Teradata	ターゲット側、完全
Vertica	ターゲット側、完全
Microsoft Azure SQL Data Warehouse	ターゲット側、完全

完全なプッシュダウンの最適化を実行するようにセッションを設定し、以下の条件のいずれかに当てはまる場合、ターゲットロジックが統合サービスによって処理されます。

- ターゲットに、ターゲット更新オーバーライドが含まれている。
- セッションは制約に基づくロード用に設定されていて、ターゲットロード順グループに複数のターゲットが含まれている。
- セッションに外部ローダが使用されている。
- アイドル状態のデータベースでビューまたはシーケンスジェネレータが生成された。

完全なプッシュダウンの最適化を設定し、ターゲット接続とソース接続に互換性がない場合、統合サービスはトランスフォーメーションロジックをすべて1つのデータベースにプッシュできません。代わりに、できるだけ多くのトランスフォーメーションロジックをソースデータベースにプッシュし、可能であれば残りのトランスフォーメーションロジックをターゲットデータベースにプッシュします。

ターゲット側のプッシュダウンの最適化を実行するようにセッションを設定し、以下の条件のいずれかに当てはまる場合、ターゲットロジックが統合サービスによって処理されます。

- ターゲットに、ターゲット更新オーバーライドが含まれている。
- ターゲットがデータベースパーティション化の設定になっている。
- セッションがバルクロード用に設定されており、ターゲットが IBM DB2、Microsoft SQL Server、Oracle、または Sybase ASE である。
- セッションに外部ローダが使用されている。統合サービスがトランスフォーメーションロジックをソースデータベースにプッシュできるようにするには、外部ローダでソース側プッシュダウンの最適化を使用してください。

共有体トランスフォーメーション

次の表に、共有体トランスフォーメーションをプッシュできる各データベースのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	ソース側、完全
Greenplum	ソース側、完全
IBM DB2	ソース側、完全
Microsoft SQL Server	ソース側、完全
Netezza	ソース側、完全
Oracle	ソース側、完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全
Sybase ASE	ソース側、完全
Teradata	ソース側、完全
Vertica	ソース側、完全
Microsoft Azure SQL Data Warehouse	ソース側、完全

以下の条件のいずれかに該当する場合、共有体トランスフォーメーションのロジックは統合サービスによって処理されます。

- 統合サービスが入力グループをすべてソースデータベースにプッシュできるとは限らない。
- 入力グループが同一のリレーショナルデータベース管理システムから発生していない。
- 共有体トランスフォーメーションの入力パイプラインの1つには、個別の共有体またはソータが含まれる。
- トランスフォーメーションが、ビューまたはシーケンスジェネレータがデータベース内で作成され、各接続が個別のデータベース上に存在することを要求するトランスフォーメーションからのダウンストリームである。

アップデートストラテジトランスフォーメーション

次の表に、アップデートストラテジトランスフォーメーションをプッシュできる各データベースのプッシュダウンタイプを示します。

データベース	プッシュダウンタイプ
Amazon Redshift	完全
Greenplum	ターゲット側
IBM DB2	完全
Microsoft SQL Server	完全
Netezza	完全
Oracle	完全
PostgreSQL	ソース側、完全
SAP HANA	ソース側、ターゲット側、完全
Snowflake	ソース側、完全
Sybase ASE	完全
Teradata	完全
Vertica	完全
Microsoft Azure SQL Data Warehouse	完全

アップデートストラテジトランスフォーメーションのロジックをデータベースにプッシュするように統合サービスを設定する際には、次のルールおよびガイドラインを使用してください。

- 更新操作でアップデートストラテジトランスフォーメーション用に生成された SQL は、複雑になる可能性があります。セッションをプッシュダウン最適化ありまたはなしで実行して、どちらの設定の方が高速かを確認します。
- 同一行に対して複数の操作が存在している場合、統合サービスとデータベースとで異なる操作が実行される可能性があります。新しい行をデータベースにプッシュしたときに削除または更新されないようにするには、トランザクション削除、トランザクション更新、トランザクション挿入の順序でソース行を処理します。
- トランスフォーメーションに複数の挿入、更新、または削除操作が含まれる場合、統合サービスにより連続して、挿入、更新、削除の SQL 文が生成、実行されます。これら 3 つの文は必要でない場合でも統合サービスによって実行されます。この結果、パフォーマンスが低下する可能性があります。
- 完全なプッシュダウン最適化が使用されているときは、拒否行が統合サービスに無視されます。拒否行は拒否ファイルに書き込まれません。

以下の条件のいずれかに当てはまる場合、統合サービスによってアップデートストラテジトランスフォーメーションが処理されます。

- 統合サービスで更新方式の式がデータベースにプッシュできない場合。例えば、データベースにプッシュできない演算子が式の中に含まれている場合、統合サービスではその式はデータベースにプッシュされません。
- トランスフォーメーションに挿入操作以外の操作が使用されていて、統合サービスが必ずしもすべてのトランスフォーメーションをデータベースにプッシュできるとは限らない。
- アップデートストラテジ式が、数値でもブールでもない値を返す。

第 6 章

リアルタイム処理

この章では、以下の項目について説明します。

- [リアルタイム処理の概要, 134 ページ](#)
- [リアルタイムデータについて, 135 ページ](#)
- [リアルタイムセッションの設定, 138 ページ](#)
- [終了条件, 138 ページ](#)
- [フラッシュ待ち時間, 139 ページ](#)
- [コミットタイプ, 140 ページ](#)
- [メッセージリカバリ, 140 ページ](#)
- [リカバリファイル, 142 ページ](#)
- [リカバリテーブル, 145 ページ](#)
- [リカバリキューおよびリカバリトピック, 146 ページ](#)
- [リカバリ無視リスト, 147 ページ](#)
- [リアルタイムセッションの停止, 147 ページ](#)
- [リアルタイムセッションのリスタートおよびリカバリ, 148 ページ](#)
- [リアルタイムセッションに関するルールおよびガイドライン, 149 ページ](#)
- [メッセージリカバリに関するルールおよびガイドライン, 150 ページ](#)
- [リアルタイム処理の例, 150 ページ](#)
- [PowerCenter リアルタイム製品, 152 ページ](#)

リアルタイム処理の概要

リアルタイム処理の動作は、リアルタイムソースに依存します。例外については、この章に注意書きされているほか、対応する製品のマニュアルにも説明が記載されています。

PowerCenter を使用して、リアルタイムでデータを処理することができます。リアルタイム処理は、リアルタイムソースからのデータのオンデマンド処理です。リアルタイムセッションでは、ターゲットに対するデータの読み込み、処理、および書き込みが連続して実行されます。セッションをリアルタイム処理に設定しない場合、デフォルトで、セッションはスケジュールされた間隔でデータを一括で読み書きします。

データをリアルタイムで処理するには、そのデータの発生元がリアルタイムソースでなければなりません。リアルタイムソースには、JMS、WebSphere MQ、TIBCO、webMethods、MSMQ、SAP、Web サービス、および PowerExchange などがあります。財務データなどの動的データに即時にアクセスしなければならない処理にリアルタイム処理を使用することができます。

PowerCenter でのリアルタイム処理を理解するには、次の概念をよく把握する必要があります。

- **リアルタイムデータ。**リアルタイムデータには、メッセージとメッセージキュー、Web サービスメッセージ、および PowerExchange 変更データキャプチャソースからの変更が含まれます。リアルタイムデータは、リアルタイムソースから派生します。
- **リアルタイムセッション。**リアルタイムセッションとは、リアルタイムソースデータを処理するセッションのことです。統合サービスがフラッシュ待ち時間の設定に基づいてリアルタイムフラッシュを生成し、すべてのトランスフォーメーションがそのフラッシュをターゲットにプロパゲートする場合、セッションはリアルタイムになります。待ち時間とは、ソースでソースデータが変更されてからセッションがそのデータをターゲットに書き込むまでの時間です。
- **リアルタイムプロパティ。**リアルタイムプロパティは、統合サービスがどのような場合にデータを処理し、データをターゲットにコミットするかを決定します。
 - **終了条件。**終了条件は、統合サービスでどのような場合にソースからのデータの読み取りを停止するかを決定します。また、セッションを連続して実行しないときは、どのような場合にセッションを終了するかを決定します。
 - **フラッシュ待ち時間。**フラッシュ待ち時間は、統合サービスでソースからリアルタイムデータをフラッシュする頻度を決定します。
 - **コミットタイプ。**コミットタイプは、統合サービスでどのような場合にリアルタイムデータをターゲットにコミットするかを決定します。
- **メッセージのリカバリ。**リアルタイムセッションが失敗した場合、メッセージをリカバリすることができません。リアルタイムセッションでメッセージのリカバリを有効にした場合、Integration Service はソースのメッセージまたはメッセージ ID をリカバリファイルまたはリカバリテーブルに格納します。セッションが失敗した場合、リカバリモードでセッションを実行して、Integration Service が処理できなかったメッセージを回復することができます。

リアルタイムデータについて

以下のタイプのリアルタイムデータを処理できます。

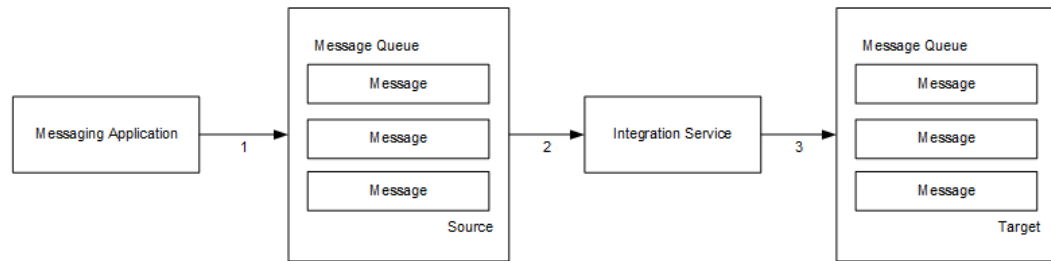
- **メッセージとメッセージキュー。**WebSphere MQ、JMS、MSMQ、SAP、TIBCO、および webMethods ソースのメッセージとメッセージキューを処理します。メッセージとメッセージキューからの読み込みが可能です。メッセージ、メッセージングアプリケーション、およびメッセージキューへの書き込みが可能です。
- **Web サービスメッセージ。**Web サービスクライアントから Web サービス Hub を介してメッセージを受信し、データを変換します。データをターゲットに書き込むことも、あるいは WebService クライアントにメッセージを送り返すこともできます。
- **PowerExchange が複数の異なるソースからキャプチャする変更データ。**i5/OS、Linux、UNIX、Windows、および z/OS システム上のさまざまなリレーショナルおよび非リレーショナルソースから PowerExchange がキャプチャする変更データを抽出します。PowerExchange 変更データキャプチャ (CDC) は、リアルタイムモードでエンタープライズ全体の変更データをキャプチャ、変換、配信するために PowerCenter と統合されています。

メッセージとメッセージキュー

Integration Service は、メッセージとキューのアーキテクチャを使用してリアルタイムデータを処理します。メッセージキューからメッセージを読み込み、メッセージデータを処理し、メッセージをメッセージキューに書き込むこともできます。

メッセージをその他のメッセージングアプリケーションに書き込むこともできます。例えば、Integration Service は JMS ソースからメッセージを読み込み、TIBCO ターゲットにデータを書き込むことができます。

次の図に、メッセージングアプリケーションおよび Integration Service でメッセージキューからのメッセージが処理される仕組みを例示します。



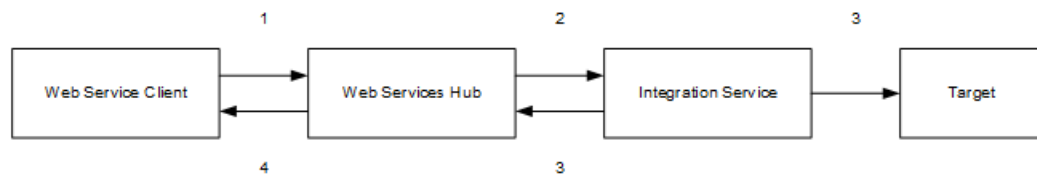
メッセージングアプリケーションおよび Integration Service は、メッセージキューからのメッセージを処理するために次のタスクを完了します。

1. メッセージングアプリケーションがメッセージをキューに追加します。
2. Integration Service がキューからメッセージを読み込み、データを抽出します。
3. Integration Service はデータを処理し、メッセージキューに応答を書き込みます。

Web サービスメッセージ

Web サービスメッセージとは、WebService クライアントからの SOAP 要求または Web Services Hub からの SOAP レスポンスです。Integration Service は Web Services Hub を介して受信したメッセージ要求を処理することにより、WebService クライアントからのリアルタイムデータを処理します。Integration Service は、Web Services Hub を介して WebService クライアントに応答を返信するか、またはデータをターゲットに書き込むことができます。

次の図に、Web サービスクライアント、Web Services Hub、および Integration Service で Web サービスメッセージが処理される仕組みを例示します。



WebService クライアント、Web Services Hub および Integration Service は、Web サービスを処理するために次のタスクを実行します。

1. WebService クライアントは、SOAP 要求を Web Services Hub に送信します。
2. Web Services Hub は SOAP 要求を処理し、この要求を Integration Service に渡します。
3. Integration Service は、このサービス要求を実行します。Integration Service は Web Services Hub に応答を送信するか、またはデータをターゲットに書き込みます。
4. Integration Service が Web Services Hub にレスポンスを送信した場合、Web Services Hub は SOAP メッセージ応答を生成し、この応答を WebService クライアントに渡します。

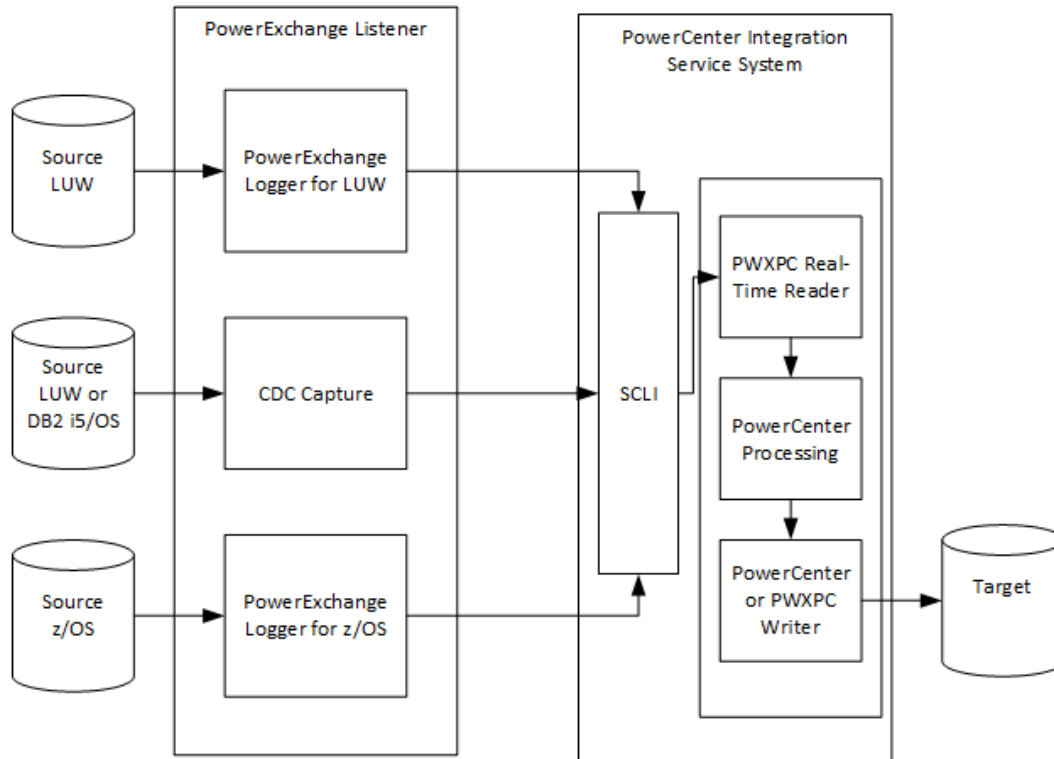
PowerExchange CDC ソースの変更データ

PowerExchange 変更データキャプチャ (CDC) を使用して、オンライントランザクション処理システムおよびデータベースへの最新の変更を、タイムリーなビジネス上の意思決定を行うためのデータウェアハウス、オペレーショナルデータストア、およびアプリケーションで使用可能にします。

PowerExchange CDC により、i5/OS、Linux、UNIX、Windows、および z/OS システム上のさまざまなリレーショナルソースおよび非リレーショナルソースのコミットされたトランザクションからの変更データをキャ

ブチャできます。PowerExchange は、選択されたソースオブジェクトに挿入、更新、および削除が発生した場合にそれらをキャプチャし、PowerExchange Client for PowerCenter (PWXPC) コンポーネントにより変更データが要求されるまで、リアルタイム変更ストリームまたは PowerExchange ロgger ログファイルに変更を保存します。

PWXPC プラグインは PowerCenter と一緒にインストールされるので、PowerExchange のローカルインストールが必要です。PWXPC は PowerCenter 統合サービスおよび Client ツールと連携し、特定の期間または継続してリアルタイムセッションを実行します。PowerExchange リスナはソースシステム上で実行される場合もソースシステム外で実行される場合もあります。次の図に、PowerExchange および PowerCenter の単純な設定を示します。



次のプロセスフローに PowerExchange と PowerCenter の相互作用の概要を示します。

1. PowerExchange は、変更が発生すると、選択したソーステーブルおよびカラムから変更データをキャプチャします。
2. PowerExchange ソースが含まれており、PWX CDC Real Time アプリケーション接続が使用されている PowerCenter ワークフローが開始されます。
3. PWXPC は PowerExchange Call Level Interface (SCLI) を経由して PowerExchange に接続され、ワークフローの代わりに PowerExchange リスナ経由で変更データを取得します。
4. PowerExchange はマッピング内のすべてのソースの変更ストリームから変更データを抽出し、そのデータを PWXPC に渡します。
5. PWXPC は変更データを統合サービスに渡します。
6. PowerCenter ワークフローは変更データを処理し、変換します。
7. PowerCenter または PowerExchange ライタは変更データを 1 つ以上のターゲットに書き込みます。

リアルタイムセッションの設定

リアルタイムでデータを処理するようにセッションを設定した場合は、セッションがソースからの読み込みを停止するタイミングを制御するセッションプロパティを設定することもできます。設定した期間にわたってセッションがメッセージの受信を停止した後、セッションがメッセージカウント制限に達したとき、あるいは設定した期間にわたってセッションがメッセージを読み込んだときに、ソースからの読み込みを停止するセッションを設定できます。また、Integration Service がターゲットにデータをコミットする方法を設定し、失敗したセッションでのメッセージのリカバリを有効にすることもできます。

リアルタイムセッション用に、次のプロパティを設定できます。

- **終了条件。** Integration Service でどのような場合にソースからの読み込みを停止しセッションを終了するかを決定する、終了条件を定義します。
- **フラッシュ待ち時間。** リアルタイムデータの読み込みおよび書き込みを行うためにフラッシュ待ち時間を使用するセッションを定義します。フラッシュ待ち時間は、セッションがデータをターゲットにコミットする頻度を決定します。
- **コミットタイプ。** リアルタイムセッション用に、ソースベースまたはターゲットベースのコミットタイプを設定できます。ソースベースのコミットを使用すると、Integration Service は、コミット間隔とフラッシュ待ち時間に基づいてメッセージをコミットします。ターゲットベースのコミットを使用する場合、Integration Service は、フラッシュ待ち時間の間隔に基づいてメッセージをコミットします。
- **メッセージリカバリ。** 失敗したセッションからのメッセージをリカバリするために、リカバリをリアルタイムセッションで有効にします。

終了条件

終了条件は、Integration Service でどのような場合にリアルタイムソースからのメッセージ読み込みを停止し、セッションを終了するかを決定します。Integration Service は終了条件に達すると、リアルタイムソースからの読み込みを停止し、読み込んだメッセージを処理して、データをターゲットにコミットします。その後、セッションが終了します。

以下の終了条件を設定することができます。

- アイドル時間
- メッセージカウント
- Reader 制限時間

複数の終了条件が設定されている場合、Integration Service は最初の条件を満たしたときにソースからの読み込みを停止します。Integration Service ではデフォルトでメッセージが連続的に読み込まれ、ソースからデータをフラッシュするタイミングがフラッシュ待ち時間を使用して決定されます。Integration Service はフラッシュ後に、終了条件カウンタをリセットします。

アイドル時間

アイドル時間は、ソースからの読み込みを停止するまでに Integration Service がメッセージの受信を待機する時間の長さ（秒単位）です。-1 は無期限を示します。

たとえば、JMS セッションのアイドル時間を 30 秒に設定した場合、Integration Service は JMS からの読み込み後、30 秒間待機します。30 秒以内に新しいメッセージが JMS に到着しなかった場合、Integration Service は JMS からの読み込みを停止します。メッセージ処理後、セッションが終了します。

メッセージカウント

メッセージカウントは、Integration Service がリアルタイムソースからの読み込みを停止する前にソースから読み込むメッセージ数です。-1 は不特定数のメッセージを示します。

たとえば、JMS セッションのメッセージカウントを 100 に設定した場合、Integration Service は 100 件のメッセージを読み込んだ後、ソースからの読み込みを停止します。メッセージ処理後、セッションが終了します。

注: メッセージカウント終了条件の名前は、Informatica 製品ごとに異なります。例えば、SAP NetWeaver 対応の PowerExchange 用のメッセージカウントはパケットカウントと呼ばれます。PowerExchange Client for PowerCenter 用のメッセージカウントは UOW カウントと呼ばれます。

Reader 制限時間

Reader 制限時間は、Integration Service がソースからの読み込みを停止するまでにリアルタイムソースからソースメッセージを読み込んだ時間数（秒単位）です。reader 制限時間を使用して、指定した期間にわたってリアルタイムソースからメッセージを読み込みます。0 は無期限を示します。

例えば、タイムリミットに 10 秒を使用した場合、Integration Service は 10 秒後にメッセージングアプリケーションからの読み込みを停止します。メッセージ処理後、セッションが終了します。

フラッシュ待ち時間

フラッシュ待ち時間を使用してセッションをリアルタイムで実行します。フラッシュ待ち時間は、Integration Service でソースからデータをフラッシュする頻度を決定します。たとえば、フラッシュ待ち時間を 10 秒に設定すると、Integration Service は 10 秒ごとにソースからデータをフラッシュします。

PowerExchange 変更データキャプチャソースからの変更データの場合、フラッシュ待ち時間間隔は、フラッシュ待ち時間および作業単位（UOW）カウントの属性によって決まります。詳細については、『*PowerCenter 用の PowerExchange インタフェース*』を参照してください。

Integration Service がリアルタイムソースからデータを読み込み、セッションがフラッシュ待ち時間を使用して設定されていると、Integration Service は次の処理を実行します。

1. Integration Service は、ソースからデータを読み込みます。
フラッシュ待ち時間間隔は、Integration Service がソースから最初のメッセージを読み込んだ時点から開始します。
2. フラッシュ待ち時間が経過すると、Integration Service は、ソースからのデータの読み込みを停止します。
3. Integration Service はメッセージを処理し、それをターゲットに書き込みます。
4. Integration Service は、次のフラッシュ待ち時間に達するまで再度ソースからデータを読み込みます。

フラッシュ待ち時間は、秒数で設定します。デフォルト値は 0 です。0 値を指定すると、フラッシュ待ち時間は無効となり、セッションはリアルタイムで実行されません。

フラッシュ待ち時間間隔は、データの動的レベル、およびユーザーに必要なデータアクセス速度によって決定されます。財務取引情報のようにデータがすぐに陳腐化する場合は、変更が発生したときにできるだけ早くターゲットテーブルが更新されるように、フラッシュ待ち時間間隔を短めに設定してください。たとえば、財務データはユーザーが数分ごとに更新する必要があるとします。一方、顧客アドレス変更内容は、更新が 1 日に一度で済みます。フラッシュ待ち時間間隔は、財務データに対しては短めに、アドレス変更内容に対しては長めに設定してください。

フラッシュ待ち時間を設定する場合には、以下の規則およびガイドラインに従ってください。

- Integration Service は、メッセージをフラッシュ待ち時間の間隔よりも長い時間にわたってバッファに入れることはしません。
- フラッシュ待ち時間の間隔を短く設定すると、Integration Service がターゲットにメッセージをコミットする頻度が増えます。
- フラッシュ待ち時間の間隔を短く指定すると、セッションでさらに多くのシステムリソースが消費される可能性があります。

コミット間隔が設定されている場合、フラッシュ待ち時間とコミット間隔の設定によって、データをターゲットにコミットするタイミングが決定されます。

コミットタイプ

Integration Service は、フラッシュ待ち時間とコミットタイプに基づいてターゲットにデータをコミットします。以下のコミットタイプを使用するセッションを設定することができます。

- **ソースベースのコミット。**ソースベースのコミットを設定する場合、Integration Service ではコミット間隔とフラッシュ待ち時間の間隔の組み合わせを使用してターゲットにデータがコミットされます。Integration Service が満たした最初の条件によって、フラッシュ待ち時間の終了がトリガされます。フラッシュ後に、カウンタがリセットされます。
たとえば、フラッシュ待ち時間を 5 秒に設定し、ソースベースのコミットの間隔を 1,000 メッセージに設定したとします。Integration Service がソースから 1,000 件のメッセージを読み込んだ後、または 5 秒後に、メッセージがターゲットにコミットされます。
- **ターゲットベースのコミット。**ターゲットベースのコミットを設定する場合、Integration Service ではコミット間隔が無視され、フラッシュ待ち時間の間隔に基づいてターゲットにデータがコミットされます。

リアルタイムセッションでターゲットに書き込む際、Integration Service ではコミットが順次処理され、データがリアルタイムでターゲットにコミットされます。データは DTM バッファメモリ内には格納されません。

関連項目：

- [「コミットポイント」 \(ページ 154\)](#)

メッセージリカバリ

リアルタイムセッションでメッセージリカバリを有効にすると、Integration Service は、失敗したセッションからの未処理のメッセージをリカバリできるようになります。Integration Service では、リカバリファイル、リカバリテーブル、リカバリキュー、またはリカバリトリップにソースメッセージまたはメッセージ ID が格納されます。セッションが失敗した場合、リカバリモードでセッションを実行して、Integration Service が処理しなかったメッセージをリカバリできます。

メッセージまたはメッセージ ID は、リアルタイムソースまたはターゲットタイプに応じて、次のストレージタイプで格納されます。

- **リカバリファイル。**メッセージまたはメッセージ ID は、指定したローカルリカバリファイルに格納されます。リアルタイムソース、および非リレーショナルターゲットまたは非キューターゲットを使用するセッションには、リカバリファイルが使用されます。

- **リカバリテーブル。**メッセージ ID は、ターゲットデータベースのリカバリテーブルに格納されます。JMS ソースまたは WebSphere MQ ソース、およびリレーショナルターゲットを使用するセッションには、リカバリテーブルが使用されます。
- **リカバリキューおよびリカバリトピック。**メッセージ ID は、リカバリキューまたはリカバリトピックに格納されます。JMS ソースまたは WebSphere MQ ソース、および JMS ターゲットまたは WebSphere MQ ターゲットを使用するセッションには、リカバリキューが使用されます。JMS ソースまたは WebSphere MQ ソース、およびトピックターゲットを使用するセッションには、リカバリトピックが使用されます。

セッションでは、ストレージタイプの組み合わせを使用できます。たとえば、JMS ソースおよび TIBCO ソースを使用するセッションには、リカバリファイルおよびリカバリテーブルが使用されます。

Integration Service でリアルタイムセッションをリカバリすると、中断箇所から操作の状態がリストアされます。リカバリファイル、リカバリテーブル、リカバリキュー、またはリカバリトピックからメッセージが読み込まれ、処理されます。その後、セッションが終了します。

リカバリ中、終了条件は Integration Service がリカバリファイル、リカバリテーブル、リカバリキュー、またはリカバリトピックから読み込むメッセージには影響しません。たとえば、セッションのメッセージカウントとアイドル時間を指定した場合、これらの条件は、Integration Service がソースから読み込むメッセージには適用されますが、Integration Service がリカバリファイル、リカバリテーブル、リカバリキュー、またはリカバリトピックから読み込むメッセージには適用されません。

セッションが特定の条件下で失敗した場合、Integration Service では、上記のストレージタイプの他に、リカバリ無視リストが使用されます。

MSMQ ソース、Web サービスメッセージ、または PowerExchange 変更データキャプチャソースからの変更データを使用するセッションには、別のリカバリ戦略が使用されます。

前提条件

JMS ソースまたは WebSphere MQ ソース、および JMS ターゲットまたは WebSphere MQ ターゲットを使用するセッションに対し、メッセージのリカバリを有効にする前に、以下の前提条件を満たします。

- JMS プロバイダまたは WebSphere MQ にリカバリキューを作成します。または、JMS プロバイダにリカバリトピックを作成します。
- コミットスコープが一致するように、メッセージキューと同じキューマネージャを使用してリカバリキューを作成します。
- リカバリキューが永続的になるように設定します。リカバリキューが永続的でない場合、データの重複が発生することがあります。

メッセージのリカバリを有効にする手順

セッションでメッセージのリカバリを有効にするには、次の手順を実行します。

1. セッションプロパティで、リカバリ戦略プロパティに [最後のチェックポイントから再開] を選択します。
2. 各パーティションポイントについて、セッションプロパティのリカバリキャッシュディレクトリを指定します。

Integration Service は、リカバリキャッシュディレクトリによって指定された場所に、メッセージを格納します。リカバリキャッシュディレクトリのデフォルト値は、\$PMCacheDir です。

リカバリファイル

Integration Service は、リカバリが有効で、次のタイプのソースとターゲットを含めたリアルタイムセッションのリカバリファイルに、メッセージまたはメッセージ ID を格納します。

- 非リレーショナルターゲット、非 JMS ターゲット、または非 WebSphere MQ ターゲットを使用する JMS ソース
- 非リレーショナルターゲット、非 JMS ターゲット、または非 WebSphere MQ ターゲットを使用する WebSphere MQ ソース
- SAP ECC ソースおよびすべてのターゲット
- webMethods ソースおよびすべてのターゲット
- TIBCO ソースおよびすべてのターゲット

Integration Service は、セッションプロパティで設定したローカルのリカバリファイルにメッセージまたはメッセージ ID を一時的に格納します。リカバリ中、Integration Service は、このリカバリファイル内のメッセージを処理し、データが失われることがないようにします。

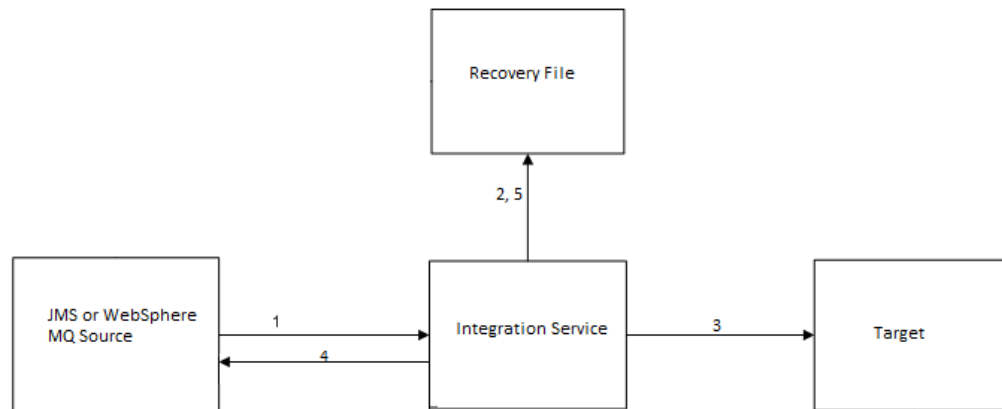
JMS および WebSphere MQ ソースのメッセージリカバリ

JMS および WebSphere MQ ソースとのセッションに対するメッセージリカバリを有効にすることで、統合サービスで処理できなかったメッセージをリカバリできます。統合サービスでは、操作の状態を中断された時点まで戻すことができます。

統合サービスは、リカバリファイルを使用してメッセージを処理するために次のタスクを実行します。

1. 統合サービスが、ソースからメッセージを読み込みます。
2. 統合サービスが、読み込んだメッセージ ID をリカバリファイルに書き込みます。統合サービスが、フラッシュ待ち時間が満了するまでステップ 1-2 を繰り返し実行します。
3. 統合サービスがメッセージを処理し、ターゲットに書き込みます。ターゲットがメッセージをコミットします。
4. 統合サービスが、メッセージを読み込んだことを確認するバッチ確認応答をソースに送信します。ソースがメッセージを削除します。
5. 統合サービスがリカバリファイルをクリアします。

次の画像に、統合サービスがリカバリファイルを使用してメッセージを処理する方法を示します。



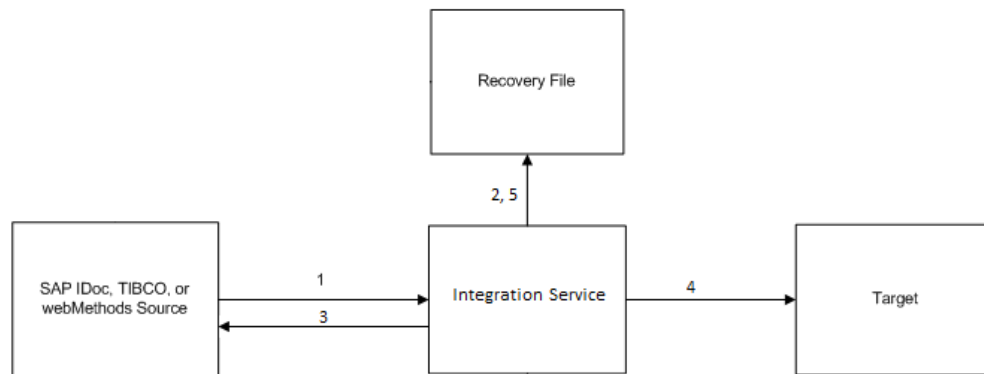
SAP IDoc、TIBCO、および webMethods ソースのメッセージリカバリ

SAP IDoc、TIBCO、および webMethods ソースとのセッションに対するメッセージリカバリを有効にすることで、統合サービスで処理できなかったメッセージをリカバリできます。統合サービスでは、操作の状態を中断された時点まで戻すことができます。

統合サービスは、リカバリファイルを使用してメッセージを処理するために次のタスクを実行します。

1. 統合サービスが、ソースからメッセージを読み込みます。
2. 統合サービスが、読み込んだメッセージをリカバリファイルに書き込みます。
3. 統合サービスが、メッセージを読み込んだことを確認する確認応答をソースに送信します。ソースがメッセージを削除します。統合サービスが、フラッシュ待ち時間が満了するまでステップ 1 - 3 を繰り返し実行します。
4. 統合サービスがメッセージを処理し、ターゲットに書き込みます。ターゲットがメッセージをコミットします。
5. 統合サービスがリカバリファイルをクリアします。

次の画像に、統合サービスがリカバリファイルを使用してメッセージを処理する方法を示します。



メッセージリカバリ

リアルタイムセッションをリカバリすると、Integration Service はキャッシュされたメッセージを読み込み、処理します。Integration Service は、キャッシュされたすべてのメッセージを読み込んだ後、セッションを終了します。

JMS および WebSphere MQ ソースを使用するセッションの場合、Integration Service はリカバリファイルのメッセージ ID を使用して、ソースからメッセージを取得します。

フラッシュ待ち時間が経過し、セッションが成功した時点で、Integration Service はリカバリファイルをクリアします。Integration Service がメッセージをターゲットにコミットしてからリカバリファイルからメッセージを削除するまでの間にセッションが失敗すると、リカバリ中にターゲットで行が重複して受け取られる可能性があります。

セッションリカバリデータのフラッシュ

リカバリデータのフラッシュは、Integration Service がオペレーティングシステムバッファ内のセッションリカバリデータをリカバリファイルにフラッシュする際に使用する処理です。Integration Service でリカバリデータをリカバリファイルに書き込めない場合に、データの喪失を防止できます。オペレーティングシステム障害、ハードウェア障害、またはファイルシステムの故障が発生した場合、Integration Service はリカバリデータの書き込みに失敗する可能性があります。リカバリデータのフラッシュは、JMS ソースまたは WebSphere MQ ソース、および非リレーショナルターゲット、非 JMS ターゲット、または非 WebSphere MQ ターゲットを含むセッションに適用されます。

Administrator ツールで、Integration Service プロパティの [セッションリカバリデータのフラッシュ] を [自動] または [はい] に設定することにより、リカバリデータがオペレーティングシステムバッファからリカバリファイルにフラッシュされるように Integration Service を設定できます。

リカバリテーブル

Integration Service でリカバリが有効化されているリアルタイムセッションのリカバリテーブルにメッセージ ID が格納されるのは、そのリアルタイムセッションに次のタイプのソースおよびターゲットが含まれる場合です。

- リレーショナルターゲットを用いた JMS ソース
- リレーショナルターゲットを用いた WebSphere MQ ソース

Integration Service は、メッセージ ID とコミット番号を各ターゲットデータベースのリカバリテーブルに一時的に格納します。コミット番号は、Integration Service からターゲットへのコミット数を示します。リカバリ中、Integration Service は、コミット番号を使用して、すべてのターゲットに同じ数のメッセージを書き込んだかどうかを判断します。メッセージ ID とコミット番号をリカバリテーブルと照合して、データの喪失や重複がないことを確認します。

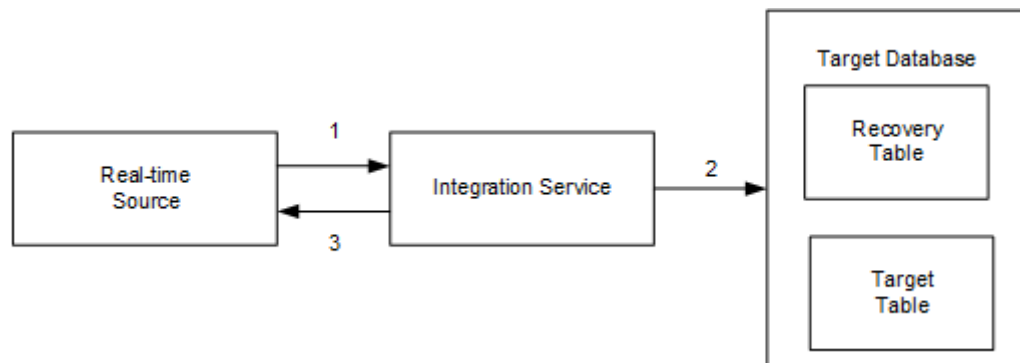
注: これらのソースは、一意のメッセージ ID を使用し、そのメッセージ ID を介してメッセージにアクセスできるようにする必要があります。

PM_REC_STATE テーブル

リカバリテーブルを使用し、リカバリが有効になっているリアルタイムセッションを実行する場合、統合サービスは、ターゲットデータベースにリカバリテーブル PM_REC_STATE を作成してメッセージ ID とコミット番号を格納します。このセッションをリカバリする際、統合サービスは、リカバリテーブル内の情報を使用して、ターゲットテーブルへのメッセージの書き込みが必要かどうかを判断します。

メッセージの処理

次の画像に、Integration Service がリカバリテーブルを使用してメッセージを処理する方法を示します。



Integration Service は、リカバリテーブルを使用したメッセージ処理を行うために以下のタスクを実行します。

1. Integration Service は、フラッシュ待ち時間に達するまで一度に 1 つずつメッセージを読み込みます。
2. Integration Service はメッセージ ID、コミット番号、およびトランスフォーメーション状態をターゲットデータベースのリカバリテーブルに書き込み、それと同時にメッセージをターゲットに書き込みます。
3. ターゲットがメッセージをコミットすると、Integration Service はすべてのメッセージが処理され、ターゲットに書き込まれたことを確認するために受信確認メッセージをリアルタイムソースに送り返します。

Integration Service は、ソースからのメッセージの読み込みを続行します。

セッションで複数のパーティションが使用される場合、これらのタスクはパーティションごとに適用されます。

メッセージリカバリ

Integration Service でリアルタイムセッションをリカバリした場合、リカバリファイル内のメッセージ ID とコミット番号を使用してすべてのターゲットにメッセージがコミットされたかが確認されます。

Integration Service ですべてのターゲットにメッセージがコミットされるのは、リカバリテーブル内にメッセージ ID が存在していて、しかもすべてのターゲットのコミット番号が同じ場合です。Integration Service はリカバリ中に、ソースに対してメッセージ処理完了の確認応答を送ります。

ターゲットのコミット番号が異なる場合、メッセージはすべてのターゲットにコミットされるとは限りません。リカバリ中、Integration Service はリカバリテーブルからメッセージ ID とトランスフォーメーション状態を読み込みます。その後、メッセージを処理し、それらのメッセージがなかったターゲットにメッセージを書き込みます。Integration Service により、リカバリテーブルからすべてのメッセージが読み込まれたときに、セッションは終了します。

Integration Service によりすべてのターゲットにメッセージがコミットされる前にセッションが失敗し、セッションをコールドスタートモードでリスタートした場合、ターゲットが重複行を受け取る可能性があります。

リカバリキューおよびリカバリトピック

Integration Service でリカバリが有効化されているリアルタイムセッションのリカバリキューまたはリカバリトピックにメッセージ ID が格納されるのは、そのリアルタイムセッションに以下のタイプのソースおよびターゲットが含まれる場合です。

- JMS ターゲットまたは WebSphere MQ ターゲットを使用する JMS ソース
- JMS ターゲットまたは WebSphere MQ ターゲットを使用する WebSphere MQ ソース

Integration Service は、JMS プロバイダまたは WebSphere MQ に作成したリカバリキューまたはリカバリトピックに一時的にメッセージ ID とコミット番号を格納します。コミット番号は、Integration Service からターゲットへのコミット数を示します。リカバリ中、Integration Service は、コミット番号を使用して、すべてのターゲットに同じ数のメッセージを書き込んだかどうかを判断します。メッセージ ID とコミット番号は、リカバリキューまたはリカバリトピックと照合して、データの喪失や重複がないことが確認されます。

Integration Service は、各セッションのすべてのキューターゲットに対して同じリカバリキューまたはリカバリトピックを使用します。セッションに対して複数のリカバリキューまたはリカバリトピックを作成すると、パフォーマンスが向上します。

セッションプロパティまたは JMS 接続オブジェクトでリカバリキュー名またはリカバリトピック名を指定しない場合、Integration Service により、リカバリ情報がリカバリファイルに格納されます。最適なパフォーマンス得るために、リカバリファイルではなく、リカバリキュー名またはリカバリトピック名を設定します。

メッセージの処理

Integration Service でリカバリキューまたはリカバリトピックを使用してメッセージを処理する方法は、リカバリテーブルを使用してメッセージを処理する方法と似ています。Integration Service は、リカバリテーブルの代わりに、リカバリキューまたはリカバリトピックにリカバリ情報を書き込みます。

メッセージリカバリ

Integration Service でリカバリキューまたはリカバリトピックからメッセージをリカバリする方法は、リカバリテーブルからメッセージをリカバリする方法と似ています。Integration Service は、リカバリテーブルの代わりに、リカバリキューまたはリカバリトピックからリカバリ情報を取得します。

リカバリ無視リスト

Integration Service は、JMS ソースまたは WebSphere MQ ソースを使用するセッションが失敗した場合、リカバリ情報をリカバリ無視リストに書き込みます。Integration Service では、ソースが受信確認メッセージを受信しなかった可能性がある場合に、リカバリ情報をこのリストに書き込みます。例えば、Integration Service によってメッセージがターゲットに書き込まれた後、受信確認メッセージがソースに送信される前に、セッションが失敗したとします。この場合、ソースでは現在のトランザクションをロールバックできますが、そのトランザクション内のメッセージは即座に利用できないことがあります。メッセージがリカバリセッションに含まれる場合、データの重複が発生することがあります。Integration Service では、データの重複を避けるためにリカバリ無視リストが作成されます。

リカバリ無視リストでは、失敗したセッションについて Integration Service がターゲットに書き込んだメッセージ ID が格納されます。Integration Service により、そのセッションに使用されるストレージタイプ（リカバリファイル、リカバリテーブル、リカバリキュー、またはリカバリトピックなど）にリカバリ無視リストが作成されます。リカバリ中、Integration Service によりリカバリ無視リストとストレージタイプが使用され、メッセージがターゲットに書き込まれたかどうかが判別されます。また、リカバリ無視リストとストレージタイプでメッセージ ID が照合され、データの重複がないことが確認されます。

セッションが失敗すると、Integration Service はリカバリ無視リストにメッセージを書き込み、タイムスタンプを追加します。デフォルトでは、Integration Service はタイムスタンプの 1 時間後にリカバリ無視リストからメッセージを削除します。Integration Service は、デフォルトの期間内にソースでメッセージを検出すると、リカバリ無視リストからメッセージを削除します。

停止したセッションまたは失敗したセッションをコールドスタートモードでリスタートすると、ターゲットで重複行を受け取ることがあります。データの重複を避けるためには、リカバリ付きでセッションをリスタートしてください。または、リカバリ無視リスト内のメッセージがソースから削除されたことが確実な場合は、セッションをコールドスタートモードをリスタートします。メッセージ ID を確認するには、セッションログを使用します。冗長データの追跡を設定している場合、Integration Service はリカバリ無視リスト内のメッセージ ID をセッションログに書き込みます。

リアルタイムセッションの停止

セッションが失敗したり、セッションを手動で停止しないかぎり、リアルタイムセッションは継続して実行されます。*pmcmd* または Workflow Monitor で stop コマンドを発行することで、セッションを停止することができます。定期的にメンテナンスを実行するためにセッションを停止する必要がある場合もあります。

リアルタイムセッションを停止すると、Integration Service は以下のリアルタイムソースに基づいてパイプライン内のメッセージを処理します。

- **JMS と WebSphere MQ。** Integration Service は stop が発行されるまでメッセージを読み込み、メッセージをターゲットに書き込みます。
- **MSMQ メッセージ、SAP メッセージ、TIBCO メッセージ、webMethods メッセージ、および Web サービスメッセージ。** Integration Service でターゲットにメッセージ全体が書き込まれないうちにユーザーがセッションを終了した場合、そのメッセージは処理されません。

JMS ソースまたは WebSphere MQ ソースを使用するリアルタイムセッションを停止すると、Integration Service は次のタスクを実行します。

1. Integration Service は、ソースからのメッセージの読み込みを停止します。
リアルタイムリカバリセッションを停止すると、Integration Service はすべてのメッセージをリカバリした後にソースからの読み込みを停止します。
2. Integration Service はパイプライン内のメッセージを処理し、それをターゲットに書き込みます。

3. Integration Service はソースに受信確認メッセージを送信します。
4. セッションをリスタートしたときのデータの重複を回避するために、Integration Service はリカバリファイルまたはリカバリをクリアします。

セッションをリスタートすると、Integration Service はソースからの読み込みを開始します。中断した時点からセッションを再開するためのセッションおよび操作のトランスフォーメーション状態をリストアします。

注: リアルタイムセッションを停止した後にそのセッションがハングする場合、セッションが停止状態に留まる可能性があります。リアルタイムセッションが停止モードに留まっている際には、そのセッションを強制終了することができます。Integration Service は、stop が発行される前に読み込んだメッセージを処理します。

リアルタイムセッションのリスタートおよびリカバリ

停止または失敗したリアルタイムセッションは、再開することができます。セッションを再開するには、セッションをリスタートするか、リカバリする必要があります。セッションの自動タスクリカバリを有効にした場合、Integration Service によるセッションの自動リカバリが可能になります。

次の節では、リアルタイムセッション固有のリカバリ情報について説明します。

リアルタイムセッションのリスタート

セッションをリスタートすると、Integration Service はリアルタイムソースに基づいてセッションを再開します。リアルタイムソースに応じてリカバリありまたはなしで、セッションをリスタートします。

タスクまたはワークフローをコールドスタートモードでリスタートできます。タスクまたはワークフローをコールドスタートモードでリスタートすると、Integration Service によりリカバリ情報が破棄され、タスクまたはワークフローがリスタートされます。

リアルタイムセッションのリカバリ

セッションのリカバリを有効にすると、失敗または強制終了したセッションのリカバリが可能になります。セッションをリカバリすると、Integration Service は中断した地点からメッセージの処理を続行します。Integration Service はリアルタイムソースに従ってメッセージをリカバリします。

Integration Service では、以下のタイプのセッションリカバリが使用されます。

- **自動リカバリ。** 終了したタスクを自動的にリカバリするようにワークフローを設定した場合、Integration Service によってセッションが再開されます。Integration Service では、未処理のデータがすべてリカバリされ、リアルタイムソースに関係なくセッションが再開されます。
- **手動リカバリ。** Workflow Monitor または Workflow Manager のメニューコマンド、または *pmcmd* コマンドを使用して、セッションをリカバリします。リアルタイムソースによっては、セッションをリカバリしてから再開する必要があります。そうしないと、失敗したセッションからのメッセージが Integration Service で処理されません。

リスタートコマンドおよびリカバリコマンド

Workflow Manager、Workflow Monitor、または *pmcmd* を使用してセッションをリスタートまたはリカバリできます。Integration Service はリアルタイムソースに基づいてセッションを再開します。

以下の表に、次のコマンドを使用してセッションをリスタートまたはリカバリしたときの動作を示します。

コマンド	説明
<ul style="list-style-type: none"> - タスクのリスタート - ワークフローのリスタート - タスクからワークフローをリスタートする 	<p>タスクまたはワークフローをリスタートします。JMS セッションおよび WebSphere MQ セッションでは、Integration Service はタスクまたはワークフローをリカバリしてからリスタートします。</p> <p>注: JMS ソース、WebSphere MQ ソース、および別のリアルタイムソースがセッションに含まれている場合、Integration Service はすべてのリアルタイムソースに対してリカバリを実行してから、タスクまたはワークフローをリスタートします。</p>
<ul style="list-style-type: none"> - タスクのリカバリ - ワークフローのリカバリ - このタスクのリカバリによるワークフローのリスタート 	<p>タスクまたはワークフローをリカバリします。</p>
<ul style="list-style-type: none"> - タスクのコールドスタート - ワークフローのコールドスタート - タスクからワークフローをコールドスタート 	<p>リカバリ情報を破棄し、タスクまたはワークフローをリスタートします。</p>

リアルタイムセッションに関するルールおよびガイドライン

リアルタイムセッションを実行する場合、次のルールおよびガイドラインに従ってください。

- マッピングにトランザクション制御トランスフォーメーションが入っていると、セッションが失敗します。
- トランザクションの生成が有効となっている任意のトランスフォーメーションがマッピングに入っていると、セッションが失敗します。
- すべての入力に対して設定されたトランスフォーメーション範囲を持つ任意のトランスフォーメーションがマッピングに入っていると、セッションが失敗します。
- 行トランスフォーメーション範囲を持ち、複数のトランザクション制御ポイントから入力を受信する任意のトランスフォーメーションがマッピングに入っていると、セッションが失敗します。
- ターゲット用にロード範囲が「すべての入力」に設定されている場合、セッションが失敗します。
- セッションをデバッグモードで実行する場合、Integration Service はフラッシュ待ち時間を無視します。
- マッピングにリレーショナルターゲットが含まれる場合、ターゲットのロードタイプをノーマルに設定します。
- マッピングに XML ターゲット定義が含まれる場合は、ターゲット定義の「コミット時」オプションで「ドキュメントへの書き込み」を選択します。
- Integration Service は、WebSphere MQ および JMS への接続失敗に対する復元性があります。他のメッセージシステムに対する復元性はありません。
- Web サービスなどで、リアルタイムセッションに要求と応答が含まれる場合、セッションログにはその要求と応答の開始時間と終了時間が含まれます。リアルタイムセッションにバブリッシュ/サブスクライブまたは P2P アーキテクチャが含まれる場合、セッションログには Integration Service によって、行がターゲットに対していつコミットされるかを説明する統計が含まれます。

メッセージリカバリに関するルールおよびガイドライン

メッセージのリカバリが有効で、以下のいずれかの条件に一致するセッションは失敗します。

- ソース定義がジョイントトランスフォーメーションのマスターソースである。
- 複数のソース定義を、同じターゲットロード順のグループに対して同時に実行されるように設定する。
- マッピングに XML ターゲット定義が含まれている。
- セッションをリスタートする前にリカバリファイルを編集し、リスタートまたは再開のリカバリ戦略を使用するセッションを実行する。
- Integration Service は、リカバリキューまたはリカバリトピックに接続できない。
- Integration Service は、リカバリメッセージをリカバリキューまたはリカバリトピックに書き込まない。

Integration Service がメッセージキューに対して読み込む、または書き込むメッセージの数がメッセージサイズの制限を超えた場合は、メッセージサイズの制限を上げるか、またはフラッシュ待ち時間を短縮してください。

リアルタイム処理の例

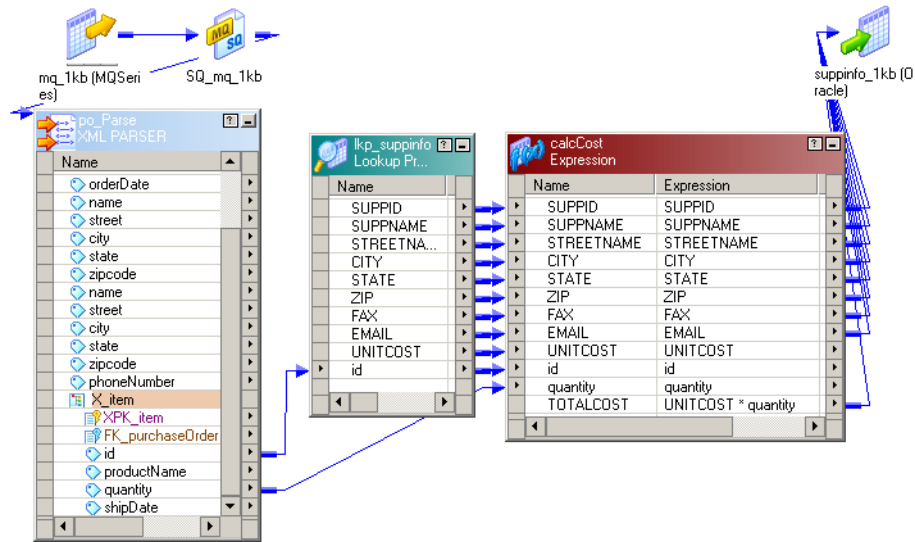
以下の例に、IBM WebSphere MQ 対応の PowerExchange および PowerCenter を使用するリアルタイムデータの処理方法を示します。

リアルタイムに購入注文を処理することを考えています。購入注文には、複数のサプライヤからの複数の品目が含まれていることがあります。ただし、購入注文にはサプライヤまたは品目原価は含まれません。購入注文を受けたら、サプライヤごとに総額を計算する必要があります。使用しているマスターデータベースには、サプライヤおよびそれに対応する品目と品目原価が含まれています。PowerCenter を使用して、品目 ID に基づいてサプライヤと品目原価を調べます。また、PowerCenter を使用して、サプライヤの総額をリレーショナルデータベースに書き込みます。

データベース管理者からは、ターゲットの更新時には 1 回のコミットで最大 1,000 メッセージを扱うことを勧められています。また、ターゲットが常に最新の状態になるように、2,000 ミリ秒ごとにターゲットを更新したいと考えています。

リアルタイムに購入注文を処理するには、マッピングを作成して設定します。

以下の図に、リアルタイムに購入注文を処理するマッピングを示します。



マッピングの例には、以下のコンポーネントが含まれます。

- **ソース。** WebSphere MQ。各メッセージは XML 形式で、それぞれに 1 つの購入注文が含まれます。
- **XML パーサトランスフォーメーション。** MQ ソース修飾子トランスフォーメーションから購入注文情報を受け取ります。XML ファイルから購入注文 ID および数量をパースします。
- **ルックアップトランスフォーメーション。** 購入注文 ID に対応するサプライヤの詳細を検索します。 サプライヤ情報、購入品目 ID、および品目原価を式トランスフォーメーションに渡します。
- **式トランスフォーメーション。** サプライヤの発注費を計算します。
- **ターゲット。** Oracle リレーショナルデータベース。 サプライヤ情報およびサプライヤの総額が含まれます。

以下のプロパティを使用して、セッションおよびワークフローを作成して設定します。

プロパティ	値
メッセージカウント	1,000
フラッシュ待ち時間間隔	2,000 ミリ秒
コミットタイプ	ソースベースのコミット
ワークフロースケジュール	継続的に実行

以下の手順は、Integration Service がリアルタイムでセッションを処理する方法を示します。

1. Integration Service は、1,000 メッセージに達するか、または 2,000 ミリ秒が経過するまで、WebSphere MQ キューからメッセージを読み込みます。どちらかの条件が満たされたら、WebSphere MQ キューからの読み込みを停止します。
2. Integration Service では、サプライヤ情報を調べて発注費を計算します。
3. Integration Service によって、サプライヤ情報および発注費が Oracle リレーショナルターゲットに書き込まれます。
4. Integration Service では、WebSphere MQ キューからのメッセージの読み込みを再開します。

5. ワークフローを継続的に実行するように設定しているため、Integration Service は手順 [1-4](#) を繰り返します。

PowerCenter リアルタイム製品

PowerCenter でのリアルタイムデータの読み込み、変換、および書き込みには、以下の製品を使用できます。

- **PowerExchange Client for PowerCenter.** PowerExchange を使用して、i5/OS、Linux、UNIX、Windows、および z/OS システム上のさまざまなリレーショナルソースおよび非リレーショナルソースから変更データをキャプチャします。PowerExchange 変更データキャプチャ (CDC) は、リアルタイムモードでエンタープライズ全体の変更データをキャプチャ、トランスフォーム、配信するために PowerCenter と統合されています。

PowerExchange Client for PowerCenter (PWXPC) ソフトウェアは PowerCenter と一緒にインストールされ、PowerCenter 統合サービスおよび PowerCenter Client ツールと連携して PowerExchange リスナーから変更データを取得します。Listener はソースシステム上で実行される場合もソースシステム外で実行される場合もあります。PowerExchange ターゲットをワークフローに含めて、i5/OS および z/OS システム上のターゲットなど、PowerCenter がサポートしないターゲットタイプに変更データを書き込むことができます。また、PowerExchange を使用してバルクデータをバッチモードで抽出してロードし、CDC ターゲットをマテリアライズまたは更新することもできます。

- **PowerCenter 用の PowerExchange for JMS.** PowerExchange for JMS は、JMS ソースからの読み込みと JMS ターゲットへの書き込みに使用します。メッセージトピックに基づいて、JMS メッセージ、JMS プロバイダメッセージキュー、または JMS プロバイダから読み込むことができます。メッセージトピックに基づいて、JMS プロバイダメッセージキューまたは JMS プロバイダに書き込むことができます。

JMS プロバイダは、メッセージベースのミドルウェアシステムであり、JMS メッセージを送受信できます。セッション中に Integration Service は Java Naming and Directory Interface (JNDI) に接続し、接続情報を決定します。Integration Service は接続情報を決定すると、JMS プロバイダに接続して JMS メッセージを読み書きします。

- **PowerCenter 用の PowerExchange for WebSphere MQ.** PowerExchange for WebSphere MQ は、WebSphere MQ メッセージキューからの読み取り、および WebSphere MQ メッセージキューまたはデータベースターゲットへの書き込みに使用します。PowerExchange for WebSphere MQ は、データの抽出およびロード処理中に、WebSphere MQ キューマネージャ、メッセージキュー、および WebSphere MQ メッセージとやり取りします。

- **PowerCenter 用の PowerExchange for TIBCO.** PowerExchange for TIBCO は、TIB/Rendezvous または AE フォーマットでの TIBCO からのメッセージの読み込みと TIBCO へのメッセージの書き込みに使用します。

Integration Service は TIBCO メッセージを TIBCO デーモンから受信し、TIBCO デーモンを介してメッセージを書き込みます。TIBCO デーモンは、ローカルエリアネットワークまたは広域ネットワークでターゲットメッセージを送信します。ターゲットリスナーはメッセージの件名に基づいて TIBCO ターゲットメッセージにサブスクライブします。

- **PowerCenter 用の PowerExchange for webMethods.** PowerExchange for webMethods は、webMethods ソースからのドキュメントの読み込みと webMethods ターゲットへのドキュメントの書き込みに使用します。

Integration Service は、webMethods ドキュメントの送信、受信、およびキューイングを行う webMethods Broker に接続します。Integration Service は、定義されたドキュメントタイプまたはクライアント ID に基づいて、webMethods ドキュメントの読み書きを行います。Integration Service は、webMethods 要求/応答ドキュメントの読み書きも行います。

- **PowerCenter 用の PowerExchange for MSMQ。**PowerExchange for MSMQ は、MSMQ ソースからの読み込みと MSMQ ターゲットへの書き込みに使用します。

Integration Service は Microsoft Messaging Queue に接続し、メッセージからデータを読み込むかまたはデータをメッセージに書き込みます。キューは public または private であり、トランザクショナルまたは非トランザクショナルです。

- **PowerCenter 用の PowerExchange for SAP NetWeaver。**PowerExchange for SAP NetWeaver は、Application Link Enabling (ALE) を使用して、送信 IDOC を使用した SAP からの読み込みを行うため、あるいは受信 IDOC を使用した SAP への書き込みを行うために使用します。

統合サービスは、送信 IDOC からの読み込みとリレーショナルターゲットへの書き込みができます。統合サービスはリレーショナルソースからデータを読み込んだり、受信 IDOC にデータを書き込んだりすることができます。統合サービスは、SAP アプリケーションデータベース内のマスターデータまたはトランザクショナルデータへの変更をリアルタイムで取得できます。

- **PowerCenter の Web Services Provider。**PowerCenter の Web Services Provider は、Web サービス Hub を介してトランスフォーメーションロジックをサービスとして公開したり、リアルタイム Web サービスを実行するためのクライアントアプリケーションを開発したりするために使用します。Web サービスクライアントからメッセージを受信してから変換し、それを PowerCenter がサポートする任意のターゲットに書き込むためのサービスマッピングを作成できます。また、Web サービスクライアントからメッセージ要求を受信し、データを変換してから Web サービスクライアントに応答を返信するための Web サービスのソース定義とターゲット定義を持つサービスマッピングを作成できます。

Web Services Hub は Web サービスクライアントから要求を受信して、それをゲートウェイに渡します。Integration Service またはリポジトリサービスは要求を処理し、応答を Web Services Hub 経由で Web サービスクライアントに送信します。

第 7 章

コミットポイント

この章では、以下の項目について説明します。

- [コミットポイントの概要, 154 ページ](#)
- [ターゲットベースのコミット, 155 ページ](#)
- [ソースベースのコミット, 155 ページ](#)
- [ユーザー定義コミット, 159 ページ](#)
- [トランザクション制御について, 162 ページ](#)
- [コミットプロパティの設定, 168 ページ](#)

コミットポイントの概要

コミットの間隔とは、セッション中に Integration Service がターゲットにデータをコミットする間隔です。コミットポイントは、コミットの間隔、コミットの間隔のタイプ、およびバッファブロックサイズの要素となります。コミットの間隔は、コミットポイントの基礎として使用する行の数です。コミットの間隔のタイプは、コミットポイントの基礎として使用する行の種類です。以下のコミットのタイプから選択することができます。

- **ターゲットベースのコミット。** Integration Service では、ターゲット行の数およびターゲットテーブルのキー制約に基づいてデータがコミットされます。コミットポイントは、バッファブロックサイズ、コミットの間隔、および Integration Service の書き込みタイムアウト設定によっても異なります。
- **ソースベースのコミット。** Integration Service ではソース行の数に基づいてデータがコミットされます。コミットポイントは、セッションプロパティで設定するコミット間隔です。
- **ユーザー定義コミット。** Integration Service では、マッピングプロパティで定義されたトランザクションに基づいてデータがコミットされます。また、セッションプロパティでコミットやロールバックのオプションを設定することもできます。

ソースベースおよびユーザー定義のコミットのセッションには、パーティション化の制約があります。複数のパーティションを持つセッションでソースベースまたはユーザー定義コミットを使用するように設定すると、パイプライン内の特定のパーティションポイントではパススルーパーティション化を選択できます。

ターゲットベースのコミット

ターゲットベースのコミットのセッションでは、Integration Service は、ターゲット行の数およびターゲットテーブルのキー制約に基づいて行をコミットします。コミットポイントは次の要因により決定されます。

- **コミット間隔。**コミットの基礎として使用する行数です。セッションプロパティでターゲットのコミット間隔を設定します。
- **書き込み待ちタイムアウト。**Writer がコミットを発行するまでの待ち時間。書き込み待ちタイムアウトは、Integration Service の設定時に設定します。
- **バッファブロック。**セッション中にデータ行を保持しているメモリのブロック。セッションプロパティにバッファブロックサイズを設定できますが、ブロックに保存する行数は設定できません。

ターゲットベースのコミットのセッションを実行すると、Integration Service は、設定されたコミット間隔の到達前、到達時、または到達後にコミットを発行する場合があります。Integration Service は、以下のプロセスによってコミットを発行します。

- コミット間隔に達しても、Integration Service は writer バッファブロックへの書き込みを継続します。writer バッファブロックがいっぱいになると、Integration Service はコミットを発行します。
- コミット間隔の到達前に writer バッファが一杯になると、Integration Service はターゲットに書き込みますが、コミット発行のタイミングを待ちます。次のいずれか条件が True の場合にコミットを発行します。
 - Integration Service の writer 待ちタイムアウトオプションで指定した時間中、writer が空状態である。
 - Integration Service がコミット間隔に到達し、さらに別の writer バッファに書き込みをしている。

注: XML ターゲットを含んでいるセッションに対してターゲットベースのコミットを選択した場合、Workflow Manager では、[マッピング] タブのトランスフォーメーションビューの [コミット時] セッションプロパティが無効にされます。

ソースベースのコミット

ソースベースのコミットのセッションの場合、Integration Service はターゲットロード順グループのアクティブソースから取り出された行の数に基づいてターゲットにデータをコミットします。これらの行は、ソース行と呼ばれます。

Integration Service はソースベースコミットのセッションを実行するときに、マッピング内のパイプラインごとにコミットソースを識別します。Integration Service は、コミットの間隔ごとに、アクティブソースからコミット行を生成します。Integration Service は、ソースベースのコミットの間隔に使用されるトランスフォーメーションの名前もセッションログに書き込みます。

Source-based commit interval based on... TRANSFORMATION_NAME

Integration Service は、アクティブソースで作成される行数よりも少ない行をターゲットにコミットする場合もあります。たとえば、アクティブソースを通じて 10,000 行を渡すソースベースコミットのセッションがあり、トランスフォーメーションのロジックによって 3,000 行が削除されるとします。Integration Service は、残りの 7,000 行がターゲットにロードされた時点でターゲットにコミットを発行します。

writer のバッファに保持される行数は、ソースベースコミットのセッションのコミットポイントには影響しません。たとえば、アクティブソースを通じて 10,000 行を渡すソースベースコミットのセッションがあるとして、この 10,000 行がターゲットにロードされると、Integration Service はコミットを発行します。セッションが正常に完了した場合、Integration Service はソース行が 10,000 行、20,000 行、30,000 行、40,000 行の時点でそれぞれコミットを発行します。

ターゲットが同じトランザクション制御単位に属する場合には、データは同時にターゲットにコミットされます。セッションが失敗したり強制終了されたりすると、トランザクション制御単位の未コミットのデータはすべて同じソース行にロールバックされます。

ターゲットが別々のトランザクション制御単位に属する場合は、コミットは各ターゲットがコミット行を受け取るたびに実行されます。セッションが失敗したり強制終了されたりすると、各ターゲットは最後のコミットポイントにロールバックされます。別々のトランザクション制御単位に属するターゲットの場合、同じソース行にロールバックされない場合があります。

注: セッションで 1 対 1 のマッピングを使用している場合、ソースベースのコミットを使うとセッションのパフォーマンスが低下する場合があります。1 対 1 のマッピングとは、ソース修飾子、XML ソース修飾子、またはアプリケーションソース修飾子トランスフォーメーションから直接ターゲットにデータを移動するマッピングです。

コミットソースの決定

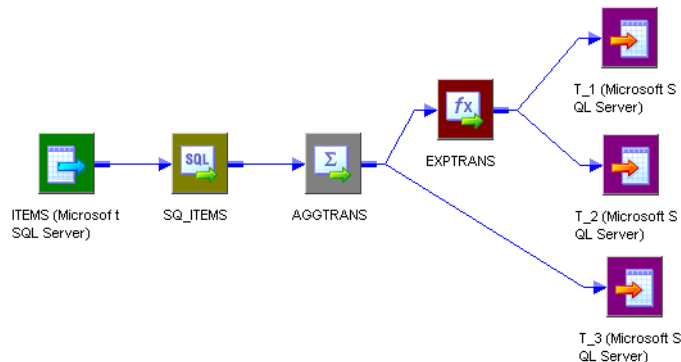
ソースベースのコミットのセッションを実行した場合、統合サービスでは、トランザクション境界がプロパゲートされないすべてのソース修飾子およびトランスフォーメーションでコミットが生成されます。これには、以下のアクティブソースが含まれます。

- ソース修飾子
- アプリケーションソース修飾子
- MQ ソース修飾子
- 1 つの出力グループからのポートのみを接続する場合の XML ソース修飾子
- ノーマライザ (VSAM)
- [すべての入力] トランスフォーメーション範囲を使用するアグリゲータ
- [すべての入力] トランスフォーメーション範囲を使用するジョイナ
- [すべての入力] トランスフォーメーション範囲を使用するリンク
- [すべての入力] トランスフォーメーション範囲を使用するソータ
- 1 つの出力グループと [すべての入力] トランスフォーメーション範囲を使用する、カスタムトランスフォーメーション
- 先行する複数のトランザクション制御ポイントに接続された 1 つの出力グループを使用する、複数入力グループトランスフォーメーション
- マプレット (上記のトランスフォーメーションのいずれかが含まれる場合)

マッピングは 1 つ以上のターゲットロード順グループを持つことができ、ターゲットロード順グループはコミットを生成する 1 つ以上のアクティブソースを持つことができます。統合サービスでは、ターゲット定義に最も近いアクティブソースによって生成されたコミットが使用されます。このコミットはコミットソースと呼ばれています。

次の図に、単一コミットソースのマッピングを示します。

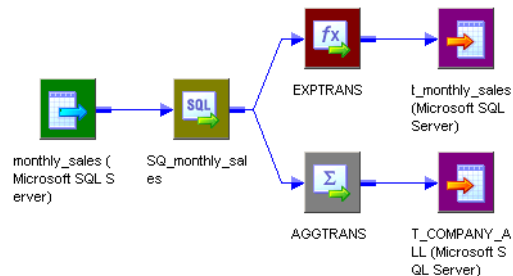
図 2. 単一のコミットソースを使用するマッピング



このマッピングには、ソース修飾子トランスフォーメーションと、[すべての入力] トランスフォーメーション範囲を使用するアグリゲータトランスフォーメーションが含まれています。アグリゲータトランスフォーメーションはソース修飾子よりもターゲットに近いので、ソースベースのコミットセッションでコミットソースとして使用されます。

次の図に、複数のコミットソースのマッピングを示します。

図 3. 複数のコミットソースを使用するマッピング



[トランスフォーメーション範囲] プロパティは [すべての入力]

このマッピングのターゲットロード順グループが使用するソースパイプラインには、ソース修飾子トランスフォーメーションから 2 つのターゲットへのブランチがあります。パイプラインブランチの 1 つには、[すべての入力] トランスフォーメーション範囲を使用するアグリゲータトランスフォーメーションが含まれ、もう 1 つのブランチには Expression トランスフォーメーションが含まれています。統合サービスは、ソース修飾子トランスフォーメーションを t_monthly_sales のコミットソースとして、アグリゲータトランスフォーメーションを T_COMPANY_ALL のコミットソースとして識別します。両方のターゲットに対して 1 つのソースベースコミットを実行しますが、それぞれが異なるコミットソースを使用します。

ソースベースコミットからターゲットベースコミットへの切り替え

Integration Service によりコミットを生成するアクティブソースからコミットを受け取らないターゲットロード順グループのターゲットが特定された場合、Integration Service では、そのターゲットのみに対して、ターゲットベースのコミットが戻ります。

Integration Service は、ソースベースのコミット間隔に使用されるトランスフォーメーションの名前を、セッションログに書き込みます。Integration Service は、ターゲットベースのコミットに切り替えると、セッションログにメッセージを書き込みます。

以下の状況では、ターゲットがコミットソースからコミットを受け取れない場合があります。

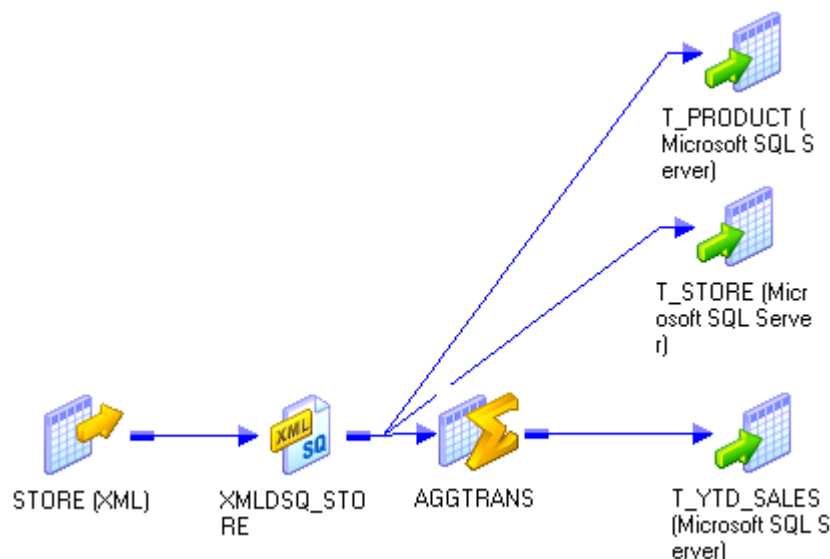
- ターゲットが XML ソース修飾子トランスフォーメーションからデータを受け取る場合で、XML ソース修飾子トランスフォーメーションの複数の出力グループをダウンストリームトランスフォーメーションに接続する場合。複数の出力グループダウンストリームを接続する場合は、XML ソース修飾子トランスフォーメーションではコミットは生成されません。
- ターゲットが、XML ソース修飾子トランスフォーメーション以外の複数の出力グループを持つアクティブソースからデータを受け取る場合。例えば、ターゲットはトランザクションを生成しないように設定したカスタムトランスフォーメーションからデータを受け取ります。複数の出力グループのアクティブソースは、コミットの生成やプロパゲートを行いません。

XML ソースをマッピング内で接続

複数の出力グループを後続のトランスフォーメーションに接続しているときは、XML ソース修飾子トランスフォーメーションはコミットを生成しません。XML ソース修飾子トランスフォーメーションをマッピング内で接続すると、マッピング内で使用するトランスフォーメーションによっては、統合サービスはこのセッションのターゲットごとに異なるコミットタイプを使用できます。

- XML ソース修飾子トランスフォーメーションとターゲット間にコミットソースを置く場合。統合サービスはコミットソースからコミットを受け取るので、ターゲットに対してソースベースのコミットを使用します。アクティブソースは、ターゲットに対するコミットソースです。
- XML ソース修飾子トランスフォーメーションとターゲット間にコミットソースを置かない場合。統合サービスはコミットを受け取らないので、ターゲットに対してターゲットベースのコミットを使用します。

次の図に、XML ソース修飾子トランスフォーメーションのマッピングを示します。



このマッピングには、複数の出力グループを後続のトランスフォーメーションに接続している XML ソース修飾子トランスフォーメーションが含まれています。複数の出力グループを後続に接続しているため、XML ソース修飾子トランスフォーメーションはコミットを生成しません。XML ソース修飾子トランスフォーメーションは、2つのリレーショナルターゲット（T_STORE と T_PRODUCT）に接続されています。したがって、これらのターゲットは、アクティブソースが生成したコミットを受け取りません。統合サービスでは、これらのターゲットへロードする際にターゲットベースのコミットを使用します。

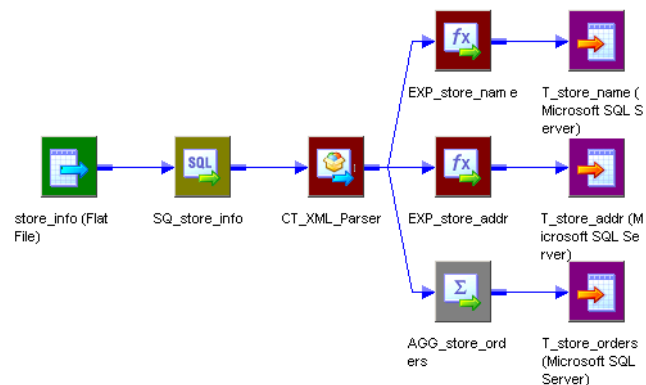
ただし、XML ソース修飾子トランスフォーメーションと T_YTD_SALES の間には、コミットを生成するアクティブソースの AGG_Sales が含まれています。統合サービスは、T_YTD_SALES へロードするときにソースベースのコミットを使用します。

複数の出力グループのあるカスタムトランスフォーメーションをマッピング内で接続

トランザクションを生成しないように設定した複数の出力グループのあるカスタムトランスフォーメーションは、コミットの生成や反映を行いません。そのため、マッピング内で使用するトランスフォーメーションによっては、統合サービスはこのセッション内のターゲットごとに異なるコミットタイプを使用できます。

- **カスタムトランスフォーメーションとターゲット間にコミットソースを置く場合。**統合サービスはアクティブソースからコミットを受け取るので、ターゲットに対してソースベースのコミットを使用します。アクティブソースは、ターゲットに対するコミットソースです。
- **カスタムトランスフォーメーションとターゲット間にコミットソースを置かない場合。**統合サービスはコミットを受け取らないので、ターゲットに対してターゲットベースのコミットを使用します。

次の図に、複数の出力グループのあるカスタムトランスフォーメーションのマッピングを示します。



このマッピングは複数の出力グループのカスタムトランスフォーメーションである CT_XML_Parser を含み、これによりソース修飾子トランスフォーメーションによって生成されたコミットが削除されます。そのため、ターゲットの T_store_name および T_store_addr は、アクティブソースが生成したコミットを受け取りません。統合サービスでは、これらのターゲットへロードする際にターゲットベースのコミットを使用します。

ただし、このマッピングにはカスタムトランスフォーメーションと T_store_orders の間に、コミットを生成するアクティブソースである AGG_store_orders が含まれています。AGG_store_orders のトランスフォーメーション範囲は「すべての入力」です。統合サービスでは、T_store_orders にロードする際にソースベースのコミットが使用されます。

注: カスタムトランスフォーメーションプロシージャがトランザクションを出力する場合、トランザクションを生成するようにカスタムトランスフォーメーションを設定できます。その際は、セッションをユーザー定義のコミット対応に設定してください。

ユーザー定義コミット

ユーザー定義のコミットセッション中に、Integration Service は、トランザクション制御トランスフォーメーションを通過する行または行のセットに基づいて、トランザクションのコミットおよびロールバックを行います。Integration Service は、トランスフォーメーションに入る行ごとにトランザクション制御式を評価します。トランザクション制御式の戻り値は、コミットまたはロールバックポイントを定義します。

トランザクションを生成するように設定されているカスタムトランスフォーメーションがマッピングに含まれている場合にも、ユーザー定義コミットセッションを作成できます。ユーザー定義コミットセッションを作成した場合、カスタムトランスフォーメーションに関連付けられた手順によりトランザクション境界が定義されます。

コミット行と評価されると、トランザクション内のすべての行がターゲットにコミットされます。ロールバック行と評価されると、トランザクションのすべての行がターゲットからロールバックされます。Integration Service は、各コミットおよびロールバックのポイントで、セッションログにメッセージを書き込みます。セッションの詳細は蓄積されます。以下のメッセージは、セッションログからのサンプルコミットメッセージです。

```
WRITER_1_1_1> WRT_8317
USER-DEFINED COMMIT POINT Wed Oct 15 08:15:29 2003
```

```
=====
WRT_8036 Target: TCustOrders (Instance Name: [TCustOrders])
WRT_8038 Inserted rows - Requested: 1003      Applied: 1003      Rejected: 0      Affected: 1023
```

Integration Service は、トランザクション内のすべての行をすべてのターゲットに書き込む場合、ターゲットごとにコミットを順次発行します。

Integration Service は、トランザクション制御式の戻り値またはエラー処理設定に基づいて、データをロールバックします。トランザクション制御式によりロールバック値が返されると、トランザクションがロールバックされます。エラーが発生した場合は、次のコミットポイントでのコミットまたはロールバックの選択が可能です。

トランザクション制御式がコミット、ロールバック、または継続以外の値に評価されると、Integration Service はセッションに失敗します。

セッション完了時に、コミット行によってバインドされなかったターゲットにデータが書き込まれる場合があります。このオープントランザクションをエンドオブファイルでコミットするか、またはロールバックすることができます。

注: ユーザー定義のコミットのセッションで一括ロードを使用している場合は、ターゲットがトランザクション境界を認識しない場合があります。ターゲット接続グループがトランザクションをサポートしない場合、Integration Service は以下のメッセージをセッションログに書き込みます。

```
WRT_8324 Warning: Target Connection Group's connection doesn't support transactions. Targets may not be loaded
according to specified transaction boundaries rules.
```

トランザクションのロールバック

Integration Service では、以下の状況でトランザクションがロールバックされます。

- **ロールバック評価。** トランザクション制御式がロールバック値を返します。
- **オープントランザクション。** ファイルの末尾でロールバックを選択します。
- **エラー時のロールバック。** Integration Service により非致命的エラーが検出された場合には、コミットトランザクションのロールバックを選択します。
- **コミット失敗時のロールバック。** トランザクション制御単位内のターゲット接続グループがコミットに失敗した場合、Integration Service によりすべてのコミットされていないデータが、最後に成功したコミットポイントにロールバックされます。

ロールバック評価

トランザクション制御式によりロールバック値が返された場合、Integration Service はそのトランザクションをロールバックし、トランザクションがロールバックされたことを示すメッセージをセッションログに書き込みます。このメッセージにはロールバックされた行数も示されます。

以下のメッセージは、トランザクション制御式によりロールバック値が返された場合に、Integration Service がセッションログに書き込むメッセージのサンプルです。

```
WRITER_1_1_1> WRT_8326 User-defined rollback processed
WRITER_1_1_1> WRT_8331 Rollback statistics
WRT_8162 =====
WRT_8330 Rolled back [333] inserted, [0] deleted, [0] updated rows for the target [TCustOrders]
```


オープントランザクションのロールバック

トランザクション制御式の最後の行が TC_CONTINUE_TRANSACTION に評価された場合、このセッションはオープントランザクションで完了します。このオープントランザクションのロールバックを選択すると、Integration Service はトランザクションをロールバックし、トランザクションがロールバックされたことを示すメッセージをセッションログに書き込みます。

以下のメッセージは、セッションプロパティで「ファイルの最後でコミット」が無効になっていることを示すサンプルメッセージです。

```
WRITER_1_1_1> WRT_8168 End loading table [TCustOrders] at: Wed Nov 05 10:21:56 2003
WRITER_1_1_1> WRT_8325 Final rollback executed for the target [TCustOrders] at end of load
```

以下のメッセージは、セッションプロパティで「ファイルの最後でコミット」が有効になっていることを示すサンプルメッセージです。

```
WRITER_1_1_1> WRT_8143
Commit at end of Load Order Group Wed Nov 05 08:15:29 2003
```

エラー時のロールバック

Integration Service が非致命的エラーを検出した場合は、トランザクションを次のコミットポイントでロールバックできます。Integration Service は非致命的エラーを検出すると、エラー行を処理し、トランザクション処理を続行します。トランザクション境界がコミット行である場合、Integration Service はトランザクション全体をロールバックし、拒否ファイルに書き込みます。

次の表で、ロールバックされたトランザクションに対する拒否ファイルの行インジケータについて説明します。

行インジケータ	説明
4	ロールバックされた挿入
5	ロールバックされた更新
6	ロールバックされた削除

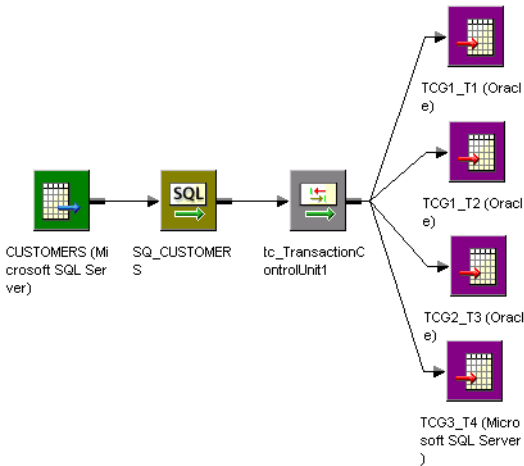
注: トランザクション制御トランスフォーメーションで行を処理する前にエラーが発生した場合、Integration Service はトランザクションをロールバックしません。

コミット失敗時のロールバック

トランザクション制御単位内のすべてのターゲットのコミットポイントに到達すると、Integration Service は各ターゲットのコミットを順次発行します。トランザクション制御単位内のターゲット接続グループのいずれかに対してコミットが失敗した場合、Integration Service は最後に成功したコミットポイントにすべてのデータをロールバックします。コミット済みトランザクションは、ロールバックできませんが、そのトランザクションは、拒否ファイルに書き込まれます。

例えば、トランザクション制御単位 1 つとターゲット接続グループ 3 つを使用してマッピングを作成するとします。ターゲット名には、ターゲット接続グループに関する情報が含まれています。TCG1_T1 は最初のターゲット接続グループと最初のターゲットを表します。

以下の図に、Integration Service がコミット失敗時にロールバックする場合の動作を示します。



Integration Service では、マッピングを処理する場合に以下のロジックが使用されます。

1. Integration Service がすべてのターゲットの 3 番目のコミットポイントに到達しました。
2. 各ターゲットに対してコミットの発行を順次開始します。
3. Integration Service によって、TCG1_T1 と TCG1_T2 に対するコミットが正常に行われます。
4. TCG2_T3 へのコミットが失敗しました。
5. Integration Service は、TCG3_T4 に対するコミットを発行しません。
6. Integration Service は、TCG2_T3 と TCG3_T4 を 2 番目のコミットポイントにロールバックします。TCG1_T1 と TCG1_T2 は 3 番目のコミットポイントでコミットに成功しているので、2 番目のコミットポイントにロールバックすることはできません。
7. Integration Service は、TCG2_T3 および TCG3_T4 からの行を拒否ファイルに書き込みます。これらは、3 番目のコミットポイントに関連付けられたロールバック行です。
8. Integration Service は、TCG2_T1 および TCG3_T2 からの行を拒否ファイルに書き込みます。これらは、3 番目のコミットポイントに関連付けられたコミット行です。

以下の表に、失敗したトランザクション制御単位のコミットされたトランザクションに対する、拒否ファイル内の行インジケータを示します。

行インジケータ	説明
7	コミットされた挿入
8	コミットされた更新
9	コミットされた削除

トランザクション制御について

PowerCenter では、トランスフォーメーション処理時およびターゲットでのデータのコミットおよびロールバック時に Integration Service が使用するトランザクションを定義できます。入力行の数の変化に応じてトランザクションを定義できます。トランザクションとは、コミット行またはロールバック行、トランザクション

境界によってバインドされた行のセットです。トランザクション境界によってバインドできない行もあります。それはオープントランザクションです。オープントランザクションをエンドオブファイルでコミットするか、ロールバックするかについては、セッションの設定時に選択できます。

Integration Service では、入力行が、トランスフォーメーションに対して一度に 1 行ずつ、すべての行に対してトランザクション内で、またはすべてのソース行に対して一度に処理されます。トランザクション内のすべての行に対するトランスフォーメーションを処理することにより、アグリゲータなどのトランスフォーメーションをリアルタイムセッションで含めることができます。

トランザクション境界は、トランザクション制御ポイントから発生します。トランザクション制御ポイントは、トランザクション境界を以下の方法で定義または再定義するトランスフォーメーションです。

- **トランザクション境界の生成。**トランザクション境界を定義するトランスフォーメーションは、セッションコミットタイプによって異なります。
 - **ターゲットベースおよびユーザー定義のコミット。**トランザクションジェネレータは、トランザクション境界を生成します。トランザクションジェネレータは、コミット行とロールバック行の両方を生成するトランスフォーメーションです。トランザクションコントロールトランスフォーメーションとカスタムトランスフォーメーションは、トランザクションジェネレータです。
 - **ソースベースのコミット。**一部のアクティブソースはコミットを生成します。アクティブソースはロールバック行を生成しません。トランザクションジェネレータは、コミット行とロールバック行を生成します。
- **入力トランザクション境界の削除。**トランスフォーメーションが入力トランザクション境界を削除し、コミットを生成しなかった場合、Integration Service ではすべての行がオープントランザクションに出力されます。コミットを生成するすべてのアクティブソースおよびトランザクションジェネレータにより、入力トランザクション境界が削除されます。

トランスフォーメーション範囲

トランスフォーメーション範囲トランスフォーメーションプロパティを使用して、統合サービスがトランスフォーメーションロジックを入力データに適用する方法を設定できます。統合サービスがトランスフォーメーションを処理する場合は、トランスフォーメーション範囲およびマッピング設定に応じて、トランザクション境界を削除するか、トランザクション境界を保持します。

トランスフォーメーション範囲には、以下の値のいずれかを選択できます。

- **行。**トランスフォーメーションロジックを、一度に 1 つのデータ行ごとに適用します。データの行が他の行に依存していない場合は [行] を選択します。複数のアップストリームのトランザクション制御ポイントに接続されたトランスフォーメーションに対して [行] を選択した場合、統合サービスではトランザクション境界が削除され、オープントランザクションとして、トランスフォーメーションからすべての行が出力されます。単一のアップストリームのトランザクション制御ポイントに接続されたトランスフォーメーションに対して [行] を選択した場合、統合サービスではトランザクション境界が保持されます。
- **トランザクション。**トランスフォーメーションロジックをトランザクションのすべての行に適用します。データの行が同一トランザクション内のすべての行に依存し、他のトランザクションの行には依存していない場合には、[トランザクション] を選択します。[トランザクション] を選択した場合、統合サービスでは入力トランザクション境界が保持されます。新しいトランザクションを受け取ると、アグリゲータキャッシュまたはルックアップキャッシュなどのキャッシュをすべてリセットします。
複数入力グループトランスフォーメーションに [トランザクション] を選択する場合は、すべての入力グループを同一の先行するトランザクション制御ポイントに接続する必要があります。
- **すべての入力。**トランスフォーメーションロジックをすべての入力データに適用します。[すべての入力] を選択した場合、統合サービスでは入力トランザクション境界が削除され、オープントランザクションとしてトランスフォーメーションからすべての行が出力されます。データの行がソース内のすべての行に依存している場合は、[すべての入力] を選択します。

以下の表に、各トランスフォーメーションに対して利用可能なトランスフォーメーション範囲値を一覧表示します。

トランスフォーメーション	行	トランザクション	すべての入力
アグリゲータ	-	オプション。	デフォルト。 トランザクション制御ポイント
アプリケーションソース修飾子	該当なし トランザクション制御ポイント	-	-
カスタム	オプション。 コミットを生成するように設定されている場合、または先行する複数のトランザクション制御点に接続されている場合には、トランザクション制御ポイント。	オプション。 コミットを生成するように設定されている場合にはトランザクション制御ポイント。	デフォルト。 常にトランザクション制御ポイント。 出力グループを 1 つ使用している場合、またはコミットを生成するように設定されている場合、コミットを生成します。上記以外の場合、オープントランザクションが生成されます。
データマスク	デフォルト。読み取り専用です。	-	-
式	デフォルト。表示されません。	-	-
エクスターナルプロシージャ	デフォルト。表示されません。	-	-
フィルタ	デフォルト。表示されません。	-	-
HTTP	デフォルト。読み取り専用です。	-	-
Java	パッシブなトランスフォーメーションの場合にはデフォルト。	アクティブなトランスフォーメーションの場合にはオプション。	アクティブなトランスフォーメーションの場合にはデフォルト。
ジョイナ	-	オプション。	デフォルト。 トランザクション制御ポイント
ルックアップ	デフォルト。表示されません。	-	-
MQ ソース修飾子	該当なし トランザクション制御ポイント	-	-

トランスフォーマー ション	行	トランザクション	すべての入力
ノーマライザ (VSAM)	該当なし トランザクション制御ポ イント	-	-
ノーマライザ (リレ ーショナル)	デフォルト。表示されま せん。	-	-
ランク	-	オプション。	デフォルト。 トランザクション制御ポイ ント
ルータ	デフォルト。表示されま せん。	-	-
ソータ	-	オプション。	デフォルト。 トランザクション制御ポイ ント
シーケンスジェネレ ータ	デフォルト。表示されま せん。	-	-
ソース修飾子	該当なし トランザクション制御ポ イント	-	-
SQL	スクリプトまたはクエリ モード SQL トランスフォー メーションのデフォル ト。	オプション。 コミットを生成する ように設定されてい る場合にはトランザ クション制御ポイン ト。	オプション。
ストアードプロシージャ	デフォルト。表示されま せん。	-	-
トランザクション制 御	デフォルト。表示されま せん。 トランザクション制御ポ イント	-	-
共有体	デフォルト。表示されま せん。	-	-
構造化されていない データ	デフォルト。読み取り専 用です。	-	-
アップデートストラ テジ	デフォルト。表示されま せん。	-	-

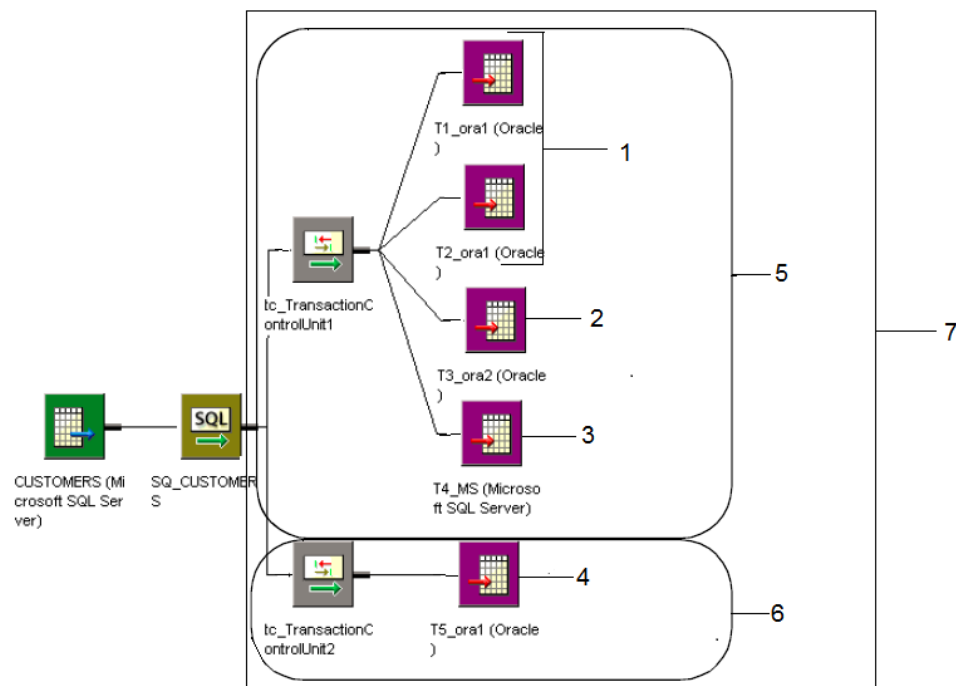
トランスフォーマー ション	行	トランザクション	すべての入力
XML ジェネレータ	-	オプション。 コミット時のフラッシュで新しいドキュメントを作成するように設定されている場合にはトランザクション。	デフォルト。表示されません。
XML パーサー	デフォルト。表示されません。	-	-
XML ソース修飾子	該当なし トランザクション制御ポイント	-	-

トランザクション制御単位について

トランザクション制御単位は、コミットを生成するアクティブソース、または有効なトランザクションジェネレータに接続されたターゲットのグループです。トランザクション制御単位は、ターゲットロード順グループのサブセットであるため、複数のターゲット接続グループを含むことができます。

トランザクション制御単位内のすべてのターゲットのコミットポイントに到達すると、Integration Service は各ターゲットのコミットを順次発行します。

次の図に、トランザクションコントロールトランスフォーメーションを設定したトランザクション制御単位を示します。



1. ターゲット接続グループ 1
2. ターゲット接続グループ 2
3. ターゲット接続グループ 3
4. ターゲット接続グループ 4
5. ターゲット接続単位 1
6. ターゲット接続単位 2
7. ターゲットのロード順グループ

この例では、T5_ora1 は T1_ora1 および T2_ora1 と同じ接続名を使用します。T5_ora1 は別のトランザクションコントロールトランスフォーメーションに接続されているので、別のトランザクション制御単位およびターゲット接続グループに属します。T5_ora1 を tc_TransactionControlUnit1 に接続すると、すべてのターゲットと同じトランザクション制御単位に含まれることになり、T1_ora1 および T2_ora1 と同じターゲット接続グループに入ります。

トランザクション制御に関する作業のルールおよびガイドライン

トランザクション制御を使用する際には、以下の規則とガイドラインについて考慮してください。

- [トランザクション] トランスフォーメーション範囲を使用するトランスフォーメーションは、1つのトランザクション制御ポイントからデータを受け取る必要があります。
- Integration Service では、トランザクショントランスフォーメーション範囲を使用するトランスフォーメーションに対して、最初のアップストリームのトランザクション制御ポイントによって定義されたトランザクション境界が使用されます。

- トランザクションジェネレータは、ターゲットに対して有効または無効にすることができます。Integration Service では、データをターゲットにロードする際に、有効なトランザクションジェネレータにより生成されたトランザクションが使用されます。
- Workflow Manager では、トランザクショントランスフォーメーション範囲を使用するアグリゲータトランスフォーメーションが含まれているセッションで、差分集計を使用できないようにします。
- すべての入力のトランスフォーメーション範囲を使用するトランスフォーメーションにより、トランザクションジェネレータは、ユーザー定義コミットセッションのターゲットに対して無効になります。
- Integration Service では、トランザクショントランスフォーメーション範囲を使用するアグリゲータ、ジョイナ、ランク、およびソータトランスフォーメーションの各トランザクションの最初にすべてのキャッシュがリセットされます。
- ソート済み入力を使用する場合は、ジョイナトランスフォーメーションにトランザクショントランスフォーメーション範囲を選択できます。
- トランザクショントランスフォーメーション範囲を使用するトランスフォーメーションでパーティションポイントを追加した場合、Workflow Manager ではデフォルトでパススルーパーティションタイプが使用されます。パーティションタイプは変更できません。

トランザクション別ターゲットファイルの作成

Integration Service が新規トランザクションを開始するたびに、個別の出力ファイルを生成できます。ターゲットフラットファイル名は動的に指定できます。

トランザクションごとに個別の出力ファイルを生成するには、FileName ポートをフラットファイルターゲット定義に追加します。マッピング内の FileName ポートを接続すると、PowerCenter ではコミットごとに個別のターゲットファイルに書き込みを行います。Integration Service では、各トランザクションの最初の行から FileName ポートの値が使用され、出力ファイルの名前が付けられます。

コミットプロパティの設定

セッションを作成するときに、コミットプロパティを設定できます。設定するプロパティは、Integration Service で実行するコミットのタイプやマッピングのタイプによって異なります。コミットプロパティは、[プロパティ] タブの [全般オプション] 設定で設定します。

以下の表に、[プロパティ] タブの [全般オプション] 設定で設定するセッションコミットプロパティを示します。

プロパティ	ターゲットベース	ソースベース	ユーザー定義
コミットタイプ	トランザクションジェネレータがない場合、または無効なトランザクションジェネレータのみがマッピングに存在する場合に、デフォルトで選択されます。	トランザクションジェネレータがない場合、または無効なトランザクションジェネレータのみがマッピングに存在する場合に、ソースベースのコミットに対して選択します。	マッピングに有効なトランザクションジェネレータがない場合に、デフォルトで選択されます。
コミット間隔	デフォルトは 10,000 です。	デフォルトは 10,000 です。	なし

プロパティ	ターゲットベース	ソースベース	ユーザー定義
ファイルの最後でコミット	エンドオブファイルでデータをコミットします。デフォルトで有効になっています。このオプションを無効にすることはできません。	エンドオブファイルでデータをコミットします。オープントランザクションがロールバックされるように設定するには、このオプションをクリアします。	エンドオブファイルでデータをコミットします。オープントランザクションがロールバックされるように設定するには、このオプションをクリアします。
Roll Back Transactions on Errors	<p>Integration Service により非致命的エラーが検出された場合は、次のコミットポイントでトランザクションのロールバックを選択できます。</p> <p>Integration Service によりトランスフォーメーションエラーが検出された際に、そのエラーがターゲットに対して有効なトランザクションジェネレータの後で発生した場合、Integration Service によりトランザクションがロールバックされます。</p>	<p>Integration Service により非致命的エラーが検出された場合は、次のコミットポイントでトランザクションのロールバックを選択できます。</p> <p>Integration Service によりトランスフォーメーションエラーが検出された際に、そのエラーがターゲットに対して有効なトランザクションジェネレータの後で発生した場合、Integration Service によりトランザクションがロールバックされます。</p>	<p>Integration Service により非致命的エラーが検出された場合は、次のコミットポイントでトランザクションのロールバックを選択できます。</p> <p>Integration Service によりトランスフォーメーションエラーが検出された際に、そのエラーがターゲットに対して有効なトランザクションジェネレータの後で発生した場合、Integration Service によりトランザクションがロールバックされます。</p>

ヒント: Microsoft SQL Server または Oracle ターゲットにバルクロードを行う場合、大きいコミット間隔を設定します。Microsoft SQL Server と Oracle ではコミットのたびに新しいバルクロードトランザクションが開始されます。コミット間隔を大きくすることで、バルクロードの回数を減らしてパフォーマンスを向上できます。

第 8 章

行エラーのロギング

この章では、以下の項目について説明します。

- [行エラーログの概要, 170 ページ](#)
- [エラーログテーブルについて, 171 ページ](#)
- [エラーログファイルについて, 177 ページ](#)
- [エラーログオプションの設定, 180 ページ](#)

行エラーログの概要

セッションを設定するときに、行エラーが一元的に記録されるように指定できます。行エラーが発生すると、Integration Service がエラー情報を記録するので、エラーの原因とソースを判断することができます。Integration Service はソース名、行 ID、現在の行のデータ、トランスフォーメーション、タイムスタンプ、エラーコード、エラーメッセージ、リポジトリ名、フォルダ名、セッション名、マッピング情報などの情報を記録します。

行エラーは、リレーショナルテーブルまたはフラットファイルに記録できます。エラーロギングを有効にした場合、Integration Service はセッションの初回実行時に、エラーテーブルまたはエラーログファイルを作成します。エラーログは、累積的に記録されます。Integration Service は、エラーログが存在する場合、エラーデータを既存のエラーログに追加します。

フラットファイルまたはリレーショナルソースのソース行データをログに出力できます。ソース行データには、エラーが発生したソース修飾子から、行データ、ソース行 ID、およびソース行タイプが含まれます。XML ファイルソースの行エラーをログに出力することはできません。XML ソースのエラーはセッションログで確認できます。

Integration Service は、複数のパーティションまたは次のいずれかのアクティブソースが存在する非パススルーパーティションポイントの後でエラーが発生した場合、Source Qualifier 内でどの行にエラーが存在するかを識別できません。

- アグリゲータ
- カスタム（アクティブなトランスフォーメーションとして設定されていること）
- ジョイナ
- ノーマライザ（パイプライン）
- ランク
- ソータ

デフォルトでは、トランスフォーメーションエラーはセッションログに、拒否された行は拒否ファイルに記録されます。エラーロギングを有効にした場合、Integration Service は、拒否ファイルを生成することも、削除

された行をセッションログに書き込むこともしません。拒否ファイルが存在しない場合、トランザクション制御トランスフォーメーションのロールバックエラーやコミットエラーが記録されません。行エラーログのほかに、行自体もセッションログに記録したい場合は、冗長データトレースを有効にします。

注: 行エラーを記録するときは、Integration Service は行ブロック単位ではなく行単位で処理を行うため、セッションのパフォーマンスが低下する場合があります。

エラーログのコードページ

Integration Service では、Integration Service プロセスのオペレーティングシステムに応じて異なる方法でエラーログファイルにデータが書き込まれます。

- **UNIX。** Integration Service では、Integration Service プロセスのコードページを使用してデータがエラーログファイルに書き込まれます。ただし、LogsInUTF-16LE の Integration Service プロパティを有効にすることにより、UTF-16LE を使用してエラーログファイルに書き込むように Integration Service を設定できます。
- **Windows。** Integration Service では、UTF-16LE エンコード形式を使用してエラーログファイルにすべての文字が書き込まれます。

エラーテーブルが存在するリレーショナルデータベースのコードページは、ターゲットのコードページのサブセットである必要があります。エラーログテーブルのコードページがターゲットのコードページのサブセットでない場合、Integration Service によりエラーログテーブルに矛盾したデータが書き込まれる場合があります。

エラーログテーブルについて

リレーショナルデータベースのエラーロギングを選択した場合、セッションの初回実行時に Integration Service によって以下のエラーテーブルが作成されます。

- **PMERR_DATA。** トランスフォーメーションの行エラーに関するデータとメタデータ、および対応するソース行を格納します。
- **PMERR_MSG。** エラーおよびエラーメッセージに関するメタデータを格納します。
- **PMERR_SESS。** セッションに関するメタデータを格納します。
- **PMERR_TRANS。** トランスフォーメーションエラーが発生した場合に、ソースポートおよびトランスフォーメーションポートに関するメタデータ（名前やデータタイプなど）を格納します。

テーブル作成先のデータベースに対し、データベース接続を指定する必要があります。特定のセッションについて既にエラーテーブルが存在する場合、これらのテーブルに対して行エラーが追加されます。

リレーショナルデータベースによるエラーロギングでは、複数のセッションで発生した行エラーを、特定のエラーテーブルに収集することができます。これを行うには、すべてのセッションに対して、エラーログテーブル名の接頭語を統一して指定します。生成されたエラーテーブルに対して Select 文を発行すれば、特定のセッションに限定してエラーデータを取得することができます。

エラーテーブルには接頭語を指定できます。エラーテーブル名に使用できる文字数は、最大 11 文字です。Oracle、Sybase、Teradata のエラーログテーブルに名前を付ける場合、19 文字を超える接頭語を指定してはいけません。これらのデータベースでは、テーブル名の最大長に 30 文字という制限があります。テーブル名の接頭語には、パラメータまたは変数を使用できます。パラメータファイルで定義可能なパラメータまたは変数タイプを使用します。たとえば、エラーログテーブル名の接頭語としてセッションパラメータ \$ParamMyErrPrefix を使用し、パラメータファイルでテーブルの接頭語に \$ParamMyErrPrefix を設定するとします。

Integration Service によって作成されるエラーテーブルには、プライマリキーと外部キーが指定されています。キーカラムは自分で指定することもできます。

PMERR_DATA

Integration Service は、行エラーが発生した場合、PMERR_DATA テーブルにエントリを挿入します。このテーブルには、トランスフォーメーションの行エラーに関するデータとメタデータ、および対応するソース行が格納されます。

以下の表に、PMERR_DATA テーブルの構造を示します。

カラム名	データタイプ	説明
REPOSITORY_GID	Varchar	リポジトリの一意の ID です。
WORKFLOW_RUN_ID	Integer	ワークフローの一意の ID です。
WORKLET_RUN_ID	Integer	ワークレットの一意の ID です。セッションがワークレットの一部ではない場合、この値は 0 になります。
SESS_INST_ID	Integer	セッションの一意の ID です。
TRANS_MAPPLET_INST	Varchar	エラーが発生したマプレットの名前です。
TRANS_NAME	Varchar	エラーが発生したトランスフォーメーションの名前です。
TRANS_GROUP	Varchar	エラーが発生した入力グループまたは出力グループの名前です。トランスフォーメーションにグループが含まれない場合、デフォルトによりメ input もまたはメ output ものいずれかになります。
TRANS_PART_INDEX	Integer	エラーが発生したトランスフォーメーションのパーティション番号を指定します。
TRANS_ROW_ID	Integer	最後のアクティブソースによって生成された行 ID を指定します。

カラム名	データタイプ	説明
TRANS_ROW_DATA	Long Varchar	<p>カラムインジケータなど、すべてのカラムデータを含む、区切り文字付きの文字列です。各種カラムインジケータを次に示します。</p> <p>D - 有効 N - NULL T - 切り詰め B - バイナリ U - データが利用不可</p> <p>カラムデータとカラムインジケータの固定区切り文字はコロン (:) になります。カラム間の区切り文字はパイプ () になります。カラム区切り文字は、エラー処理設定で上書きできます。</p> <p>エラーテーブルでは、すべてのカラムデータがテキスト文字列に変換されます。バイナリデータの場合は、Integration Service はカラムインジケータのみ使用します。</p> <p>この値は複数行にまたがる場合があります。データが 2000 バイトを超える場合は、Integration Service によって新しい行が作成されます。各カラムエラーエントリの行番号は、LINE_NO カラムに格納されます。</p>
SOURCE_ROW_ID	Integer	<p>ソース修飾子が、読み込んだ各行に割り当てる値です。Integration Service が行を識別できなかった場合、この値は-1 になります。</p>
SOURCE_ROW_TYPE	Integer	<p>行インジケータは、挿入、更新、またはリジェクトのいずれのマークが行に付いていたかを示します。</p> <p>0 - 挿入 1 - 更新 2 - 削除 3 - リジェクト</p>

カラム名	データタイプ	説明
SOURCE_ROW_DATA	Long Varchar	<p>カラムインジケータなど、すべてのカラムデータを含む、区切り文字付きの文字列です。各種カラムインジケータを次に示します。</p> <p>D - 有効 O - オーバーフロー N - NULL T - 切り詰め B - バイナリ U - データが利用不可</p> <p>カラムデータとカラムインジケータの固定区切り文字はコロン (:) になります。カラム間の区切り文字はパイプ () になります。カラム区切り文字は、エラー処理設定で上書きできます。</p> <p>Integration Service は、すべてのカラムデータをエラーテーブルまたはエラーファイルのテキスト文字列に変換します。バイナリデータの場合は、Integration Service はカラムインジケータのみ使用します。</p> <p>この値は複数行にまたがる場合があります。データが 2000 バイトを超える場合は、Integration Service によって新しい行が作成されます。各カラムエラーエントリの行番号は、LINE_NO カラムに格納されます。</p>
LINE_NO	Integer	SOURCE_ROW_DATA および TRANS_ROW_DATA において、複数行にまたがる各行エラーエントリの行番号を指定します。
注: テーブルを結合するには、太字のカラム名を使用します。		

PMERR_MSG

Integration Service は、行エラーが発生した場合、PMERR_MSG テーブルにエントリを挿入します。このテーブルには、エラーおよびエラーメッセージに関するメタデータが格納されます。

以下の表に、PMERR_MSG テーブルの構造を示します。

カラム名	データタイプ	説明
REPOSITORY_GID	Varchar	リポジトリの一意の ID です。
WORKFLOW_RUN_ID	Integer	ワークフローの一意の ID です。
WORKLET_RUN_ID	Integer	ワークレットの一意の ID です。セッションがワークレットの一部ではない場合、この値は 0 になります。
SESS_INST_ID	Integer	セッションの一意の ID です。
MAPPLET_INST_NAME	Varchar	トランスフォーメーションが属しているマプレットです。トランスフォーメーションがマプレットの一部ではない場合、この値は N/A になります。

カラム名	データタイプ	説明
TRANS_NAME	Varchar	エラーが発生したトランスフォーメーションの名前です。
TRANS_GROUP	Varchar	エラーが発生した入力グループまたは出力グループの名前です。トランスフォーメーションにグループが含まれない場合、デフォルトによりメ input もまたはメ output ものいずれかになります。
TRANS_PART_INDEX	Integer	エラーが発生したトランスフォーメーションのパーティション番号を指定します。
TRANS_ROW_ID	Integer	最後のアクティブソースによって生成された行 ID を指定します。
ERROR_SEQ_NUM	Integer	各トランスフォーメーショングループの行単位のエラー数を示すカウンタです。セッションに複数のパーティションが存在する場合、Integration Service は、このカウンタをパーティションごとに保持します。 たとえば、パーティション 1 のトランスフォーメーションで 3 つのエラーが、パーティション 2 のトランスフォーメーションで 2 つのエラーが生成された場合、パーティション 1 における ERROR_SEQ_NUM の値は 1、2、3 に、パーティション 2 における ERROR_SEQ_NUM の値は 1 および 2 になります。
ERROR_TIMESTAMP	日付/時刻	エラー発生時の Integration Service のタイムスタンプです。
ERROR_UTC_TIME	Integer	エラー発生時刻を協定世界時（グリニッジ標準時）で示します。
ERROR_CODE	Integer	エラーによって生成されるエラーコードです。
ERROR_MSG	Long Varchar	エラーメッセージです。複数行にまたがる場合があります。データが 2000 バイトを超える場合は、Integration Service によって新しい行が作成されます。各カラムエラーエントリの行番号は、LINE_NO カラムに格納されます。
ERROR_TYPE	Integer	発生したエラーのタイプです。Integration Service は、次の値を使用します。 1 - Reader エラー 2 - Writer エラー 3 - トランスフォーメーションエラー
LINE_NO	Integer	ERROR_MSG において、複数行にまたがる各行エラーエントリの行番号を指定します。
注: テーブルを結合するには、太字のカラム名を使用します。		

PMERR_SESS

リレーショナルデータベースによるエラーロギングを選択した場合、エントリが PMERR_SESS テーブルに挿入されます。このテーブルには、エラーが発生したセッションに関するメタデータが格納されます。

以下の表に、PMERR_SESS テーブルの構造を示します。

カラム名	データタイプ	説明
REPOSITORY_GID	Varchar	リポジトリの一意の ID です。
WORKFLOW_RUN_ID	Integer	ワークフローの一意の ID です。
WORKLET_RUN_ID	Integer	ワークレットの一意の ID です。セッションがワークレットの一部ではない場合、この値は 0 になります。
SESS_INST_ID	Integer	セッションの一意の ID です。
SESS_START_TIME	日付/時刻	セッション開始時の Integration Service のタイムスタンプです。
SESS_START_UTC_TIME	Integer	セッション開始時刻を協定世界時（グリニッジ標準時）で示します。
REPOSITORY_NAME	Varchar	セッション格納先のリポジトリ名です。
FOLDER_NAME	Varchar	マッピングおよびセッションが置かれたフォルダを指定します。
WORKFLOW_NAME	Varchar	ログgingsの対象となるセッションを実行するワークフローを指定します。
TASK_INST_PATH	Varchar	完全修飾セッション名です。複数行にまたがる場合があります。セッション名に対応する新しい行が Integration Service によって作成されます。また、修飾されたセッション名では、各ワークレットについて新しい行が作成されます。たとえば、WL1.WL2.S1 というセッション名があるとします。この名前の各構成コンポーネントは、次のように新しい行に出力されます。 WL1 WL2 S1 カラム番号は LINE_NO カラムに出力されます。
MAPPING_NAME	Varchar	セッションが使用するマッピングを指定します。
LINE_NO	Integer	TASK_INST_PATH において、複数行にまたがる各行エラーエントリの行番号を指定します。
注: テーブルを結合するには、太字のカラム名を使用します。		

PMERR_TRANS

Integration Service は、トランスフォーメーションエラーが発生した場合、PMERR_TRANS テーブルにエントリを挿入します。このテーブルには、ソースポートおよびトランスフォーメーションポートのメタデータ（名前やデータ型など）が格納されます。

以下の表に、PMERR_TRANS テーブルの構造を示します。

カラム名	データタイプ	説明
REPOSITORY_GID	Varchar	リポジトリの一意の ID です。
WORKFLOW_RUN_ID	Integer	ワークフローの一意の ID です。
WORKLET_RUN_ID	Integer	ワークレットの一意の ID です。セッションがワークレットの一部ではない場合、この値は 0 になります。
SESS_INST_ID	Integer	セッションの一意の ID です。
TRANS_MAPPLET_INST	Varchar	マプレットのインスタンスを指定します。
TRANS_NAME	Varchar	エラーが発生したトランスフォーメーションの名前です。
TRANS_GROUP	Varchar	エラーが発生した入力グループまたは出力グループの名前です。トランスフォーメーションにグループが含まれない場合、デフォルトによりメ input もまたはメ output モのいずれかになります。
TRANS_ATTR	Varchar	エラーが発生した入力グループまたは出力グループのポート名およびデータ型が列挙されます。ポート名とデータ型のペアがカンマ区切りで、「portname1:datatype, portname2:datatype」のように記録されます。 この値は複数行にまたがる場合があります。データが 2000 バイトを超える場合は、そのトランスフォーメーション属性に対応する新しい行が作成され、カラム番号が LINE_NO カラムに出力されます。
SOURCE_MAPPLET_INST	Varchar	ソースが存在するマプレットの名前です。
SOURCE_NAME	Varchar	ソース修飾子の名前です。ソース修飾子以外のアクティブソースまたは複数のパーティションを持つ非パススルーパーティションポイントの下流で行エラーが発生した場合、N/A が記録されます。
SOURCE_ATTR	Varchar	ソース修飾子内のエラーが発生した接続済みフィールドを列挙します。複数のフィールドでエラーが発生した場合、各フィールド名は新しい行に入力されます。カラム番号は LINE_NO カラムに出力されます。
LINE_NO	Integer	TRANS_ATTR および SOURCE_ATTR において、複数行にまたがる各行エラーエントリの行番号を指定します。
注: テーブルを結合するには、太字のカラム名を使用します。		

エラーログファイルについて

セッションで発生したすべてのエラーを収集するには、エラーログファイルを作成します。エラーログファイルは、ラインがカラムごとに区切られたシーケンシャルファイルです。固有のエラーログファイル名を指定することにより、ワークフローの各セッションについて独立したログファイルを作成できます。1 セッションについて行エラーを分析する際は、1 つのエラーログファイルを使用します。

エラーログファイルでは、エラーロギングカラムが二重パイプ (||) で区切られます。デフォルトでは、行データが一重パイプ (|) で区切られます。データカラム区切り文字は、[Data Column Delimiter] エラーログオプションの設定で変更できます。

エラーログファイルは、次のような構造になっています。

[Session Header]
[Column Header]
[Column Data]

セッションヘッダには、PMERR_SESS テーブルに格納される情報と類似したセッション実行情報が含まれます。カラムヘッダには、データカラム名が含まれます。カラムデータには、行データおよびエラーメッセージ情報が含まれます。

以下の表に、エラーログファイルのカラムを示します。

ログファイルカラムヘッダ	説明
トランスフォーメーション	マッピング内のエラーが発生したトランスフォーメーションの名前です。
トランスフォーメーションマプレット名	当該のトランスフォーメーションが含まれるマプレットの名前です。この情報が利用できない場合、N/A が出力されます。
トランスフォーメーショングループ	エラーが発生した入力グループまたは出力グループの名前です。トランスフォーメーションにグループが含まれない場合、デフォルトによりメ input もまたはメ output ものいずれかになります。
パーティションインデックス	エラーが発生したトランスフォーメーションのパーティション番号を指定します。
トランスフォーメーション行 ID	エラー行の行 ID を指定します。
エラーシーケンス	各トランスフォーメーショングループの行単位のエラー数を示すカウンタです。セッションに複数のパーティションが存在する場合、Integration Service は、このカウンタをパーティションごとに保持します。 たとえば、パーティション 1 のトランスフォーメーションで 3 つのエラーが、パーティション 2 のトランスフォーメーションで 2 つのエラーが生成された場合、パーティション 1 における ERROR_SEQ_NUM の値は 1、2、3 に、パーティション 2 における ERROR_SEQ_NUM の値は 1 および 2 になります。
エラータイムスタンプ	エラー発生時の Integration Service のタイムスタンプです。
エラー UTC 時間	エラー発生時刻を協定世界時（グリニッジ標準時）で示します。
エラーコード	エラーメッセージに対応するエラーコードです。
エラーメッセージ	エラーメッセージです。
エラータイプ	発生したエラーのタイプです。Integration Service は、次の値を使用します。 1 - Reader エラー 2 - Writer エラー 3 - トランスフォーメーションエラー

ログファイルカラムヘッダ	説明
トランスフォーメーションデータ	<p>カラムインジケータなど、すべてのカラムデータを含む、区切り文字付きの文字列です。各種カラムインジケータを次に示します。</p> <p>D - 有効 O - オーバーフロー N - NULL T - 切り詰め B - バイナリ U - データが利用不可</p> <p>カラムデータとカラムインジケータの固定区切り文字はコロン (:) になります。カラム間の区切り文字はパイプ () になります。カラム区切り文字は、エラー処理設定で上書きできます。</p> <p>エラーファイルでは、すべてのカラムデータがテキスト文字列に変換されます。バイナリデータの場合は、Integration Service はカラムインジケータのみ使用します。</p>
ソース名	<p>ソース修飾子の名前です。ソース修飾子以外のアクティブソースまたは複数のパーティションを持つ非パススルーパーティションポイントの下流で行エラーが発生した場合、N/A が記録されます。</p>
ソース行 ID	<p>ソース修飾子が、読み込んだ各行に割り当てる値です。Integration Service が行を識別できなかった場合、この値は-1 になります。</p>
ソース行タイプ	<p>行インジケータは、挿入、更新、またはリジェクトのいずれのマークが行に付いていたかを示します。</p> <p>0 - 挿入 1 - 更新 2 - 削除 3 - リジェクト</p>
ソースデータ	<p>カラムインジケータなど、すべてのカラムデータを含む、区切り文字付きの文字列です。各種カラムインジケータを次に示します。</p> <p>D - 有効 O - オーバーフロー N - NULL T - 切り詰め B - バイナリ U - データが利用不可</p> <p>カラムデータとカラムインジケータの固定区切り文字はコロン (:) になります。カラム間の区切り文字はパイプ () になります。カラム区切り文字は、エラー処理設定で上書きできます。</p> <p>Integration Service は、すべてのカラムデータをエラーテーブルまたはエラーファイルのテキスト文字列に変換します。バイナリデータの場合は、Integration Service はカラムインジケータのみ使用します。</p>

エラーログオプションの設定

セッションプロパティの「設定オブジェクト」タブで、セッションごとにエラーロギングを設定します。エラーロギングを有効にするとき、リレーショナルデータベースまたはフラットファイルにエラーログを作成するように指定できます。エラーロギングを有効にしない場合、エラーログは作成されません。

ヒント: Workflow Manager を使って、「設定オブジェクト」タブに適用する再利用可能な属性のセットを作成します。

エラーロギングオプションを設定するには：

1. 「セッション」タスクをダブルクリックして、セッションのプロパティを開きます。
2. 「設定オブジェクト」タブを選択します。
3. エラーログタイプを指定します。

以下の表に、「設定オブジェクト」タブのエラーロギング設定を示します。

エラーログオプション	説明
エラーログタイプ	作成するエラーログのタイプを指定します。[リレーショナルデータベース]、[フラットファイル]、または[なし]を指定できます。デフォルトは[なし]です。 注: XML ファイルソースの行エラーをログに出力することはできません。XML ソースのエラーはセッションログで確認できます。
エラーログ DB 接続	リレーショナルログに対するデータベース接続を指定します。リレーショナルデータベースによるロギングを有効にする場合、このオプションは省略できません。
エラーログテーブル接頭辞	リレーショナルログに使用するテーブル名の接頭語を指定します。この接頭語に対し 11 文字が付加されます。Oracle および Sybase では、テーブル名に 30 文字の制限があります。テーブル名が 30 文字を超えている場合、セッションが失敗します。 エラーログテーブル名の接頭語には、パラメータまたは変数を使用できます。パラメータファイルで定義可能なパラメータまたは変数タイプを使用します。
エラーログファイルディレクトリ	エラーを出力するディレクトリを指定します。デフォルトでは、エラーログファイルディレクトリは\$PMBadFilesDir\\u306b なります。フラットファイルによるロギングを有効にする場合、このオプションは省略できません。
エラーログファイル名	エラーログファイル名を指定します。エラーログファイル名に使用できる文字数の上限は 255 文字です。デフォルトでは、エラーログファイル名は PLError.log になります。フラットファイルによるロギングを有効にする場合、このオプションは省略できません。
ログ行データ	トランスフォーメーション行データをログに出力するかどうかを指定します。エラーログを有効にすると、Integration Service は、デフォルトで、トランスフォーメーション行データをログを記録します。このプロパティを無効にした場合、トランスフォーメーション行データのフィールドには、N/A または-1 が出力されます。

エラーログオプション	説明
ログソース行データ	<p>ソース行データをログに出力しないように選択した場合、あるいは、ソース行データが利用できない場合、カラムのデータ型に応じて N/A または -1 が出力されます。</p> <p>ソース行データを収集する必要がない場合は、Integration Service のパフォーマンスを向上させるため、このオプションを無効にすることを検討してください。</p>
データカラム区切り文字	<p>文字列型のソース行データおよびトランスフォーメーショングループ行データに使用する区切り文字です。デフォルトでは、Integration Service は区切り文字にパイプ () を使用します。該当する行データに対する区切り文字と同じものがエラーロギングカラムに使用されていないか確認してください。もし同じ区切り文字が使用されていると、エラーログファイルの読み込みが困難な場合があります。</p>

4. [OK] をクリックします。

第 9 章

ワークフローリカバリ

この章では、以下の項目について説明します。

- [ワークフローリカバリの概要, 182 ページ](#)
- [操作の状態, 183 ページ](#)
- [リカバリオプション, 186 ページ](#)
- [ワークフローのサスペンド, 188 ページ](#)
- [ワークフローリカバリの設定, 189 ページ](#)
- [タスクリカバリの設定, 190 ページ](#)
- [セッションの再開, 192 ページ](#)
- [再現可能なデータに関する作業, 194 ページ](#)
- [ワークフローとタスクのリカバリの手順, 198 ページ](#)
- [セッションリカバリに関するルールおよびガイドライン, 199 ページ](#)

ワークフローリカバリの概要

ワークフローのリカバリを行うことにより、ワークフローとワークフロータスクの処理を中断した時点から継続することができます。Integration Service が操作のワークフローステートにアクセスできる場合は、ワークフローをリカバリできます。操作のワークフローステートには、ワークフロー内のタスクのステータスとワークフロー変数の値が含まれます。Integration Service は、このステートを、ワークフローの設定に基づいてメモリまたはディスクに格納します。

- **リカバリの有効化。**ワークフローのリカバリを有効にする場合、Integration Service によって、ワークフローの操作の状態が共有場所に保存されます。ワークフローが終了、停止、または強制終了した場合に、そのワークフローをリカバリできます。ワークフローが実行中である必要はありません。
- **サスペンド。**エラー発生時にサスペンド状態にするようにワークフローを設定した場合、Integration Service によって、ワークフローの操作の状態がメモリに格納されます。タスクが失敗した場合、一時停止されたワークフローをリカバリできます。タスクエラーを修正し、ワークフローをリカバリすることができます。

Integration Service はワークフロー内のタスクを、そのタスクのリカバリ戦略に基づいてリカバリします。デフォルトでは、[セッション] タスクと [コマンド] タスクのリカバリ戦略は、タスクを失敗させて引き続きワークフローを実行するというものです。[セッション] タスクと [コマンド] タスクのリカバリ戦略は設定することができます。他のすべてのタスクの戦略は、タスクをリスタートするというものです。

高可用性オプションがある場合、ワークフローを実行しているサービスプロセスが異なるノードにフェイルオーバーすると、PowerCenter はそのワークフローを自動的にリカバリします。タスクの終了時にそのタスクを

自動的にリカバリするように、実行中のワークフローを設定することができます。データベース接続が中断した後も、PowerCenter はセッションとワークフローをリカバリします。

セーフモードで実行されている場合、Integration Service はリカバリ用に設定されているワークフローの操作の状態を格納します。ワークフローが失敗し、Integration Service がバックアップノードにフェイルオーバーした場合、Integration Service では自動的にワークフローをリカバリしません。Integration Service に対する適切な特権を持っている場合は、ワークフローを手動でリカバリできます。

操作の状態

ワークフローまたはセッションをリカバリすると、Integration Service は操作のワークフローステートまたはセッションステートをリストアして、リカバリ処理の開始場所を決定します。Integration Service は、操作のワークフローステートを、ワークフローの設定に基づいてメモリまたはディスクに格納します。Integration Service は、操作のセッションステートをセッションの設定方法に基づいて格納します。

ワークフローの操作の状態

ワークフローのリカバリまたは一時停止を有効にする場合、Integration Service は操作のワークフローステートを格納します。ワークフローが一時停止になると、操作のステートはメモリに保存されます。

リカバリのワークフローを有効にする場合、Integration Service は、操作のワークフローステートを共有場所 \$PMStorageDir に格納します。Integration Service は、停止、強制終了、または終了したワークフローをリカバリするために、操作のワークフローステートをリストアすることができます。リカバリを実行するとき、中断した時点からワークフローをリカバリするために操作のステートをリストアします。ワークフローが完了すると、Integration Service は共有フォルダから操作のワークフローステートを削除します。

操作のワークフローステートには、次の情報が含まれます。

- アクティブなサービス要求
- 完了したタスクと実行中のタスクのステータス
- ワークフロー変数の値

コンカレントワークフローを実行すると、\$PMStorageDir 内のワークフローリカバリストレージファイルに、インスタンス名またはワークフロー実行 ID が付加されます。

ワークフローのリカバリを有効にする場合、Integration Service は、デフォルトでは操作のセッションステートを格納しません。操作のセッションステートが保存されるように、セッションのリカバリ戦略を設定することができます。

操作のセッション状態

最後のチェックポイントから再開するようにセッションのリカバリ戦略を設定すると、Integration Service は、操作のセッションステートを共有場所 \$PMStorageDir に格納します。Integration Service はまた、リレーショナルターゲットのリカバリ情報をターゲットデータベーステーブルに保存します。Integration Service は、リカバリを実行するときに、中断した時点からセッションをリカバリするために操作のステートをリストアします。ターゲットリカバリデータを使用して、ターゲットテーブルのリカバリ方法を決定します。

操作のワークフローステートを保存しない場合でも、操作のセッションステートを保存するようにセッションを設定できます。セッションをリカバリするかまたはセッションからワークフローをリカバリすることができます。

操作のセッションステートには、次の情報が含まれます。

- **ソース。** ソースからの出力が確定的で再現可能でない場合、Integration Service では SQL クエリからの結果が、\$PMStorageDir 内の共有ストレージファイルに保存されます。
- **トランスフォーメーション。** Integration Service では、\$PMStorageDir にチェックポイントが作成され、リカバリセッションを実行するときにパイプラインの処理の開始場所が決定されます。
差分アグリゲータトランスフォーメーションを持つセッションを実行した場合、Integration Service によりセッションの開始時に\$PMCacheDir にアグリゲータキャッシュファイルのバックアップが作成されます。Integration Service では、セッションリカバリの実行開始時に、バックアップキャッシュが初期キャッシュへプロモートされます。
- **リレーショナルターゲットのリカバリデータ。** Integration Service ではリカバリ情報が、ターゲットデータベース内のリカバリテーブルに書き込まれ、セッションが中断した場合にターゲットにコミットされている最後の行が決定されます。

ターゲットリカバリテーブル

Integration Service は、再開リカバリ戦略を持つセッションを実行する際に、ターゲットデータベースシステムのリカバリテーブルに書き込みます。セッションをリカバリする場合、Integration Service はリカバリテーブル内の情報を使用して、ターゲットテーブルへのデータのロードをどこから開始するかを決定します。

Integration Service でリカバリテーブルが作成されるようにする場合、ターゲットデータベース接続に設定されたデータベースユーザー名に対してテーブル作成特権を与えます。Integration Service でリカバリテーブルが作成されないように設定する場合は、リカバリテーブルを手動で作成します。

Integration Service は、以下のリカバリテーブルをターゲットデータベースに作成します。

- **PM_RECOVERY。** セッション実行のためのターゲットロード情報が含まれています。Integration Service は、各セッションが成功した後でこのテーブルから情報を削除し、後続のセッションの開始時に情報を初期化します。
- **PM_TGT_RUN_ID。** Integration Service がデータベース上の各ターゲットの特定に使用する情報が含まれています。この情報は、セッション実行後も次の実行まで維持されます。このテーブルを手動で作成する場合は、セッションが正常にリカバリされるように、行を作成し LAST_TGT_RUN_ID に 0 以外を入力する必要があります。
- **PM_REC_STATE。** Integration Service がリアルタイムセッションのリカバリ時にターゲットテーブルへのメッセージ書き込みの必要があるかどうかの判断に使用する情報が含まれています。

セッションのリカバリ前にリカバリテーブルを編集または削除した場合、Integration Service はそのセッションをリカバリできません。リカバリを無効にした場合、Integration Service ではターゲットデータベースからリカバリテーブルが削除されません。リカバリテーブルは手動で削除する必要があります。

以下の表に、PM_RECOVERY の形式を示します。

カラム名	データ型
REP_GID	VARCHAR(240)
WFLOW_ID	INTEGER
WFLOW_RUN_ID	INTEGER
WFLOW_RUN_INS_NAME	VARCHAR (240)
SUBJ_ID	INTEGER

カラム名	データ型
TASK_INST_ID	INTEGER
TGT_INST_ID	INTEGER
PARTITION_ID	INTEGER
TGT_RUN_ID	INTEGER
RECOVERY_VER	INTEGER
CHECK_POINT	INTEGER
ROW_COUNT	INTEGER

以下の表に、PM_TGT_RUN_ID の形式を示します。

カラム名	データ型
LAST_TGT_RUN_ID	INTEGER

以下の表に、PM_REC_STATE の形式を示します。

カラム名	データ型
OWNER_TYPE_ID	INTEGER
REP_GID	VARCHAR(240)
FOLDER_ID	INTEGER
WFLOW_ID	INTEGER
WFLOW_RUN_INS_NAME	VARCHAR (240)
WLET_ID	INTEGER
TASK_INST_ID	INTEGER
WID_INST_ID	INTEGER
GROUP_ID	INTEGER
PART_ID	INTEGER
PLUGIN_ID	INTEGER
APPL_ID	VARCHAR(38)
SEQ_NUM	INTEGER
VERSION	INTEGER

カラム名	データ型
CHKP_NUM	INTEGER
STATE_DATA	VARCHAR(1024)

Oracle では、INTEGER データタイプの代わりに、NUMBER データタイプが使用されます。

注: コンカレントリカバリセッションが同じターゲットデータベースに書き込む場合、Integration Service で PM_RECOVERY でデッドロックが検出されることがあります。デッドロック時に PM_RECOVERY への書き込みを再試行するために、[デッドロック時のセッションのリトライ] オプションを設定し、セッションのデッドロックを再試行できます。

関連項目：

- [「PM_REC_STATE テーブル」 \(ページ 145\)](#)

ターゲットリカバリテーブルの作成

ターゲットリカバリテーブルは、手動で作成することができます。Informatica 提供の SQL スクリプトは、次のディレクトリにあります。

<PowerCenter installation_dir>\server\bin\RecoverySQL

次のいずれかのスクリプトを実行して、ターゲットデータベースにリカバリテーブルを作成します。

スクリプト	データベース
create_schema_db2.sql	IBM DB2
create_schema_inf.sql	Informix
create_schema_ora.sql	Oracle
create_schema_sql.sql	Microsoft SQL Server
create_schema_syb.sql	Sybase
create_schema_ter.sql	Teradata

リカバリオプション

リカバリを実行するには、マッピング、ワークフロータスク、およびワークフローをリカバリ用に設定する必要があります。

以下の表に、リカバリ用に設定できるオプションを示します。

オプション	場所	説明
Suspend Workflow on Error	ワークフロー	ワークフロー内のタスクが失敗したときにワークフローを一時停止します。失敗したタスクを修正し、サスペンド状態のワークフローをリカバリすることができます。
サスペンド時のメール	ワークフロー	ワークフローがサスペンド状態である際に、メールを送信します。
Enable HA Recovery	ワークフロー	操作のワークフローステートを共有場所に保存します。ワークフローのリカバリを有効にするための高可用性は必要ありません。
終了したタスクの自動リカバリ	ワークフロー	ワークフロー実行中に、終了した [セッション] タスクと [コマンド] タスクをリカバリします。高可用性オプションが必要です。
自動リカバリの試行最大数	ワークフロー	Integration Service がセッションまたはコマンドタスクのリカバリを試みる回数。
リカバリ戦略	セッション、コマンド	[セッション] タスクまたは [コマンド] タスクのリカバリ戦略。Integration Service によって、ワークフローリカバリ中にセッションまたはコマンドタスクがリカバリされる方法と、セッションリカバリ中にセッションがリカバリされる方法を決定します。
Fail Task If Any Command Fails	コマンド	タスクのいずれかのコマンドが失敗した場合に [コマンド] タスクが失敗できるようにします。このオプションを設定しない場合、コマンドが失敗してもタスクは実行を継続します。このオプションを [エラー時のワークフローのサスペンド] とともに使用して、タスク内のコマンドが失敗した場合にワークフローをサスペンド状態にすることができます。
出力が確定的	トランスフォーメーション	トランスフォーメーションが常に同じ入力データから同じデータのセットを生成することを示します。Integration Service が最後のチェックポイントからセッションを再開できるのは、出力が再現可能で決定性のある場合です。リレーショナルソース修飾子に対してこのオプションを [出力が再現可能] オプションとともに有効にした場合、Integration Service では SQL 結果は共有ストレージに保存されません。
出力が再現可能	トランスフォーメーション	トランスフォーメーションが行を生成する順序がセッション間で同じかどうかを示します。出力が再現可能で確定的である場合、Integration Service によってセッションを最後のチェックポイントから再開できます。リレーショナルソース修飾子に対してこのオプションを [出力が確定的] オプションとともに有効にした場合、Integration Service では SQL 結果は共有ストレージに保存されません。

警告: トランスフォーメーションに再現性および決定性を設定している場合、データの再現性を保証するのはユーザーの責任です。リカバリしようとしているセッションのトランスフォーメーションから再現性および決定性を持つデータが生成されない場合、リカバリプロセスによりデータが破損することがあります。

ワークフローのサスペンド

ワークフロー内のタスクが失敗した場合は、ワークフローをサスペンド状態にしてエラーを修復し、ワークフローをリカバリできます。ワークフロープロパティの「タスクのエラー発生時にサスペンドする」オプションが有効になっている場合、Integration Service では、そのワークフローがサスペンド状態になります。必要に応じて、Integration Service によりワークフローがサスペンド状態になった場合に電子メールが送信されるように、サスペンド時のメールを設定できます。

エラー発生時にサスペンド状態にするようにワークフローを設定した場合、Integration Service により、次のいずれかのタスクが失敗した際にワークフローがサスペンド状態になります。

- セッション
- コマンド
- ワークレット
- 電子メール

タスクがワークフロー内で失敗した場合、Integration Service はそのパスにあるタスクの実行を停止します。Integration Service では、失敗したタスクの出力リンクが評価されません。ワークフロー内でほかに実行中のタスクがない場合、Workflow Monitor はワークフローのステータスをメサスペンド状態モと表示します。

高可用性オプションが導入済みの場合、自動タスクリカバリの設定に応じてワークフローが一時停止されます。エラー発生時にサスペンド状態にするようにワークフローが設定されていて、自動タスクリカバリが有効化されていない場合、タスクの失敗後にワークフローが一時停止されます。自動タスクリカバリが有効化されている場合は、まず指定されたりカバリ限度までタスクのリスタートが試行された後、失敗したタスクをリスタートできないときはワークフローが一時停止されます。

タスクが失敗したときにワークフロー内で 1 つ以上のタスクがまだ実行中である場合、Integration Service は失敗したタスクの実行を停止して、他のパスにあるタスクの実行を続行します。Workflow Monitor はワークフローのステータスを「サスペンド中」と表示します。

ワークフローの状態が「サスペンド中」または「サスペンド状態」である場合、ターゲットデータベースエラーなどのエラーを修復した後で Workflow Monitor からワークフローをリカバリできます。ワークフローをリカバリした場合、Integration Service では失敗したタスクがリスタートされ、ワークフロー内の残りのタスクの評価が続行されます。既に正常に完了しているタスクは Integration Service によって実行されません。

注: 一時停止したワークフロー、または一時停止したワークフロー内のタスクを編集すると、リポジトリで不整合が発生する可能性があります。

ワークフローを一時停止するには：

1. Workflow Designer で、ワークフローを開きます。
2. 「ワークフロー」 - 「編集」をクリックします。
3. 「全般」タブで、「タスクのエラー発生時にサスペンドする」を有効にします。
4. 「OK」をクリックします。

サスペンド時のメールの設定

ワークフローが一時停止した場合、Integration Service によってメールが送信されるようにワークフローを設定することができます。サスペンド時のメール用に、既存の再利用可能なメールタスクを選択します。タスクが失敗した場合、Integration Service によってワークフローのサスペンドが開始され、サスペンド時のメールが送信されます。Integration Service でワークフローのサスペンドをしているときに別のタスクが失敗した場合、サスペンド時のメールを再度受け取ることはありません。

ワークフローを再開した後で別のタスクが失敗した場合は、Integration Service によってサスペンド時のメールが送信されます。

ワークフローリカバリの設定

ワークフローをリカバリ用に設定するには、リカバリのワークフローを有効にするかまたはタスクエラー発生時におけるワークフローの一時停止を設定する必要があります。ワークフローをリカバリ用に設定した場合は、そのワークフローが停止、強制終了、終了、または一時停止した場合にこれをリカバリすることができます。

以下の表に、リカバリ可能な各ワークフローステータスを示します。

ステータス	説明
強制終了	Workflow Monitor または <i>pmcmd</i> を使用して、ワークフローを強制終了します。また、Administrator ツールでサービスプロセスを無効にするときに、実行中のすべてのワークフローを強制終了することを選択することもできます。強制終了されたワークフローは、そのワークフローのリカバリを有効にした場合、リカバリすることができます。強制終了されたワークフローは、Workflow Monitor または <i>pmcmd</i> を使用してリカバリすることができます。
停止	Workflow Monitor または <i>pmcmd</i> を使用してワークフローを停止します。また、Administrator ツールでサービスまたはサービスプロセスを無効にするときに、実行中のすべてのワークフローを停止することを選択することもできます。停止したワークフローは、そのワークフローのリカバリを有効にした場合、リカバリすることができます。停止したワークフローは、Workflow Monitor または <i>pmcmd</i> を使用してリカバリすることができます。
サスペンド状態	タスクは失敗します。ワークフローは、タスクエラー発生時に一時停止するように設定されます。複数のタスクが実行中の場合、Integration Service は、実行中のすべてのタスクが成功または失敗したときにワークフローを一時停止にします。タスクの失敗を引き起こしたエラーを修正してから、リカバリを実行することができます。 デフォルトでは、ワークフローはタスクが失敗した後も継続的に実行されます。タスクが失敗した場合にワークフローが一時停止するようにするには、タスクエラー発生時に一時停止するようにワークフローを設定します。
ターミネート済み	ワークフローを実行しているサービスプロセスが予期せずシャットダウンします。そのワークフローを実行しているすべてのノードでタスクが終了します。ワークフロー内のタスクが終了し、高可用性オプションがない場合は、ワークフローが終了することがあります。リカバリのワークフローを有効にした場合は、終了したワークフローをリカバリすることができます。高可用性オプションがある場合、サービスプロセスは別のノードにフェイルオーバーし、ワークフローリカバリが開始されます。
注: 失敗したワークフローとは、失敗によって完了したワークフローです。失敗したワークフローはリカバリできません。	

停止、強制終了、および終了されたワークフローのリカバリ

ワークフローのリカバリを有効にする場合、Integration Service は、ワークフロー実行中に操作のワークフローステートをファイルに保存します。停止、終了、または強制終了したワークフローをリカバリすることができます。ワークフローの [プロパティ] タブで、リカバリを有効にしてください。

一時停止されたワークフローのリカバリ

ワークフロー内のタスクが失敗した場合にワークフローが一時停止するように設定できます。デフォルトでは、ワークフローはタスクが失敗した場合も継続的に実行されます。タスク失敗時にワークフローを一時停止し、失敗したタスクを修正してから、そのワークフローをリカバリすることができます。ワークフローを一時停止すると、操作のワークフローステータスはメモリに残されます。タスクの失敗を引き起こしたエラーを修正し、中断した時点からワークフローをリカバリすることができます。タスクが再び失敗した場合、Integration Service はワークフローを再びサスペンド状態にします。一時停止されたワークフローは、リカバリすることはできません。ワークフロープロパティの [全般] タブで、ワークフローを一時停止するように設定してください。

タスクが一時停止したときにメールを発信するようにワークフローを設定することもできます。

タスクリカバリの設定

ワークフローをリカバリするときには、Integration Service は、タスクごとのリカバリ戦略に基づいてタスクをリカバリします。タスクによっては、リカバリ戦略が失敗タスクになっておりワークフローを継続、最後のチェックポイントから再開、またはタスクをリスタートする場合があります。

ワークフローリカバリを有効にする場合、強制終了または停止するタスクをリカバリすることができます。ネットワークまたはサービスプロセスの失敗が原因で終了したタスクを、リカバリすることができます。エラー発生時にサスペンド状態にするようにワークフローを設定した場合、そのワークフローをリカバリするときに、失敗したタスクをリカバリすることができます。

以下の表に、リカバリ可能な各タスクステータスを示します。

ステータス	説明
強制終了	Workflow Monitor または <i>pmcmd</i> を使用して、ワークフローまたはタスクを強制終了します。また、Administrator ツールでサービスまたはサービスプロセスを無効にする場合に、実行中のすべてのワークフローの強制終了を選択することもできます。マッピング条件に基づいて強制終了するように、セッションを設定することもできます。 Workflow Monitor のワークフローをリカバリしてタスクをリカバリするか、 <i>pmcmd</i> を使用してワークフローをリカバリすることができます。
停止	Workflow Monitor または <i>pmcmd</i> を使用して、ワークフローまたはタスクを停止します。また、Administrator ツールでサービスまたはサービスプロセスを無効にするときに、実行中のすべてのワークフローを停止することを選択することもできます。 Workflow Monitor のワークフローをリカバリしてタスクをリカバリするか、 <i>pmcmd</i> を使用してワークフローをリカバリすることができます。
失敗	エラーが原因で Integration Service はタスクの実行に失敗しました。タスク失敗時に一時停止になるようにワークフローが設定されている場合、ワークフローリカバリを使用して、失敗したタスクをリカバリできます。ワークフローが一時停止になっていない場合は、単にセッションをリカバリするかまたはワークフローをセッションからリカバリすることにより、失敗したタスクをリカバリできます。 Workflow Monitor でエラーを修正してからワークフローをリカバリするか、または <i>pmcmd</i> を使用してワークフローをリカバリすることができます。
ターミネート済み	Integration Service は予期せず停止するか、またはマスタサービスプロセスへのネットワーク接続を喪失します。Workflow Monitor でワークフローをリカバリするか、または Integration Service のリスタート後に <i>pmcmd</i> を使用してワークフローをリカバリすることができます。

タスクのリカバリ戦略

ワークフロー内の各タスクにはリカバリ戦略があります。Integration Service は、ワークフローをリカバリするときに、リカバリ戦略に基づいてタスクをリカバリします。

- **タスクのリスタート。** Integration Service は、ワークフローをリカバリするときに、リスタートの戦略を使用して設定されているリカバリ可能な各タスクをリスタートします。リスタートリカバリ戦略を使用して [セッション] タスクと [コマンド] タスクを設定できます。他のすべてのタスクは、リスタートリカバリ戦略をデフォルトで持っています。

- **タスクを失敗してワークフローを続行します。** Integration Service では、ワークフローがリカバリされる際に、タスクはリカバリされません。タスクのステータスは失敗となりますが、Integration Service は引き続きワークフローを実行します。

ワークフローを完了したいがタスクをリカバリしたくない場合は、失敗リカバリ戦略を設定します。[セッション] タスクと [コマンド] タスクを失敗タスクで設定し、ワークフローのリカバリ戦略を継続することができます。

- **最後のチェックポイントから再開します。** Integration Service では、停止、強制終了、または終了したセッションが最後のチェックポイントからリカバリされます。再開戦略を使用して、セッションタスクを設定できます。

以下の表に、タスクタイプのリカバリ戦略を示します。

タスクタイプ	リカバリ戦略	コメント
Assignment	タスクを再開します。	-
Command	タスクを再開します。 タスクを失敗してワークフローを続行します。	デフォルトは [タスクを失敗してワークフローを続行] です。
コントロール	タスクを再開します。	-
ディシジョン	タスクを再開します。	-
電子メール	タスクを再開します。	Integration Service は E-Mail を重複して送信することがあります。
Event-Raise	タスクを再開します。	-
Event-Wait	タスクを再開します。	-
Session	最後のチェックポイントから再開します。 タスクを再開します。 タスクを失敗してワークフローを続行します。	デフォルトは [タスクを失敗してワークフローを続行] です。
Timer	タスクを再開します。	タスクまたはワークフローの開始時点から起算した時間差指定を使用する場合は、元の値から経過時間を差し引いてタイマを設定します。
ワークレット	なし	Integration Service はワークレットをリカバリしません。ワークレットのセッションは、Workflow Monitor でワークレットを展開し、[タスクのリカバリ] を選択してリカバリできます。

[コマンド] タスク戦略

[コマンド] タスクを設定する場合、リカバリ戦略をリスタートまたは失敗に選択できます。

- **タスクを失敗してワークフローを続行します。** コマンドタスクのエラーで、ワークフローがサスペンド状態にする場合は、失敗戦略を使用してタスクを設定する必要があります。コマンドタスクに複数のコマンドがある際に、失敗戦略を設定する場合、いずれのコマンドが失敗した場合でもタスクが失敗するように設定する必要があります。

- **タスクをリスタートします。** Integration Service では、ワークフローがリカバリされる際に、リスタートの戦略を使用して設定されているコマンドタスクがリスタートされます。

コマンドタスクのプロパティページでリカバリ戦略を設定します。

セッションタスク戦略

セッションをリカバリ用に設定すると、ワークフローをリカバリするときにそのセッションをリカバリできます。または、残りのワークフローを実行することなく、セッションをリカバリすることができます。

セッションを設定するとき、失敗、リスタート、または再開のリカバリ戦略を選択できます。

- **最後のチェックポイントから再開します。** Integration Service によって、操作のセッション状態が保存されターゲットリカバリテーブルが保持されます。セッションが強制終了、停止、または終了した場合、Integration Service によって保存されたリカバリ情報が使用され、中断ポイントからセッションが再開されます。

セッションによってマッピング変数が使用されている場合は、そのセッションに再開戦略を設定することはできません。

- **タスクのリスタート。** Integration Service は、ワークフローをリカバリすると、セッションを再実行します。[タスクのリスタート] を使用してリカバリする場合は、ターゲットに部分的にロードされたデータを削除するかまたは重複行をスキップするようにマッピングを設計する必要がある場合もあります。
- **タスクを失敗してワークフローを続行します。** Integration Service によってワークフローがリカバリされる際、セッションはリカバリされません。セッションのステータスは失敗となりますが、Integration Service は引き続きワークフローを実行します。

[Session] タスクの [プロパティ] ページでリカバリ戦略を設定してください。

終了したタスクの自動的なリカバリ

高可用性オプションがある場合、終了したタスクの自動リカバリを設定できます。自動タスクリカバリを有効にした場合、Integration Service はワークフローがまだ実行中の場合、終了した [セッション] タスクと [コマンド] タスクをユーザー介入なしにリカバリします。タスクのリカバリを Integration Service が試みる回数を設定します。ワークフロープロパティで自動タスクリカバリを有効にします。

セッションの再開

最後のチェックポイントから再開するようにセッションリカバリを設定した場合、Integration Service によって、セッションリカバリの処理を開始する場所を決定するために \$PMStorageDir にチェックポイントが作成されます。Integration Service は、セッションを再開するときに、各ソース、ターゲット、およびトランスフォーメーションの状態など、操作のセッションステートをリストアします。Integration Service は、処理に必要なソースデータの量を確認します。

Integration Service がセッションを再開するときには、リカバリセッションが元のセッションと同じデータを生成する必要があります。最後のチェックポイントから再開するようにリカバリを設定した場合、セッションは無効ですが、セッションは再現可能なデータを生成できません。

Integration Service は、FTP ソースなどのフラットファイルソースをリカバリできます。フラットファイルと FTP ターゲットを切り詰めるかまたは追加することができます。

最後のチェックポイントからセッションをリカバリすると、Integration Service は操作のセッションステートをリストアし、実行可能なリカバリのタイプを確認します。

- **差分。** Integration Service は、中断した時点からデータの処理を開始します。中断前に処理した行の読み込みまたは変換を行いません。デフォルトでは、Integration Service は差分リカバリを実行しようとします。
- **フル。** 差分リカバリを実行できない場合、Integration Service はすべてのソース行を再び読み込み、すべてのトランスフォーメーションロジックを実行します。Integration Service は、最後のコミットポイントでターゲットへの書き込みを開始します。いずれかのセッションコンポーネントでフルリカバリが必要な場合、Integration Service はそのセッションについてフルリカバリを実行します。

以下の表に、Integration Service によって、どのような時にセッションの設定に応じて差分リカバリまたはフルリカバリが実行されるのかを示します。

コンポーネント	差分リカバリ	フルリカバリ
コミットタイプ	セッションはソースベースのコミットを使用します。マッピングは、コミットを生成するトランスフォーメーションを含みません。	セッションは、ターゲットベースのコミットまたはユーザー定義のコミットを使用します。
トランスフォーメーション範囲	トランスフォーメーションは、トランザクションをプロパゲートします。トランスフォーメーションスコープは、[トランザクション] または [行] である必要があります。	少なくとも 1 個のトランスフォーメーションが [すべて] トランスフォーメーション範囲で設定されています。
ファイルソース	ファイルソースは差分読み込みをサポートします。	なし
FTP ソース	差分読み込みを可能にするには、FTP サーバーがシーク操作をサポートしている必要があります。	FTP サーバーはシーク操作をサポートしません。
リレーショナルソース	出力が再現可能で決定性がある場合、リレーショナルソースは差分読み込みをサポートします。出力が再現可能で決定性がない場合、Integration Service は SQL 結果をストレージファイルにステージングすることにより、リレーショナルソースの差分読み込みをサポートします。	なし
VSAM ソース	なし	Integration Service により、フルリカバリが実行されます。
XML ソース	なし	Integration Service により、フルリカバリが実行されます。
XML ジェネレータトランスフォーメーション	XML ジェネレータトランスフォーメーションのトランスフォーメーションスコープは [トランザクション] に設定されている必要があります。	なし
XML ターゲット	XML ターゲットは、コミットで新しい XML ドキュメントを生成するように設定されている必要があります。	なし

再現可能なデータに関する作業

最後のチェックポイントから再開するようにリカバリを設定する場合、リカバリセッションは元のセッションと同じ順序で同じデータを生成できる必要があります。

セッションを検査すると、Workflow Manager によって、再現性および決定性を持つデータを生成するようにトランスフォーメーションが設定されていることが検査されます。最後のチェックポイントから再開するようにリカバリを設定していても、再現性および決定性を持つデータ用にトランスフォーメーションが設定されていない場合、セッションは無効です。

すべてのターゲットが再現可能なデータを以下のマッピングオブジェクトから受け取る場合、セッションデータは再現可能です。

- **ソース。** ソースからの出力データは、元の実行とリカバリ実行の間で再現可能です。
- **トランスフォーメーション。** 各トランスフォーメーションからターゲットへの出力データは再現可能です。

ソースの再現性

各ソースが一連の同一のデータを生成する場合で、出力の順序がセッション間で再現可能な場合は、最後のチェックポイントからセッションを再開できます。ソースデータは、セッション内のソースのタイプに基づいて再現可能です。

リレーショナルソース

リレーショナルソースは、ワークフロー間で異なるデータまたは順序の異なるデータを生成する場合があります。最後のチェックポイントから再開するようにリカバリを設定すると、Integration Service は、リカバリ用の出力順序を保証するために SQL 結果をキャッシュファイルに格納します。

SQL 結果がワークフロー間で同じになると分かっている場合、データが再現可能で決定性があることを示すようにソース修飾子を設定することができます。リレーショナルソース出力に決定性があり、出力が常に再現可能な場合、Integration Service は SQL 結果をキャッシュファイルに格納しません。リレーショナル出力が再現可能でない場合、マッピング内のトランスフォーメーションが順番付けられたデータを常に生成する場合は、Integration Service はキャッシュファイルの作成をスキップできます。

SDK ソース

SDK ソースが再現可能なデータを生成する場合、SDK ソース修飾子トランスフォーメーションで「出力が確定的」と「出力が再現可能」を有効にすることができます。

フラットファイルソース

フラットファイルはセッション実行とリカバリ実行の間で変わりません。ソースファイルを変更した後にセッションをリカバリした場合は、予期できない結果がリカバリセッションで発生する可能性があります。

トランスフォーメーションの再現性

セッション内のトランスフォーメーションがセッション実行とリカバリ実行の間で同じデータを生成する場合、最後のチェックポイントから再開するようにセッションを設定できます。すべてのトランスフォーメーションには、再現可能なデータを作成できるかどうかを決定するプロパティがあります。出力に決定性があり出力が再現可能な場合、トランスフォーメーションはセッション実行とリカバリ実行の間で同じデータを作成できます。

警告: トランスフォーメーションに再現性および決定性を設定している場合、データの再現性を保証するのはユーザーの責任です。リカバリしようとしているセッションのトランスフォーメーションから再現性および決定性を持つデータが生成されない場合、リカバリプロセスによりデータが破損することがあります。

出力が確定的

常に同じ出力データが同じ入力データから作成された場合に、トランスフォーメーションは決定性出力を作成します。

出力が再現可能

トランスフォーメーションは、セッション間で同じ順序で行を生成する場合、再現可能なデータを作成します。トランスフォーメーションはトランスフォーメーションのタイプ、トランスフォーメーションの設定、またはマッピングの設定に基づいて、再現可能なデータを作成します。

トランスフォーメーションは、次の状況で再現可能なデータを作成します。

- **常時。** 入力データの順序がセッションの実行間で矛盾している場合でも、出力データの順序はセッションの実行間で一貫しています。
- **入力順を基準。** トランスフォーメーションによって、すべての入力グループの入力データの順序がセッション実行間で一貫している場合は、セッション実行間で再現可能なデータが作成されます。入力グループからの入力データが順序付けられていない場合、出力は順序付けられません。

トランスフォーメーションが入力順序に依存して再現可能なデータを作成する場合、セッションの検査中に Workflow Manager はマッピングを検査して、トランスフォーメーションが再現可能なデータを作成できるかどうか確認します。例えば、式トランスフォーメーションによって、再現可能なデータを受け取った場合にだけ再現可能データが生成されます。

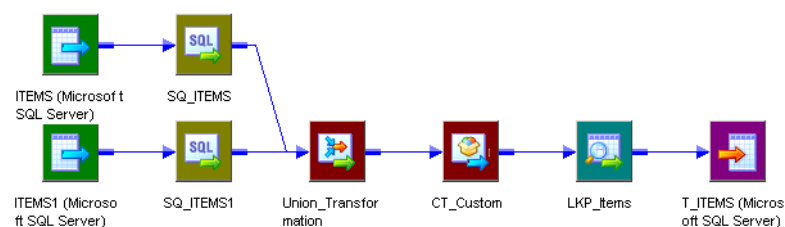
- **不可能。** 出力データの順序はセッションの実行間で矛盾します。

リカバリ用のマッピングの設定

セッション内のトランスフォーメーションがセッション実行とリカバリ実行の間で同じデータを作成できるように、マッピングを設定できます。マッピングに再現可能なデータを作成しないトランスフォーメーションが含まれている場合は、その直後に再現可能なデータを常に作成するトランスフォーメーションを追加できます。

例えば、再現可能なデータを作成しないトランスフォーメーションを、入力順序に依存して再現可能なデータを作成するトランスフォーメーションに直接接続します。データが再現可能でなければ、最後のチェックポイントから再開するようにリカバリを設定することはできません。セッションをリカバリ可能にするために、再現可能なデータを作成しないトランスフォーメーションの後に、再現可能なデータを常に作成するトランスフォーメーションを追加できます。

次の図に、最後のチェックポイントからの再開でリカバリできないマッピングを示します。

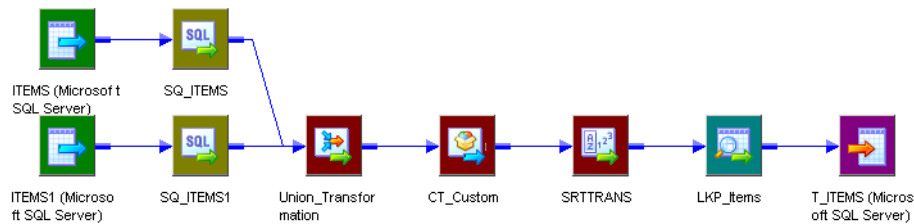


上図のマッピングは、再現可能なデータを作成するソース修飾子トランスフォーメーションを2つ含んでいます。再現可能なデータを作成しない共有体トランスフォーメーションとカスタムトランスフォーメーションがあります。ルックアップトランスフォーメーションは、再現可能なデータを受け取った場合、再現可能データ

を作成します。そのため、ターゲットは再現可能なデータを受け取れないので、リカバリを再開するようにセッションを設定することはできません。

リカバリを再開できるようにマッピングを一部変更することができます。出力行が重複しないように設定されたソータトランスフォーメーションを、再現可能なデータを出力しないトランスフォーメーションの直後に追加します。カスタムトランスフォーメーションの後にソータトランスフォーメーションを追加します。

次の図に、ソータトランスフォーメーションをカスタムトランスフォーメーションに接続したマッピングを示します。



ルックアップトランスフォーメーションは、再現可能なデータをソータトランスフォーメーションから受け取っているため、再現可能データを作成します。

以下の表に、いつ再現可能なデータをトランスフォーメーションが作成するかを示します。

トランスフォーメーション	再現可能なデータ
アグリゲータ	常に作成する。
アプリケーションソース修飾子	入力順に基づく。
カスタム	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。
データマスキング	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。再現可能なデータを生成するには、ポートごとに再現可能なマスキングまたはキーマスキングを設定します。
Expression	入力順に基づく。
エクスターナルプロシージャ	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。
Filter	入力順に基づく。
HTTP	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。
ジョイナ	入力順に基づく。
Java	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。
Lookup、動的	常に作成する。ルックアップソースは、セッション内のターゲットと同じである必要があります。
Lookup、静的	入力順に基づく。
MQ ソース修飾子	常に作成する。

トランスフォーメーション	再現可能なデータ
Normalizer、パイプライン	入力順に基づく。
Normalizer、VSAM	常に作成する。Normalizer は、一意なプライマリキーの形式でソースデータを生成します。セッションを再開すると、そのセッションは、正常に完了した場合とは異なるキー値を生成する場合があります。
ランク	常に作成する。
ルータ	入力順に基づく。
シーケンスジェネレータ	常に作成する。Integration Service は、現在の値をリポジトリに格納します。
Sorter（出力行が重複しない設定）	常に作成する。
Sorter（出力行が重複しない設定なし）	入力順に基づく。
Source Qualifier、フラットファイル	常に作成する。
ソース修飾子、リレーショナル	入力順に基づく。トランスフォーメーションをソースデータに従って設定します。データが再現可能でない場合、Integration Service はそのデータをステージングします。
SQL トランスフォーメーション	入力順に基づく。トランスフォーメーションをソースデータに従って設定します。
ストアドプロシージャ	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。
トランザクションコントロール	入力順に基づく。
Union	作成しない。
Unstructured Data	入力順に基づく。トランスフォーメーションプロシージャの動作に従って、プロパティを設定します。
アップデートストラテジ	入力順に基づく。
XML ジェネレータ	常に作成する。
XML パーサ	入力順に基づく。トランスフォーメーションをソースデータに従って設定します。
XML ソース修飾子	常に作成する。

以下のトランスフォーメーションに「出力が再現可能」および「出力が確定的」プロパティを設定することも、あるいはこれらのトランスフォーメーションの直後に再現可能なデータを生成するトランスフォーメーションを追加することもできます。

- アプリケーションソース修飾子
- カスタム

- エクスターナルプロシージャ
- ソース修飾子、リレーショナル
- ストアドプロシージャ

ワークフローとタスクのリカバリの手順

ワークフローをリカバリ用に設定すると、そのワークフローをリカバリすることができます。セッションのリカバリ戦略を設定すると、セッションをリカバリすることができます。セッションのリカバリ戦略を設定する場合、セッションをリカバリするためにワークフローのリカバリを有効にする必要はありません。

次のいずれかの方法を使ってワークフローまたはタスクをリカバリできます。

- **ワークフローのリカバリ。** 中断ポイントからワークフローの処理を続行します。
- **セッションのリカバリ。** セッションをリカバリしますが、ワークフローの残りはリカバリしません。
- **セッションからのワークフローのリカバリ。** セッションをリカバリし、ワークフローの処理を続行します。

Integration Service でオペレーティングシステムのプロファイルが使用されている場合、Integration Service でセッションまたはワークフローの実行に使用されるものと同じオペレーティングシステムのプロファイルを使用して、セッションまたはワークフローをリカバリします。

ワークフローまたはタスクをリカバリなしでリスタートする場合、コールドスタートモードを使用できます。リアルタイムセッションのリカバリ動作は、リアルタイムソースに応じて異なります。

ワークフローのリカバリ

ワークフローをリカバリすると、Integration Service は操作のワークフローステートをリストアし、失敗したところから処理を続行します。Integration Service は、失敗したタスクをリカバリするためにタスクのリカバリ戦略を使用します。

タスクが失敗したときにワークフローが一時停止するように設定することにより、またはワークフロープロパティでリカバリを有効にすることにより、ワークフローをリカバリ用に設定します。

Workflow Manager、Workflow Monitor、または *pmcmd* を使用してワークフローをリカバリすることができます。ワークフローをリカバリする場合、Integration Service によって既存のセッションログにログイベントが追加されます。

Workflow Monitor を使用したワークフローのリカバリ

Workflow Monitor を使用してワークフローをリカバリするには：

1. Workflow Monitor でワークフローを選択します。
2. ワークフローを右クリックし、[リカバリ] を選択します。

Integration Service によって失敗したタスクがリカバリされ、ワークフローの残りの部分が実行されます。

pmcmd recoverworkflow コマンドを使用してワークフローをリカバリすることもできます。

セッションのリカバリ

失敗、終了、強制終了、または停止したセッションを、ワークフローをリカバリすることなくリカバリすることができます。ワークフローが完了すると、残りのワークフローを実行することなく、セッションをリカバリ

することができます。セッションをリカバリするには、最後のチェックポイントからのリスタートまたは再開のリカバリ戦略を設定する必要があります。Integration Service はタスクのリカバリ戦略に基づいてセッションをリカバリします。セッションをリカバリするために、ワークフローをサスペンド状態にしたり、ワークフローのリカバリを有効にしたりする必要はありません。セッションをリカバリする場合、Integration Service により別のセッションログが作成されます。

Workflow Monitor を使用してセッションをリカバリするには：

1. Workflow Monitor でワークフローをダブルクリックして展開し、タスクを表示します。
2. セッションを右クリックし、[タスクのリカバリ] を選択します。

Integration Service によって、失敗したセッションがリカバリ戦略に従ってリカバリされます。

`pmcmd starttask` を `-recover` オプション付きで使用して、セッションをリカバリすることもできます。

セッションからのワークフローのリカバリ

セッションが停止、強制終了、または終了し、ワークフローが完了しない場合、セッションのリカバリ戦略を設定していれば、セッションからワークフローをリカバリすることができます。セッションをリカバリすると、Integration Service はそのリカバリ戦略を使用し、セッションをリカバリしてワークフローを続行します。ワークフローをサスペンド状態にしたり、ワークフローのリカバリを有効にしたりしない場合でも、セッションをリカバリすることができます。セッションからワークフローをリカバリする場合、Integration Service により、別のセッションログが作成されます。

Workflow Monitor でセッションからワークフローをリカバリするには：

1. Workflow Monitor でワークフローをダブルクリックして展開し、セッションを表示します。
2. セッションを右クリックし、[このタスクのリカバリによるワークフローのリスタート] を選択します。

Integration Service によって、失敗したセッションがリカバリ戦略に従ってリカバリされます。

セッションからワークフローをリカバリするには、`pmcmd startworkflow` を `-recovery` オプション付きで使用します。

注：ワークレット内のセッションをリカバリするには、ワークレットを展開し、タスクのリカバリを選択します。

セッションリカバリに関するルールおよびガイドライン

セッションをリカバリする場合には、以下のルールおよびガイドラインを使用します。

- Integration Service は、セッションリカバリを実行するときに新しいセッションログを作成します。
- セッションは、最後に成功した実行のパフォーマンス統計をレポートします。
- シードパラメータを指定する場合は、乱数生成 (RAND) 関数を使用するトランスフォーメーションを含むセッションをリカバリできます。
- セッションリカバリの実行時に、マッピング変数が開始値にリセットされます。

最後のチェックポイントから再開するようにするためのリカバリの設定

最後のチェックポイントから再開するようにリカバリを設定するときは、次のルールとガイドラインに従う必要があります。

- トランスフォーメーションごとにパススルーパーティション化を使用する必要があります。
- グリッド上で実行されるセッションの場合は、最後のチェックポイントから再開するようにリカバリを設定することはできません。
- 完全なプッシュダウンの最適化にセッションを設定すると、Integration Service は、そのセッションをデータベース上で実行します。その結果、セッションが失敗すると、差分リカバリが実行されません。SQL オーバーライドを含んだセッションのリカバリを実行する場合、Integration Service はビューを削除し、再作成する必要があります。
- ワークフローまたはセッションを中断された実行とリカバリ実行の間で変更したときは、予期しない結果が得られることがあります。Integration Service では、変更されたワークフローに対するリカバリは防止されません。ワークフローまたはタスクが前回実行後に変更された場合、リカバリのワークフローログまたはセッションログはメッセージを表示します。
- 最後のチェックポイントからセッションを再開する場合、セッション実行前のコマンドと実行前 SQL コマンドは 1 回しか実行されません。実行前または実行後のコマンドまたは SQL コマンドが失敗すると、Integration Service は、リカバリ中にそのコマンドを再度実行します。再度実行できるようにコマンドを設計してください。
- セッションがリレーショナルターゲットへの書き込みを一括モードで行った場合、再開するようにそのセッションを設定することはできません。

リカバリ不可能なワークフローまたはタスク

場合によっては、Integration Service がワークフローまたはタスクをリカバリできないことがあります。以下の状況では、ワークフローまたはタスクをリカバリすることができません。

- **パーティション数を変更した場合。**セッションが失敗した後でパーティション数を変更した場合、リカバリセッションは失敗します。
- **中断されたタスクに、失敗リカバリ戦略がある場合。**コマンドまたはセッションのリカバリが失敗し、ワークフローのリカバリが続行されるように設定した場合、タスクをリカバリすることはできません。
- **リカバリストレージファイルが欠如している場合。**\$PMStorageDir にリカバリストレージファイルが見つからないか、\$PMStorageDir の定義が当初の実行からリカバリ実行に変更された場合、Integration Service はセッションまたはワークフローのリカバリに失敗します。
- **リカバリテーブルが空であるかターゲットデータベース内に存在しない場合。**Integration Service では、以下の状況でリカバリセッションが中止されます。
 - Integration Service で作成されたテーブルを削除した。
 - Integration Service がリカバリ情報をテーブルから削除した直後に、リカバリが有効になっているセッションが失敗した。

以下の状況でリカバリを実行すると、整合性のないデータが生成される場合があります。

- **初期セッション後にソースまたはターゲットが変更された場合。**セッションをリカバリする前にインデックスを削除または作成するか、ソースまたはターゲットテーブルのデータを編集した場合、Integration Service が存在しない行または繰り返し行を返すことがあります。
- **初期セッションの失敗後に、ソースまたはターゲットコードページが変更された場合。**ソースまたはターゲットコードページを変更した場合、Integration Service によって不正なデータが返されることがあります。そのコードページと元のコードページとの間で両方向の互換性がある場合は、リカバリを実行できません。

第 10 章

停止と強制終了

この章では、以下の項目について説明します。

- [停止と強制終了の概要, 201 ページ](#)
- [エラーのタイプ, 202 ページ](#)
- [Integration Service によるセッションの失敗の処理, 203 ページ](#)
- [ワークフローの停止または強制終了, 203 ページ](#)
- [停止または強制終了の手順, 204 ページ](#)

停止と強制終了の概要

タスク、ワークフロー、またはワークレットは、いつでも停止または強制終了することができます。

タスクの停止や強制終了とまったく同じように、セッションを停止したり強制終了したりできます。マッピングブロックで `ABORT()` 関数を使ってセッションを強制終了することもできます。セッションエラーが発生した場合、Integration Service によってセッションが早期に停止されることがあります。セッションにエラーしきい値を設定するか、マッピングで `ABORT` 関数を使用するか、あるいは Integration Service にセッションの停止を要求することによって、停止位置を制御できます。Integration Service により致命的なエラー（ターゲットデータベースへの接続が失われた場合など）が検出された場合、停止位置の制御はできません。

エラーの結果、セッションが失敗した場合、ワークフローをリカバリしてセッションをリカバリできます。

ワークフローを停止した場合、Integration Service は、ワークフローで現在実行中のすべてのタスクの停止を試みます。ワークフローにワークレットが含まれている場合、Integration Service は、ワークレット内の現在実行中のすべてのタスクの停止も試みます。Integration Service でワークレットを停止できない場合は、ワークフローを強制終了する必要があります。

Integration Service は、以下のタスクを完全に停止できます。

- セッション
- コマンド
- タイマ
- Event-Wait
- ワークレット

複数のコマンドを含んでいるコマンドタスクを停止した場合、Integration Service は現在のコマンドの実行を完了し、残りのコマンドを実行しません。Integration Service は、電子メールタスクなどのタスクは停止できません。例えば、停止コマンドを発行した際に既に Integration Service が電子メールを送信していた場合、Integration Service はワークフローの実行を停止する前に、電子メールの送信を完了します。

リポジトリサービスプロセスがシャットダウンした場合、Integration Service はワークフローを強制終了します。

エラーのタイプ

セッションエラーには、致命的なものと非致命的なものがあります。最初の発生でセッションの強制的な終了が行われないエラーが、非致命的エラーです。致命的なエラーは、Integration Service がソース、ターゲット、またはリポジトリにアクセスできない場合に発生します。

しきい値のエラー

非致命的エラーの発生が指定した回数に達したときにセッションが停止するように選択することができます。最初の発生でセッションの強制的な終了が行われないエラーが、非致命的エラーです。エラーのしきい値は、セッションプロパティで [Stop on Error (エラー発生時に停止)] オプションを使って指定できます。このオプションを有効にした場合、Integration Service では reader、writer、およびトランスフォーメーションの各スレッドで発生した非致命的エラーがカウントされます。

Integration Service では、ソースの読み込み、データの変換、およびターゲットへの書き込み時に、独立したエラー数が保持されます。セッションプロパティで [エラー時の停止] オプションを設定した場合、Integration Service では以下の非致命的エラーがカウントされます。

- **reader エラー。** ソースデータベースまたはソースファイルの読み込み中に Integration Service によって検出されたエラー。Unicode モードでセッションを実行している場合は、reader エラーしきい値に不揃いエラーを含めることができます。
- **writer エラー。** ターゲットデータベースまたはターゲットファイルへの書き込み中に Integration Service によって検出されたエラー。writer エラーしきい値には、キー制約違反、非 NULL フィールドへの NULL のロード、およびデータベーストリガ応答を含むことができます。
- **トランスフォーメーションエラー。** データの変換中に Integration Service によって検出されたエラー。トランスフォーメーションエラーしきい値には、変換エラー、および ERROR として設定したすべての条件 (NULL 入力など) を含むことができます。

1 つのパイプラインに複数のパーティションを作成した場合、Integration Service ではパーティションごとに別々のエラーしきい値が保持されます。Integration Service はすべてのパーティションのエラーしきい値に達した場合、セッションを停止します。writer が 1 つ以上のパーティションからのデータの書き込みを続行する場合がありますが、正常なリカバリを実行する機能には影響を与えません。

注: 非ラインシーケンシャル VSAM ファイルで不揃いエラーが発生した場合、Integration Service ではエラーしきい値が 1 に設定され、セッションが停止します。

致命的なエラー

致命的なエラーは、Integration Service がソース、ターゲット、またはリポジトリにアクセスできない場合に発生します。これには、データをロードするためのデータベース容量不足など、接続またはターゲットデータベースエラーが含まれます。セッションでノーマライズまたはシーケンスジェネレータトランスフォーメーションが使用されている場合、Integration Service がリポジトリ内でシーケンス値を更新することができず、致命的なエラーが発生します。

セッションでノーマライズまたはシーケンスジェネレータトランスフォーメーションが使用されておらず、Integration Service でリポジトリとの接続が失われた場合、セッションは停止しません。セッションは完了しますが、Integration Service はセッション統計をリポジトリに記録できません。

セッションは、Workflow Manager または *pmcmd* から停止できます。

セッションを、Workflow Manager から強制終了できます。指定されたトランスフォーメーションエラーが Integration Service によって検出された場合には、マッピングロジックで ABORT 関数を使用してセッションを強制終了させることもできます。

Integration Service によるセッションの失敗の処理

Integration Service では、セッションが失敗した原因のエラーまたはイベントに応じて、さまざまな方法でセッションエラーが処理されます。

以下の表に、セッションが失敗する際の Integration Service の動作を示します。

セッションエラーの原因	Integration Service の動作
<ul style="list-style-type: none">- reader エラーによってエラーのしきい値に達した- Workflow Manager または <i>pmcmd</i> を使用した停止コマンド	<p>Integration Service により以下のタスクが実行されます。</p> <ul style="list-style-type: none">- 読み込みを停止する。- データ処理を継続する。- ターゲットへのデータの書き込みおよびコミットを継続する。 <p>Integration Service がデータの処理やコミットを完了できない場合は、強制終了コマンドを発行してセッションを停止する必要があります。</p>
Workflow Manager を使用した強制終了コマンド	<p>Integration Service により以下のタスクが実行されます。</p> <ul style="list-style-type: none">- 読み込みを停止する。- データ処理を継続する。- ターゲットへのデータの書き込みおよびコミットを継続する。 <p>Integration Service で 60 秒以内にデータの処理とコミットを完了できない場合、Integration Service は DTM プロセスを中止し、セッションを終了します。</p>
<ul style="list-style-type: none">- データベースからの致命的なエラー- writer エラーによってエラーのしきい値に達した	<p>Integration Service により以下のタスクが実行されます。</p> <ul style="list-style-type: none">- 読み込みと書き込みを停止する。- ターゲットデータベースに書き込んでないデータをすべてロールバックする。 <p>セッションが致命的なエラーのために停止した場合、コミットまたはロールバックが成功する場合としない場合があります。</p>
<ul style="list-style-type: none">- トランスフォーメーションエラーによってエラーのしきい値に達した- ABORT ()- トランザクション制御式の無効な評価	<p>Integration Service により以下のタスクが実行されます。</p> <ul style="list-style-type: none">- 読み込みを停止する。- 行に強制終了行としてのフラグを設定し、データの処理を継続する。- 強制終了行に達するまで、ターゲットデータベースへの書き込みを継続する。- コミットの間隔に基づいてコミットを発行する。- ターゲットデータベースに書き込んでないデータをすべてロールバックする。

ワークフローの停止または強制終了

ワークフローの制御タスクを使用することにより、Integration Service によってワークフローを停止または強制終了する時間と方法を指定できます。ワークフローの開始後、Workflow Monitor または *pmcmd* を使用し

ワークフローの停止または強制終了ができます。停止または強制終了コマンドは、ワークフローの実行中のいつでも発行できます。

ワークフローを停止または強制終了するには、以下のいずれかの操作を行います。

- ワークフロー内で [Control] タスクを使用する。
- Workflow Monitor で停止または強制終了コマンドを発行する。
- pmcmd で停止または強制終了コマンドを発行する。

タスクの停止または強制終了

Workflow Monitor から、ワークフロー内のタスクを停止または強制終了することができます。タスクを停止または強制終了した場合、Integration Service はタスクの処理を停止します。停止または強制終了されたタスクのパス内にある他のタスクは、Integration Service によって処理されません。ワークフロー内の同時実行されるタスクの処理は、Integration Service によって続行されます。Integration Service がそのタスクを停止できない場合は、タスクを強制終了できます。

タスクを強制終了した場合、Integration Service はタスク上のプロセスを中止します。タスクを強制終了した場合、Integration Service によってワークフロー内の同時実行されるタスクの処理が続行されます。

また、ワークレットを停止または強制終了することもできます。Integration Service ではワークレットの停止と強制終了が、タスクの停止と強制終了の場合と同様に行われます。Integration Service は、ワークフロー内の同時実行されるタスクを実行する際に、ワークレットを停止します。また、ワークレット内のタスクを停止または強制終了することもできます。

セッションタスクの停止または強制終了

停止コマンドを発行する際に Integration Service がセッションタスクを実行していた場合、Integration Service はデータの読み込みを停止します。Integration Service により、ターゲットへのデータの書き込みやコミットの処理は続行されます。Integration Service がデータの処理やコミットを完了できない場合、強制終了コマンドを発行できます。

Integration Service では、Integration Service に 60 秒のタイムアウト期間があること以外は、セッションタスクの強制終了コマンドが停止コマンドのように処理されます。Integration Service は、データ処理やコミットをタイムアウト期間内に完了できない場合、DTM プロセスを中止し、セッションを終了します。

停止または強制終了の手順

Workflow Monitor では、タスク、ワークフロー、またはワークレットをいつでも停止または強制終了することができます。ワークフロー内のタスクを停止する場合、Integration Service はそのタスクと、そのパスにある他のすべてのタスクの実行を停止します。同時実行されるタスクの実行は、Integration Service によって続行されます。Integration Service がタスクの処理を停止できない場合は、タスクを強制終了する必要があります。Integration Service がタスクを強制終了する場合、Integration Service は DTM プロセスを中止し、タスクを終了します。

リアルタイムセッションの動作は、リアルタイムソースによって異なります。

Workflow Monitor でワークフロー、タスク、またはワークレットを停止または強制終了するには：

1. ナビゲータ内で、停止または強制終了したいタスク、ワークフロー、またはワークレットを選択します。
2. [タスク] - [停止] または [タスク] - [強制終了] をクリックします。

Workflow Monitor で、停止または強制終了コマンドのステータスがアウトプットウィンドウに表示されます。

第 11 章

コンカレントワークフロー

この章では、以下の項目について説明します。

- [コンカレントワークフローの概要, 205 ページ](#)
- [一意のワークフローインスタンスの設定, 206 ページ](#)
- [同一名のコンカレントワークフローの設定, 206 ページ](#)
- [パラメータおよび変数の使用, 208 ページ](#)
- [コンカレントワークフローの設定手順, 209 ページ](#)
- [コンカレントワークフローの開始および停止, 209 ページ](#)
- [コンカレントワークフローの監視, 211 ページ](#)
- [セッションログおよびワークフローログの表示, 211 ページ](#)
- [コンカレントワークフローに関するルールおよびガイドライン, 212 ページ](#)

コンカレントワークフローの概要

コンカレントワークフローとは、複数インスタンスとして並列実行できるワークフローのことです。ワークフローインスタンスとは、ワークフローの表現です。コンカレントワークフローの設定時には、ワークフローの単一インスタンスを Integration Service が複数回並列に実行できるようにするか、ワークフローの一意インスタンスを並列実行されるように定義します。

コンカレントワークフローを設定するには、次のいずれかのワークフローオプションを使用します。

- **同一インスタンス名を持つコンカレントワークフローの許可。** 1 つのワークフローインスタンスを複数回並列に実行されるように設定します。各インスタンスのソースパラメータ、ターゲットパラメータ、および変数パラメータは同じです。Integration Service は各インスタンスを実行 ID で識別します。実行 ID は、実行済みのワークフローインスタンスを特定する番号です。
- **一意のワークフローインスタンスの同時実行の設定。** 各ワークフローインスタンスの名前を定義して、インスタンスのワークフローパラメータファイルを設定します。パラメータファイル内には、さまざまなソース、ターゲット、および変数を定義することができます。

コンカレントワークフローを実行した場合、各ワークフローのワークフロー名およびインスタンス名が Workflow Monitor に表示されます。ワークフローに一意のインスタンス名が付いていない場合、実行済みの各コンカレントワークフローの同じワークフロー名が Workflow Monitor に表示されます。

Integration Service では、インスタンス名、または実行 ID とタイムスタンプのどちらかをワークフロー名およびセッションログ名に追加して、コンカレントワークフローの一意のログファイルが作成されます。

一意のワークフローインスタンスの設定

ワークフローの複数インスタンスを設定して、各インスタンスを一度に並列実行することができます。ワークフローインスタンスの設定時には、インスタンスの一意名を指定し、インスタンスのワークフローパラメータファイルを設定します。

ワークフローインスタンスを、別のソースとターゲットを持つワークフローを実行するように設定します。たとえば、所属組織が3部門から売上データを受け取るとします。売上データを読み取ってデータベースに書き込むワークフローを作成します。そのワークフローに対して3つのインスタンスを設定します。各インスタンスには、処理対象の売上ファイルを定義する別個のワークフローパラメータファイルがあります。ワークフローのインスタンスはすべて並列実行できます。

ワークフローの開始時には、実行対象のインスタンスを選択することができます。コンカレントワークフローが一意のインスタンスで実行されるように設定されている場合、インスタンスを並列に実行することができます。1つのインスタンスを複数回並列に実行するには、ワークフローを同じインスタンス名で実行されるように設定します。

ワークフローインスタンスのインスタンス名によるリカバリ

Workflow Monitor または *pmcmd* からワークフローインスタンスをリカバリできます。ワークフローのリカバリを有効にすると、Integration Service でワークフロー実行 ID がリカバリストレージファイル名に追加されます。

コンカレントワークフローをリカバリする際には、リカバリ対象のインスタンスを識別してください。Workflow Monitor で、リカバリ対象のインスタンスを右クリックします。*pmcmd* を使用してリカバリする際には、インスタンス名パラメータを入力します。

同一インスタンス名のコンカレントインスタンスの実行に関するルールおよびガイドライン

同一インスタンス名のコンカレントインスタンスを実行する際は、次の規則およびガイドラインに従ってください。

- Integration Service ではコンカレントワークフロー実行間で変数が上書きされます。
- ワークフローは *pmcmd* から実行 ID で強制終了できます。
- ワークフロータスクは *pmcmd* から実行 ID で強制終了できます。
- Workflow Monitor ではインスタンスごとの実行 ID は表示されません。実行 ID は、ワークフローログ、セッションログ、および Workflow Monitor の [Run Properties (実行プロパティ)] パネルにが表示されます。
- 同一インスタンス名のコンカレントワークフローが実行される設定にした場合、ログファイル名には常にタイムスタンプが組み込まれます。

同一名のコンカレントワークフローの設定

一意のインスタンス名を定義しなくても、ワークフローを並列に実行できます。同じワークフロー名のインスタンスを複数実行することができます。PowerCenter 統合サービスは、各ワークフローインスタンスを実行識別子番号 (実行 ID) で区別します。各ワークフロー実行には、一意の実行 ID が付けられます。PowerCenter 統合サービスはワークフローログ名、セッションログ名、リカバリファイル名などの一時ファイル名に実行 ID を付加して、ワークフローごとに個別のファイルを作成します。

ワークフローをリアルタイムソース（例えば、メッセージキューや Web サービス）から読み出すときは、コンカレントワークフローを同じインスタンス名で実行します。例えば、複数のプロジェクトチームからのデータを管理するとします。ソースデータおよびターゲットを判別するメッセージキューからデータを読み込むワークフローを作成します。インスタンスを複数回並列に実行して、別々の接続パラメータをメッセージキューからワークフローインスタンスに渡すことができます。

コンカレント Web サービスワークフローの実行

Web サービスワークフローを実行すると、Integration Service で複数のワークフローインスタンスを実行するパフォーマンスが向上します。ワークフローを Web サービスとして実行されるように設定するときは、ハブ上で実行されるワークフローインスタンスの数、および新規ワークフローインスタンスの起動タイミングを設定します。

ワークフローを Web サービスとして有効にすると、Workflow Designer で同一ワークフロー名のワークフローの並列実行が有効になります。Web Services Hub では、Web サービス用に設定された 1 ハブあたりの最大実行数、およびサービス時間プロパティに基づいて、Web サービスの新規インスタンスの起動タイミングが決定されます。

Web サービスワークフローインスタンスは、Web Services Hub によって起動されたときに、他のワークフローインスタンスと同じ名前が付けられます。

注: ワークフローを Web サービスとして有効にすると、Workflow Designer ではデフォルトで、ワークフローの並列実行が有効になります。

同一名のワークフローインスタンスの設定

同一インスタンス名のワークフローの並列実行を有効にした場合は、そのワークフロー用にワークフローインスタンスおよびパラメータファイルも設定できます。各インスタンスは並列して複数回起動することができます。

たとえば、ワークフローを定義して 2 つのインスタンスを作成しておく、そのワークフローを開始して、両方のインスタンスを実行することができます。ワークフローを再度開始して、同一インスタンスを並列に実行することができます。

並列実行されている 4 つのインスタンスは、Workflow Monitor のタスクビューに表示されます。

```
wf_sales [Instance1]
wf_sales [Instance2]
wf_sales [Instance1]
wf_sales [Instance2]
```

同一名のワークフローインスタンスのリカバリ

ワークフローのリカバリを有効にすると、ワークフローリカバリストレージファイルに実行 ID が付加されます。*pmcmd* を使用して同じ名前のワークフローをリカバリできます。Workflow Monitor を使用してリカバリすることはできません。コンカレントワークフローをリカバリする場合、実行 ID パラメータを入力する必要があります。

コンカレントワークフローのリカバリ時には、リカバリ対象のインスタンスを特定する必要があります。Workflow Monitor で、リカバリ対象のインスタンスを右クリックします。リカバリに *pmcmd* を使用するときは、実行 ID パラメータを入力します。

注: ワークフローでリレーショナルターゲットが更新される場合、最終チェックポイントからセッションをリカバリすることはできません。

同一インスタンス名のコンカレントインスタンスの実行に関するルールおよびガイドライン

同一インスタンス名のコンカレントインスタンスを実行する際は、次の規則およびガイドラインに従ってください。

- Integration Service ではコンカレントワークフロー実行間で変数が上書きされます。
- ワークフローは *pmcmd* から実行 ID で強制終了できます。
- ワークフロータスクは *pmcmd* から実行 ID で強制終了できます。
- Workflow Monitor ではインスタンスごとの実行 ID は表示されません。実行 ID は、ワークフローログ、セッションログ、および Workflow Monitor の [Run Properties (実行プロパティ)] パネルにが表示されます。
- 同一インスタンス名のコンカレントワークフローが実行される設定にした場合、ログファイル名には常にタイムスタンプが組み込まれます。

パラメータおよび変数の使用

競合を防ぐには、ワークフローインスタンスごとにパラメータファイルを設定します。

以下の表に、コンカレントワークフローの設定パラメータのリストを一覧表示します。

パラメータタイプ	パラメータ名
データベース接続	<i>\$DBConnectionName</i>
ソースファイル	<i>\$InputFileName</i>
ターゲットファイル	<i>\$OutputFileName</i>
拒否ファイル	<i>\$BadFileName</i>
ルックアップファイル	<i>\$LookupFileName</i>

Integration Service では、ワークフロー変数がワークフロー実行インスタンス名によって持続されます。

実行インスタンス名または実行 ID へのアクセス

一意のインスタンス名でワークフローを並列に実行するように設定した場合、Integration Service ではワークフロー実行インスタンスは実行インスタンス名で区別されます。各ワークフローインスタンスはワークフロー名と実行インスタンス名を組み合わせで定義されるため、複数のワークフローに対して同一の実行インスタンス名を設定することができます。同一のインスタンス名でワークフローを並列に実行するように設定した場合、Integration Service ではワークフロー実行インスタンスは実行 ID で区別されます。

組み込みの変数 *\$PMWorkflowRunInstanceName* および *\$PMWorkflowRunId* によって、ワークフロー実行インスタンス名および実行 ID が文字列値として返されます。これらの変数は読み取り専用です。ワークフローまたはマッピングでこれらの変数にアクセスして、ワークフローインスタンスの名前または実行 ID を取得できます。これらの変数は、式、ファイル監視イベント、またはデータに適用できます。また、これらの変数を使用して、一意のファイル名を設定することもできます。

例えば、事前定義済み [イベント待ち] タスクを作成して、インジケータファイルが表示された後に削除するとします。\$PMWorkflowRunInstanceName を使用して、ファイル名を定義します。一意のインスタンス名の付いた 2 つのコンカレントワークフローを実行すると、各ワークフロー [イベント待ち] タスクは別個のインジケータファイルを待って削除します。

注: 並列実行が有効化されていないワークフローを実行した場合、\$PMWorkflowRunInstanceName には値が代入されません。

コンカレントワークフローの設定手順

ワークフローの作成時、または編集時には、ワークフローの並列実行を有効にすることができます。

ワークフローの並列実行を有効にするには：

1. Workflow Manager で、ワークフローを開きます。
2. ワークフローの [全般] タブで、並列実行を有効にします。
同一インスタンス名のワークフローの並列実行が有効になります。
3. いくつかのインスタンス名を設定するには、[並列実行の設定] をクリックします。
[並列実行の設定] ダイアログボックスが表示されます。
4. 次のいずれかのオプションを選択します。
 - **一意のインスタンス名を持つもののみ同時実行を許可。** インスタンス名が一意である場合、Integration Service ではコンカレントワークフローを実行できます。
 - **同一インスタンス名を持つものの同時実行を許可。** Integration Service では、同じ名前を持つコンカレントワークフローを実行できます。
5. 必要に応じて、[追加] ボタンをクリックして、ワークフローインスタンスの名前を追加します。
ワークフローインスタンスの名前では、大文字と小文字は区別されません。インスタンス名の文字が Workflow Designer によって検査されます。インスタンス名に使用できない特殊文字は、次のとおりです。
\$. + - = ~ ` ! % ^ & * () [] { } ' \ " ; : / ? , < > \\ | \t \r \n
6. 必要に応じて、インスタンス用ワークフローパラメータファイルへのパスを入力します。ワークフローインスタンスごとに異なるソース、ターゲット、または変数を使用するには、インスタンスごとにパラメータファイルを設定します。
7. [OK] をクリックします。

コンカレントワークフローの開始および停止

コンカレントワークフローは、Workflow Designer または Workflow Monitor で開始することができます。ワークフローは *pmcmd* から開始することもできます。一意のワークフローインスタンスを実行するには、ワークフローの開始時に、実行するインスタンスを選択します。

Workflow Designer からのワークフローインスタンスの開始

実行するワークフローインスタンスは、Workflow Designer からワークフローを開始する際に選択することができます。定義済みインスタンスが 1 つ以上あるワークフローを開始するには、次の手順に従います。

Workflow Designer からワークフローインスタンスを起動するには：

1. ワークフローが格納されているフォルダを開きます。
2. ナビゲータで、開始するワークフローを選択します。
3. ワークフローを右クリックし、[Workflow Advanced の開始] を選択します。
4. 起動するワークフロー実行インスタンスを選択します。デフォルトでは、すべてのインスタンスが選択されています。ワークフローインスタンスをすべてクリアしてから、起動するインスタンスを選択することができます。
5. [OK] をクリックして、ワークフローインスタンスを起動します。

Workflow Monitor に各コンカレントワークフロー名およびインスタンス名が表示されます。

単一のコンカレントワークフローの開始

コンカレントワークフローに一意的インスタンス名が付いていない場合、またはユーザーが設定済みインスタンスの実行を望まない場合は、ワークフローの開始に Workflow Designer を使用することができます。[ワークフローの開始] オプションを使用してワークフローを開始した場合、Integration Service でのワークフロー実行には、[プロパティ] タブおよび [変数] タブで定義された属性および変数が使用されます。設定済みワークフローインスタンスはすべて Integration Service で実行されません。

1 つのコンカレントワークフローインスタンスを起動するには：

1. ワークフローが格納されているフォルダを開きます。
2. ナビゲータで、開始するワークフローを選択します。
3. ナビゲータ内でワークフローを右クリックし、[ワークフローの開始] を選択します。

[プロパティ] タブおよび [変数] タブから参照された属性を使用して、1 つのワークフローインスタンスが実行されます。

コマンドラインからのコンカレントワークフローの開始

コマンドラインからは一度に 1 つのワークフローインスタンスを起動できます。 `pmcmd startworkflow` コマンドには、インスタンス名を指定するパラメータがあります。コマンドラインからワークフローを開始してインスタンス名パラメータを入力すると、Integration Service によってワークフローのインスタンスが起動されます。複数のワークフローインスタンスを実行するには、`pmcmd startworkflow` コマンドを複数回実行します。

`startworkflow` にインスタンス名パラメータを入力しないと、Integration Service でのワークフロー実行に属性および変数が使用されます。設定済みワークフローインスタンスはすべて Integration Service で実行されません。

コマンドラインからのワークフローインスタンスの作成

`pmcmd` を使用してワークフローを開始する場合、インスタンスを動的に作成できます。インスタンス名およびパラメータファイル名を入力します。インスタンス名が設定されていない場合、インスタンスは Integration Service で生成されます。リポジトリ内のインスタンスの変数は Integration Service による永続化が可能ですが、そのインスタンスは [並列実行の設定] ダイアログボックスには表示されません。

コンカレントワークフローの停止または強制終了

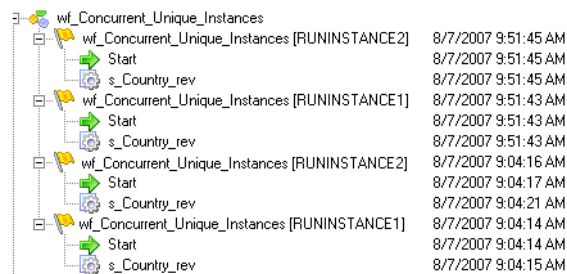
コンカレントワークフローは、Workflow Monitor または `pmcmd` から停止または強制終了することができます。ワークフローを Workflow Monitor から停止または強制終了するには、ナビゲータでワークフローを右クリックして、[停止] または [強制終了] を選択します。Workflow Monitor で、停止または強制終了コマンドのステータスがアウトプットウィンドウに表示されます。

コンカレントワークフローを *pmcmd* から停止または強制終了するには、コマンドラインでインスタンス名またはワークフロー実行 ID パラメータを入力して、ワークフローインスタンスを指定します。コンカレントワークフロー内のタスクを停止または強制終了するには、ワークフローインスタンスの名前、または停止するタスクを含むコンカレントワークフローの実行 ID を入力します。コンカレントワークフローに一意のインスタンス名が付いていない場合は、ワークフローログまたは Workflow Monitor のワークフロー実行プロパティにワークフロー実行 ID を調べることができます。

コンカレントワークフローの監視

コンカレントワークフローを実行すると、実行された各ワークフローのワークフロー名が Workflow Monitor に表示されます。ワークフローに一意のインスタンス名が付いている場合、Workflow Monitor にインスタンス名がワークフロー名付きで表示されます。

以下の図に、Workflow Monitor のタスクビューに表示されるコンカレントワークフロー名およびインスタンス名を示します。



ガントチャートビューでコンカレントワークフローを表示した場合、Workflow Monitor にワークフローの名前またはワークフローインスタンスの名前ごとに 1 つのタイムラインが表示されます。ワークフローに一意のインスタンス名 (RunInstance1、RunInstance2 など) が付いている場合、ワークフロー実行ごとのインスタンス名が Workflow Monitor に表示されます。タイムウィンドウをスクロールして、特定のワークフロー実行の情報を表示することができます。

セッションログおよびワークフローログの表示

Integration Service では設定された同時実行に基づいて、コンカレントワークフローセッションおよびワークフローログファイルに名前が付けられます。

- **一意のインスタンス名。** Integration Service によって、ログファイル名にインスタンス名が追加されます。
- **同一名のインスタンス。** Integration Service により、ログファイル名に実行 ID およびタイムスタンプが追加されます。

ワークフローログには実行 ID およびワークフロータイプが書き込まれます。ワークフロータイプは、ワークフローがコンカレントワークフローであるかどうかを示します。

以下に例を示します。

Workflow SALES_REV started with run id [108], run instance name [WF_CONCURRENT_SALES1], run type [Concurrent Run with Unique Instance Name].

各セッションログには、ワークフロー実行 ID およびインスタンス名を示すエントリも含まれます。

Workflow: [SALES_REV] Run Instance Name: [WF_CONCURRENT_SALES1] Run Id: [108]

注: エラー重要度レベルを警告にしたときにすべてのワークフローログメッセージが表示されない場合は、ワークフローログのエラー重要度レベルを変更してください。PowerCenter Integration Service プロセスの詳細プロパティで、ログレベルを警告から情報に変更します。

一意のワークフローインスタンス用のログファイル

一意のインスタンス名のワークフローを並列実行するように設定しておく、Integration Service によってインスタンスごとにログが作成されます。各ログファイル名には、インスタンス名が組み込まれます。

```
<workflow_name>.<workflow_instance_name>  
<session_name>.<workflow_instance_name>
```

たとえば、ワークフローログファイル名が wf_store_sales.log、インスタンス名が store1_workflow の場合、Integration Service でバイナリワークフローログファイルおよびテキストワークフローログファイル用に作成されるログファイル名は、次のようになります。

```
wf_store_sales.log.store1_workflow.bin  
wf_store_sales.log.store1_workflow
```

ログファイルの上書きを回避するために、ログファイルをアーカイブするか、またはログファイルをタイムスタンプで保存することができます。

同一名のワークフローインスタンス用のログファイル

ワークフローを同一インスタンス名で同時に実行するように設定した場合、Integration Service によってインスタンスごとにログファイルが作成されます。各ログファイル名にはデフォルトで、実行 ID およびタイムスタンプが組み込まれます。

```
<workflow_name>.<runID>.<timestamp>  
<session_name>.<run ID>.<timestamp>
```

例えば、ワークフローログファイル名が wf_store_sales.log、実行 ID が 845 で、2007 年 7 月 12 日 11:20:45 にワークフローが実行された場合、Integration Service でバイナリワークフローログファイルおよびテキストワークフローログファイル用に作成されるログファイル名は、次のようになります。

```
wf_store_sales.log.845.20070712112045.bin  
wf_store_sales.log.845.20070712112045
```

ワークフローを同一インスタンス名で並列実行されるように設定してインスタンス名を定義すると、Integration Service によってログファイル名にインスタンス名およびタイムスタンプが付加されます。以下に例を示します。

```
<workflow_name>.<instance_name>.<run ID>.20070712112045.bin  
<session_name>.<instance_name>.<run ID>.20070712112045.bin
```

Integration Service では、インスタンス名および実行 ID がワークフローログに書き込まれます。以下に例を示します。

```
Workflow wf_Stores started with run ID[86034], run instance name[Store1_workflow]
```

コンカレントワークフローに関するルールおよびガイドライン

コンカレントワークフローに使用される規則およびガイドラインは、次のとおりです。

- パラメータファイル内のワークフロー実行インスタンスは、参照することができません。インスタンスごとに別々のパラメータを使用するには、個別のパラメータファイルを設定する必要があります。

- 複数のコンカレントワークフローインスタンスに対して同じキャッシュファイル名を使用しても、各ワークフローインスタンスは有効です。ただし、キャッシュへの書き込み中に競合が発生した場合は、セッションが失敗します。
- コンカレントワークフローを実行 ID またはインスタンス名でリスタートするには、*pmcmd*を使用します。
- ワークフローの複数インスタンスを設定して、ワークフローをスケジュールしておく、スケジュールされた時刻にすべてのインスタンスが Integration Service によって実行されます。インスタンスをそれぞれ別個のスケジュールで実行することはできません。
- ワークレットの [全般] タブでは、ワークレットを並列に実行するように設定します。
- 親ワークフローの並列実行を有効にした場合は、ワークレットの並列実行を有効にする必要があります。そうしないと、ワークフローは無効です。
- ワークレットの並列実行を有効にして、そのワークレットを 2 つの非コンカレントワークフロー内に配置することができます。Integration Service では 2 つのワークレットを並列実行できます。
- 並列実行が有効化された 2 つのワークフローは、同一のワークレットを実行できます。同一ワークレット（ただし、そのワークレットに永続化された変数がない場合）の 2 つのインスタンスは、1 つのワークフローで実行できます。
- ワークレット内のセッションは、同一インスタンス名の別のワークレット内のセッション（ただし、そのセッションに永続化された変数が含まれていない場合）と並列実行できます。

コンカレントワークフローに関する制約のあるトランスフォーメーションは、次のとおりです。

- **アグリゲータトランスフォーメーション。**コンカレントワークフローには、差分集計を使用できません。セッションが失敗します。
- **ルックアップトランスフォーメーション。**コンカレントワークフローのルックアップトランスフォーメーションには、以下の規則およびガイドラインが使用されます。
 - コンカレントワークフローに対して、静的または動的なルックアップキャッシュを使用できます。
 - キャッシュが非永続的な場合、Integration Service によって、ワークフロー実行 ID が接頭語としてキャッシュファイル名に追加されます。
 - キャッシュが名前のない永続キャッシュの場合、Integration Service によって、実行インスタンス名が接頭語としてキャッシュファイル名に追加されます。
 - キャッシュが動的で名前がない永続キャッシュであり、現在のワークフローが同一インスタンス名による同時実行を許可するように設定されている場合、セッションは失敗します。
 - ルックアップキャッシュ名がパラメータ化されている場合、Integration Service によって、キャッシュファイルにパラメータ値に基づいた名前が付けられます。実行インスタンスごとに異なるファイル名を渡します。
- **シーケンスジェネレータトランスフォーメーション。**コンカレントワークフロー用に同一シーケンス番号のセットが生成されるのを回避するには、シーケンスジェネレータトランスフォーメーション内のキャッシュ値の数を設定します。

第 12 章

グリッド処理

この章では、以下の項目について説明します。

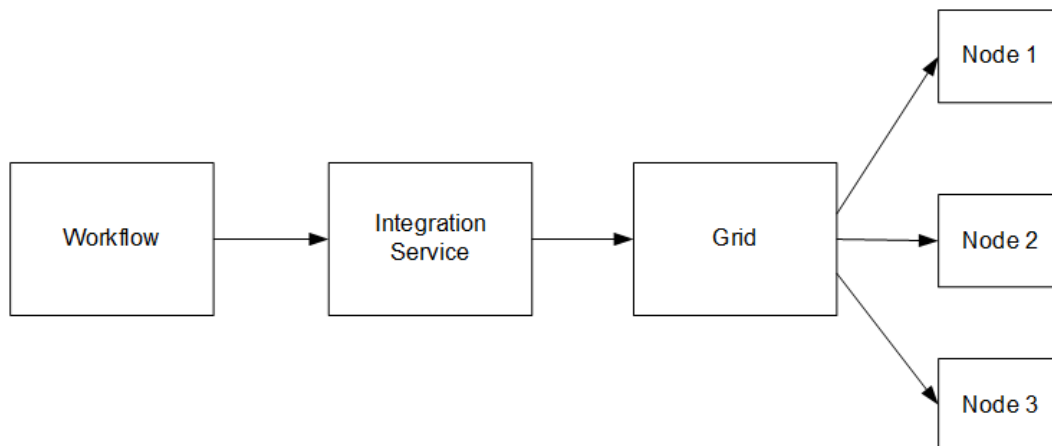
- [グリッド処理の概要, 214 ページ](#)
- [グリッド上でのワークフローの実行, 215 ページ](#)
- [グリッド上でのセッションの実行, 216 ページ](#)
- [パーティショングループに関する作業, 216 ページ](#)
- [グリッドの接続とリカバリ, 219 ページ](#)
- [グリッド上で実行するためのワークフローまたはセッションの設定, 219 ページ](#)

グリッド処理の概要

PowerCenter ドメインに複数のノードが含まれている場合、グリッド上で実行するようにワークフローとセッションを設定できます。グリッド上でワークフローを実行する場合、統合サービスによりグリッドの各使用可能なノードでサービスプロセスが実行され、パフォーマンスと拡張性が向上します。グリッド上でセッションを実行する場合、統合サービスによりグリッド内の各ノード上で複数の DTM プロセスにセッションスレッドが分散され、パフォーマンスと拡張性が向上します。

Administrator ツールで、グリッドの作成と統合サービスの設定を行います。グリッド上でワークフローを実行するには、グリッドと関連付けられた統合サービスで実行するようにワークフローを設定します。グリッド上でセッションを実行するには、グリッド上で実行するようにセッションを設定します。

次の図に、グリッド上でワークフローを実行する場合のワークフローとノード間の関係を示します。



統合サービスにより、実行するためのワークフローまたはセッションの、以下の設定方法に基づいて、ワークフロータスクとセッションスレッドが分散されます。

- **グリッド上でのワークフローの実行。** 統合サービスでは、グリッド内のノード全体にワークフローが分散します。また、グリッド内のノード全体にワークフロー内でセッション、コマンド、および定義済みイベント待ちタスクも分散します。
- **グリッド上でのセッションの実行。** 統合サービスでは、グリッド内のノード全体にセッションスレッドが分散します。

注: グリッド上でワークフローを実行するには、[サーバーグリッド] オプションを使用している必要があります。グリッド上でワークフローを実行するには、[サーバーグリッド] オプションを使用している必要があります。

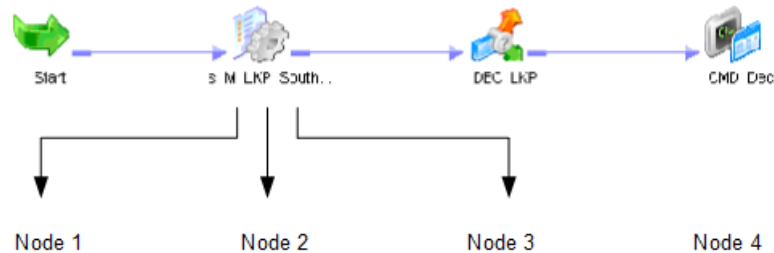
グリッド上でのワークフローの実行

グリッド上でワークフローを実行する場合、マスタサービスプロセスによって、他のノードに分散されるセッションタスク、コマンドタスク、および定義済みイベント待ちタスク以外のワークフローおよびすべてのタスクが実行されます。マスタサービスプロセスは、ワークフローが実行される統合サービスプロセスであり、他のノードで実行されているサービスプロセスを監視し、ロードバランサを実行します。スケジューラはマスタサービスプロセスノード上で実行されるので、マスタサービスプロセスノードのために日付および時間が使用され、スケジュール設定されたワークフローが開始されます。

ロードバランサは、グリッド内のノードにセッションタスク、コマンドタスク、および定義済みイベント待ちタスクをディスパッチする統合サービスのコンポーネントです。ロードバランサでは、ノードの可用性に基づいてタスクが分散します。統合サービスがリソースをチェックするように設定されている場合、ロードバランサではリソースの可用性に基づいても各タスクが分散します。

例えば、ワークフローに [セッション] タスク、[ディシジョン] タスク、および [コマンド] タスクが含まれているとします。[セッション] タスクのリソース要件を指定します。グリッドにはノードが4つ含まれており、ノード4は使用できないとします。マスタサービスプロセスは、[スタート] タスクと [ディシジョン] タスクを実行します。ロードバランサは、リソースの可用性とノードの可用性に基づいて、グリッド上の各ノードに [セッション] タスクと [コマンド] タスクを分散します。

次の図に、グリッド上のノードに分散されるワークフローを示します。



1. Reader スレッドは、リソースが使用可能なノード 1 で実行されます。
2. Transformation スレッドは、リソースが使用可能なノード 2 で実行されます。
3. Writer スレッドは、リソースが使用可能なノード 3 で実行されます。
4. ノード 4 は使用できません。このため、このノードではスレッドは実行されません。

グリッド上でのセッションの実行

グリッド上でセッションを実行する場合、マスタサービスプロセスによって、グリッド上でワークフローを実行する場合に行われるように、他のノードに分散されるセッションタスク、コマンドタスク、および定義済みイベント待ちタスク以外のワークフローおよびすべてのタスクが実行されます。スケジューラはマスタサービスプロセスノード上で実行されるので、マスタサービスプロセスノードのために日付および時間が使用され、スケジュール設定されたワークフローが開始されます。さらに、ロードバランサは、異なるノードで実行されている DTM プロセスにセッションスレッドを分散します。

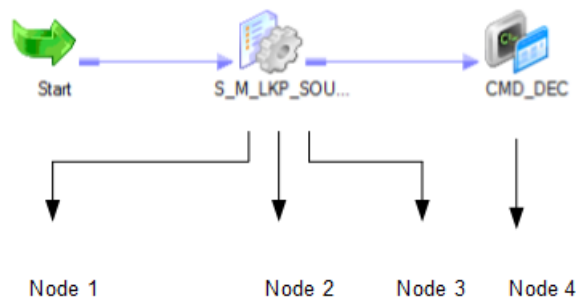
グリッド上でセッションを実行する場合、ロードバランサにより以下の要素に基づいてセッションスレッドが分散されます。

- **ノードの可用性。** ロードバランサによって、タスクのディスパッチに対して現在どのノードが実行され、有効であり、使用可能であるかが確認されます。
- **リソースの可用性。** 統合サービスがリソースをチェックするように設定されている場合、セッションでマッピングオブジェクトに必要なリソースを持つノードが特定されます。
- **パーティション化の設定。** ロードバランサにより、パーティション化の設定に基づいて、別々のノードにセッションスレッドのグループがディスパッチされます。

ワークフローに、実行に時間がかかるセッションが含まれている場合、グリッド上で実行するようにセッションを設定した方がよいことがあります。

例えば、ワークフローに、パーティションが 1 つ設定されたセッションが含まれているとします。負荷を分散するには、グリッド上で実行するようにセッションを設定し、リソースをチェックするように統合サービスを設定します。ロードバランサでは、reader、writer、およびトランスフォーメーションのスレッドが、グリッド内の各ノードで実行される DTM プロセスに分散します。reader スレッドにはリソースが必要であるため、ロードバランサにより、リソースが使用可能なノード上の DTM プロセスにこれらのスレッドが分散されます。

次の図に、グリッド内の各ノードで実行されている DTM プロセスに分散されるセッションスレッドを示します。



1. Reader スレッドは、リソースが使用可能なノードで実行されます。
2. Transformation スレッドは、使用可能なノード上で実行されます。
3. Writer スレッドは、使用可能なノード上で実行されます。
4. コマンドタスクは使用可能なノード上で実行されます。

パーティショングループに関する作業

グリッド上でセッションを実行すると、Data Transformation Manager (DTM) プロセスにより、パーティショングループと呼ばれるセッションスレッドのグループが形成されます。パーティショングループとは、単一の DTM プロセスで実行される reader、writer、またはトランスフォーメーションの各スレッドのグループを指します。パーティショングループには、1 つ以上のパイプラインステージが含まれています。パイプラインステージとは、任意の 2 つのパーティションポイント間で実行されるパイプラインの一区間です。トランスフ

オーメーションの中には、グリッドを越えてパーティションできないものがあります。グリッドを越えてトランスフォーメーションをパーティションできない場合、DTM は、トランスフォーメーションスレッドに単一のパーティショングループを作成して、単一のノードでこれらのスレッドを実行します。

リソース要件を使用しないパーティショングループの作成

セッションに複数のパーティションが設定されている場合、DTM は、パーティション設定に基づいてパーティショングループを形成します。

例えば、あるセッションを 2 つのパーティションで設定したとします。この場合、DTM が各パーティション内のスレッドごとにパーティショングループを作成し、ロードバランサがこれらのグループを 2 つのノードに分散します。パーティショングループ 1 はノード 1 で、パーティショングループ 2 はノード 2 で実行されます。

次の図は、2 つのパーティションを含むセッションの、2 つのパーティショングループを示しています。



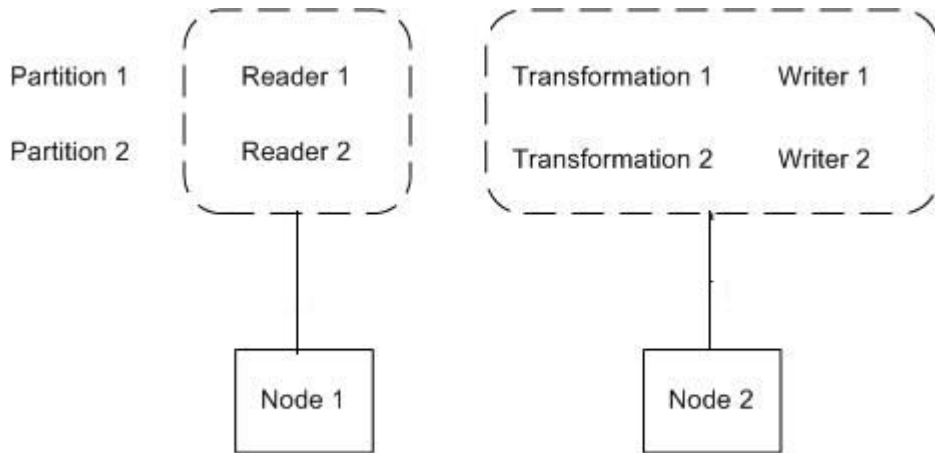
リソース要件を使用するパーティショングループの形成

マッピングオブジェクトにリソース要件を指定する場合、DTM プロセスは、特定のノードで使用できるリソースに基づいてパーティショングループを作成します。例えば、特定のノードで、セッションのソースファイルを使用でき、ソース修飾子トランスフォーメーションにリソース要件が指定されている場合、DTM プロセスは、この要件に基づいてパーティショングループを形成します。

ソース修飾子トランスフォーメーションのリソース要件を満たすために、DTM プロセスは、読み取りスレッドからパーティショングループを作成します。ロードバランサは、リソースを使用できるノードに読み取りスレッドを分散させます。

注: ロードバランサによって、必須リソースの使用可能なノードで、スレッドが分散されるようにするには、統合サービスがリソースをチェックするように設定する必要があります。

次の図に、パーティション化の設定とリソースの可用性に基づいて分散される2つのパーティショングループを示します。



パーティショングループの作成に関するルールおよびガイドライン

Integration Service によって、以下のルールとガイドラインが使用されパーティショングループが作成されます。

- Integration Service によって、パーティショングループの数は、グリッド内のノードの数の制限されます。
- トランスフォーメーションをローカル形式でパーティションできる場合、DTM プロセスは、トランスフォーメーションスレッドのパーティショングループを1つ形成して、1つのDTM プロセスでそのグループを実行します。以下の各トランスフォーメーションはローカル形式でパーティションされます。
 - ローカル形式でパーティションするよう設定されたカスタムトランスフォーメーション
 - エクスターナルプロシージャトランスフォーメーション
 - キャッシュされたルックアップトランスフォーメーション
 - 未ソートジョイナトランスフォーメーション
 - ローカル形式でパーティションするよう設定された SDK Reader または Writer トランスフォーメーション

キャッシュに関する作業

Integration Service によって、アグリゲータ、ランク、ジョイナ、ソータ、およびルックアップの各トランスフォーメーションにインデックスとデータキャッシュが作成されます。セッションに複数のパーティションが含まれている場合、トランスフォーメーションスレッドを、グリッド内の複数のノードに分散できます。これらのトランスフォーメーションスレッドに単一のデータキャッシュとインデックスキャッシュを作成するには、ルートディレクトリとキャッシュディレクトリが、グリッド内のノードすべてで同じディレクトリを指していることを確認します。

Integration Service によって、共有ディレクトリにルックアップトランスフォーメーションのキャッシュが作成される場合、最初のパーティショングループにキャッシュが構築されます。そして、後続のパーティショングループによってこのキャッシュが使用されます。ルックアップトランスフォーメーションキャッシュファイルに共有ディレクトリを設定しないと、個別のノード上にある各サービスプロセスは、データベースまたはソースファイルからデータをフェッチしてキャッシュを作成します。ソースデータが頻繁に変更される場合、個別のノードで作成された各キャッシュが矛盾することがあります。

関連項目：

- [「セッションのキャッシュ」](#) (ページ 307)

グリッドの接続とリカバリ

グリッド上でワークフローまたはセッションを実行する場合、サービスプロセスと DTM プロセスは、異なるノード上で実行されます。ネットワーク障害が発生すると、個別のノード上で実行されている各プロセス間の接続が失われることがあります。サービスが予期せずシャットダウンすることがあります。あるいは、ワークフローまたはセッションの実行中に Integration Service またはサービスプロセスを無効にできます。これらの状況での Integration Service のフェイルオーバーおよびリカバリの動作は、サービスプロセスが無効にされたか、シャットダウンしたか、接続が失われたかによって異なります。リカバリ動作は、以下の要素によっても異なります。

- **高可用性オプション。** 高可用性を使用している場合、ノードまたはサービスがシャットダウンすると、ワークフローは別のノードにフェイルオーバーされます。高可用性を使用していない場合、別のノードでワークフローを手動で再起動すればリカバリできます。
- **リカバリ戦略。** エラー発生時にサスペンド状態にするようにワークフローを設定できます。ワークフロー内のタスクにリカバリ戦略を設定できます。ワークフローが一時停止する場合、リカバリ動作は、ワークフロー内のタスクごとに設定したリカバリ戦略によって異なります。
- **シャットダウンモード。** Integration Service またはサービスプロセスを無効にした場合、サービスが、サービス上で実行されるプロセスを完了、強制終了、または停止するように指定できます。Integration Service を無効にするか、サービスプロセスを無効にするかで、動作が異なります。また、マスタサービスプロセスを無効にするか、Worker Service プロセスを無効にするかでも動作が異なります。Integration Service またはサービスプロセスも、予期せずシャットダウンすることがあります。この場合、フェイルオーバーとリカバリ動作は、シャットダウンするサービスプロセスと設定済みリカバリ戦略により異なります。
- **実行モード。** グリッド上でワークフローが実行される場合、Integration Service は、別のノード上でワークフローとタスクがリカバリできます。グリッド上でセッションが実行される場合、再開リカバリ戦略を設定できません。
- **操作モード。** Integration Service がセーフモードで実行されている場合、セッションおよびワークフローでリカバリは無効になっています。

注: Integration Service がグリッド内で実行されている場合は、セーフモードでフェイルオーバーするように設定することはできません。

グリッド上で実行するためのワークフローまたはセッションの設定

グリッド上でセッションまたはワークフローを実行する前に、グリッドを複数のノードに割り当て、Integration Service はグリッド上で実行するように設定する必要があります。Administrator ツールで、グリッドの作成と Integration Service の割り当てを行います。ドメイン管理者でこれらの設定を確認することが必要な場合があります。

グリッド上でワークフローまたはセッションを実行するには、以下のプロパティおよび設定を使用します。

- **ワークフロープロパティ。** ワークフロープロパティの [全般] タブで、ワークフローを実行する Integration Service を割り当てます。グリッド上で実行するように Integration Service が設定されていることを確認します。
- **セッションプロパティ。** グリッド上でセッションを実行するには、セッションプロパティの [設定オブジェクト] タブで、グリッド上でセッションを実行できるようにします。
- **リソース要件。** セッション、コマンド、定義済みイベント待ちタスクの [全般] タブで、リソース要件を設定します。

グリッド上で実行するためのワークフローまたはセッションの設定に関するルールおよびガイドライン

グリッドで実行するようにセッションまたはワークフローを設定する際には、以下のルールおよびガイドラインを使用します。

- グリッドでセッションを実行するには、オペレーティングシステムとビットモードがグリッドの各ノードで同一であることを確認します。ノードが異なるオペレーティングシステムまたはビットモードで動作している場合、セッションはそのグリッドで実行できない場合があります。
- サービスプロセス変数をオーバーライドする場合には、Integration Service が、入力ファイル、キャッシュ、ログ、ストレージ、および一時ディレクトリ、およびソースとターゲットファイルディレクトリにアクセスできることを確認します。
- セッション、コマンド、定義済みイベント待ちタスクが特定のノードで実行することを確認するには、Integration Service がリソースをチェックしてタスクにリソース要件を指定するように設定します。
- マッピングオブジェクトのセッションスレッドが特定のノードで実行することを確認するには、Integration Service がリソースをチェックしてそのオブジェクトにリソース要件を指定するように設定します。
- キャッシュファイルを作成するセッションを実行する場合、共有場所を使用するようにルートディレクトリとキャッシュディレクトリを設定して、キャッシュファイル間での一貫性を確認します。
- ジョイナトランスフォーメーションでパーティションポイントを追加し、このトランスフォーメーションが 1:n のパーティション化として設定されている場合、Integration Service が、共有場所内にキャッシュを構築することを確認します。明細パイプライン用のキャッシュを共有する必要があります。
- ルックアップトランスフォーメーションでパーティションポイントを追加し、パーティションタイプがハッシュ自動キーでない場合、Integration Service が、共有場所内にキャッシュを構築することを確認します。
- 動的パーティションを使用するセッションを実行して、グリッド内のノード全体にセッションスレッドを分散する場合は、セッション用に動的パーティション化を設定して「グリッドのノード数に基づく」方法を使用します。
- デバッグセッションはグリッド上で実行できません。
- グリッド上で実行するセッションでは、レジュームリカバリ戦略を設定できません。
- 実行に時間がかかるセッションを使用する場合、グリッド上で実行するようにセッションを設定します。
- 複数のコンカレントセッションを使用している場合、グリッド上で実行するようにワークフローを設定します。
- グリッド上で永続プロファイルセッションを実行できますが、グリッド上で一時プロファイルセッションを実行することはできません。
- シーケンスジェネレータトランスフォーメーションを使用する場合、キャッシュされる値の数を増加して、マスタおよびワーカ DTM プロセスとリポトリ間で必要な通信の回数を減らします。

- グリッド上でワークフローまたはセッションを実行する場合にログビューアが必ずログイベントを正確に順序付けることができるように、時間同期化ソフトウェアを使用して、グリッドのノードが確実に同期化された日時を使用するようにします。
- Windows 環境でワークフローが [E-Mail] タスクを使用する場合、[E-Mail] タスクを確実に実行できるように、各ノードで同じ Microsoft Outlook プロファイルを設定します。

第 13 章

ロードバランサ

この章では、以下の項目について説明します。

- [ロードバランサの概要, 222 ページ](#)
- [ワークフローへのサービスレベルの割り当て, 222 ページ](#)
- [タスクへのリソースの割り当て, 223 ページ](#)

ロードバランサの概要

ロードバランサによって、ノード上で実行されている Integration Service プロセスにタスクがディスパッチされます。ワークフローを実行する場合、ロードバランサによって、ワークフロー内のセッション、コマンド、定義済みイベント待ちタスクがディスパッチされます。Integration Service がリソースをチェックするように設定されている場合、ロードバランサによって、タスク要件とリソース可用性が一致し、タスクを実行するのに最適なノードが特定されます。単一のノードまたは複数ノード全体にタスクをディスパッチする場合があります。

タスクを実行できるノードを特定するために、ロードバランサではタスクが必要とするリソースを各ノードで使用可能なリソースと一致させます。ロードバランサにより、タスクを受け取った順序でタスクがディスパッチされます。ディスパッチするセッションタスクおよびコマンドタスクの数が、Integration Service がその時点で実行できる数を超えている場合、ロードバランサはそのタスクをディスパッチキューに入れます。ノードが利用可能になったとき、キューから待機しているタスクは、ロードバランサにより、ワークフローのサービスレベルによって決定される順序でディスパッチされます。

Workflow Manager を使用して、リソースおよびサービスレベルを割り当てます。以下のタスクを実行できます。

- **サービスレベルの割り当て。**ワークフローにサービスレベルを割り当てます。サービスレベルでは、ディスパッチを待機しているワークフロータスク間の優先度が設定されます。
- **リソースの割り当て。**タスクにリソースを割り当てます。セッション、コマンド、および定義済みのイベント待ちの各タスクは、成功させるために PowerCenter リソースを必要とします。Integration Service がリソースをチェックするように設定されている場合、ロードバランサではリソースが利用可能になっているノードにタスクをディスパッチします。

ワークフローへのサービスレベルの割り当て

サービスレベルによって、ロードバランサがディスパッチキューからタスクをディスパッチする順序が決定します。複数のタスクがディスパッチを待機している場合、ロードバランサによって、優先度の高いタスクの後

に優先度の低いタスクがディスパッチされます。Administrator ツールでサービスレベルを作成し、ディスパッチ優先度を設定します。

ワークフロープロパティの [全般] タブで、サービスレベルをワークフローに割り当てます。

タスクへのリソースの割り当て

PowerCenter リソースとは、タスクを成功させるためにタスクが必要とするデータベース接続、ファイル、ディレクトリ、ノード名、およびオペレーティングシステムのタイプのことです。ロードバランサでは、リソースを使用しタスクをディスパッチできます。Integration Service が、グリッド上で実行するように設定されていない場合、またはリソースをチェックするように設定されていない場合、リソース要件はロードバランサに無視されます。そのノード上で実行されているマスタ Integration Service プロセスにすべてのタスクがディスパッチされます。

Integration Service がグリッド上で実行されている場合、およびリソースをチェックするように設定されている場合、ロードバランサではリソースを使用しタスクをディスパッチします。Integration Service ではワークフローのタスクが必要とするリソースをグリッド内の各ノードで使用できるリソースと一致させ、どのノードがタスクを実行できるか判断します。ロードバランサによって、セッション、コマンド、および定義済みのイベント待ちの各タスクが、使用可能なリソースを持つノードに分散されます。例えば、セッションにより予約語ファイルのファイルリソースが必要な場合、ロードバランサによって、このセッションが、このファイルにアクセスできるノードにディスパッチされます。必須リソースが利用可能なノードを Integration Service が特定できない場合、タスクは失敗します。

Administrator ツールでは、各ノードで使用可能なリソースを定義します。リソースには、定義済みリソースとユーザー定義リソースがあります。定義済みリソースには、ノードで使用できる接続、ノード名、およびオペレーティングシステムのタイプが含まれています。ユーザー定義リソースには、ファイル/ディレクトリリソースおよびカスタムリソースが含まれています。

タスクのプロパティでは、PowerCenter リソースを、これらのリソースを必要とする再利用不可能なタスクに割り当てます。再利用可能なタスクにリソースを割り当てることはできません。

以下の表に、リソースタイプと、リソースタイプに割り当て可能なリポジトリオブジェクトを一覧表示します。

リソースタイプ	定義済み/ ユーザー定義	リソースを使用するリポジトリオブジェクト
カスタム	ユーザー定義	[セッション]、[コマンド]、および定義済みの [Event Wait] タスクのインスタンス、およびセッション内のすべてのマッピングオブジェクト。
ファイル/ディレクトリ	ユーザー定義	[セッション]、[コマンド]、および定義済みの [Event Wait] タスクのインスタンス、およびセッション内の次のマッピングオブジェクト。 <ul style="list-style-type: none"> - ソース修飾子 - アグリゲータトランスフォーメーション - カスタムトランスフォーメーション - エクスターナルプロシージャトランスフォーメーション - ジョイナトランスフォーメーション - ルックアップトランスフォーメーション - ソータトランスフォーメーション - カスタムトランスフォーメーション - Java トランスフォーメーション - HTTP トランスフォーメーション - SQL トランスフォーメーション - 共有体トランスフォーメーション - ターゲット
ノード名	定義済み	[セッション]、[コマンド]、および定義済みの [Event Wait] タスクのインスタンス、およびセッション内のすべてのマッピングオブジェクト。
オペレーティングシステムタイプのタイプ	定義済み	[セッション]、[コマンド]、および定義済みの [Event Wait] タスクのインスタンス、およびセッション内のすべてのマッピングオブジェクト。

リポジトリオブジェクトに適用されないリソースタイプを割り当てようとすると、Workflow Manager は、以下のエラーメッセージを表示します。

The selected resource cannot be applied to this type of object. Please select a different resource.

Workflow Manager により、接続リソースが割り当てられます。リレーショナル、FTP、または外部ローダー接続を使用する場合は、Workflow Manager によって接続リソースが、セッションインスタンス内のソース、ターゲット、およびトランスフォーメーションに割り当てられます。Workflow Manager では接続リソースを手動で割り当てることができません。

リソースをタスクインスタンスに割り当てするには：

1. Worklet Designer または Workflow Designer でタスクプロパティを開きます。
[Event Wait] タスクの場合、リソースを割り当てることができるのは、タスクが定義済みのイベントを待機している場合に限ります。
2. [全般] タブの [編集] をクリックします。
3. [リソースの編集] ダイアログボックスの [追加] をクリックして、リソースを追加します。
4. [リソースの選択] ダイアログボックスで、リソースを割り当てるオブジェクトを選択します。[リソース] リストに、Integration Service を実行するノードで使用できるリソースが表示されます。

5. 割り当てるリソースを選択した後、[選択] をクリックします。
6. [リソースの編集] ダイアログボックスで [OK] をクリックします。

第 14 章

ワークフロー変数

この章では、以下の項目について説明します。

- [ワークフロー変数の概要, 226 ページ](#)
- [定義済みワークフロー変数, 227 ページ](#)
- [ユーザー定義ワークフロー変数, 231 ページ](#)
- [ワークレット変数の使用, 234 ページ](#)
- [ワークレットでの変数値の割り当て, 235 ページ](#)

ワークフロー変数の概要

変数を作成してワークフロー内で使用すると、値を参照したり情報を記録したりできます。例えば、[ディシジョン] タスク内で変数を使用して、直前のタスクが正常に実行されたかどうかを判定するとします。直前のタスクが正常に実行されたと判定された場合は、次のタスクを実行できます。直前のタスクが正常に実行されたと判定されなかった場合は、ワークフローを停止することができます。

以下のタイプのワークフロー変数を使用します。

- **定義済みワークフロー変数。** Workflow Manager によって、ワークフロー内のタスク用に定義済みワークフロー変数が提供されます。
- **ユーザー定義ワークフロー変数。** ワークフローを作成する際に、ユーザー定義ワークフロー変数を作成します。

以下のタイプのタスクを設定する際に、ワークフロー変数を使用します。

- **割り当てタスク。** ユーザー定義のワークフロー変数に値を割り当てるために、割り当てタスクを使用します。例えば、カレント値に 1 を加えるように変数を設定することで、ユーザー定義のカウンタ変数を増やすことができます。
- **ディシジョンタスク。** ディシジョンタスクによって、Integration Service でどのようにワークフローが実行されるか決定されます。例えば、最初のセッションが正常に終了した場合のみ 2 番目のセッションを実行するために、ステータス変数が使用されます。
- **リンク。** リンクによって各ワークフロータスクが接続されます。ワークフローにブランチを作成するために、リンクでワークフロー変数を使用します。例えば、ディシジョンタスクの後に、判定条件が真に評価された場合のリンクと偽に評価された場合のリンクを作成することができます。
- **タイマータスク。** タイマータスクによって、Integration Service でワークフロー内の次のタスクがいつ実行されるか指定されます。Integration Service によって次のタスクの実行が開始される時刻を指定するには、ユーザー定義の日付/時刻変数を使用します。

変数を使用する式を作成するには、式エディタを使用します。式を作成するときは、[組み込み] タブで組み込み変数を選択できます。ユーザー定義変数は、[ユーザー定義] タブで選択できます。[関数] タブには、ワー

クフロー変数といっしょに使用する関数が含まれます。ポイントアンドクリック方法を使用して、変数を使用する式を入力します。

ユーザー定義ワークフロー変数および組み込みワークフロー変数を使用した式を作成するには、以下のキーワードを使用します。

- AND
- OR
- NOT
- TRUE
- FALSE
- NULL
- SYSDATE

定義済みワークフロー変数

ワークフローごとに一式の組み込み変数があり、それらを使ってワークフローやタスクの状況进行评估します。以下のタイプの組み込みワークフロー変数を使用します。

- **タスク固有の変数。** Workflow Manager は、ワークフロー内のタスクごとに一式のタスク固有の変数を定義しています。リンク条件にタスク固有の変数を使用して、ワークフローの実行中に Integration Service がたどるパスを制御します。Workflow Manager は、式エディタ内のタスク名の下にタスク固有の変数をリストします。
- **組み込み変数。** ワークフローで組み込み変数を使用して、実行時情報またはシステム情報（フォルダ名、統合サービス名、システム日付、ワークフロー開始時刻など）を返します。Workflow Manager では、式エディタの組み込みノードの下に組み込み変数が一覧表示されます。

ヒント: Integration Service でログファイルのエラー重要度レベルを、[トレース] に設定した場合、ワークフローログに、ワークフロー変数の値が表示されます。このロギングレベルは、トラブルシューティングの場合にのみ使用されます。

以下の表に、Workflow Manager で使用可能なタスク固有のワークフロー変数を一覧表示します。

タスク固有の変数	説明	タスクタイプ	データ型
Condition	判定条件式の評価結果。 タスクが失敗した場合、Workflow Manager は、Condition に NULL を設定したままにします。 構文の例： <code>\$Dec_TaskStatus.Condition = <TRUE FALSE NULL any integer></code>	判断	整数
EndTime	関連タスクが終了した日付および時刻。精度は秒です。 構文の例： <code>\$s_item_summary.EndTime > TO_DATE('11/10/2004 08:13:25')</code>	すべてのタスク	日付/時刻

タスク固有の変数	説明	タスクタイプ	データ型
ErrorCode	<p>関連タスクに対して最後に出力されるエラーコード。エラーがない場合、タスク完了時に ErrorCode は 0 に設定されます。</p> <p>構文の例：</p> <pre>\$s_item_summary.ErrorCode = 24013</pre> <p>注: タスクがこの最終エラーメッセージで失敗することが多い場合、この変数を使用します。</p>	すべてのタスク	整数
ErrorMsg	<p>関連タスクに対して最後に出力されるエラーメッセージ。エラーがない場合、タスク完了時に ErrorMsg は空の文字列に設定されます。</p> <p>構文の例：</p> <pre>\$s_item_summary.ErrorMsg = 'PETL_24013 Session run completed with failure'</pre> <p>Nstring 型の変数の最大文字数は 600 文字です。</p> <p>注: タスクがこの最終エラーメッセージで失敗することが多い場合、この変数を使用します。</p>	すべてのタスク	Nstring
FirstErrorCode	<p>セッションにおける最初のエラーメッセージのエラーコード。</p> <p>エラーがない場合、セッション完了時に FirstErrorCode は 0 に設定されます。</p> <p>構文の例：</p> <pre>\$s_item_summary.FirstErrorCode = 7086</pre>	セッション	整数
FirstErrorMsg	<p>セッションにおける最初のエラーメッセージ。</p> <p>エラーがない場合、タスク完了時に FirstErrorMsg に空の文字列が設定されます。</p> <p>構文の例：</p> <pre>\$s_item_summary.FirstErrorMsg = 'TE_7086 Tscrubber: Debug info... Failed to evalWrapUp'</pre> <p>Nstring 型の変数の最大文字数は 600 文字です。</p>	セッション	Nstring
PrevTaskStatus	<p>前回実行されたワークフローのタスクの状態。次の状態があります。</p> <ul style="list-style-type: none"> - ABORTED - FAILED - STOPPED - SUCCEEDED <p>前回のタスク状態を評価する式を記述するときに、上記のキーワードを使用します。</p> <p>構文の例：</p> <pre>\$Dec_TaskStatus.PrevTaskStatus = FAILED</pre>	すべてのタスク	整数
SrcFailedRows	<p>ソースからの読み込みに失敗した行の合計数。</p> <p>構文の例：</p> <pre>\$s_dist_loc.SrcFailedRows = 0</pre>	セッション	整数

タスク固有の変数	説明	タスクタイプ	データ型
SrcSuccessRows	ソースからの読み込みが成功した行の合計数。 構文の例： \$s_dist_loc.SrcSuccessRows > 2500	セッション	整数
StartTime	関連タスクが開始された日付および時刻。精度は秒です。 構文の例： \$s_item_summary.StartTime > TO_DATE('11/10/2004 08:13:25') 注: SESSSTARTTIME は、統合サービスがセッションを初期化した後でセッションを実行するノードの現在の日付と時刻を返します。マッピングまたはマップレットで SESSSTARTTIME を使用する場合、同じセッションで StartTime と SESSSTARTTIME の値は異なります。	すべてのタスク	日付/時刻
Status	ワークフロー内の前回実行タスクの状態。次の状態があります。 - ABORTED - DISABLED - FAILED - NOTSTARTED - STARTED - STOPPED - SUCCEEDED 現在のタスク状態を評価する式を記述するときに、上記のキーワードを使用します。 構文の例： \$s_dist_loc.Status = SUCCEEDED	すべてのタスク	整数
TgtFailedRows	ターゲットへの書き込みに失敗した行の合計数。 構文の例： \$s_dist_loc.TgtFailedRows = 0	セッション	整数
TgtSuccessRows	ターゲットに正常に書き込まれた行の合計数。 構文の例： \$s_dist_loc.TgtSuccessRows > 0	セッション	整数
TotalTransErrors	トランスフォーメーションエラーの合計数。 構文の例： \$s_dist_loc.TotalTransErrors = 5	セッション	整数

Status を除くすべての組み込みワークフロー変数のデフォルト値は NULL です。ワークフロー内でまだ実行されていないタスクから組み込み変数が検出されると、デフォルト値の NULL が使用されます。したがって、まだ実行されていないタスクに依存する式およびリンク条件は有効です。Status のデフォルト値は NOTSTARTED です。

式における定義済みワークフロー変数の使用

式にワークフロー変数を使用するときは、Integration Service が式を評価して True または False を返します。条件が True と評価された場合、Integration Service は次のタスクを実行します。Integration Service はワークフローログに以下のメッセージのようなエントリを書き込みます。

INFO : LM_36506 : (1980|1040) Link [Session2 --> Session3]: condition is TRUE for the expression [\$Session2.PrevTaskStatus = SUCCEEDED].

式エディタは、組み込みワークフロー変数を [組み込み] タブに表示します。Workflow Manager では、タスク固有の変数がタスク別にグループ化されて表示されるとともに、ビルトイン変数が [ビルドイン] ノードに表示されます。式内で変数を使用するには、目的の変数をダブルクリックします。式エディタは、タスク固有の変数を以下の形式で [式] フィールドに表示します。

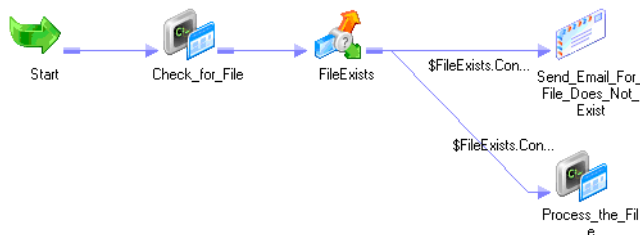
`$<TaskName>.<predefinedVariable>`

ワークフロー内の条件の評価

判定条件式の結果を評価するには、リンク条件の条件を使用します。

以下の図に、条件を使用したリンク条件のあるワークフローを示します。

図 4. 条件変数の例



FileExist ディシジョンタスクの判定条件式は、`$Check_for_file.Status = SUCCEEDED` です。マッピングには次の 2 つのリンク条件が含まれます。`$FileExists.Condition = False` によって電子メールタスクが実行され、`$FileExists.Condition = True` によってコマンドタスク `Process_the_File` が実行されます。

ワークフローを実行する際に、統合サービスはリンク条件を評価し、FileExists ディシジョンタスクの判定条件式に基づいた値を返します。統合サービスでは、Check_for_File タスクの結果に応じて、電子メールタスクまたはコマンドタスクが実行されます。

ワークフロー内のタスクステータスの評価

ワークフロー内の前回実行タスクの状態をテストするには、リンク条件の Status を使用します。

以下の図に、ステータスを使用したリンク条件のあるワークフローを示します。

図 5. ステータス変数の例



ワークフローを実行するときに、Integration Service はリンク条件 `$Session2.Status = SUCCEEDED` を評価し、セッション 2 のステータスに基づいた値を返します。

ワークフロー内の前回実行のタスクステータスの評価

Integration Service が前回実行したワークフロー内のタスクの状態をテストするには、リンク条件の PrevTaskStatus を使用します。

ワークフロー内のタスクを無効にする場合は、PrevTaskStatus を使用します。条件に無効タスクを使用していない場合は、Status と PrevTaskStatus は同じ値を返します。

以下の図に、PrevTaskStatus を使用したリンク条件のあるワークフローを示します。

図 6. PrevTaskStatus 変数の例



ワークフローを実行するときに、Integration Service は無効となったセッション 2 をスキップします。Integration Service は、リンク条件 `$Session2.PrevTaskStatus = SUCCEEDED` が評価されると、セッション 1 の状態に基づいた値を返します。

ヒント: セッション 2 を無効にしないと、Integration Service はセッション 2 の状態に基づいた値を返します。セッション 2 を有効または無効にするときに、リンク条件を変更する必要はありません。

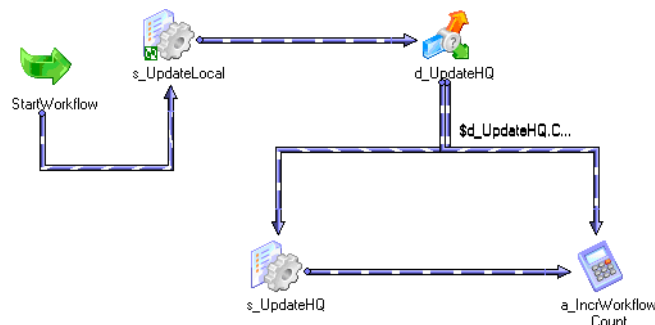
ユーザー定義ワークフロー変数

ワークフロー内に変数を作成できます。ワークフロー内に自分で変数を作成した場合、変数はそのワークフロー内でのみ有効になります。作成した変数は、そのワークフロー内のタスクで使用します。ユーザー定義のワークフロー変数は、編集や削除が可能です。

ユーザー定義の変数は、自分で指定した条件に基づいてワークフローの判定を行う必要がある場合に使用します。例えば、注文データベースにデータを毎晩ロードするワークフローを作成するとします。さらに、このデータのサブセットを定期的に、ローカルの注文データベースを 10 回更新するごとに、本社側のデータベースにロードする必要があるとします。ローカルデータベースを更新するセッションと本社のデータベースを更新するセッションを個別に作成します。

以下の図にワークフローを示します。

図 7. ワークフロー変数を使用するワークフロー



ここで、ユーザー定義の変数を使用して、本社にある注文データベースを更新するセッションをいつ実行するかを決定します。

ユーザー定義ワークフロー変数を設定するには、以下の手順を実行します。

1. ワークフローが実行された回数を表す持続型ワークフロー変数`$$WorkflowCount`を作成します。
2. [Start] タスクと両方のセッションをワークフローに追加します。
3. ローカル注文データベースを更新するセッションの後に [ディシジョン] タスクを配置します。
ワークフローの実行回数が 10 で均等に割りきれられるかどうかを確認する判定条件を設定します。このためには、MOD 関数を使用します。
4. [割り当て] タスクを作成して、`$$WorkflowCount` 変数を 1 だけ増分します。
5. 判定条件が真に評価された場合、本社側のデータベースを更新するセッションに [ディシジョン] タスクをリンクします。判定条件が偽に評価された場合、[ディシジョン] タスクを [割り当て] タスクにリンクします。

条件を使用してワークフロー変数を設定する場合、ワークフローを実行するたびに、ローカルデータベースを更新するセッションが実行されます。本社側のデータベースを更新するセッションは、ワークフローを 10 回実行するごとに実行されます。

ワークフロー変数の初期値とカレント値

理論的には、Integration Service は、ワークフローの実行中にワークフロー変数に関して以下の 2 つの異なる値を保持します。

- ワークフロー変数の初期値
- ワークフロー変数のカレント値

初期値とは、ワークフローの開始時に変数を持つ値です。初期値には、変数のパラメータファイルに定義されている値、ワークフローの前の実行によってリポジトリに保存されている値、変数に対してユーザーが定義した初期値、変数のデータ型に基づくデフォルト値などが使用されます。

Integration Service は、以下の順に変数の初期値を探します。

1. パラメータファイルに定義されている値
2. リポジトリに保存されている値（変数が持続型の場合）
3. ユーザーによって指定されたデフォルト値
4. データ型に基づくデフォルト値

例えば、ワークフロー内にワークフロー変数を作成してデフォルト値を入力した一方で、その変数の値をパラメータファイルに定義していないとします。このワークフローを初めて実行する場合、Integration Service は変数の初期値をユーザー定義のデフォルト値として評価します。

変数を *持続型* として宣言した場合、Integration Service は、ワークフローの実行が終了したときにこの変数の値をリポジトリに保存します。次回同じワークフローを実行するときは、リポジトリに保存されている値が変数の初期値として評価されます。

変数が *非持続型* の場合、Integration Service は変数の値を保存しません。次回同じワークフローを実行するときは、ユーザーによって指定されたデフォルト値が変数の初期値として評価されます。

リポジトリに保存されている値をワークフローの実行前に上書きしたい場合は、パラメータファイルにその変数の値を定義する必要があります。パラメータファイルにワークフロー変数の値が定義されている場合、Integration Service は、リポジトリに保存されている値や、変数に設定されている初期値ではなく、この値を使用します。

カレント値とは、ワークフローの進行にともなって変数に保持される値です。ワークフローの開始時の変数のカレント値は初期値と等しくなります。変数の値を更新する [Assignment] タスクを作成している場合は、ワークフローの進行に合わせて変数の値が変化します。

変数が持続型の場合、Integration Service は、ワークフローの実行が正常終了したときに変数のカレント値をリポジトリに保存します。ワークフローが完了できなかった場合、Integration Service は、リポジトリ内の変数の値を更新しません。

Integration Service は、ワークフロー変数ごとに、リポジトリに保存された値をワークフローログに記録します。

データタイプのデフォルト値

他の方法によって変数の初期値を特定できない場合、統合サービスは変数のデータタイプに基づいてそのデフォルト値を使用します。

以下の表に、ユーザー定義ワークフロー変数の各データタイプのデフォルト値を示します。

データタイプ	Workflow Manager のデフォルト値
Date/Time	1/1/1753 00:00:00.000000000 A.D.
Double	0
Integer	0
Nstring	空の文字列

ユーザー定義ワークフロー変数の作成

ワークフローのワークフロー変数は、ワークフロープロパティで作成することができます。

ワークフロー変数を作成するには：

1. Workflow Designer で、新しいワークフローを作成するか、または既存のワークフローを編集します。
2. [変数] タブを選択します。
3. [追加] をクリックします。
4. 以下の表に示す情報を入力し、[OK] をクリックします。

フィールド	説明
名前	変数名。正しい形式は <code>\$\$VariableName</code> です。ワークフロー変数名は大文字/小文字を区別しません。 ユーザー定義ワークフロー変数に単独のドル記号（\$）を使用してはなりません。単独のドル記号（\$）は、組み込みワークフロー変数用に予約されています。
データ型	変数のデータ型。以下のデータ型の中から選択できます。 <ul style="list-style-type: none">- 日付/時刻- ダブル- Integer- Nstring
パシステント	変数が持続型かどうか。ワークフローの実行の間で変数の値を保持したい場合は、このオプションを有効にします。

フィールド	説明
デフォルト値	<p>変数のデフォルト値。パラメータファイルで変数に値を設定しておらず、リボジトリに値が格納されていない場合、セッション中、Integration Service は変数にこの値を使用します。</p> <p>Date/Time 型の変数で使用する形式は、以下のとおりです。</p> <ul style="list-style-type: none"> - MM/DD/RR - MM/DD/YYYY - MM/DD/RR HH24:MI - MM/DD/YYYY HH24:MI - MM/DD/RR HH24:MI:SS - YYYY/MM/DD HH24:MI:SS - MM/DD/RR HH24:MI:SS.MS - MM/DD/YYYY HH24:MI:SS.MS - MM/DD/RR HH24:MI:SS.US - MM/DD/YYYY HH24:MI:SS.US - MM/DD/RR HH24:MI:SS.NS - MM/DD/YYYY HH24:MI:SS.NS <p>以下の区切り記号を使用することができます。ダッシュ (-)、スラッシュ (/)、バックスラッシュ (\)、コロン (:)、ピリオド (.)、およびスペース。</p> <p>Integration Service では、余分な空白は無視されます。年または時間の「HH12」形式には、1 桁または 3 桁の値を使用できません。</p> <p>Nstring 型の変数は、最大で 600 文字の長さを持つことができます。</p>
デフォルト値=NULL	<p>変数のデフォルト値が NULL かどうか。デフォルト値が NULL の場合、このオプションを有効にします。</p>
説明	<p>変数に関する説明。</p>

5. 新しいワークフロー変数のデフォルト値を検査するには、[検査] をクリックします。
6. [適用] をクリックして新しいワークフロー変数を保存します。
7. [OK] をクリックします。

ワークレット変数の使用

ワークレット変数はワークフロー変数に似ています。ワークレットは、タスクと同じ定義済み変数のセットを持ちます。また、ユーザー定義のワークレット変数を作成することもできます。ユーザー定義のワークフロー変数と同様に、ユーザー定義のワークレット変数には、持続型と非持続型があります。

永続ワークレット変数

ユーザー定義のワークレット変数には、持続型と非持続型があります。持続型ワークレット変数を作成するには、変数を作成するときに [パーシステント] を選択します。永続ワークレット変数を作成した場合、ワークレット変数は、次回 Integration Service によって親ワークフローでワークレットが実行される時も保持されます。

たとえば、持続型変数を持つワークレットがあるとします。ワークレットを2回実行するために、ワークフロー内にワークレットのインスタンスが2つあるとします。ワークレットの1つ目のインスタンスは Worklet1、2つ目のインスタンスは Worklet2 です。

ワークフローの実行時、持続型ワークレット変数は Worklet1 での値を保持してそれが Worklet2 における初期値となります。Integration Service で Worklet2 が実行された後、Integration Service によって永続変数の値がリポジトリに保持され、次回このワークフローを実行するときにこの値が使用されます。

ワークレット変数は、同じワークフローが実行されるときだけその値を持続します。ワークレット変数は、別のワークフローにあるワークレットのインスタンスが使用された場合はその値を持続しません。

初期値のオーバーライド

それぞれのワークレットインスタンスごとに、ワークフロー変数をワークレット変数に割り当てることで、ワークレット変数の初期値を上書きすることができます。

ワークレット変数の初期値を上書きするには：

1. Workflow Designer の作業領域内でワークレットのインスタンスをダブルクリックします。
2. [変数] タブで、プリワークレット変数の割り当てで [追加] ボタンをクリックします。
3. [ユーザー定義ワークレット変数] フィールドの右端のボタンをクリックし、ワークレット変数を選択します。
4. [適用] をクリックします。

このワークレットインスタンス内のワークレット変数には、初期値として選択されたワークフロー変数が含まれます。

ワークレット変数の使用に関するルールおよびガイドライン

ワークレット変数を使用する際には、以下のルールおよびガイドラインを使用します。

- ワークレットで親ワークフロー変数を使用することはできません。
- ワークフロー変数の値をワークレット変数に割り当てて初期化することができます。
- 親ワークフローでユーザー定義のワークレット変数を使用することはできません。
- ワークフローの他のタスクに定義済みの変数を使えるのと同様に、定義済みのワークレット変数を親ワークフローで使用することはできます。

ワークレットでの変数値の割り当て

ワークレットの実行前または実行後に変数の値を更新できます。この機能により、1つのワークレットから同じワークフロー内の別のワークレット、または親ワークレットに情報を渡すことができます。たとえば、同じカウンタをインクリメントする必要がある2つのワークレットを含むワークフローがあるとします。1つ目のワークレットでカウンタをインクリメントし、更新後のカウンタ値を2つ目のワークレットに渡し、そのカウンタを再び2つ目のワークレットでインクリメントすることができます。

また、情報をワークレットから再利用不可能なセッションに、あるいは再利用不可能なセッションからワークレットに渡すこともできます。この場合、ワークレットとセッションが共に同じワークフローあるいは親ワークレットであることが条件です。再利用可能なワークレットと再利用不可能なワークレット内の変数を割り当てることができます。

ワークレットの実行前に変数を割り当てるか、またはワークレットの実行後に変数を割り当てるかによって、異なる変数値を更新できます。ワークレットの実行前または実行後に次のタイプの変数を更新できます。

- **ワークレット実行前の変数割り当て。**ワークレットの実行前にユーザー定義のワークレット変数を更新できます。これらのワークレット変数に、親ワークフロー変数またはワークレット変数の値、あるいはワークフローまたは親ワークレットの他のタスクからのマッピング変数の値を割り当てることができます。

ワークレットの親からの値でワークレット変数を更新できます。したがって、ワークレットがワークフロー内の別のワークレットである場合、そのワークフロー変数の値ではなく、親ワークレット変数からの値を割り当てることができます。

- **ワークレット実行後の変数割り当て。**ワークレットの完了後、親ワークフローまたはワークレット変数を更新できます。これらの変数に、ユーザー定義のワークレット変数の値を割り当てることができます。

ワークレットを編集するときには「変数」タブで変数を割り当てます。

ワークレット間での変数値の受け渡し

1つのワークレットから同じワークフロー内の後続ワークレットまたは親ワークレットに値を渡すために、ワークレット内の変数値を割り当てます。例えば、ワークフローに `wklt_CreateCustList` と `wklt_UpdateCustOrders` の2つのワークレットが含まれているとします。Worklet `wklt_UpdateCustOrders` は、`wklt_CreateCustList` で更新されたワークレット変数の値を使用する必要があります。

以下の図にワークフローを示します。



`wklt_CreateCustList` から `wklt_UpdateCustOrders` にワークレット変数を渡すには、以下の手順を実行します。

1. 例えば、ワークレット変数 `$$URLString1` を使用するようにワークレット `wklt_CreateCustList` を設定します。
2. 例えば、ワークレット変数 `$$URLString2` を使用するようにワークレット `wklt_UpdateCustOrders` を設定します。
3. 例えば、ワークフロー変数 `$$PassURLString` を使用するようにワークフローを設定します。
4. ワークレットの完了後、ワークレット変数 `$$URLString1` の値を、ワークフロー変数 `$$PassURLString` に割り当てるようにワークレット `wklt_CreateCustList` を設定します。
5. ワークレットを開始する前に、ワークフロー変数 `$$PassURLString` の値を、ワークレット変数 `$$URLString2` に割り当てるようにワークレット `wklt_UpdateCustOrders` を設定します。

変数割り当ての設定

ワークレットを編集するときには「変数」タブで変数を割り当てます。ワークレットの実行前または実行後に、次のタイプの変数に値を割り当てます。

- **ワークレット実行前の変数割り当て。**ユーザー定義のワークレット変数を、親ワークフローまたはワークレット変数の値、あるいはワークフロー内の他のタスクからのマッピング変数またはこのワークレットの前に実行している親ワークレットの値で更新します。
- **ワークレット実行後の変数割り当て。**親ワークフローおよびワークレット変数を、ユーザー定義ワークレット変数の値で更新します。

ワークレット内の変数を割り当てるには：

1. 変数を割り当てるワークレットを編集します。

2. [変数] タブをクリックします。
3. 変数割り当てタイプを選択します。
 - **ワークレット実行前の変数割り当て。**ワークレットの実行前にユーザー定義のワークレット変数に値を割り当てます。
 - **ワークレット実行後の変数割り当て。**ワークレットの完了後に親ワークフローおよびワークレット変数に値を割り当てます。
4. 変数割り当てフィールドの [編集] ボタンをクリックします。
5. プリワークレット変数またはポストワークレット変数の割り当て領域で、追加ボタンをクリックして、変数割り当て文を追加します。
6. [ユーザー定義ワークレット変数] フィールドおよび [親ワークフロー/ワークレット変数] フィールドのオープンボタンをクリックして、値を読み込む変数または値を割り当てる変数を選択します。プリワークレット変数の割り当てでは、これらのフィールドにパラメータと変数名を入力する場合があります。Workflow Manager では、パラメータと変数名は検査されません。

Workflow Manager は、割り当て文の右側から割り当て文の左側の変数に値を割り当てます。したがって、変数割り当て文が“\$\$SiteURL_WFVar=\$\$SiteURL_WkltVar,”の場合、Workflow Manager は\$SiteURL_WkltVar の値を\$\$SiteURL_WFVar に割り当てます。
7. 5 から 6 までの手順を繰り返して、さらに変数割り当て文を追加します。

変数割り当て文を削除するには、割り当て文のフィールドの 1 つをクリックして、[カット] ボタンをクリックします。
8. [OK] をクリックします。

第 15 章

セッションのパラメータおよび変数

この章では、以下の項目について説明します。

- [セッションパラメータに関する作業, 238 ページ](#)
- [セッションのマッピングパラメータおよび変数, 244 ページ](#)
- [セッションでのパラメータ値および変数値の割り当て, 245 ページ](#)

セッションパラメータに関する作業

セッションパラメータは、データベース接続やソースファイルとターゲットファイルなど、セッションの実行ごとに変更できる値を示します。

セッションパラメータには、ユーザー定義パラメータまたは組み込みパラメータのいずれかがあります。セッションまたはワークフローのプロパティでユーザー定義のセッションパラメータを使用し、パラメータファイルでその値を定義します。セッションを実行する際は、Integration Service ではパラメータファイル内のパラメータをセッション内のパラメータと一致させます。セッションプロパティ値にパラメータファイルの値を使用します。パラメータファイルでは、フォルダ名とセッション名は大文字/小文字が区別されます。

例えば、ログファイルにセッションログを書き込むことができます。セッションプロパティで、セッションログファイル名として\$PMSessionLogFile を使用し、パラメータファイルで\$PMSessionLogFile を TestRun.txt に設定します。セッションを実行する際に、Integration Service によって、TestRun.txt と名付けられたセッションログが作成されます。

ユーザー定義のセッションパラメータにはデフォルト値がないので、パラメータファイルでこれらを定義する必要があります。Integration Service によってユーザー定義のセッションパラメータの値が検索できない場合、セッションが失敗するか、空の文字列をデフォルト値とするか、または実行時にパラメータの展開に失敗します。

pmcmd を使用してセッションを開始すると、各種パラメータファイルを使用して 1 つのセッションを実行できます。*pmcmd* を使用して設定したパラメータファイルは、セッションまたはワークフロープロパティでパラメータファイルを上書きします。

組み込みセッションパラメータを使用して、フォルダ名、サービス名、セッション実行統計などのランタイム情報を取得します。組み込みセッションパラメータは、セッション実行後のシェルコマンド、SQL コマンド、およびメールメッセージの中に使用できます。それらのセッションパラメータは、Designer および Workflow Manager でセッションパラメータを受け入れる入力フィールドに使用できます。Integration Service は組み込みセッションパラメータの値を設定します。組み込みセッションパラメータ値は、パラメータファイル内に

は定義することができません。これらのパラメータは、Integration Service でセッションの実行時に展開されます。

以下の表に、ユーザー定義のセッションパラメータを示します。

パラメータタイプ	命名規則	説明
セッションログファイル	\$PMSessionLogFile	セッション実行間のセッションログの名前を定義します。
パーティションの数	\$DynamicPartitionCount	セッションのパーティション数を定義します。
ソースファイル	\$InputFileName	ソースファイル名を定義します。 適切なプレフィックスを使用してパラメータ名を定義します。
ルックアップファイル	\$LookupFileName	ルックアップファイル名を定義します。 適切なプレフィックスを使用してパラメータ名を定義します。
ターゲットファイル	\$OutputFileNames	ターゲットファイル名を定義します。 適切なプレフィックスを使用してパラメータ名を定義します。
拒否ファイル	\$BadFileName	拒否ファイル名を定義します。 適切なプレフィックスを使用してパラメータ名を定義します。
データベース接続	\$DBConnectionName	ソース、ターゲット、ルックアップ、またはストア ドプロシージャのリレーショナルデータベース接続 を定義します。 適切なプレフィックスを使用してパラメータに名前 を付けます。
外部ローダー接続	\$LoaderConnectionName	外部ローダー接続を定義します。 適切なプレフィックスを使用してパラメータ名を定義 します。
FTP 接続	\$FTPConnectionName	FTP 接続を定義します。 適切なプレフィックスを使用してパラメータ名を定義 します。
キュー接続	\$QueueConnectionName	メッセージキューのデータベース接続を定義します。 適切なプレフィックスを使用してパラメータ名を定義 します。

パラメータタイプ	命名規則	説明
ソースまたはターゲットアプリケーション接続	<i>\$AppConnectionName</i>	ソースアプリケーションおよびターゲットアプリケーションへの接続を定義します。 適切なプレフィックスを使用してパラメータ名を定義します。
一般のセッションパラメータ	<i>\$ParamName</i>	他のセッションプロパティを定義します。たとえば、このパラメータを使用して、テーブルオーナー名、テーブル名の接頭語、FTP ファイルまたはディレクトリ名、ルックアップキャッシュファイル名接頭語、またはメールアドレスを定義できます。このパラメータを使用して、ソース、ルックアップ、ターゲット、およびリジェクトファイル名を定義できますが、セッションログファイル名やデータベース接続は定義できません。 適切なプレフィックスを使用してパラメータ名を定義します。

以下の表に、組み込みセッションパラメータを示します。

パラメータタイプ	命名規則	説明
フォルダ名	<i>\$PMFolderName</i>	フォルダ名を返します。
Integration Service 名	<i>\$PMIntegrationServiceName</i>	Integration Service 名を返します。
マッピング名	<i>\$PMMappingName</i>	マッピング名を返します。
Repository Service 名	<i>\$PMRepositoryServiceName</i>	リポジトリサービス名を返します。
リポジトリユーザ名	<i>\$PMRepositoryUserName</i>	リポジトリユーザ名を返します。
セッション名	<i>\$PMSessionName</i>	セッション名を返します。
セッション実行モード	<i>\$PMSessionRunMode</i>	セッション実行モード（ノーマルまたはリカバリ）を返します。
ソース側の影響を受けた行の数	<i>\$PM.SourceQualifierName@numAffectedRows</i>	Integration Service で名前付き Source Qualifier から正常に読み込まれた行の数が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ソース側の適用された行の数	<i>\$PM.SourceQualifierName@numAppliedRows</i>	Integration Service で名前付き Source Qualifier から正常に読み込まれた行の数が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。

パラメータタイプ	命名規則	説明
ソース側の拒否された行の数	<code>\$PM.SourceQualifierName@numRejectedRows</code>	Integration Service で名前付き Source Qualifier からの読み込み中に削除された行の数が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ソーステーブル名	<code>\$PM.SourceName@TableName</code>	名前付きソースインスタンスのテーブル名が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ターゲット側の影響を受けた行の数	<code>\$PM.TargetName@numAffectedRows</code>	Integration Service で名前付きターゲットインスタンスに対する特定操作の影響が及んだ行の数が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ターゲット側の適用された行の数	<code>\$PM.TargetName@numAppliedRows</code>	Integration Service で名前付きターゲットインスタンスに正常に適用された行の数が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ターゲット側の拒否された行の数	<code>\$PM.TargetName@numRejectedRows</code>	Integration Service で名前付きターゲットインスタンスへの書き込み中に拒否された行の数が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ターゲットテーブル名	<code>\$PM.TargetName@TableName</code>	名前付きターゲットインスタンスのテーブル名が返されます。 適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。
ワークフロー名	<code>\$PM.WorkflowName</code>	ワークフロー名を返します。
ワークフロー実行 ID。	<code>\$PM.WorkflowRunId</code>	ワークフロー実行 ID を返します。
ワークフロー実行インスタンス名	<code>\$PM.WorkflowRunInstanceName</code>	ワークフロー実行インスタンス名を返します。

適切なプレフィックスおよびサフィックスを使用してパラメータ名を定義します。例えば、ソースインスタンスが「Customers」という名前の場合、ソーステーブル名を示すパラメータは `$PM.Customers@TableName` となります。ソース修飾子が「SQ_Customers」という名前の場合、影響を受けた行のソース数を示すパラメータは `$PMSQ_Customers@numAffectedRows` となります。

セッションログ名の変更

ファイルにログイベントを書き込むようにセッションを設定できます。セッションプロパティで、セッションログファイルディレクトリは、デフォルトでサービスプロセス変数\$PMSessionLogDirとなります。セッションログファイル名は、デフォルトでは、\$PMSessionLogFileとなります。

パラメータファイルで、\$PMSessionLogFile を TestRun.txt に設定します。Administrator ツールで、\$PMSessionLogDir を \\server\infa_shared\SessLogs と定義します。Integration Service によってセッションが実行される場合、\\server\infa_shared\SessLogs ディレクトリに TestRun.txt という名前のセッションログファイルが作成されます。

ターゲットファイルおよびディレクトリの変更

セッションのターゲットファイルとディレクトリを変更するには、セッションプロパティでターゲットファイルパラメータを使用します。[出力ファイル名] フィールドには、ディレクトリ名とファイル名が含まれているパスを入力できます。[出力ファイル名] フィールドにディレクトリを含める場合、出力ファイルディレクトリをクリアする必要があります。Integration Service によって、出力ファイルディレクトリと出力ファイル名が連結され、ターゲットファイルの場所が特定されます。

たとえば、セッションでファイルパラメータを使用して内部 Web ログと外部 Web ログを読み取ります。内部 Web ログセッションの結果と外部 Web ログセッションの結果を別個の場所に書き込むとします。

セッションのプロパティで、ターゲットファイルに\$OutputFileName と名前を付け、[出力ファイルディレクトリ] フィールドをクリアします。パラメータファイルでは、内部 Web ログセッションのターゲットファイルを作成するのに、\$OutputFileName を「E:\internal_weblogs\November_int.txt」に設定します。そのセッションが完了したら、今度は外部 Web ログセッション用に\$OutputFileName を「F:\external_weblogs\November_ex.txt」に設定します。

ターゲットごとに異なるパラメータファイルを作成し *pmcmd* を使用すれば、特定のパラメータファイルでセッションを開始できます。このパラメータファイルは、セッションプロパティのパラメータファイル名を上書きします。

ファイルでのソースパラメータの変更

パラメータファイルでセッションプロパティに複数のパラメータを定義し、セッションでこれらのパラメータの 1 つを使用できます。セッションプロパティでパラメータ名を変更し、別のパラメータ値を指定して再度セッションを実行できます。

たとえば、パラメータファイル内に\$InputFile_Products という名前のセッションパラメータを作成します。パラメータ値を"products.txt"に設定します。同じパラメータファイル内に、\$InputFile_Items と呼ばれる別のパラメータを作成します。パラメータ値を items.txt に設定します。

セッションプロパティでソースファイル名を\$InputFile_Products に設定すると、Integration Service で products.txt が読み込まれます。ソースファイル名を\$InputFile_Items に変更すると、Integration Service で items.txt が読み込まれます。

接続パラメータの変更

ソース、ターゲット、ルックアップテーブル、またはストアドプロシージャを個別に指定してセッションを再実行するには、接続パラメータを使用します。接続パラメータは、任意のセッションのセッションプロパティで作成できます。パラメータで接続を参照できます。すべての接続セッションパラメータの名前には必ず、先頭に接頭語を指定し、その後に任意の英数字や下線文字を付けます。

たとえば、2 つのリレーショナルソースから読み込むセッションを実行するとします。あるソースにはデータベース接続「Marketing」を使用してアクセスし、他のソースには「Sales」という接続でアクセスするとします。セッションプロパティで、「\$DBConnection_Source」という名前のソースデータベース接続パラメータ

を作成します。パラメータファイル内で\$DBConnection_Source を Marketing と定義したあと、セッションを実行します。次のセッション実行のパラメータファイルで\$DBConnection_Source を Sales に設定します。

ソースまたはターゲットの接続を上書きする接続パラメータを使用すれば、パラメータファイル内の接続属性を上書きすることができます。ソースインスタンスまたはターゲットインスタンスに対して非リレーショナル接続パラメータを使用しているならば、接続属性の上書きが可能です。パラメータファイル内に接続を定義すると、接続属性を定義する特定のユーザー定義セッションパラメータが Integration Service によって検索されます。たとえば、\$FTPConnectionMyFTPConn という FTP 接続パラメータを作成し、それをパラメータファイル内に定義するとします。パラメータファイル内の次のパラメータが、Integration Service によって検索されます。

- \$Param_FTPConnectionMyFTPConn_Remote_Filename
- \$Param_FTPConnectionMyFTPConn_Is_Staged
- \$Param_FTPConnectionMyFTPConn_Is_Transfer_Mode_ASCII

これらのパラメータの値がいずれも定義されていない場合、接続オブジェクト用に定義された値が Integration Service で使用されます。

上書き可能な接続属性は、次のテンプレートファイル内にリストされています。

<PowerCenter Installation Directory>/server/bin/ConnectionParam.prm

ランタイム情報の取得

フォルダ名、Integration Service 名、ソーステーブル名、ターゲットテーブル名などのランタイム情報を取得するには、組み込みセッションパラメータを使用します。組み込みセッションパラメータは、セッション実行後のシェルコマンド、SQL コマンド、およびメールメッセージの中に使用できます。それらのセッションパラメータは、Designer および Workflow Manager でセッションパラメータを受け入れる入力フィールドに使用できます。

たとえば、セッション"s_UpdateCustInfo"の完了後に、セッション後の電子メールを送信するとします。この電子メールには、ソース修飾子"SQ_Customers"およびターゲット"T_CustInfo"に関するセッション実行統計を含めます。電子メールの本文に、次のテキストを入力します。

```
Statistics for session $PMSessionName
Integration service: $PMIntegrationServiceName
Source number of affected rows: $PMSQ_Customers@numAffectedRows
Source number of dropped rows: $PMSQ_Customers@numRejectedRows
Target number of affected rows: $PMT_CustInfo@numAffectedRows
Target number of applied rows: $PMT_CustInfo@numAppliedRows
Target number of rejected rows: $PMT_CustInfo@numRejectedRows
```

電子メール変数を使用して、セッション名、Integration Service 名、ロードされた行数、および拒否された行数を取得することもできます。

ファイルパラメータとデータベース接続パラメータの作成に関するルールおよびガイドライン

セッションファイルパラメータとデータベース接続パラメータは、各種ファイルとデータベースに対してセッションを実行する柔軟性を提供します。

ファイルパラメータを作成する場合には、次の規則およびガイドラインに従ってください。

- ノードのリソースとしてパラメータファイルを定義する場合、パラメータファイルにアクセスできるノードで Integration Service がセッションを実行していることを確認します。ノードのリソースを定義し、リソースをチェックするように Integration Service を設定し、そのリソースを要求するようにセッションを編集します。

- ファイルパラメータを作成する場合、英数字と下線文字を使用します。たとえば、ソースファイルパラメータ名を指定するには、`$InputFile_Data` などの `$InputFileName` を使用します。
- 特定タイプのセッションファイルパラメータにはすべて、明確な名前を付ける必要があります。たとえば、ソースファイルパラメータを 2 つ作成する場合、`$SourceFileAccts` および `$SourceFilePrices` という名前を付けます。
- ファイルでパラメータを定義すると、Integration Service のいずれかのローカルディレクトリを参照することができます。
- ファイルの場所を定義するパラメータを使用します。ファイルの場所を定義するエントリをセッションプロパティでクリアします。パラメータファイルに、ファイルのフルパスを入力します。
- セッションを実行するたびにパラメータファイルのパラメータ値を変更できます。または、複数のパラメータファイルを作成することもできます。複数のパラメータファイルを使用する場合は、`-paramfile` オプションまたは `-localparamfile` オプションを指定した `pmcmd Startworkflow` コマンドを使用して、使用するパラメータファイルを指定します。

データベース接続パラメータを作成する場合には、次の規則およびガイドラインに従ってください。

- リレーショナルソース、ターゲット、ルックアップ、およびストアドプロシージャの各接続を変更できます。
- パラメータを定義すると、リポジトリ内の任意のデータベース接続を参照できます。
- セッションで、複数の接続に同じ `$DBConnection` パラメータを使用します。

セッションのマッピングパラメータおよび変数

セッションのプロパティのマッピングパラメータを使用して、所定のマッピング属性を変更できます。たとえば、トランスフォーメーションオーバーライド内のマッピングパラメータを使用して、ソース修飾子トランスフォーメーション内のフィルタまたはユーザー定義の結合を上書きできます。

セッション内でマッピング変数を使用している場合は、セッションを編集してリポジトリに格納されている任意の変数値をクリアすることができます。変数値を取り消した場合、次回セッションを実行する際に Integration Service によりパラメータファイル内の値が使用されます。セッションでパラメータファイルが使用されない場合、Integration Service ではセッション実行前の変数の割り当てで割り当てられた値が使用されます。割り当てられた値がない場合、Integration Service ではマッピングに定義されている初期値が使用されます。

リポジトリに格納されているマッピング変数の値を表示または削除するには：

1. Workflow Manager のナビゲータウィンドウで、[セッション] タスクを右クリックし、[パーシステント値の表示] を選択します。
変数の名前と値を確認できます。
2. [リセット] をクリックして既存の変数値を削除します。
3. [OK] をクリックします。

セッションでのパラメータ値および変数値の割り当て

再利用不可能なセッションの実行前または実行後に、特定のパラメータ値および変数の値を更新できます。この機能により、1つのセッションから同じワークフローまたはワークレット内の別のセッションへと情報を渡すことができます。たとえば、同じカウンタをインクリメントする必要がある2つのセッションを含むワークフローがあるとします。1つ目のセッションでカウンタをインクリメントし、更新後のカウンタ値を2つ目のセッションに渡し、再度2つ目のセッションでカウンタをインクリメントすることができます。または、同じWebサイトにアクセスするセッションを含むワークレットがあるとします。最初のセッションがWebサイトからセッションIDを取得するように、次にそのセッションID値を後続のセッションに渡すように設定できます。

また、セッションとワークレットが同じワークフローまたは親ワークレットにある限り、セッションからワークレットに、あるいはワークレットからセッションに情報を渡すこともできます。

注: 再利用可能なセッションでパラメータおよび変数を割り当てることはできません。

更新可能なパラメータおよび変数のタイプは、セッション実行前に割り当てるか、またはセッション実行後に割り当てるかによって異なります。セッション実行前または実行後に更新できるのは、次のパラメータおよび変数タイプです。

- **セッション実行前の変数割り当て** セッション実行前に、マッピングパラメータ、マッピング変数、およびセッションパラメータを更新できます。これらのパラメータおよび変数には、親ワークフローまたはワークレット内のワークフローまたはワークレット変数の値を割り当てることができます。したがって、セッションがワークフロー内のワークレット内にある場合は、ワークフロー変数ではなくワークレット変数から値を割り当てることができます。
セッション実行前の変数割り当てでは、ワークレット変数を更新することはできません。
- **セッション実行後の成功時の変数割り当て** 親ワークフローまたは親ワークレット内のワークフロー変数やワークレット変数は、セッションが正常に完了した後に更新できます。これらの変数には、マッピングパラメータおよびマッピング変数の値を割り当てることができます。
- **セッション実行後の失敗時の変数割り当て** セッションが失敗した場合に、親ワークフローまたは親ワークレット内のワークフロー変数やワークレット変数を更新できます。これらの変数には、マッピングパラメータおよびマッピング変数の値を割り当てることができます。

パラメータおよび変数はセッションプロパティの「コンポーネント」タブで割り当てます。

セッション間でのパラメータ値および変数値の受け渡し

1つのセッションから同じワークフローやワークレット内の以降のセッションに値を渡すために、セッション内でパラメータ値および変数値を割り当てることができます。例えば、ワークフローに、2つのセッション `s_NewCustomers` と `s_MergeCustomers` が含まれているとします。Session `s_MergeCustomers` は、`s_NewCustomers` で更新されたマッピング変数の値を使用する必要があります。

以下の図にワークフローを示します。



`s_NewCustomers` から `s_MergeCustomers` にマッピング変数値を渡すには、次の手順を実行します。

1. 例えば、マッピング変数 `$$Count1` を使用するように、セッション `s_NewCustomers` に関連付けられたマッピングを設定します。

- 例えば、マッピング変数`$$Count2`を使用するように、セッション `s_MergeCustomers` に関連付けられたマッピングを設定します。
- 例えば、ユーザー定義のワークフロー変数`$$PassCountValue`を使用するように、ワークフローを設定します。
- セッションが正常に完了した後にマッピング変数`$$Count1`の値をワークフロー変数`$$PassCountValue`に割り当てるように、セッション `s_NewCustomers` を設定します。
- セッションの開始前にワークフロー変数`$$PassCountValue`の値をマッピング変数`$$Count2`に割り当てるように、セッション `s_MergeCustomers` を設定します。

変数割り当ての設定

ワークレットを編集するときには「変数」タブで変数を割り当てます。ワークレットの実行前または実行後に、次のタイプの変数に値を割り当てます。

- **ワークレット実行前の変数割り当て。**ユーザー定義のワークレット変数を、親ワークフローまたはワークレット変数の値、あるいはワークフロー内の他のタスクからのマッピング変数またはこのワークレットの前に実行している親ワークレットの値で更新します。
- **ワークレット実行後の変数割り当て。**親ワークフローおよびワークレット変数を、ユーザー定義ワークレット変数の値で更新します。

ワークレット内の変数を割り当てるには：

- 変数を割り当てるワークレットを編集します。
- 「変数」タブをクリックします。
- 変数割り当てタイプを選択します。
 - **ワークレット実行前の変数割り当て。**ワークレットの実行前にユーザー定義のワークレット変数に値を割り当てます。
 - **ワークレット実行後の変数割り当て。**ワークレットの完了後に親ワークフローおよびワークレット変数に値を割り当てます。
- 変数割り当てフィールドの「編集」ボタンをクリックします。
- プリワークレット変数またはポストワークレット変数の割り当て領域で、追加ボタンをクリックして、変数割り当て文を追加します。
- 「ユーザー定義ワークレット変数」フィールドおよび「親ワークフロー/ワークレット変数」フィールドのオープンボタンをクリックして、値を読み込む変数または値を割り当てる変数を選択します。プリワークレット変数の割り当てでは、これらのフィールドにパラメータと変数名を入力する場合があります。Workflow Manager では、パラメータと変数名は検査されません。
Workflow Manager は、割り当て文の右側から割り当て文の左側の変数に値を割り当てます。したがって、変数割り当て文が“`$$SiteURL_WFVar=$$SiteURL_WkltVar,`”の場合、Workflow Manager は`$$SiteURL_WkltVar`の値を`$$SiteURL_WFVar`に割り当てます。
- 5 から 6 までの手順を繰り返して、さらに変数割り当て文を追加します。
変数割り当て文を削除するには、割り当て文のフィールドの 1 つをクリックして、「カット」ボタンをクリックします。
- 「OK」をクリックします。

第 16 章

パラメータファイル

この章では、以下の項目について説明します。

- [パラメータファイルの概要, 247 ページ](#)
- [パラメータおよび変数のタイプ, 248 ページ](#)
- [パラメータおよび変数を使用する場所, 249 ページ](#)
- [パラメータファイル内の接続属性のオーバーライド, 257 ページ](#)
- [パラメータファイル構造体, 258 ページ](#)
- [パラメータファイル名および場所の設定, 261 ページ](#)
- [パラメータファイルの例, 263 ページ](#)
- [パラメータファイルの作成に関するガイドライン, 264 ページ](#)
- [パラメータおよびパラメータファイルのトラブルシューティング, 265 ページ](#)
- [パラメータおよびパラメータファイルに関するヒント, 266 ページ](#)

パラメータファイルの概要

パラメータファイルは、パラメータと変数、およびそれらに関連する値のリストです。これらの値はサービス、サービスプロセス、ワークフロー、ワークレット、またはセッションのプロパティを定義します。Integration Service により、これらの値は、パラメータファイルを使用するワークフローまたはセッションを実行する際に適用されます。

パラメータファイルを使用すると、セッションまたはワークフローを実行するたびに、パラメータや変数の値を柔軟に変更することができます。複数のサービス、サービスプロセス、ワークフロー、ワークレット、およびセッションに関する情報を 1 つのパラメータファイルに含めることができます。また、複数のパラメータファイルを作成し、セッションまたはワークフローを実行するたびに異なるファイルを使用することもできます。Integration Service により、ワークフローまたはセッションの開始時にパラメータファイルが読み込まれ、ファイルに定義されたパラメータおよび変数の開始値が決定されます。パラメータファイルはワードパッドやメモ帳などのテキストエディタを使って作成できます。

パラメータファイルを使用する際は、以下に注意してください。

- **パラメータおよび変数のタイプ。**1 つのパラメータファイル内に、さまざまなタイプのパラメータと変数を定義できます。サービス変数、サービスプロセス変数、ワークフロー変数とワークレット変数、セッションパラメータ、およびマッピングパラメータとマッピング変数などが含まれます。
- **パラメータファイルに設定可能なプロパティ。**パラメータおよび変数を使用して、Designer および Workflow Manager 内に多数のプロパティを定義します。例えば、リレーショナルターゲットインスタンスの更新の上書きとしてセッションパラメータを入力し、パラメータファイルの UPDATE 文にこのパラメータを設定できます。パラメータは、Integration Service でセッションの実行時に展開されます。

- **パラメータファイル構造体。**パラメータ名または変数名、およびパラメータ値または変数値を「*name=value*」形式で1行に入力することで、パラメータファイルのパラメータまたは変数に値を割り当てます。パラメータおよび変数のグループの先頭には、パラメータまたは変数を適用するサービス、サービスプロセス、ワークフロー、ワークレット、またはセッションを識別する見出しを付ける必要があります。
- **パラメータファイルの場所。**ワークフローまたはセッションで使用するパラメータファイルを指定します。ワークフローまたはセッションのプロパティ、あるいは *pmcmd* コマンドラインに、パラメータファイルの名前とディレクトリを入力できます。

パラメータおよび変数のタイプ

パラメータファイルには、さまざまなタイプのパラメータおよび変数を含めることができます。パラメータファイルを使用するセッションまたはワークフローの実行時には、Integration Service によってパラメータファイルが読み込まれ、ファイル内に定義されたパラメータおよび変数が展開されます。

パラメータファイルには、以下のタイプのパラメータおよび変数を定義できます。

- **サービス変数。**電子メールアドレス、ログファイル数、エラーしきい値など、Integration Service の一般的なプロパティを定義します。\$PMSuccessEmailUser、\$PMSessionLogCount、\$PMSessionErrorThreshold などは、サービス変数の例です。パラメータファイルで定義したサービス変数値は、Administrator ツールで設定された値をオーバーライドします。
- **サービスプロセス変数。**各 Integration Service プロセスの Integration Service ファイルのディレクトリを定義します。\$PMRootDir、\$PMSessionLogDir、\$PMBadFileDir などは、サービスプロセス変数の例です。パラメータファイルで定義したサービスプロセス変数値は、Administrator ツールで設定された値をオーバーライドします。Integration Service がオペレーティングシステムのプロファイルを使用する場合、オペレーティングシステムのプロファイルで指定されているオペレーティングシステムのユーザーは、サービスプロセス変数に定義されたディレクトリへのアクセス権を持つ必要があります。
- **ワークフロー変数。**ワークフロー内のタスク条件とレコード情報を評価します。たとえば、[ディシジョン] タスク内でワークフロー変数を使用して、直前のタスクが正常に実行されたかどうかを判定できます。ワークフローで、\$TaskName.PrevTaskStatus は、組み込みワークフロー変数で、\$\$VariableName は、ユーザー定義のワークフロー変数です。
- **ワークレット変数。**ワークレット内のタスク条件とレコード情報を評価します。親ワークフロー内では定義済みのワークレット変数を使用できます。ただし、ワークレット内では親ワークフローのワークフロー変数を使用することはできません。ワークレットで、\$TaskName.PrevTaskStatus は、定義済みのワークレット変数で、\$\$VariableName は、ユーザー定義のワークレット変数です。
- **セッションパラメータ。**データベース接続またはファイル名など、セッション間で変更できる値を定義します。\$PMSessionLogFile および\$ParamName は、ユーザー定義のセッションパラメータです。
- **マッピングパラメータ。**消費税率など、セッション全体で一定の値を定義します。マッピングまたはマプレットで宣言される\$\$ParameterName は、ユーザー定義のマッピングパラメータです。
- **マッピング変数。**セッション中に変更できる値を定義します。Integration Service は、セッションが正常に実行されるたびに、そのセッションの最後にマッピング変数の値をリポジトリに保存し、その値を次回セッションを実行するときに使用します。マッピングまたはマプレットで宣言される\$\$VariableName は、マッピング変数です。

パラメータファイルには、以下のタイプの変数を定義することはできません。

- **\$Source 接続変数および\$Target 接続変数。**リレーショナルソース、リレーショナルターゲット、ルックアップテーブル、またはストアドプロシージャ用のデータベースの場所を定義します。

- **電子メール変数。** ロードされる行の数、セッションの完了時間、読み込み統計および書き込み統計など、電子メールメッセージ内のセッション情報を定義します。
- **ローカル変数。** アグリゲータ、式、およびランクトランスフォーメーション内の変数ポート内のデータを一時的に格納します。
- **組み込み変数。** Integration Service 名やシステム日付などのランタイム情報またはシステム情報を返す変数。
- **トランザクション制御変数。** データベース行の処理中にトランザクションをコミットまたはロールバックする条件を定義します。
- **ABAP プログラム変数。** SAP 構造体、SAP 構造体のフィールド、または ABAP プログラムの値を表します。

パラメータおよび変数を使用する場所

パラメータおよび変数を使用して、Designer や Workflow Manager のプロパティに値を割り当てたり、サービスおよびサービスプロセスのプロパティを上書きしたりできます。例えば、パラメータを使用して、ルックアップキャッシュファイル名の接頭語、または FTP 接続のデフォルトリモートディレクトリを指定できます。

プロパティが SQL 文または SQL コマンドの場合、SQL 文または SQL コマンド内にパラメータおよび変数を使用するか、あるいはプロパティの入力フィールドにパラメータまたは変数を入力して、そのパラメータまたは変数をパラメータファイル内の文またはコマンド全体に設定することができます。

例えば、リレーショナルターゲットの上書きでパラメータまたは変数を使用するとします。リレーショナルターゲットの上書きの UPDATE ステートメント内にパラメータまたは変数を入力して、パラメータファイルの適切な見出しの下にパラメータまたは変数を定義できます。または、パラメータファイルで UPDATE ステートメントを定義するには、以下の手順を実行します。

1. Designer で、ターゲットインスタンスを編集し、[更新の上書き] フィールドにセッションパラメータ \$ParamMyOverride を入力し、マッピングを保存します。
2. Workflow Manager で、パラメータファイルを使用するようにワークフローまたはセッションを設定します。
3. パラメータファイルの適切な見出しの下にある SQL UPDATE 文に \$ParamMyOverride を設定します。

また、パラメータファイルを使用して、Administrator ツールで定義したサービスおよびサービスプロセスのプロパティをオーバーライドすることもできます。例えば、セッションログディレクトリ \$PMSessionLogDir を上書きすることができます。これには、パラメータファイルを使用するようにワークフローまたはセッションを設定し、パラメータファイルで新しいファイルパスに \$PMSessionLogDir を設定します。

パラメータおよび変数は、以下の PowerCenter オブジェクトに指定できます。

- **ソース。** ソースに関連する入力フィールドにパラメータおよび変数を使用できます。
- **ターゲット。** ターゲットに関連する入力フィールドにパラメータおよび変数を使用できます。
- **トランスフォーメーション。** トランスフォーメーションに関連する入力フィールドでパラメータおよび変数を使用できます。
- **タスク:** Workflow Manager のタスクに関連する入力フィールドでパラメータおよび変数を使用できます。
- **セッション。** [Session] タスクに関連する入力フィールドでパラメータおよび変数を使用できます。
- **ワークフロー。** ワークフローに関連する入力フィールドでパラメータおよび変数を使用できます。
- **接続。** 接続オブジェクトに関連する入力フィールドでパラメータおよび変数を使用できます。

以下の表に、パラメータおよび変数を指定できるソースに関連する入力フィールドを一覧表示します。

ソースタイプ	フィールド	有効なパラメータおよび変数のタイプ
リレーショナル	ソーステーブル名	ワークフロー変数、ワークレット変数、セッションパラメータ、マッピングパラメータ、およびマッピング変数。 Workflow Manager のセッションプロパティ（[マッピング] タブ）でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。
PeopleSoft	SetID、 有効日、 ツリー名、 制御値設定、 抽出日	すべて。
TIBCO	TIB/Adapter SDK リポジトリ URL	サービスおよびサービスプロセス変数。
Web サービス	エンドポイント URL	マッピングパラメータとマッピング変数 Workflow Manager のセッションプロパティ（[マッピング] タブ）でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。

以下の表に、パラメータおよび変数を指定できるターゲットに関連する入力フィールドを一覧表示します。

ターゲットタイプ	フィールド	有効なパラメータおよび変数のタイプ
リレーショナル	更新オーバーライド セッション実行前および実行後の SQL コマンド	すべて。 Workflow Manager のセッションプロパティ（[マッピング] タブ）でパラメータおよび変数をオーバーライドする際に、これらのフィールドにパラメータおよび変数を指定することができます。
リレーショナル	ターゲットテーブル名	ワークフロー変数、ワークレット変数、セッションパラメータ、マッピングパラメータ、およびマッピング変数。 Workflow Manager のセッションプロパティ（[マッピング] タブ）でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。
XML	キャッシュディレクトリ	サービスおよびサービスプロセス変数。 Workflow Manager のセッションプロパティ（[マッピング] タブ）でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。

ターゲットタイプ	フィールド	有効なパラメータおよび変数のタイプ
TIBCO	TIB/Adapter SDK リポジトリ URL	サービスおよびサービスプロセス変数。
Web サービス	エンドポイント URL	マッピングパラメータとマッピング変数。 Workflow Manager のセッションプロパティ ([マッピング] タブ) でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。

以下の表に、パラメータおよび変数を指定できるトランスフォーメーションに関連する入力フィールドを一覧表示します。

トランスフォーメーションのタイプ	フィールド	有効なパラメータおよび変数のタイプ
式エディタを使用可能なトランスフォーメーション	トランスフォーメーション式	マッピングパラメータとマッピング変数。
アグリゲータ、ジョイナ、ルックアップ、ランク、XML ジェネレータ	キャッシュディレクトリ	サービスおよびサービスプロセス変数。 Workflow Manager のセッションプロパティ ([マッピング] タブ) でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。
アグリゲータ、ジョイナ、ルックアップ、ランク、ソータ	キャッシュサイズ	マッピングパラメータ。 Workflow Manager のセッションプロパティ ([マッピング] タブ) でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。
カスタム、エクスターナルプロシージャ、HTTP、XML パーサー	実行時位置	サービスおよびサービスプロセス変数。 Workflow Manager のセッションプロパティ ([マッピング] タブ) でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。
データマスキング	シード	マッピングパラメータとマッピング変数。
エクスターナルプロシージャ	初期化プロパティ	サービスおよびサービスプロセス変数。
HTTP	ベース URL	マッピングパラメータとマッピング変数。
ルックアップ	SQL オーバーライド キャッシュファイル名のプレフィックス	すべて。 Workflow Manager のセッションプロパティ ([マッピング] タブ) でパラメータおよび変数をオーバーライドする際に、このフィールドにパラメータおよび変数を指定することができます。

トランスフォーマー セッションのタイプ	フィールド	有効なパラメータおよび変数のタイプ
ルックアップ	接続情報	セッションパラメータ <code>\$DBConnectionName</code> および <code>\$AppConnectionName</code> 、接続変数 <code>\$Source</code> および <code>\$Target</code> 、マッピングパラメー タおよび変数。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。
ソータ	デフォルトワークディレクトリ	サービスおよびサービスプロセス変数。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。
Source Qualifier（リ レーショナルソース）	SQL クエリ ユーザー定義結合 ソースフィルタ条件。 セッション実行前および実行後 の SQL コマンド	すべて。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。
SQL	スクリプトファイル名	マッピングパラメータとマッピング変数。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。
ストアードプロシージャ	呼び出しテキスト（未接続のストアードプロシージャ）	すべて。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。
ストアードプロシージャ	接続情報	セッションパラメータ <code>\$DBConnectionName</code> 、 接続変数 <code>\$Source</code> および <code>\$Target</code> 。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。
Web サービスコンシューマ	エンドポイント URL	マッピングパラメータとマッピング変数。 Workflow Manager のセッションプロパティ （[マッピング] タブ）でパラメータおよび変 数をオーバーライドする際に、このフィール ドにパラメータおよび変数を指定すること ができます。

以下の表に、パラメータおよび変数を指定できる Workflow Manager のタスクに関連する入力フィールドを一覧表示します。

タスクタイプ	フィールド	有効なパラメータおよび変数のタイプ
割り当てタスク	Assignment (ユーザ定義変数および式)	ワークフロー変数とワークレット変数
[Command] タスク	Command (名前およびコマンド)	サービス、サービスプロセス、ワークフロー、およびワークレット変数
[Command] タスク	セッション実行前および実行後のシェルコマンド	すべて
[Decision] タスク	Decision 名 (評価する条件)	ワークフロー変数とワークレット変数
[Email] タスク	メールユーザ名、件名、および本文	サービス、サービスプロセス、ワークフロー、およびワークレット変数
[Event Wait] タスク	トリガファイル名 (定義済みイベント)	サービス、サービスプロセス、ワークフロー、およびワークレット変数
リンク	リンク条件	サービス、サービスプロセス、ワークフロー、およびワークレット変数
Session	the table (ページ 253) を参照してください。	
[Timer] タスク	時刻指定: ワークフロー変数から開始時刻を決定する	ワークフロー変数とワークレット変数

以下の表に、パラメータおよび変数を指定できるセッションに関連する入力フィールドを一覧表示します。

タブ	フィールド	有効なパラメータおよび変数のタイプ
[プロパティ] タブ。	セッションログファイル名	組み込みセッションパラメータ \$PMSessionLogFile。
[プロパティ] タブ。	セッションログファイルディレクトリ	サービスおよびサービスプロセス変数。
[プロパティ] タブ。	パラメータファイル名	ワークフローとワークレット変数。
[プロパティ] タブ。	\$Source 接続値および\$Target 接続値	セッションパラメータ \$DBConnectionName および \$AppConnectionName、接続変数 \$Source および \$Target。
[プロパティ] タブ。	プッシュダウン最適化セッションプロパティ	マッピングパラメータ \$\$PushdownConfig。
[設定オブジェクト] タブ	セッションログカウント数	サービス変数 \$PMSessionLogCount。
[設定オブジェクト] タブ	セッションエラーしきい値	サービス変数 \$PMSessionErrorThreshold。
[設定オブジェクト] タブ	リレーショナルエラーログのテーブル名の接頭語	すべて。

タブ	フィールド	有効なパラメータおよび変数のタイプ
[設定オブジェクト] タブ	エラーログファイル名およびエラーログファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[設定オブジェクト] タブ	動的パーティション化のパーティション数	組み込みセッションパラメータ \$DynamicPartitionCount。
[マッピング] タブ	マッピングで設定するプロパティを上書きするトランスフォーメーションプロパティ	プロパティによって変わります。詳細については、 the table (ページ 250) を参照してください。
[マッピング] タブ	リレーショナル接続値	セッションパラメータ\$DBConnectionName、接続変数\$Source および\$Target。
[マッピング] タブ	キュー接続値	セッションパラメータ \$QueueConnectionName。 この接続タイプの場合、パラメータファイル内の接続属性はオーバーライドできます。
[マッピング] タブ	FTP 接続値	セッションパラメータ\$FTPConnectionName。 この接続タイプの場合、パラメータファイル内の接続属性はオーバーライドできます。
[マッピング] タブ	アプリケーション接続値	セッションパラメータ\$AppConnectionName。 この接続タイプの場合、パラメータファイル内の接続属性はオーバーライドできます。
[マッピング] タブ	外部ローダー接続値	セッションパラメータ \$LoaderConnectionName。 この接続タイプの場合、パラメータファイル内の接続属性はオーバーライドできます。
[マッピング] タブ	FTP リモートファイル名	すべて。
[マッピング] タブ	Lookup ソースファイル名およびLookup ソースファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	セッション実行前および実行後の SQL コマンド（ソースおよびターゲット）	すべて。
[マッピング] タブ	ファイルソースおよびファイルターゲット用のコードページ	ワークフロー変数、ワークレット変数、セッションパラメータ\$ParamName。
[マッピング] タブ	ソース入力ファイル名およびソース入力ファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	ソース入力ファイルコマンド	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	リレーショナルソースのテーブル所有者名	すべて。

タブ	フィールド	有効なパラメータおよび変数のタイプ
[マッピング] タブ	ターゲットマージファイル名およびターゲットマージファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	ターゲットマージコマンド	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	ターゲットヘッダおよびターゲットフッタのコマンド	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	ターゲット出力ファイル名およびターゲット出力ファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	ターゲット拒否ファイル名およびターゲット拒否ファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	ターゲットテーブル名接頭語	すべて。
[マッピング] タブ	Teradata FastExport 一時ファイル	サービスおよびサービスプロセス変数。
[マッピング] タブ	Teradata 外部ローダ用の制御ファイルの上書き	すべて。
[マッピング] タブ	WebSphere MQ、JMS、SAP ALE IDoc、TIBCO、webMethods、Web Service Provider ソース用のリカバリキャッシュディレクトリ	サービスおよびサービスプロセス変数。
[マッピング] タブ	継続サブスクリプション名	セッションパラメータ \$Param 名。
[マッピング] タブ	MQ ソース修飾子フィルタ条件	すべて。
[マッピング] タブ	SAP ステージングファイル名および SAP ステージングファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[マッピング] タブ	SAP ソースファイルディレクトリ	サービス変数、サービスプロセス変数、ワークフロー変数、ワークレット変数、セッションパラメータ。
[コンポーネント] タブ	セッション後の Email (ユーザー名、件名、および本文)	すべて。
[コンポーネント] タブ	セッション実行後に発信されるメールの添付ファイル名	すべて。

以下の表に、パラメータおよび変数を指定できるワークフローに関連する入力フィールドを一覧表示します。

タブ	フィールド	有効なパラメータおよび変数のタイプ
[プロパティ] タブ。	ワークフローログファイル名およびワークフローログファイルディレクトリ	サービス、サービスプロセス、ワークフロー、およびワークレット変数。
[プロパティ] タブ。	ワークフローログカウント数	サービス変数\$PMWorkflowLogCount。
[全般] タブ	サスペンド時の Email (ユーザー名、件名、および本文)	サービス、サービスプロセス、ワークフロー、およびワークレット変数。

以下の表に、パラメータおよび変数を指定できる Workflow Manager の接続オブジェクトに関連する入力フィールドを一覧表示します。

接続タイプ	フィールド	有効なパラメータおよび変数のタイプ
リレーショナル	データベースユーザー名、パスワード	セッションパラメータ\$ParamName。 pmpasswd コマンドラインプログラムを使用し、CRYPT_DATA 暗号化タイプを指定して、パラメータファイル内のパスワードを暗号化します。
リレーショナル: ソース、ターゲット、ルックアップ、ストアードプロシージャ	接続およびトランザクション環境 SQL	すべて。
FTP	ホストマシンのユーザー名、パスワード	セッションパラメータ\$ParamName。 pmpasswd コマンドラインプログラムを使用し、CRYPT_DATA 暗号化タイプを指定して、パラメータファイル内のパスワードを暗号化します。
FTP	デフォルトリモートディレクトリ	すべて。
アプリケーション	アプリケーションユーザー名、パスワード	セッションパラメータ\$ParamName。 pmpasswd コマンドラインプログラムを使用し、CRYPT_DATA 暗号化タイプを指定して、パラメータファイル内のパスワードを暗号化します。
アプリケーション: Web サービスコンシューマ	エンドポイント URL	セッションパラメータ\$ParamName、マッピングパラメータと変数。
アプリケーション: HTTP	ベース URL	セッションパラメータ\$ParamName。

接続タイプ	フィールド	有効なパラメータおよび変数のタイプ
アプリケーション： JMS	JMS 接続先	セッションパラメータ <i>\$ParamName</i> 。
ローダ	データベースユーザー名、パスワード	セッションパラメータ <i>\$ParamName</i> 。 pmpasswd コマンドラインプログラムを使用し、CRYPT_DATA 暗号化タイプを指定して、パラメータファイル内のパスワードを暗号化します。

パラメータファイル内の接続属性のオーバーライド

セッションパラメータを使用してソースまたはターゲットの接続を定義する場合、パラメータファイル内の接続属性を上書きすることができます。*\$FTPConnectionName*、*\$QueueConnectionName*、*\$LoaderConnectionName*、または *\$AppConnectionName* セッションパラメータを使用します。

パラメータファイル内に接続を定義すると、接続属性を定義する特定のユーザー定義セッションパラメータが Integration Service によって検索されます。たとえば、*\$QueueConnectionMyMQ* というメッセージキュー接続パラメータを作成して、パラメータファイル内の “[s_MySession]” セクションで定義するとします。Integration Service は、パラメータファイル内のこのセクションを検索し、「rows per message (メッセージあたりの行数)」パラメータ *\$Param_QueueConnectionMyMQ_Rows_Per_Message* を探します。

PowerCenter をインストールすると、FTP、キュー、ローダ、およびアプリケーション接続用の上書き可能な接続属性をリストした *ConnectionParam.prm* という名前のテンプレートファイルが、インストールプログラムによって作成されます。*ConnectionParam.prm* ファイルは、次のディレクトリに配置されています。

```
<PowerCenter Installation Directory>/server/bin
```

パラメータファイル内に接続を定義する際には、適切な接続タイプのテンプレートをコピーしてパラメータファイル内に貼り付けてください。その後、パラメータ値を指定します。

たとえば、パラメータファイル内に FTP 接続の接続属性を上書きするには、次の手順を実行します。

1. セッションまたはワークフローをパラメータファイルで実行されるように設定します。
2. セッションプロパティの [マッピング] タブで、[接続] ノード内のソースインスタンスまたはターゲットインスタンスを選択します。
3. [値] フィールドで [開く] ボタンをクリックして、セッションパラメータを使用するように接続を設定します。たとえば、FTP 接続には *\$FTPConnectionMyFTPConn* を使用します。
4. テキストエディタで *ConnectionParam.prm* テンプレートファイルを開き、属性を上書きする対象の接続タイプのセクションまでスクロールします。たとえば、FTP 接続の場合、「接続タイプ：FTP」セクションに位置付けます。

```
Connection Type : FTP
```

```
-----
```

```
...
```

```
Template
```

```
=====
```

- ```
$FTPConnection<VariableName>=
$Param_FTPConnection<VariableName>_Remote_Filename=
$Param_FTPConnection<VariableName>_Is_Staged=
$Param_FTPConnection<VariableName>_Is_Transfer_Mode_ASCII=
```
5. 上書きする接続属性のテンプレートテキストをコピーします。例えば、[リモートファイル名] 属性および [Is Staged] 属性を上書きするには、次の行をコピーします。
- ```
$FTPConnection<VariableName>=
$Param_FTPConnection<VariableName>_Remote_Filename=
$Param_FTPConnection<VariableName>_Is_Staged=
```
6. テキストをパラメータファイルに貼り付けます。<VariableName>を接続名で置き換えて、パラメータ値を指定します。以下に例を示します。
- ```
[MyFolder.WF:wf_MyWorkflow.ST:s_MySession]

$FTPConnectionMyFTPConn=FTP_Conn1
$Param_FTPConnectionMyFTPConn_Remote_Filename=ftp_src.txt
$Param_FTPConnectionMyFTPConn_Is_Staged=YES
```
- 注:** 等号符号の前または後にあるスペースまたは引用符は、パラメータ名または値の一部として Integration Service に解釈されます。
- 属性の値を定義しない場合、接続オブジェクト用に定義された値が Integration Service で使用されます。

## パラメータファイル構造体

パラメータファイルには、パラメータと変数、および割り当てられた値のリストが含まれます。パラメータおよび変数はパラメータファイルの各セクションでグループ化されます。各セクションの先頭には、パラメータまたは変数を定義する、Integration Service プロセス、ワークフロー、ワークレット、またはセッションを識別する見出しが付けられます。見出しのすぐ下でパラメータや変数を定義します。このとき各パラメータや変数を新しい行に入力します。各セクション内では、パラメータおよび変数を任意の順序で記述できます。

パラメータまたは変数の定義は、「名前=値」の形式で入力します。例えば、以下の行では、サービス変数 \$PMSuccessEmailUser とセッションパラメータ \$ParamTgtOverride に値が割り当てられます。

```
$PMSuccessEmailUser=rsmith@mail.com
$ParamTgtOverride=UPDATE T_SALES SET DATE_SHIPPED = :TU.DATE_SHIPPED, TOTAL_SALES = :TU.TOTAL_SALES
WHERE :TU.EMP_NAME = EMP_NAME and EMP_NAME = 'MIKE SMITH'
```

Integration Service では、行の先頭から最初の等号までの間のすべての文字はパラメータ名、最初の等号から行末までの間のすべての文字はパラメータ値として解釈されます。したがって、パラメータ名と等号の間にスペースを入力した場合、そのスペースは Integration Service によってパラメータ名の一部として解釈されます。行に複数の等号が含まれている場合、最初の等号の後にあるすべての等号は Integration Service によってパラメータ値の一部として解釈されます。

**警告:** Integration Service では、パラメータファイルを使用してワークフローを実行する際に、フォルダ、ワークフロー、およびセッションの名前を修飾するためにピリオド (.) が使用されます。フォルダ名にピリオド (.) が含まれる場合、Integration Service により名前が適切に修飾されず、ワークフローが失敗します。

## パラメータファイルのセクション

パラメータおよび変数は、パラメータファイルのどのセクションにも定義できます。ワークフロー、ワークレットまたはセッションセクションにサービスまたはサービスプロセス変数を定義する場合、その変数はタスクを実行するサービスプロセスに適用されます。同様に、セッションセクションにワークフロー変数を定義する場合、そのワークフロー変数の値はセッションを実行するときのみ適用されます。

以下の表に、パラメータファイルの各セクションを定義するパラメータファイルの見出しと、各セクションに定義するパラメータおよび変数の範囲を示します。

| 見出し                                                                                                                              | 範囲                                                                         |
|----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| [グローバル]                                                                                                                          | すべての Integration Services、Integration Service プロセス、ワークフロー、ワークレット、およびセッション。 |
| [サービス:サービス名]                                                                                                                     | このサービスを実行する名前付きの Integration Service、ワークフロー、ワークレット、およびセッション。               |
| [サービス:サービス名.ND:ノード名]                                                                                                             | このサービスプロセスを実行する名前付きの Integration Service プロセス、ワークフロー、ワークレット、およびセッション。      |
| [フォルダ名.WF:ワークフロー名]                                                                                                               | 名前付きワークフローおよびワークフロー内のすべてのセッション。                                            |
| [フォルダ名.WF:ワークフロー名.WT:ワークレット名]                                                                                                    | 名前付きワークレットおよびワークレット内のすべてのセッション。                                            |
| [フォルダ名.WF:ワークフロー名.WT:ワークレット名.WT:ワークレット名...]                                                                                      | ネストしたワークレットおよびネストしたワークレット内のすべてのセッション。                                      |
| [フォルダ名.WF:ワークフロー名.ST:セッション名]<br>-または-<br>[フォルダ名.WF:ワークフロー名.WT:ワークレット名.ST:セッション名]<br>-または-<br>[フォルダ名.セッション名]<br>-または-<br>[セッション名] | 名前付きセッション。                                                                 |

各見出しはパラメータファイルで1つのみ作成します。パラメータファイルに同じ見出しを何度も指定した場合、Integration Service では最初の見出しの下の子セクションの情報が使用され、後続の同一の見出しの下の子セクションの情報は無視されます。例えば、パラメータファイルに以下のような同一の見出しが含まれていたとします。

```
[HET_TGTS.WF:wf_TCOMMIT1]
$$platform=windows
...
[HET_TGTS.WF:wf_TCOMMIT1]
$$platform=unix
$DBConnection_ora=0ra2
```

ワークフロー wf\_TCOMMIT では、マッピングパラメータ `$$platform` の値は「unix」ではなく「windows」で、セッションパラメータ `$DBConnection_ora` は定義されていません。

パラメータファイル内の複数のセクションで同じパラメータや変数を定義すると、最も小さいスコープのパラメータまたは変数が、より大きいスコープのパラメータまたは変数よりも優先されます。たとえば、パラメータファイルには以下のセクションが含まれます。

```
[HET_TGTS.WF:wf_TGTS_ASC_ORDR]
$DBConnection_ora=Ora2
[HET_TGTS.WF:wf_TGTS_ASC_ORDR.ST:s_TGTS_ASC_ORDR]
$DBConnection_ora=Ora3
```

セッション `s_TGTS_ASC_ORDR` では、セッションパラメータ `$DBConnection_ora` の値は「Ora3」です。ワークフロー内の他のすべてのセッションでは、「Ora2」となります。

## コメント

パラメータファイルにはコメントを含めることができます。Integration Service では、有効でない見出しの行は無視されます。また、等号文字 (=) は含まれません。以下にパラメータファイルのコメントの例を示します。

```

Created 10/11/06 by JSmith.
*** Update the parameters below this line when you run this workflow on Integration Service Int_01. ***
; This is a valid comment because this line contains no equals sign.
```

## NULL 値

パラメータファイルのパラメータおよび変数には NULL 値を割り当てることができます。パラメータおよび変数に NULL 値を割り当てると、Integration Service は、パラメータのタイプまたは変数のタイプに応じて、次の場所から値を取得します。

- **サービスおよびサービスプロセス変数。** Integration Service により Administrator ツール内で使用される値です。
- **ワークフローとワークレット変数。** Integration Service により、リポジトリに保存されている値（変数が永続の場合）、ユーザー固有のデフォルト値、またはデータタイプデフォルト値が使用されます。
- **セッションパラメータ。** セッションパラメータにはデフォルト値はありません。Integration Service がセッションパラメータの値を検出できない場合、セッションが失敗するか、空の文字列をデフォルト値とするか、実行時にパラメータの展開に失敗する可能性があります。たとえば、Integration Service はセッションパラメータ `$DBConnectionName` が定義されていないセッションに失敗します。
- **マッピングパラメータとマッピング変数。** Integration Service により、リポジトリに保存された値（マッピング変数のみ）、設定された初期値、またはデータタイプデフォルト値が使用されます。

NULL 値を割り当てするには、マッピングパラメータまたは変数の値を "<null>" に設定するか、値を空欄のままにします。たとえば、以下の行では、サービスプロセス変数 `$PMBadFileDir` と `$PMCacheDir` に NULL 値を割り当てます。

```
$PMBadFileDir=<null>
$PMCacheDir=
```

## サンプルのパラメータファイル

以下のテキストは、1 つの Integration Service のサービス変数および 4 つのワークフローのパラメータを含むパラメータファイルからの抜粋です。

```

File created by RSmith 11/12/2005

[Service:IntSvs_01]
$PMSuccessEmailUser=pcadmin@mail.com
$PMFailureEmailUser=pcadmin@mail.com
[HET_TGTS.WF:wf_TCOMMIT_INST_ALIAS]
$$platform=unix
```

```
[HET_TGTS.WF:wf_TGTS_ASC_ORDR.ST:s_TGTS_ASC_ORDR]
$$platform=unix
$$DBConnection_oracle=0ra2
$ParamAscOrderOverride=UPDATE T_SALES SET CUST_NAME = :TU.CUST_NAME, DATE_SHIPPED = :TU.DATE_SHIPPED,
TOTAL_SALES = :TU.TOTAL_SALES WHERE CUST_ID = :TU.CUST_ID
[ORDERS.WF:wf_PARAM_FILE.WT:WL_PARAM_Lvl_1]
$$DT_WL_lvl_1=02/01/2005 01:05:11
$$Double_WL_lvl_1=2.2
[ORDERS.WF:wf_PARAM_FILE.WT:WL_PARAM_Lvl_1.WT:NWL_PARAM_Lvl_2]
$$DT_WL_lvl_2=03/01/2005 01:01:01
$$Int_WL_lvl_2=3
$$String_WL_lvl_2=ccccc
```

## パラメータファイル名および場所の設定

ワークフローまたはセッションを開始する場合は、パラメータファイルを使用して、パラメータ値および変数値を Integration Service に渡します。ワークフローまたはセッションのプロパティ、あるいは *pmcmd* コマンドラインに、パラメータファイル名とディレクトリを指定できます。Integration Service でオペレーティングシステムプロファイルを使用している場合、オペレーティングシステムプロファイルに指定されているオペレーティングシステムのユーザー名は、パラメータファイルへのアクセス権を持っている必要があります。

*pmcmd* で使用するパラメータファイルは、ワークフローまたはセッションプロパティでパラメータファイルを上書きします。*pmcmd* コマンドラインにパラメータファイル名を入力しない場合、Integration Service では、ワークフローおよびワークフロー内のすべてのセッションのワークフロープロパティで指定したパラメータファイルが使用されます。*pmcmd* コマンドラインまたはワークフロープロパティでパラメータファイル名を入力しない場合、Integration Service では、セッションプロパティで指定したパラメータファイルが使用されます。

## ワークフローまたはセッションでのパラメータファイルの使用

ワークフローまたはセッションのプロパティで、パラメータファイルの名前とディレクトリを指定できます。ワークフローが並列実行されるように設定されている場合、各ワークフロー実行インスタンスのセッションに別々のパラメータファイルを使用するときは、セッションパラメータファイル名としてワークフロー変数またはワークレット変数を指定します。

ワークフローまたはセッションにパラメータファイルを指定しても、Integration Service が指定したパラメータファイルを特定できない場合、ワークフローまたはセッションは失敗します。

### ワークフロープロパティ内のパラメータファイルの入力

ワークフローのプロパティでパラメータファイルを指定するには：

1. Workflow Manager でワークフローを開きます。
2. [ワークフロー] - [編集] をクリックします。  
[ワークフローの編集] ダイアログボックスが表示されます。
3. [プロパティ] タブをクリックします。
4. [Parameter Filename (パラメータファイル名)] フィールドにパラメータファイルの場所と名前を入力します。  
ダイレクトパスまたはサービスプロセス変数のいずれかを入力できます。Integration Service オペレーティングシステムに有効な区切り文字を使用します。PowerCenter 環境を高可用性に設定した場合は、サービスプロセス変数をパスに含めます。
5. [OK] をクリックします。

## セッションプロパティ内のパラメータファイルの入力

セッションのプロパティでパラメータファイルを指定するには：

1. Workflow Manager でセッションを開きます。  
[Edit Tasks (タスクの編集)] ダイアログボックスが表示されます。
2. [プロパティ] タブをクリックし、[General Options] 設定を開きます。
3. [Parameter Filename (パラメータファイル名)] フィールドにパラメータファイルの場所と名前を入力します。

ダイレクトパスまたはサービスプロセス変数を入力できます。Integration Service オペレーティングシステムに有効な区切り文字を使用します。PowerCenter 環境を高可用性に設定した場合は、サービスプロセス変数をパスに含めます。

また、ユーザー定義のワークフロー変数またはワークレット変数も入力することができます。ワークフローパラメータファイル内にセッションパラメータファイル名を定義するためのワークフロー変数またはワークレット変数を入力します。

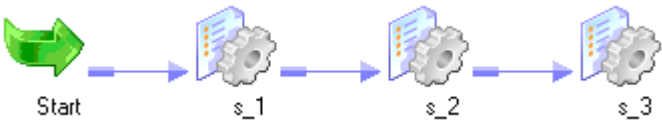
4. [OK] をクリックします。

## 変数を使用したセッションパラメータファイルの指定

セッションパラメータファイル名として、ワークフロー変数またはワークレット変数を入力できます。ワークフローが並列実行されるように設定されている場合、各ワークフロー実行インスタンスのセッション用に別々のパラメータ値および変数値を定義するときは、ワークフロー変数またはワークレット変数をセッションパラメータファイル名として入力してください。

ワークフロー内のセッション用にワークフローパラメータファイルおよびセッションパラメータファイルを定義する場合、Integration Service ではワークフローパラメータファイルを使用するため、セッションパラメータファイルが無視されます。変数を使用してセッションパラメータファイル名を定義する場合は、セッションパラメータファイル名を定義し、ワークフローパラメータファイルに\$PMMergeSessParamFile=TRUE を設定する必要があります。\$PMMergeSessParamFile プロパティを設定すると、Integration Service でセッションパラメータファイルおよびワークフローパラメータファイルの両方が読み込まれます。

例えば、3 つのセッションを含む 2 つのコンカレントインスタンスを実行するようにワークフローを設定したとします。



1 番目と 2 番目のワークフローインスタンスでは、セッションに次のセッションパラメータファイルを使用します。

| セッション | セッションパラメータファイル名<br>(1 番目のワークフロー実行インスタンス) | セッションパラメータファイル名<br>(2 番目のワークフロー実行インスタンス) |
|-------|------------------------------------------|------------------------------------------|
| s_1   | s_1Inst1.txt                             | s_1Inst2.txt                             |
| s_2   | s_2Inst1.txt                             | s_2Inst2.txt                             |
| s_3   | s_3Inst1.txt                             | s_3Inst2.txt                             |

ワークフロー変数を作成して、セッションパラメータファイル名を格納できます。例えば、ユーザー定義のワークフロー変数`$$s_1ParamFileName`、`$$s_2ParamFileName`、および`$$s_3ParamFileName`を作成することができます。各セッションのセッションプロパティで、パラメータファイル名をワークフロー変数に設定してください。

| セッション | セッションパラメータファイル名<br>セッションプロパティ内 |
|-------|--------------------------------|
| s_1   | \$\$s_1ParamFileName           |
| s_2   | \$\$s_2ParamFileName           |
| s_3   | \$\$s_3ParamFileName           |

各ワークフローインスタンスのワークフローパラメータファイルで、各ワークフロー変数を正しいセッションパラメータファイル名に設定し、`$PMMergeSessParamFile=TRUE`を設定します。

セッションパラメータファイル名として変数を使用した場合、セッションパラメータファイルとワークフローパラメータファイルの両方に同じパラメータまたは変数が定義されているときは、Integration Service では次のルールに従ってパラメータ値および変数値が設定されます。

- ワークフローパラメータファイルにもセッションパラメータファイルにも同一セッションにパラメータまたは変数が定義されている場合、Integration Service で使用されるのはワークフローパラメータファイル内の値です。
- セッションパラメータファイルのセッションセクション、およびワークフローパラメータファイルのワークフローセクションの両方にパラメータまたは変数が定義されている場合、Integration Service で使用されるのはセッションパラメータファイル内の値です。

## pmcmd でのパラメータファイルの使用

`pmcmd startworkflow` または `starttask` コマンドで、パラメータファイルを使用します。これらのコマンドを使用して、ワークフローまたはセッションを開始するときに使用するパラメータファイルを指定できます。

`pmcmd -paramfile` オプションは、セッションまたはワークフローを実行するときに使用するパラメータファイルを定義します。`-localparamfile` オプションは、Integration Service マシン上のパラメータファイルにアクセスできない場合に参照できるローカルマシン上のパラメータファイルを定義します。

次のコマンドは、`myfile.txt` というパラメータファイルを使用してワークフロー A を開始します。

```
pmcmd startworkflow -uv USERNAME -pv PASSWORD -s SALES:6258 -f east -w wSalesAvg -paramfile '$\PMRootDir/myfile.txt' workflowA
```

次のコマンドは、`myfile.txt` というパラメータファイルを使用してタスク A を開始します。

```
pmcmd starttask -uv USERNAME -pv PASSWORD -s SALES:6258 -f east -w wSalesAvg -paramfile '$\PMRootDir/myfile.txt' taskA
```

## パラメータファイルの例

このセクションの例では、パラメータファイルの使用が必要なセッションについて説明します。別の状態および時間値を指定してセッションを再実行することもできます。この例では、設定可能なパラメータおよび変数を示し、パラメータ値と変数値をリストし、さらにセッションを再実行する際の変更について説明します。



プロダクションフォルダに s\_MonthlyCalculations というセッションがあります。このセッションではセッションパラメータを使用してソースファイルおよびターゲットデータベースに接続し、セッションログファイルに書き込みます。セッションに失敗した場合、Integration Service によって電子メールメッセージが pccadmin@mail.com に送信されます。このセッションでは、文字列マッピングパラメータ \$\$State と日付/時刻マッピング変数 \$\$Time を使用しています。\$\$State には「MA」と設定します。リポジトリ内の \$\$Time の初期値は「9/30/2005:04:00」ですが、この値を「10/1/2005 05:04:11」と上書きします。

次の表に、s\_MonthlyCalculations セッションのパラメータおよび変数を示します。

| パラメータまたは変数のタイプ          | パラメータまたは変数の名前         | 定義                           |
|-------------------------|-----------------------|------------------------------|
| サービス変数                  | \$PMFailureEmailUser  | pccadmin@mail.com            |
| 文字列マッピングパラメータ           | \$\$State             | MA                           |
| 日付/時刻マッピング変数            | \$\$Time              | 05/10/01 05:04:11            |
| ソースファイル（セッションパラメータ）     | \$InputFile1          | Sales.txt                    |
| データベース接続（セッションパラメータ）    | \$DBConnection_Target | Sales                        |
| セッションログファイル（セッションパラメータ） | \$PMSessionLogFile    | d:/session logs/firstrun.txt |

セッションのパラメータファイルには、フォルダ名とセッション名、各パラメータと変数を記述します。

```
[Production.s_MonthlyCalculations]
$PMFailureEmailUser=pccadmin@mail.com
$$State=MA
$$Time=10/1/2005 05:04:11
$InputFile1=sales.txt
$DBConnection_target=sales
$PMSessionLogFile=D:/session logs/firstrun.txt
```

次のセッション実行時には、パラメータファイルを編集して状態を MD に変更し、\$\$Time 変数を削除することができます。これにより、Integration Service は前回のセッションでリポジトリに格納された変数の値を使用できるようになります。

## パラメータファイルの作成に関するガイドライン

パラメータファイルを作成するには、以下のルールおよびガイドラインを使用します。

- **セッションパラメータをすべてリスト。**セッションパラメータにはデフォルト値はありません。Integration Service がセッションパラメータの値を検出できない場合、セッションが失敗するか、空の文字列をデフォルト値とするか、実行時にパラメータの展開に失敗する可能性があります。セッションパラメータ名の大文字/小文字は区別されません。
- **必要なマッピングパラメータおよびマッピング変数をすべてリスト。**マッピングパラメータおよび変数の値は、マッピング内のパラメータおよび変数の開始値となります。マッピングパラメータ名および変数名の大文字/小文字は区別されません。
- **一意でないセッション名へのフォルダ名の入力。**リポジトリ内にセッション名が複数存在する場合は、セッションの場所を示すためにフォルダ名を入力します。



- **マップレット内のパラメータと変数の前にマップレット名を配置。**次の形式を使用します。

```
mapplet_name.parameter_name=value
mapplet2_name.variable_name=value
```

- **複数のパラメータファイルの使用。**ワークフロー、ワークレット、およびセッションに対し、個別にパラメータファイルを割り当てます。これらすべてのタスクに同じパラメータファイルを指定するか、複数のパラメータファイルを作成することができます。
- **パラメータ値を定義する際の不必要な改行やスペースの不使用。**Integration Service では、余分なスペースはパラメータ名または値の一部として解釈されます。
- **日時の値に対する正しい日付形式の使用。**日時の値には、以下の日付形式を使用します。
  - MM/DD/RR
  - MM/DD/YYYY
  - MM/DD/RR HH24:MI
  - MM/DD/YYYY HH24:MI
  - MM/DD/RR HH24:MI:SS
  - YYYY/MM/DD HH24:MI:SS
  - MM/DD/RR HH24:MI:SS.MS
  - MM/DD/YYYY HH24:MI:SS.MS
  - MM/DD/RR HH24:MI:SS.US
  - MM/DD/YYYY HH24:MI:SS.US
  - MM/DD/RR HH24:MI:SS.NS
  - MM/DD/YYYY HH24:MI:SS.NS

以下の区切り記号を使用することができます。ダッシュ (-)、スラッシュ (/)、バックスラッシュ (\)、\uff09、コロン (:)、ピリオド (.)、およびスペース。Integration Service では、余分な空白は無視されます。年または時間の「HH12」形式には、1 桁または 3 桁の値を使用できません。
- **パラメータや変数の値を引用符で囲まない。**Integration Service では、最初の等号の後はすべて値の一部として解釈されます。
- **エラーログテーブル名のプレフィックスに対する適切な長さのパラメータまたは変数値の使用。**エラーログテーブル名のプレフィックスにパラメータまたは変数を使用する場合は、Oracle、Sybase、または Teradata のエラーログテーブルの名前を付ける際に 19 文字を超えるプレフィックスを指定しないようにします。エラーテーブル名に使用できる文字数は、最大 11 文字です。Oracle、Sybase、Teradata データベースでは、テーブル名の最大長に 30 文字という制限があります。パラメータまたは変数名は、19 文字以上を指定できます。

## パラメータおよびパラメータファイルのトラブルシューティング

セッションのパラメータファイル内にセクションが存在しますが、Integration Service がそのセクションを読み込んでいないように思われます。

Workflow Manager で表示されるとおりにフォルダ名とセッション名が正しく入力されているかどうかを確認してください。また、ユーザー定義のセッションパラメータには、適切なプレフィックスを使用してください。

ソースファイルパラメータを使用してソースファイルとその場所を指定しようとしても、Integration Service がソースファイルを見つけられません。

セッションプロパティでソースファイルディレクトリが消去されていることを確認してください。Integration Service により、ソースファイルディレクトリとソースファイル名が連結され、ソースファイルの場所が特定されます。

また、Integration Service のローカルにディレクトリが入力されており、オペレーティングシステムで適切な区切り文字が使用されていることを確認してください。

パラメータファイルを使用してワークフローを実行すると、セッションの 1 つが必ず失敗します。

パラメータファイルに定義されていないパラメータがそのセッションで使用されている可能性があります。Integration Service では、パラメータファイルを使用して、ワークフロー内のすべてのセッションが開始されます。セッションのプロパティをチェックし、パラメータファイル内ですべてのセッションパラメータが正しく定義されていることを確認してください。

パラメータファイルを使用するワークフローまたはセッションを実行しましたが、失敗しました。Integration Service は、リカバリ実行中にどのパラメータ値および変数値を使用しますか。

サービス変数、サービスプロセス変数、セッションパラメータ、およびマッピングパラメータについては、Integration Service は、これらが存在する場合にはパラメータファイルで指定された値を使用します。パラメータファイルに値が指定されていない場合、Integration Service はリカバリストレージファイルに保存された値を使用します。ワークフロー、ワークレット、マッピング変数については、Integration Service は、リカバリストレージファイルに保存された値を常に使用します。

## パラメータおよびパラメータファイルに関するヒント

1 つのパラメータファイルを使用して、関連するセッションのパラメータ情報をグループ化します。

同じデータベース接続やディレクトリを使用する複数のセッションがある場合は、それらを 1 つのパラメータファイルで定義します。接続またはディレクトリを変更する場合、1 つのパラメータファイルを編集するだけですべてのセッションの情報を更新することができます。

定期的に行うセッションに対しては、`pmcmd` と複数のパラメータファイルを組み合わせます。

サイクル内のセッションパラメータを再利用する場合があります。たとえば、あるセッションを売上げデータベースに対して毎日実行しますが、同じセッションを売上げおよびマーケティングデータベースに対して週に 1 回実行するとします。セッションの実行ごとに異なるパラメータファイルを作成することができます。週次セッションを実行するたびにセッションプロパティでパラメータファイルを変更する代わりに、`pmcmd` を使用して、セッションを開始するときに使用するパラメータファイルを指定します。

拒否ファイルパラメータとセッションログパラメータをターゲットファイルパラメータまたはターゲットデータベース接続パラメータといっしょに使用します。

ターゲットファイルパラメータまたはターゲットデータベース接続パラメータをセッションで使用する場合は、拒否ファイルパラメータを使うことで、拒否ファイルを追跡できます。同様に、セッションログパラメータを使うことで、セッションログをターゲットマシンに書き込むことができます。

リソースを使用して、パラメータファイルへのアクセス権を持つノード上でセッションが実行されることを確認します。

Administrator ツールで、パラメータファイルへのアクセス権を持つ各ノードにファイルリソースを定義し、リソースをチェックするように Integration Service を設定できます。次に、パラメータファイルを使用するセッションを編集し、リソースを割り当てます。ワークフローを実行すると、Integration Service により、リソースが使用可能なノード上で必須リソースを使用してセッションが実行されます。

セッションのワークフロー変数の初期値をセッションセクションで定義することにより、これらの値を上書きできます。

ワークフロー変数の値を変更する [Assignment] タスクがワークフローに含まれる場合、ワークフロー内の次のセッションでは、最新の変数の値がセッションの初期値として使用されます。セッションの初期値を上書きするには、パラメータファイルのセッションセクションで新しい変数の値を定義します。

パラメータおよび変数を定義する際に、他のパラメータおよび変数を使用することができます。

たとえば、パラメータファイルで、セッションパラメータ \$PMSessionLogFile をサービスプロセス変数を使用して以下のように定義できます。

```
$PMSessionLogFile=$PMSessionLogDir/TestRun.txt
```

## 第 17 章

# FastExport

この章では、以下の項目について説明します。

- [FastExport の使用の概要, 268 ページ](#)
- [手順 1。FastExport 接続の作成, 269 ページ](#)
- [手順 2 reader の変更, 271 ページ](#)
- [手順 3。ソース接続の変更, 271 ページ](#)
- [手順 4。制御ファイルのオーバーライド \(オプション\) , 272 ページ](#)
- [FastExport の使用に関するルールおよびガイドライン, 273 ページ](#)

## FastExport の使用の概要

FastExport は複数の Teradata セッションを使用して、Teradata データベースから大容量のデータを素早くエクスポートするツールです。FastExport を使用して Teradata ソースを読み込む PowerCenter セッションを作成できます。

FastExport を使用するには、Teradata ソースデータベースとのマッピングを作成します。マッピングには、1 つのソース修飾子トランスフォーメーションで結合された同じ Teradata ソースデータベースからの複数のソース定義を含めることができます。そのセッション中は、Relational reader に代わって FastExport reader を使用します。セッションでエクスポートする Teradata テーブルとの FastExport 接続を使用します。

FastExport では、何をエクスポートするかを定義する制御ファイルを使用します。セッションを開始した場合、Integration Service により、FastExport 接続属性から制御ファイルが作成されます。Teradata テーブルに SQL オーバーライドを作成する場合、Integration Service では、SQL が使用され制御ファイルが生成されます。セッションプロパティで制御ファイルを定義することにより、セッションの制御ファイルをオーバーライドできます。

Integration Service により、FastExport メッセージがセッションログに書き込まれ、FastExport パフォーマンスに関する情報が FastExport ログに書き込まれます。PowerCenter は、FastExport ログを、Temporary File Name セッション属性で定義されたフォルダに保存します。FastExport ログのデフォルト拡張子は、.log です。

セッションで FastExport を使用するには、以下の手順を実行します。

1. Workflow Manager で FastExport 接続を作成し、接続属性を設定します。
2. セッションを開いて、[Reader] を Relational から Teradata FastExport Reader に変更します。
3. 接続タイプを変更し、セッションに対して FastExport 接続を選択します。
4. 必要に応じて、テキストエディタで FastExport 制御ファイルを作成し、リポジトリに保存します。

# 手順 1。FastExport 接続の作成

Workflow Manager で FastExport 接続を作成します。FastExport 接続を編集すると、その接続を使用するセッションはすべて、この更新された接続を使用します。

FastExport 接続を作成するには：

1. Workflow Manager で、[接続] - [アプリケーション] をクリックします。  
[接続ブラウザ] ダイアログボックスが表示されます。
2. [新規作成] をクリックします。
3. Teradata FastExport 接続を選択し、[OK] をクリックします。
4. FastExport 接続の名前を入力します。
5. データベースユーザー名を入力します。
6. データベースユーザー名のパスワードを入力するか、あるいは [パスワードでパラメータを使用] をクリックしてデータベースパスワードにセッションパラメータ `$ParamName` を使用します。  
[パスワードでパラメータを使用] を有効にした場合は、ワークフローまたはセッションパラメータファイルでパスワードを定義し、`pmpasswdCRYPT_DATA` オプションを使用してそのパスワードを暗号化します。
7. FastExport で Teradata ソースの読み込みに使用するコードページを選択します。  
FastExport では、`fexpcodepagemapfile.dat` ファイルが使用され、FastExport でサポートされる Teradata 文字セットにコードページ名がマッピングされます。このファイルにコードページが含まれていること、割り当てられた文字セットが Teradata データベースで有効になっていることを確認します。
8. FastExport 属性を入力して、[OK] をクリックします。

以下の表に、Teradata FastExport 接続に設定する属性を示します。

| 属性       | デフォルト値 | 説明                                                                                                                                                                                                                                                                                   |
|----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TDPID    | なし     | Teradata データベース ID。                                                                                                                                                                                                                                                                  |
| Tenacity | 4      | FastExport が Teradata データベースへのログオンを試行する時間の長さです (単位：時間)。FastExport がログオンしようとしたが、最大数の Teradata セッションがすでに実行されている場合、FastExport は、SLEEP オプションで定義された時間待機します。SLEEP 時間が経ったら、FastExport は、再度 Teradata データベースにログオンしようとします。<br>FastExport は、必要な数のセッションにログオンするか、TENACITY 時間を超えるまで、このプロセスを繰り返します。 |
| 最大セッション数 | 1      | FastExport ジョブ 1 つあたりの FastExport セッションの最大数。[最大セッション数] は 1 以上で、さらにシステムのアクセスモジュールプロセス (AMP) の合計数以下でなければなりません。                                                                                                                                                                         |
| スリープ     | 6      | FastExport がログインを再試行するまでの待ち時間 (単位：分)。FastExport は、ログインが成功するか、Tenacity 時間が経過するまでログインの試行を繰り返します。                                                                                                                                                                                       |
| ブロックサイズ  | 64000  | エクスポートしたデータに使用する最大ブロックサイズ。                                                                                                                                                                                                                                                           |
| データ暗号化   | 無効     | FastExport のデータ暗号化を有効にします。バージョン 8 Teradata クライアントを使用すればデータ暗号化を使用できます。                                                                                                                                                                                                                |

| 属性         | デフォルト値        | 説明                                                                                                                                                                                                                                                                                                                       |
|------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Logtable 名 | FE_<ソーステーブル名> | ログテーブル名をリスタートします。FastExport ユーティリティでは、リスタートログテーブルでこの情報を使用して、Teradata データベース障害またはクライアントシステム障害が原因で停止したジョブをリスタートします。各 FastExport ジョブでは、個別のログテーブルを使用する必要があります。指定したテーブルが存在していない場合、FastExport ユーティリティは、このテーブルを作成し、リスタートログとして使用します。<br>PowerCenter では、FastExport のリスタートはサポートされていません。ただし、出力をステージングする場合、FastExport を手動でリスタートできます。 |
| 実行可能名      | fexp          | ソースデータの読み込みに使用する Teradata コマンド。デフォルト値を使用します。                                                                                                                                                                                                                                                                             |
| データベース名    | なし            | 接続する Teradata データベースの名前。統合サービスは、テーブル名のプレフィックスとしてデータベース名を使用して SQL 文を生成します。                                                                                                                                                                                                                                                |

## コードページのマッピングファイルの確認

FastExport 接続を作成する場合、FastExport で Teradata ソースの読み込みに使用する PowerCenter のコードページを選択します。FastExport では、fexpcodepagemapfile.dat ファイルが使用され、FastExport でサポートされる Teradata 文字セットに PowerCenter のコードページ名がマッピングされます。例えば、「MS Windows Latin 1 (ANSI)、Latin1 のスーパーセット」を接続コードページとして選択した場合、PowerCenter では「MS1252」という名前が付けられたコードページが使用され、Teradata では「Latin1252\_0A」という名前が付けられた文字セットが使用されます。

fexpcodepagemapfile.dat ファイルでは、*Teradata International Character Set Support* に指定されたの中から最適な Teradata 文字セットを PowerCenter コードページにマップします。Teradata 文字セットは、使用する前にデータベースで有効にしておく必要があります。デフォルトでは、以下の文字セットのみが Teradata データベースで有効になっています。

- ASCII
- EBCDIC
- UTF8
- UTF16

FastExport 接続に選択する PowerCenter のコードページが fexpcodepagemapfile.dat ファイルに存在しない場合、または割り当てられた Teradata 文字セットがデータベースで有効になっていない場合は、Integration Service でセッションが失敗します。

fexpcodepagemapfile.dat ファイルは、<PowerCenter installation directory>\server\bin にあります。このファイルに FastExport 接続で選択する PowerCenter コードページが含まれていること、割り当てた文字セットが Teradata データベースで有効になっていることを確認してください。追加の PowerCenter コードページを Teradata 文字セットに割り当てたり、既存のマッピングを変更したりするには、テキストエディタを使用します。PowerCenter コードページを Teradata 文字セットに割り当てるには、以下の形式で単一行に名前を入力します。

<PowerCenter\_code\_page> = <Teradata\_character\_set>.

例: MS1252 = Latin1252\_0A

このファイルを編集する際には、以下のルールおよびガイドラインを使用します。

- コメントを指定する場合は、感嘆符 (!) で行を開始します。
- 行は、空白スペースのみやタブ文字のみでは構成できません。
- このファイルで単一の PowerCenter のコードページが複数の Teradata 文字セットにマッピングされている場合、FastExport ではファイルで最後に割り当てられた文字セットが使用されます。

**注:** Teradata では、Unicode の UTF-16LE エンコードについて、ビッグエンディアンとリトルエンディアンが区別されません。UTF-16LE の文字を処理する場合は、Workflow Manager で FastExport 接続を作成する際に「Unicode の UTF-16LE エンコード（プラットフォームエンディアン）」のコードページを選択します。

## 手順 2 reader の変更

Teradata のデフォルト reader は Relational reader です。FastExport を使用するには、reader を Teradata FastExport reader に変更します。

## 手順 3。ソース接続の変更

セッションで FastExport を使用するには、Teradata ソース接続を Teradata FastExport 接続に変更します。セッション属性によっては上書きすることができます。

以下の表に、FastExport で変更できるセッション属性を示します。

| 属性             | デフォルト値       | 精度                                                                                                                    |
|----------------|--------------|-----------------------------------------------------------------------------------------------------------------------|
| ステージング済み       | 無効           | 有効にすると、FastExport はデータをステージファイルに書き込みます。無効にすると、FastExport はデータを名前付きパイプに書き込みます。                                         |
| 小数秒の精度         | 0            | タイムスタンプ内の小数点以下のミリ秒の精度。0 - 6 を入力できます。例えば、精度 6 のタイムスタンプは、hh:mm:ss.ss.ss.ss です。ミリ秒精度は、Teradata データベース内の設定と一致している必要があります。 |
| 一時ファイル         | \$PMTempDir\ | PowerCenter は、一時ファイル名を使用して、ログファイル、制御ファイル、およびステージングする出力ファイルの名前を生成します。ファイルの完全パスを入力します。                                  |
| 制御ファイルのオーバーライド | 空白           | 制御ファイルのテキストデータ。この属性を使用して、統合サービスによってセッションに作成された制御ファイルをオーバーライドします。                                                      |



## 手順 4。制御ファイルのオーバーライド（オプション）

デフォルトで、統合サービスにより、FastExport でセッションを実行する場合、セッションおよび接続のプロパティに基づいて、FastExport 制御ファイルが生成されます。統合サービスにより、生成した制御ファイルが一時ファイルディレクトリに保存され、次回セッションを実行する際にオーバーライドされます。

統合サービスにより生成された制御ファイルをオーバーライドできます。制御ファイルをオーバーライドする場合、Workflow Designer により制御ファイルがリポジトリに保存されます。セッションを実行する場合、統合サービスにより保存された制御ファイルが使用されます。

各 FastExport 文は、次の基準を満たしている必要があります。

- 改行します。
- 先頭にはピリオド (.) を指定します。
- 終わりにはセミコロン (;) を指定します。

以下の表に、PowerCenter で使用できる制御ファイル文を示します。

| 制御ファイル文                          | 説明                                         |
|----------------------------------|--------------------------------------------|
| .LOGTABLE utillog;               | リスタートログテーブル名。                              |
| LOGON tdpz/user,pswd;            | データベースログイン文字列。データベース、ユーザー名、およびパスワードを指定します。 |
| BEGIN EXPORT                     | 最初のエクスポートコマンド。                             |
| .SESSIONS 20;                    | Teradata セッションの数。                          |
| .EXPORT OUTFILE ddname2;         | エクスポートするデータの宛先ファイル。                        |
| SELECT EmpNo, Hours FROM charges | データを選択する SQL 文。                            |
| WHERE Proj_ID = 20               | -                                          |
| ORDER BY EmpNo ;                 | -                                          |
| .END EXPORT ;                    | エクスポートタスクの終わりを示し、エクスポートプロセスを開始します。         |
| LOGOFF ;                         | データベースからの切断。                               |

制御ファイルをオーバーライドする手順

1. テキストエディタで制御ファイルを作成します。
2. 制御ファイルのテキストをクリップボードにコピーします。
3. 制御ファイルのテキストを [Control File Override] フィールドに貼り付けます。

Workflow Manager では、制御ファイルの構文は検証しません。セッションの実行時に Teradata が制御ファイルの構文を検証します。制御ファイルが無効だと、セッションは失敗に終わります。

**ヒント:** 制御ファイルを読み込み専用に変更することで、その制御ファイルを各セッションに使用することができます。統合サービスでは、読み込み専用ファイルは上書きされません。



# FastExport の使用に関するルールおよびガイドライン

PowerCenter で FastExport を使用する場合、以下のルールおよびガイドラインに従ってください。

- Teradata に SQL オーバーライドを使用する場合、PowerCenter は、この SQL オーバーライドを使用して FastExport 制御ファイルを作成します。SQL オーバーライドを使用しない場合、PowerCenter は、ソース修飾子内の接続されたポートに基づいて制御ファイルを生成します。
- UNIX MP-RAS オペレーティングシステムでは、FastExport は、最大 2 GB のエクスポートファイルサイズをサポートしています。他のオペレーティングシステムでは、ファイルサイズに制限はありません。
- エクスポートされたデータファイルを結合することはできません。
- セッション実行前 SQL コマンドおよび FastExport を使用した場合、セッションは失敗します。

## 第 18 章

# 外部データのロード

この章では、以下の項目について説明します。

- [外部データのロードの概要, 274 ページ](#)
- [外部ローダーの動作, 275 ページ](#)
- [IBM DB2 へのロード, 276 ページ](#)
- [Oracle へのロード, 283 ページ](#)
- [Sybase IQ へのロード, 284 ページ](#)
- [Teradata へのロード, 287 ページ](#)
- [セッション内での外部データのロードの設定, 296 ページ](#)
- [外部データのロードのトラブルシューティング, 298 ページ](#)

## 外部データのロードの概要

IBM DB2、Oracle、Sybase IQ、および Teradata の外部ローダを使ってセッションターゲットファイルをそれぞれのデータベースにロードするように、セッションを設定することができます。外部ローダを使用すると、ファイルまたはパイプから情報を直接ロードすることによって、SQL コマンドを実行して同じデータをデータベースに挿入するよりも、セッションのパフォーマンスを高めることができます。

1 つのセッションで複数の外部ローダを使用してください。たとえば、マッピングに 2 つのターゲットが含まれている場合には、Oracle 外部ローダ接続と Sybase IQ 外部ローダ接続を使用するセッションを作成できます。

### はじめに

外部ローダーを実行する前に、以下の作業を実行します。

- **制約の無効化。** ロードを実行する前に、データを受け取るテーブルに組み込まれている制約を無効にしておきます。制約を無効にする方法の詳細については、データベースのマニュアルを参照してください。
- **データベースロギングのオフまたは無効化。** 高いパフォーマンスを維持するために、コミット間隔を増やし、データベースロギングをオフにすることができます。ただし、失敗したセッションのデータベースのリカバリを実行するには、データベースロギングをオンにしておく必要があります。
- **コードページの設定。** IBM DB2、Oracle、Sybase IQ、および Teradata データベースサーバーは、ターゲットフラットファイルのコードページと同じコードページを使用する必要があります。Integration Service によって、ターゲットフラットファイルのコードページを使用して制御ファイルとターゲットフラットファイルが作成されます。ターゲットフラットファイルに対して 7 ビット ASCII 以外のコードページを使用する場合は、Integration Service を Unicode データ移動モードで実行します。

- **外部ローダー接続をリソースとして設定。**Integration Service がグリッド上で実行するように設定されている場合、外部ローダーが利用可能なノード上に外部ローダー接続をリソースとして設定します。

## 外部ローダーの動作

外部ローダーを使用するセッションを実行する場合、Integration Service により制御ファイルとターゲットフラットファイルが作成されます。制御ファイルには、データ形式や外部ローダーに対するロード指示などの情報が格納されます。制御ファイルの拡張子は.ctl です。制御ファイルおよびターゲットフラットファイルは、ターゲットファイルディレクトリに格納されます。

セッションを実行する場合、Integration Service によりターゲットファイルが削除され、再作成されます。外部ローダーは制御ファイルを使用してセッション出力をデータベースにロードします。日時データは、次の方法で Integration Service で処理された後、データベースにロードされます。

- セッションがサブ秒切り捨ての設定になっている場合、Integration Service での日時データ処理には精度 19 が使用されます。
- セッションがサブ秒切り捨ての設定になっていない場合、Integration Service での日時データ処理は、ターゲットフラットファイル内に指定されている精度に基づきます。精度は 19～29 の範囲です。サブ秒の切り捨ては、指定された精度に基づきます。
- ターゲットファイル内に指定された精度は、データベース用に指定されている精度を越える場合、データベース用に指定されている最大精度に制限されます。

Integration Service は、すべての外部データのロードが完了するまで待ってから、セッション実行後に実行するコマンドを実行し、エクスターナルプロシージャを実行し、セッション実行後に発信されるメールを送信します。

Integration Service により、外部ローダーの初期化と完了のメッセージがセッションログに書き込まれます。外部ローダーのパフォーマンスの詳細については、外部ローダーのログをチェックしてください。ローダーは、ターゲットフラットファイルと同じディレクトリにログを保存します。外部ローダーのログにはデフォルトで.ldrlog という拡張子が付きます。

外部ローダーの動作は、選択したデータロード方法によって異なります。名前付きパイプまたはフラットファイルにデータをロードできます。

## 名前付きパイプへのデータのロード

データがパイプに出力された時点で、外部ローダーは即座にデータベースへのデータのロードを開始します。ローダーはロードが完了した時点で、即座に名前付きパイプを削除します。

UNIX では、Integration Service により、設定されたターゲットファイル名に従って名付けられた名前付きパイプに書き込まれます。

Windows では、Integration Service により、指定された形式を使用して名前付きパイプにデータが書き込まれます。

```
\\.\pipe\
```

パイプ名は設定されたターゲットファイル名と同じです。

## フラットファイルへのデータのステージング

Windows または UNIX でフラットファイルにデータをステージングする場合、Integration Service によって、設定されたターゲットファイル名に従って名付けられたフラットファイルにデータが書き込まれます。

Integration Service によってターゲットフラットファイルにすべてのデータが書き込まれた後で、外部ローダーによって、ターゲットデータベースへのデータのロードが開始されます。外部ローダーは、ターゲットフラットファイルをデータベースにロードしたあとで、このファイルを削除しません。ターゲットファイルのディレクトリに、ターゲットフラットファイルのサイズに必要な容量があることを確認してください。

**注:** Integration Service では、フラットファイルにデータをステージングするときに、ポートの位取りに基づいて数値が丸められます。名前付きパイプにデータをロードする外部ローダーを使用している場合、またはターゲットをノーマルロードに設定している場合、結果は丸められません。

Integration Service がフラットファイルターゲットにすべてのデータを書き込む前にセッションが強制終了または失敗した場合、外部ローダーは起動されません。Integration Service がフラットファイルターゲットにすべてのデータを書き込んだ後にセッションが強制終了または失敗した場合、外部ローダーはターゲットデータベースへのデータのロードを完了してから終了します。

## 外部ローダーを使用したセッションのパーティション化

フラットファイルターゲットを使用するセッションに複数のパーティションを設定する場合、Integration Service によってパーティションごとにフラットファイルが 1 つずつ作成されます。一部の外部ローダーは、複数のファイルからデータをロードすることができません。複数のパーティションがあるセッションで外部ローダーを使用するときは、ご使用の外部ローダーに従ってターゲットパーティションタイプを設定する必要があります。

複数のファイルからデータをロードできる外部ローダーを使用する場合は、フラットファイルターゲットで使用可能なパーティションタイプを選択できます。パーティションごとに外部ローダー接続も選択してください。Integration Service がパーティションごとに出力ファイルを作成し、外部ローダーによって各ターゲットファイルからの出力がデータベースにロードされます。次のローダーを使用する場合は、任意のターゲットパーティションタイプを使用してください。

- Oracle (パラレルロードが有効)
- Teradata TPump

複数のファイルからロードできないローダーを使用する場合は、ラウンドロビンパーティション化を使用して、データを単一のターゲットファイルヘルパーティングします。パーティションごとに外部ローダー接続を選択してください。ただし、Integration Service によってこのローダー接続が最初のパーティションに使用されます。Integration Service によって単一の出力ファイルが作成され、外部ローダーによってターゲットファイルからの出力がデータベースにロードされます。ターゲットに他のパーティションタイプを選択した場合、Integration Service のセッションは失敗します。次のローダーを使用する場合は、ラウンドロビンのターゲットパーティションタイプを使用してください。

- IBM DB2 EE
- IBM DB2 EEE Autoloader
- Oracle (パラレルロードが無効)
- Sybase IQ
- Teradata MultiLoad
- Teradata Fastload

## IBM DB2 へのロード

IBM DB2 ターゲットにロードする際は、IBM DB2 EE または IBM DB2 EEE 外部ローダーを使用します。どちらの外部ローダーも、ターゲットに対して挿入操作と置換操作を行います。ロード操作のリスタートまたは終了

も行うことができます。どちらの外部ローダーも、データをパーティション化し、パーティション化されたデータを対応するデータベースパーティションへ同時にロードすることができます。

## IBM DB2 EE 外部ローダー

IBM DB2 EE 外部ローダーを使用して、次のいずれかのデータベースにロードします。

- IBM DB2 EE バージョン 8.x
- IBM DB2 EEE バージョン 8.x
- IBM DB2 バージョン 9.x

IBM DB2 EE 外部ローダーによって、Integration Service インストールディレクトリにある次のいずれかの実行可能ファイルが呼び出されます。

- **db2load**。バージョン 9.5 より前の IBM DB2 クライアント用に使用します。
- **db2load95**。IBM DB2 クライアントバージョン 9.5 以降に使用します。

外部ローダー接続を作成する場合は、Integration Service プロセスが実行されるマシン上にインストールされている IBM DB2 クライアントバージョンに応じて、実行可能ファイル名を指定します。

IBM DB2 EE 外部ローダーでは、Integration Service からリモートであるマシン上の IBM DB2 サーバーにデータをロードできます。

### LOB データの処理

IBM DB2 EE 外部ローダーでは、LOB データ (Blob、Clob、Dbclob データなど) をロードすることができません。IBM DB2 EE 外部ローダーを使用するセッションを実行し、ソースに LOB データが含まれる場合は、以下のマッピング設定に応じて、外部ローダーでは残りのデータを正常にターゲットにロードできます。

- **LOB ポートが接続されていない状態**。外部ローダーによって、残りのデータが正常にターゲットにロードされます。
- **LOB ポートが接続されている状態**。データベースバージョン 8.x にロードする場合、外部ローダーにより LOB データは NULL としてロードされ、残りのデータが正しくロードされます。データベースバージョン 9.x にロードする場合、外部ローダーではどのデータもロードされません。拒否された行は、外部ローダーによって外部ローダーログに記録されます。

## IBM DB2 EEE 外部ローダー

IBM DB2 EEE 外部ローダーを使用して、IBM DB2 EEE バージョン 8.x データベースにロードします。IBM DB2 EEE 外部ローダーは、IBM DB2 Autoloader プログラムを起動してデータをロードします。Autoloader プログラムは、db2atld 実行ファイルを使用します。IBM DB2 EEE ローダーによって、IBM DB2 サーバーは Integration Service をホストする同じマシン上にあることが要求されます。

**注:** IBM DB2 EEE サーバーが Integration Service からリモートであるマシン上に存在する場合は、IBM DB2 EE 外部ローダーを使用するか、リレーショナルデータベース接続を使用して IBM DB2 EEE データベースに接続します。IBM DB2 ターゲットのためにデータベースパーティション化を使用します。データベースパーティション化を使用した場合、Integration Service によって、テーブルのパーティション情報のために IBM DB2 システムに対してクエリが行われ、パーティション化データがターゲットデータベースで対応するノードにロードされます。

## IBM DB2 EEE 外部ローダーに関するルールおよびガイドライン

外部ローダーを使用して IBM DB2 にロードする場合には、以下の規則およびガイドラインに従ってください。

- IBM DB2 外部ローダーは、区切りフラットファイルからロードします。ターゲットテーブルのカラムの幅が、すべてのデータを格納するのに十分かどうか確認してください。
- IBM DB2 クライアント認証を使用する接続の場合は、外部ローダー接続を作成する際に PmNullUser ユーザー名と PmNullPasswd を入力してください。PowerCenter によって、接続ユーザー名が PmNullUser で IBM DB2 データベースに接続する際に、IBM DB2 クライアント認証が使用されます。
- 複数のパーティションを伴うセッションについては、ラウンドロビンのパーティションタイプを使用して、データを単一のターゲットファイルにルーティングします。
- 同じパイプライン内の複数のターゲットで IBM DB2 外部ローダーを使用するように設定した場合は、各ローダーによりターゲットデータベース上の異なるテーブルスペースにロードされる必要があります。
- データベーステーブルにデータをロードするための正しい権限レベルと特権が必要です。

## 操作モードの設定

外部ローダーが実行するロードの種類は、IBM DB2 の操作モードによって異なります。IBM DB2 EE または IBM DB2 EEE 外部ローダーは、次のいずれかの操作モードで動作するように設定できます。

- **挿入。**既存のテーブルデータを変更せずに、ロードしたデータをテーブルに追加します。
- **置き換え。**テーブルから既存のデータをすべて削除して、ロードしたデータを挿入します。テーブルおよびインデックスの定義は変更されません。
- **リスタート。**以前に中断されたロード操作をリスタートします。
- **終了。**以前に中断したロード操作を終了して、（整合点を超過している場合でも）操作を開始ポイントまでロールバックします。テーブルスペースは通常状態に戻り、外部ローダーによってすべてのテーブルオブジェクトは一貫した状態になります。

## オーソリティ、特権、権限の設定

IBM DB2 特権により、データベースリソースの作成およびアクセスが許可されます。オーソリティレベルにより、特権をグループ化し、データベースマネージャに対するより高いレベルの保守およびユーティリティ操作を実行することが可能になります。特権とオーソリティにより、データベースマネージャとそのデータベースオブジェクトへのアクセスを制御します。必要な特権またはオーソリティを持っているオブジェクトにアクセスできます。

テーブルにデータをロードするためには、次のいずれかのオーソリティが必要です。

- SYSADM オーソリティ
- DBADM オーソリティ
- データベースに対する LOAD オーソリティおよび次のいずれかの特権
  - ロードユーティリティを INSERT モード、TERMINATE モード、または RESTART モードで起動する場合、テーブルに対する INSERT 特権。
  - ロードユーティリティを REPLACE モード、TERMINATE モード、または RESTART モードで起動する場合、テーブルに対する INSERT および DELETE 特権。

さらに、適切な読み込みアクセスおよび読み込み/書き込み権限を持っている必要があります。

- データベースインスタンスのオーナーは、外部ローダー入力ファイルに対する読み込みアクセス権を持っている必要があります。

- IBM DB2 が Windows 上のサービスとして動作している場合には、サービス開始アカウントに、LAN リソース（ドライブ、ディレクトリ、ファイルなど）を使用する読み込み/書き込み権限を持つユーザーアカウントを設定する必要があります。
- IBM DB2 EEE にロードする場合には、データベースインスタンスのオーナーがロードダンプファイルおよびロード一時ファイルに対する書き込み権限を持っている必要があります。

## IBM DB2 EE 外部ローダーの属性の設定

IBM DB2 EE 外部ローダーでは、ロード先となる以下のデータベースに基づき、単一のログまたは複数のログが作成されます。

- **IBM DB2 EE バージョン 8.x またはパーティション化されていない IBM DB2 バージョン 9.x。** 外部ローダーによって、ターゲットフラットファイルと同じディレクトリ内に、拡張子.ldrlog を持つ単一の外部ローダーログが作成されます。
- **IBM DB2 EEE バージョン 8.x またはパーティション化された IBM DB2 バージョン 9.x。** 外部ローダーによって、ターゲットフラットファイルと同じディレクトリ内に、複数の外部ローダーログが作成されます。ローダーログには以下の拡張子が付いています。
  - ldrlog.load.number。ロードエージェント外部ローダープロセスによって作成されたものです。Load Agent によってログファイルが 2 つ作成されます。
  - ldrlog.part.パーティション番号。パーティション化のエージェント外部ローダープロセスによって作成されたものです。パーティション化のエージェントでは、ターゲットテーブル内のパーティションの数に基づき、複数のファイルを作成できます。
  - ldrlog.prep.パーティション番号。パーティション化前のエージェント外部ローダープロセスによって作成されたものです。パーティション化前のエージェントでは、ターゲットテーブル内のパーティションの数に基づき、複数のファイルを作成できます。
  - ldrlog。IBM DB2 EE 外部ローダーによって作成されたものです。

以下の表に、IBM DB2 EE 外部ローダー接続の属性を示します。

| 属性                         | デフォルト値  | 説明                                                                                                                                                                                                                                                                          |
|----------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Opmode                     | 挿入      | IBM DB2 外部ローダーの操作モード。次のいずれかの操作モードを選択します。 <ul style="list-style-type: none"> <li>- 挿入</li> <li>- 置換</li> <li>- Restart</li> <li>- 終了</li> </ul>                                                                                                                              |
| External Loader Executable | db2load | IBM DB2 EE 外部ローダー実行ファイルの名前。Integration Service プロセスが実行されるマシン上にインストールされた IBM DB2 クライアントバージョンに基づき、次のいずれかのファイル名を入力します。 <ul style="list-style-type: none"> <li>- db2load。バージョン 9.5 より前の IBM DB2 クライアントに使用します。</li> <li>- db2load95。IBM DB2 クライアントバージョン 9.5 以降に使用します。</li> </ul> |
| DB2 Server Location        | リモート    | Integration Service から見た IBM DB2 データベースサーバの場所。データベースサーバが、Integration Service が稼動しているマシン上に存在している場合は、[Local] を選択します。データベースサーバがほかのマシン上に存在している場合は、[Remote] を選択します。                                                                                                              |



| 属性          | デフォルト値 | 説明                                                                                                                                                                                            |
|-------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Is Staged   | 無効     | データのロード方法。データベースへのロード前に、フラットファイルのステージング領域にデータをロードするには、[Is Staged] を選択します。デフォルトでは、データは名前付きパイプを使用してデータベースにロードされます。                                                                              |
| Recoverable | 有効     | forward recovery が有効な場合は、テーブルスペースをバックアップ保留状態にします。forward recovery を無効にすると、IBM DB2 テーブルスペースはバックアップ保留状態に設定されません。IBM DB2 テーブルスペースがバックアップ保留状態にある場合は、テーブルスペースに関する操作を行う前にデータベースを完全にバックアップする必要があります。 |

## IBM DB2 EE 外部ローダーを使用した空白スペースのロード

IBM DB2 EE 外部ローダーを通して空白スペースをロードする必要がある場合は、セッションを設定する必要があります。ステージングしたモードでは、オプションのダブル引用符を使用するようにフラットファイルを設定します。ステージングしていないモードでは、以下の行を制御ファイルに追加します。

MODIFIEDBY = keepblanks

制御ファイルを読み込み専用に設定します。

## IBM DB2 EE 外部ローダの戻りコード

IBM DB2 EE 外部ローダは、ロード操作が成功したか失敗したかを戻りコードで示します。Integration Service によって、外部ローダーの戻りコードがセッションログに書き込まれます。戻りコード (0) は、ロード操作が成功したことを示します。外部ローダーによってロード操作が正常に完了された場合、Integration Service によってセッションログに次のメッセージが書き込まれます。

WRT\_8029 External loader process <external loader name> exited successfully.

それ以外の戻りコードは、ロード操作が失敗したことを示します。Integration Service によって、以下のエラーメッセージがセッションログに書き込まれます。

WRT\_8047 Error: External loader process <external loader name> exited with error <return code>.

以下の表に、IBM DB2 EE 外部ローダーの戻りコードを示します。

| コード | 説明                                                     |
|-----|--------------------------------------------------------|
| 0   | 外部ローダ操作が正常に完了しました。                                     |
| 1   | 外部ローダが制御ファイルを見つけられません。                                 |
| 2   | 外部ローダが外部ローダログファイルを開くことができませんでした。                       |
| 3   | 制御ファイルが他のプロセスによってロックされているため、外部ローダが制御ファイルにアクセスできませんでした。 |
| 4   | IBM DB2 データベースがエラーを返しました。                              |



## IBM DB2 EEE 外部ローダの属性の設定

IBM DB2 EEE 外部ローダは、データベースにロードするときに異なるロードモードを使用するように設定できます。ロードモードにより、IBM DB2 EEE 外部ローダがデータベースの複数パーティションにまたがってデータをどのようにロードするかが決まります。IBM DB2 EEE 外部ローダは、次のロードモードを使用するように設定できます。

- **分割およびロード。**データをパーティション化して、対応するデータベースパーティションを使用して同時にロードします。
- **分割専用。**データをパーティション化して、指定された分割ファイルディレクトリ内のファイルに出力を書き込みます。
- **ロード専用。**データをパーティション化しません。対応するデータベースパーティションを使用して既存の分割ファイルにデータをロードします。
- **分析。**すべてのデータベースパーティション間で均一に分散した最適パーティションマップを生成します。外部ローダーを分析モードで実行した後に、分割およびロードモードで実行する場合、外部ローダーは最適パーティションマップを使用してデータをパーティション化します。

IBM DB2 EEE 外部ローダは、ロード先のデータベースパーティションの数に基づいて複数のログを作成します。各パーティションについて、外部ローダはパーティション番号に対応した番号を外部ローダログファイル名に付加します。IBM DB EEE 外部ローダログのファイル名は、「<ファイル名>.ldrlog.<パーティション番号/2>」という形式で指定します。

Integration Service では、IBM DB2 EEE 外部ローダーログがアーカイブされたり、上書きされることはありません。外部ローダーの実行時に同じ名前の外部ローダーログが存在する場合、外部ローダーによって既存の外部ローダーログファイルの最後に新しい外部ローダーログメッセージが追加されます。外部ローダーログファイルは手動でアーカイブ、または削除する必要があります。

IBM DB2 EEE 外部ローダーの戻りコードの詳細については、IBM DB2 のマニュアルを参照してください。

以下の表に、IBM DB2 EEE 外部ローダー接続の属性を示します。

| 属性                         | デフォルト値  | 説明                                                                                                                                     |
|----------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------|
| Opmode                     | 挿入      | IBM DB2 外部ローダの操作モード。次のいずれかの操作モードを選択します。 <ul style="list-style-type: none"><li>- 挿入</li><li>- 置換</li><li>- リスタート</li><li>- 終了</li></ul> |
| External Loader Executable | db2atld | IBM DB2 EEE 外部ローダ実行ファイルの名前。                                                                                                            |
| Split File Location        | なし      | 分割ファイルの場所。SPLIT_ONLY ロードモードを設定した場合、外部ローダは分割ファイルを作成します。                                                                                 |
| Output Nodes               | なし      | ロード操作を行うデータベースパーティション。                                                                                                                 |
| Split Nodes                | なし      | データをどのように分割するかを決定するデータベースパーティション。この属性を指定しないと、外部ローダが最適な分割方法を決定します。                                                                      |

| 属性                | デフォルト値         | 説明                                                                                                                                                                                              |
|-------------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mode              | Split and load | 外部ローダがデータをロードするときに使用するロードモード。次のいずれかのロードモードを選択します。<br><ul style="list-style-type: none"> <li>- Split and load</li> <li>- Split only</li> <li>- Load only</li> <li>- 分析</li> </ul>                |
| Max Num Splitters | 25             | 分割プロセスの最大数。                                                                                                                                                                                     |
| 強制                | いいえ            | 開始時に一部のターゲットパーティションまたはテーブルスペースがオフラインであることを検出した場合でも、外部ローダ操作を強制的に継続します。                                                                                                                           |
| ステータス Interval    | 100            | 外部ローダログに進捗メッセージを書き込む前に外部ローダがロードするデータのサイズ（単位：MB）。1～4,000MB の値を指定します。                                                                                                                             |
| ポート               | 6000-6063      | 外部ローダが IBM DB2 サーバーとの内部通信用ソケットを作成するために使用する、TCP ポートの範囲。                                                                                                                                          |
| Check Level       | Nocheck        | 入力または出力時にレコードの切り詰めをチェックします。                                                                                                                                                                     |
| マップファイル出力         | なし             | パーティションマップを指定するファイルの名前。カスタマイズしたパーティションマップを使用するには、この属性を指定します。外部ローダを分析ロードモードで実行する場合に、カスタマイズしたパーティションマップを生成してください。                                                                                 |
| マップファイル入力         | なし             | 外部ローダを [Analyze] ロードモードで実行する場合のパーティションマップの名前。外部ローダを [Analyze] モードで実行したい場合は、この属性を指定する必要があります。                                                                                                    |
| トレース              | 0              | データ変換プロセスのダンプとハッシュ値の出力を見たい場合に、外部ローダがトレースする行数。                                                                                                                                                   |
| ステージング済み          | 無効             | データのロード方法。データベースへのロード前に、フラットファイルのステージング領域にデータをロードするには、[Is Staged] を選択します。それ以外の場合、データは名前付きパイプを使用してデータベースにロードされます。                                                                                |
| 日付形式              | mm/dd/yyyy     | 日付形式。ターゲット定義で定義する日付形式と一致する必要があります。IBM DB2 は、次の日付形式をサポートします。<br><ul style="list-style-type: none"> <li>- MM/DD/YYYY</li> <li>- YYYY-MM-DD</li> <li>- DD.MM.YYYY</li> <li>- YYYY-MM-DD</li> </ul> |

# Oracle へのロード

Oracle ターゲットにロードする際は、Oracle SQL Loader を使用してターゲットに対して挿入、更新、および削除操作を実行してください。

Oracle 外部ローダーでは、データベースによって拒否されたデータのための拒否ファイルが作成されます。拒否ファイルの拡張子は .ldrreject です。ローダーは、ターゲットファイルディレクトリに拒否ファイルを保存します。

## Oracle 外部ローダーに関するルールおよびガイドライン

外部ローダーを使用して Oracle にロードする場合には、以下の規則およびガイドラインに従ってください。

- Oracle 外部ローダーを選択した場合は、デフォルトの外部ローダーの実行ファイル名は sqlload です。これはほとんどの UNIX プラットフォームでは正しい名前ですが、Windows を使っている場合は、Oracle のマニュアルを調べて、外部ローダー実行ファイルの名前を確認してください。
- Oracle OS 認証を使用する接続の場合は、外部ローダー接続を作成する際に PmNullUser ユーザー名と PmNullPasswd を入力してください。PowerCenter は接続ユーザー名が PmNullUser で Oracle データベースへの接続を要求しているときに、Oracle OS 認証を使用します。
- Oracle 外部ローダーのターゲットフラットファイルは、固定長または区切りファイルです。
- パーティション化されたターゲットに書き込むときに最適なパフォーマンスを得るには、[ダイレクトパス] を選択してください。詳細については、Oracle のマニュアルを参照してください。
- Oracle 10.x または Oracle 11.x ターゲットのタイムスタンプカラムにサブ秒データを書き込むようにセッションを設定した場合、Integration Service によってデフォルトで書き込まれるサブ秒データはマイクロ秒までです。精度を高めるには、制御ファイルを編集し、タイムスタンプ精度を変更します。たとえば、TIMESTAMP(9)を指定してナノ秒を処理します。
- 最適なパフォーマンスを得るには、以下のガイドラインに従って、パーティション化されたターゲットとパーティション化されていないターゲットの設定を決定します。

| ターゲット                                                                     | ロード方法   | パラレルロード | ロードモード |
|---------------------------------------------------------------------------|---------|---------|--------|
| パーティション化                                                                  | ダイレクトパス | 有効化     | 付加     |
| パーティション化                                                                  | 通常パス    | 有効化     | なし     |
| 非パーティション化                                                                 | なし      | 無効化*    | なし     |
| * パラレルモードを無効にした場合、データを単一のターゲットファイルヘッディングするには、ラウンドロビンパーティション化を選択する必要があります。 |         |         |        |

## Oracle へのマルチバイトデータのロード

マルチバイトデータを Oracle へロードするときは、固定長ファイルのデータ精度はバイト単位、区切りファイルのデータ精度は文字単位です。ターゲットテーブルのカラムの幅が、すべてのデータを格納するのに十分かどうか確認してください。

Oracle は、Nchar などの文字単位のデータ型をサポートしています。その場合、精度は文字単位になります。Nchar データ型を使用する場合は、最大文字数に K を掛けます。K は、選択したターゲットコードページで 1 文字に含まれる最大のバイト数です。これにより、Integration Service ではターゲットファイルをロードする前にデータを切り詰めることがなくなります。

## Oracle 外部ローダーの属性の設定

以下の表に、Oracle 外部ローダー接続の属性を示します。

| 属性                         | デフォルト値        | 説明                                                                                                                                                                                                                                                    |
|----------------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| エラー数の上限値                   | 1             | 外部ローダーがロード操作を中止するまでに許可されるエラー数。                                                                                                                                                                                                                        |
| ロードモード                     | 付加            | 外部ローダーがデータをロードするときに使用するロードモード。次のいずれかのロードモードを選択します。 <ul style="list-style-type: none"><li>- 付加</li><li>- 挿入</li><li>- 置換</li><li>- 切り詰め</li></ul>                                                                                                      |
| ロード方法                      | 通常パスを使用       | 外部ローダーがデータをロードするときに使用する方式。次のいずれかのロード方式を選択します。 <ul style="list-style-type: none"><li>- 通常パスを使用</li><li>- ダイレクトパスを使用（リカバリ可能）。</li><li>- ダイレクトパスを使用（リカバリ可能）。</li></ul>                                                                                   |
| パラレルロードを有効にする              | パラレルロードを有効にする | Oracle 外部ローダーが、パーティション化された Oracle ターゲットテーブルに並行してデータをロードするかどうかを決定します。 <ul style="list-style-type: none"><li>- パーティション化されたターゲットにロードする場合は、[パラレルロードを有効にする] を選択します。</li><li>- パーティション化されていないターゲットにロードする場合は、[Do Not Enable Parallel Load] を選択します。</li></ul> |
| Rows Per Commit            | 10000         | [ダイレクトパスを使用] ロード方式の場合、この属性はロード操作に対するバインド配列内の行数を指定します。[Use Direct Path] ロード方式の場合、この属性は、外部ローダーがデータベースにデータを保存する前にターゲットフラットファイルから読み込む行数を指定します。                                                                                                            |
| External Loader Executable | sqlload       | 外部ローダー実行ファイルの名前。                                                                                                                                                                                                                                      |
| ログファイル名                    | なし            | 外部ローダーログファイルのパスと名前。                                                                                                                                                                                                                                   |
| ステージング済み                   | 無効            | データのロード方法。データベースへのロード前に、フラットファイルのステージング領域にデータをロードするには、[Is Staged] を選択します。それ以外の場合、データは名前付きパイプを使用してデータベースにロードされます。                                                                                                                                      |

## Sybase IQ へのロード

Sybase IQ にロードする際は、Sybase IQ 外部ローダーを使用して挿入操作を実行します。Integration Service では、マルチバイトデータを Sybase IQ ターゲットにロードすることができます。Integration Service では、Sybase IQ サーバーが Integration Service と同じマシン上にある場合、または異なるマシン上にある場合でもフラットファイルに書き込むことができます。Integration Service が Sybase IQ データベ

スサーバーに対してローカルである場合、Integration Service では名前付きパイプに書き込むことができません。

## Sybase IQ 外部ローダーに関するルールおよびガイドライン

外部ローダーを使用して Sybase IQ にロードする場合には、以下の規則およびガイドラインに従ってください。

- ターゲットテーブルがプライマリキー制約に違反していないことを確認する必要があります。
- Sybase IQ 外部ローダーを使用するには、読み込み/書き込みアクセス権を持つ Sybase IQ ユーザーを事前に設定しておかなければなりません。
- Sybase IQ 外部ローダーのターゲットフラットファイルは、固定長または区切りファイルです。
- Sybase IQ 外部ローダーはターゲットに対して更新操作や削除操作を実行できません。
- 複数のパーティションを伴うセッションについては、ラウンドロビンのパーティションタイプを使用して、データを単一のターゲットファイルにルーティングします。
- Integration Service および Sybase IQ サーバーが異なるマシン上にある場合は、Integration Service をホストするマシンから Sybase IQ サーバーをホストするマシンへドライブをマッピングまたはマウントします。

## Sybase IQ へのマルチバイトデータのロード

マルチバイトデータを Sybase IQ ターゲットにロードする場合は、以下のガイドラインを使用します。

### 区切りフラットファイルターゲット

区切りフラットファイルの場合、データ精度は文字単位です。ターゲットにマルチバイト文字データを挿入する場合、マルチバイトデータ用に追加の精度を確保する必要はありません。Sybase IQ では、オプションの引用符を使用できません。区切りターゲットフラットファイルを持っている場合は、[引用符] に [なし] を選択する必要があります。

マルチバイトデータを Sybase IQ にロードする場合、NULL キャラクタおよび区切り文字はそれぞれ 4 バイトまでにすることができます。区切り文字が通常の文字として読み取られないようにするために、区切り文字の各バイトは 0x40 未満の ASCII 値でなければなりません。

### 固定幅フラットファイルターゲット

固定長フラットファイルの場合、データ精度は文字単位ではなくバイト単位です。固定長フラットファイルターゲットにマルチバイトデータをロードする場合は、マルチバイトデータを許容できるように精度を設定します。精度が小さくてマルチバイトデータに対応できない場合、Integration Service により、行が拒否ファイルに書き込まれます。

## Sybase IQ 外部ローダーの属性の設定

PowerCenter では、Sybase IQ に外部ローダー接続タイプを使用します。Sybase IQ データベースのログイン資格情報には接続文字列属性を指定します。

Sybase IQ 15.x の接続文字列には、以下の属性を含める必要があります。

`uid=user ID; pwd=password; eng=Sybase IQ database server name`

例えば、次の接続文字列を使用できます。

`uid=qasrvr65;pwd=qasrvr65;eng=SUNQA2SybaseIQ`

**注:** 接続文字列に引用符を使用すると、セッションが失敗する場合があります。

以下の表に、Sybase IQ 外部ローダー接続の属性を示します。

| 属性                         | デフォルト値                                                   | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Block Factor               | 10000                                                    | ターゲット Sybase テーブル内の 1 ブロックあたりのレコード数。外部ローダーは、バージョン 15.x までの Sybase IQ に限り、固定長フラットファイルターゲットのロード操作に Block Factor 属性を適用します。                                                                                                                                                                                                                                                                                                                                                                                               |
| ブロックサイズ                    | 50000                                                    | Sybase データベース操作に使用されるブロックのサイズ。外部ローダーは、バージョン 15.x までの Sybase IQ に限り、区切りフラットファイルターゲットのロード操作に Block Size 属性を適用します。                                                                                                                                                                                                                                                                                                                                                                                                       |
| Checkpoint                 | 有効                                                       | 利用可能にすると、Sybase IQ データベースはテーブルを正常にロードしたあとでチェックポイントを発行します。利用不可にすると、データベースはチェックポイントを発行しません。                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Notify Interval            | 1000                                                     | Sybase IQ 外部ローダーが外部ローダーログにステータスメッセージを書き込む前にロードする行数。                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Server Datafile Directory  | なし                                                       | <p>統合サービスをホストするマシンからアクセスできる、Sybase IQ サーバーの場所。ターゲットファイルの場所を指定すると、Sybase IQ サーバーはそのファイルのアクセスに失敗することがあります。</p> <p>データベースサーバーのインストールディレクトリに対してこの属性を指定します。</p> <p>ディレクトリが Windows システムにある場合は、ディレクトリパスにバックスラッシュ (\) を使用します。</p> <p>D:\mydirectory\inputfile.out</p> <p>ディレクトリが UNIX システムにある場合は、ディレクトリパスにフォワードスラッシュ (/) を使用します。</p> <p>/mydirectory/inputfile.out</p> <p>データベースサーバーインストールをホストするマシンに対応する構文を使用して、ディレクトリパスを入力します。たとえば Integration Service が Windows マシン上にあり、Sybase IQ サーバが UNIX マシン上にある場合は、UNIX の構文を使用します。</p> |
| External Loader Executable | Sybase 15.x: dbisql - host<hostname> - port<port number> | Sybase IQ 外部ローダー実行ファイルの名前。Sybase IQ 外部ローダー接続を作成する場合、Workflow Manager は外部ローダー実行ファイルの名前をデフォルトで dbisql に設定します。別の名前の実行ファイルを使用する場合は、[外部ローダー実行可能] フィールドを更新する必要があります。外部ローダー実行ファイルのディレクトリがシステムパスに含まれていない場合は、このフィールドにファイルのパスとファイル名を入力する必要があります。                                                                                                                                                                                                                                                                                |
| Is Staged                  | 有効                                                       | データのロード方法。データベースへのロード前に、フラットファイルのステージング領域にデータをロードするには、[Is Staged] を選択します。名前付きパイプからデータをロードするには、この属性をクリアします。統合サービスが Sybase IQ データベースに対してローカルな場合、統合サービスは名前付きパイプに書き込むことができます。                                                                                                                                                                                                                                                                                                                                             |

# Teradata へのロード

Teradata ターゲットにロードする場合、次のいずれかの外部ローダーを使用します。

- **Multiload**. 大量の増分ロードのための挿入、更新、削除および upsert 操作を行います。単一のパーティションでセッションを実行する際にこのローダーを使用します。マルチロードによりテーブルレベルのロックが取得されるので、オフラインローディングに適しています。
- **TPump**. 比較的少量の変更のための挿入、更新、削除および upsert 操作を行います。複数のパーティションでセッションを実行する際にこのローダーを使用します。TPump によってテーブルに関する行ハッシュロックが取得されるので、TPump がテーブルにロードしているときも他のユーザーはそのテーブルにアクセスできます。
- **FastLoad**. 大量の初期ロードのための挿入、または大量の切り詰めおよび再ロード操作のための挿入を行います。単一のパーティションでセッションを実行する際にこのローダーを使用します。セカンダリインデックスが定義されていない空のターゲットテーブルでこのローダーを使用します。

Teradata 外部ローダーを使用して更新または upsert 操作を行う場合、Mapping Designer の「ターゲット更新の上書き」オプションを使用して、外部ローダー制御ファイルの UPDATE 文をオーバーライドします。upsert では、外部ローダー制御ファイルの INSERT 文は変更されません。

## Teradata 外部ローダーに関するルールおよびガイドライン

外部ローダーを使用して Teradata にロードする際には、以下のルールおよびガイドラインを使用します。

- Integration Service では、Teradata 外部ローダーを使用して固定長フラットファイルおよび区切りフラットファイルを Teradata データベースにロードすることができます。すべての Teradata ロードは改行 (\n) 文字を使用して個々のレコードを区切るため、Teradata ロードの区切り文字として改行文字を使用することはできません。
- セッションに 1 つのパーティションが含まれる場合、ターゲット出力ファイル名は、拡張子を含めて最大 27 文字です。セッションに複数のパーティションが含まれる場合、ターゲット出力ファイル名は、拡張子を含めて最大 25 文字です。
- Teradata 外部ローダーを使用してバイナリデータをロードすることはできません。
- 名前付きパイプを使用して Teradata にロードする場合、チェックポイント操作が外部ローダーによって実行されないように、チェックポイントの値を 0 に設定してください。
- 使用しているローダーに従って、エラーテーブル名、ログテーブル名、または作業テーブル名を指定できます。エラーデータベース名、ログデータベース名、または作業データベース名も指定できます。
- セッションプロパティで制御ファイルを上書きできます。
- Teradata を使用する場合、制御ファイルにパスワードが出力されないように、データベースのパスワードとして PmNullPasswd を入力することができます。代わりに、Integration Service により制御ファイルにパスワード用に空の文字列が書き込まれます。

## 制御ファイルのオーバーライド

セッションのローダー接続を編集するときに、制御ファイルを上書きできます。ローダー接続で編集できない一部のローダープロパティを変更したいときなどに、制御ファイルをオーバーライドできます。たとえば、制御ファイル中のトレースオプションを設定できます。

制御ファイルを上書きするときは、Workflow Manager が制御ファイルをリポジトリに保存します。制御ファイルの属性をクリアするまで、Integration Service では、セッションの実行時および各後続セッションの実行時に保存された制御ファイルが使用されます。制御ファイルを編集した後にターゲットまたはローダー接続設定を変更した場合は、これらの変更は制御ファイルに含まれません。これらの変更を含める場合は、制御ファイルを再度生成して編集する必要があります。



制御ファイルを上書きしない場合、セッションを実行するたびに、Integration Service によってセッションプロパティとローダープロパティに基づいて、新規制御ファイルが生成されます。Integration Service によって、制御ファイルが出力ファイルディレクトリに生成されます。セッションが実行されるたびに制御ファイルを上書きします。

**注:** Workflow Manager では、制御ファイルの構文は検証しません。セッションの実行時に Teradata が制御ファイルの構文を検証します。制御ファイルが無効だと、セッションは失敗に終わります。

編集済みの制御ファイルを表示するには、制御ファイルエディタを開きます。

制御ファイルを上書きするには：

1. Workflow Manager で、セッションプロパティを開きます。
2. [マッピング] タブをクリックし、[トランスフォーメーション] ビューを開きます。
3. [ターゲット] ノードをクリックします。
4. [接続] 設定の [値] フィールドで、[変更] をクリックします。
5. [Control File Content Override (制御ファイルの内容の上書き)] フィールドで [開く] をクリックします。  
[Control File Editor (制御ファイルエディタ)] ダイアログボックスが表示されます。
6. [生成] をクリックします。  
Workflow Manager はセッションプロパティとローダープロパティに基づいて制御ファイルを生成します。
7. 生成された制御ファイルを編集し、[OK] をクリックして変更を保存します。

## 制御ファイルでのユーザー変数の作成

MultiLoad または TPump 外部ローダー属性を設定する場合は、ユーザー変数を作成できます。ユーザー変数は、制御ファイルで使用するカスタム定義された置き換え変数です。ユーザー変数は、接続オブジェクト属性で利用できない可能性があるセッション固有の情報を取得します。多くの場合、ユーザー変数はロード前またはロード後の処理に使用されます。

ユーザー変数名および置き換え値は、接続オブジェクト内に定義します。制御ファイルで、置換変数の接頭語とユーザー変数名を対応するコマンドに追加します。セッションを実行すると、Integration Service は制御ファイル内の置換変数の接頭語とユーザー変数名を、置き換え値に置き換えます。制御ファイルを編集した後に置き換え値を変更した場合、制御ファイルは新しい値を使用します。

ユーザー変数を作成する場合には、次の規則およびガイドラインに従ってください。

- ユーザー変数を作成する場合は、以下の構文を使用します。  
`<User_Variable_Name>=<Substitution_Value>`
- ユーザー変数名や置き換え値にスペースを含めると、セッションは失敗します。
- 制御ファイルにユーザー変数を追加する場合は、以下の構文を使用します。  
`:CF.<User_Variable_Name>`

### 例

Integration Service がターゲットにデータをロードした後、システム日付を出力ファイルに表示できます。接続オブジェクトでは、以下のユーザー変数を設定します。

```
OutputFileName=output_file.txt
```

制御ファイルで、以下のように設定します。

```
DISPLAY '&SYSDATE' TO FILE ':CF.OutputFileName'
```

セッションを実行すると、Integration Service は「:CF.OutputFileName」を制御ファイルの「output\_file.tx」に置き換えます。



## Teradata MultiLoad 外部ローダーの属性の設定

MultiLoad 外部ローダーで作業する場合は、以下の規則およびガイドラインに従ってください。

- ターゲット上での挿入、更新、削除および更新挿入操作を行います。また、データドリブンモードを使用し、アップデイトストラテジトランスフォーメーションまたはカスタムトランスフォーメーションに基づいて、挿入、更新、および削除操作を実行することができます。
- 複数のパーティションを伴うセッションについては、ラウンドロビンのパーティションタイプを使用して、データを単一のターゲットファイルにルーティングします。
- データベースで許可されている同時実行セッションの最大数を超える数のセッションを起動すると、セッションがハングすることがあります。[Tenacity] および [Sleep] に最小値を設定すると、セッションがハングではなく、失敗するようにできます。

Teradata MultiLoad 外部ローダーの属性を設定するには、[接続] - [ローダー] をクリックし、[タイプ] を選択し、[編集] をクリックします。

以下の表に、Teradata MultiLoad 外部ローダーに対して設定する属性を示します。

| 属性       | デフォルト値 | 説明                                                                                                                                                                                                                                               |
|----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TDPID    | なし     | Teradata データベース ID。                                                                                                                                                                                                                              |
| データベース名  | なし     | ベース名（オプション）。データベース名を指定しない場合、Integration Service によって、マッピングで定義したターゲットテーブルデータベース名が使用されます。                                                                                                                                                          |
| 日付形式     | なし     | 日付形式。接続オブジェクトの日付形式は、ターゲット定義で定義する日付形式と一致する必要があります。Integration Service は、次の日付形式をサポートします。 <ul style="list-style-type: none"><li>- DD/MM/YYYY</li><li>- MM/DD/YYYY</li><li>- YYYY/DD/MM</li><li>- YYYY/MM/DD</li></ul>                               |
| エラー数の上限値 | 0      | MultiLoad が MultiLoad エラーテーブルに書き込めるリジェクトレコードの合計数。一意性違反は、リジェクトレコードとして数えられません。<br>エラー制限が 0 の場合は、リジェクトされるレコード数に制限がないことを意味します。                                                                                                                       |
| チェックポイント | 10,000 | チェックポイント間の間隔。間隔は、次の値に設定できます。 <ul style="list-style-type: none"><li>- 60 以上。MultiLoad は、このレコード数を処理するごとにチェックポイント操作を実行します。</li><li>- 1-59。MultiLoad は、指定された間隔（単位：分）でチェックポイント操作を実行します。</li><li>- 0。MultiLoad は、インポート作業中にチェックポイント操作を実行しません。</li></ul> |
| Tenacity | 10,000 | MultiLoad が必要なセッションにログインしようと試みる時間（単位：時間）。ログインが失敗した場合、MultiLoad は [Sleep] 属性に指定された時間（単位：分）だけ待ってから、ログインを再試行します。<br>MultiLoad は、ログインが成功するか、または [Tenacity] 属性に指定された時間が経過するまで、ログインの試行を繰り返します。                                                         |

| 属性                         | デフォルト値 | 説明                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ロードモード                     | Upsert | <p>SQL コマンドを生成するモード。Insert、Delete、Update、Upsert、Data Driven のいずれかを選択します。</p> <p>データドリブンのロードを選択すると、Integration Service は、アップデートストラテジまたはカスタムトランスフォーメーションの指示に従って、挿入、削除、または更新の行にフラグをたてるか決定します。Integration Service はアップデートストラテジを示すカラムをターゲットファイルまたは名前付きパイプに出力します。制御ファイルでは、これらの値を使用しターゲットに対してどのようにデータをロードするか決定します。Integration Service では、以下の値を使用してアップデートストラテジを指定します。</p> <p>0 - 挿入<br/>1 - 更新<br/>2 - 削除</p> |
| Drop Error Tables          | 有効     | 次のセッションを開始する前に MultiLoad エラーテーブルを削除します。テーブルを削除するにはこのオプションを選択します。                                                                                                                                                                                                                                                                                                                                      |
| External Loader Executable | mload  | Teradata 外部ローダ実行ファイルのファイル名と必要に応じてファイルパス。外部ローダ実行ファイルのディレクトリがシステムパスに含まれていない場合は、絶対パスを入力する必要があります。                                                                                                                                                                                                                                                                                                        |
| 最大セッション数                   | 1      | <p>MultiLoad ジョブ 1 つあたりの MultiLoad セッションの最大数。[Max Sessions] は 1~32,767 までの値でなければなりません。</p> <p>複数の MultiLoad セッションを実行すると、クライアントとデータベースが使用するリソースの量が増えます。そのため、この値を小さな数に設定すると、パフォーマンスが向上します。</p>                                                                                                                                                                                                          |
| スリープ                       | 6      | <p>MultiLoad がログインを再試行するまでの待ち時間（単位：分）。MultiLoad は、ログインが成功するか、または [Tenacity] 属性に指定された時間が経過するまで、ログインの試行を繰り返します。</p> <p>[Sleep] には 0 より大きな値を指定する必要があります。0 を指定すると、MultiLoad はエラーメッセージを発行し、デフォルトの値（6 分）を使用します。</p>                                                                                                                                                                                         |
| ステージング済み                   | 無効     | データのロード方法。データベースへのロード前に、フラットファイルのステージング領域にデータをロードするには、[Is Staged] を選択します。それ以外の場合、データは名前付きパイプを使用してデータベースにロードされます。                                                                                                                                                                                                                                                                                      |
| Error Database             | なし     | Error Database Name この属性を使用して、デフォルトのエラーデータベース名を上書きします。データベース名を指定しない場合、Integration Service によって、ターゲットテーブルデータベースが使用されます。                                                                                                                                                                                                                                                                                |
| ワークテーブルデータベース              | なし     | 作業テーブルデータベースの名前。この属性を使用して、デフォルトの作業テーブルデータベース名を上書きします。データベース名を指定しない場合、Integration Service によって、ターゲットテーブルデータベースが使用されます。                                                                                                                                                                                                                                                                                 |

| 属性           | デフォルト値 | 説明                                                                                                                    |
|--------------|--------|-----------------------------------------------------------------------------------------------------------------------|
| ログテーブルデータベース | なし     | ログテーブルデータベースの名前。この属性を使用して、デフォルトのログテーブルデータベース名を上書きします。データベース名を指定しない場合、Integration Service によって、ターゲットテーブルデータベースが使用されます。 |
| ユーザー変数       | なし     | デフォルトの制御ファイルで使用するユーザー定義変数。                                                                                            |

以下の表に、セッションプロパティで Teradata MultiLoad 外部ローダー接続オブジェクトを上書きする際に設定する属性を示します。

| 属性         | デフォルト値 | 説明                                                                                                                          |
|------------|--------|-----------------------------------------------------------------------------------------------------------------------------|
| エラーテーブル 1  | なし     | 最初のエラーテーブルのテーブル名。この属性を使用して、デフォルトのエラーテーブル名を上書きします。エラーテーブル名を指定しない場合、Integration Service により ET_<target_table_name> が使用されます。   |
| エラーテーブル 2  | なし     | 2 番目のエラーテーブルのテーブル名。この属性を使用して、デフォルトのエラーテーブル名を上書きします。エラーテーブル名を指定しない場合、Integration Service により、UV_<target_table_name> が使用されます。 |
| ワークテーブル    | なし     | 作業テーブル名は、デフォルトの作業テーブル名を上書きします。作業テーブル名を指定しない場合、Integration Service により WT_<target_table_name> が使用されます。                       |
| ログテーブル     | なし     | ログテーブル名は、デフォルトのログテーブル名を上書きします。ログテーブル名を指定しない場合、Integration Service により ML_<target_table_name> が使用されます。                       |
| 制御ファイルの上書き | なし     | 制御ファイルのテキストこの属性を使用して、Integration Service が Teradata へロードするときの制御ファイルを上書きします。                                                 |

## Teradata TPump 外部ローダーの属性の設定

ターゲット上での挿入、更新、削除および更新挿入操作を行えます。また、データドリブンモードを使用し、アップデートストラテジトランスフォーメーションまたはカスタムトランスフォーメーションに基づいて、挿入、更新、および削除操作を実行することができます。

複数のパーティションを含むセッションを実行する場合は、各パーティションについて Teradata TPump 外部ローダーを選択してください。

Teradata TPump 外部ローダーの属性を設定するには、[接続] - [ローダー] をクリックし、[タイプ] を選択し、[編集] をクリックします。

以下の表に、Teradata TPump 外部ローダーに対して設定する属性を示します。

| 属性                         | デフォルト値 | 説明                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TDPID                      | なし     | Teradata データベース ID。                                                                                                                                                                                                                                                                                                                                                                         |
| データベース名                    | なし     | ベース名（オプション）。データベース名を指定しない場合、Integration Service によって、マッピングで定義したターゲットテーブルデータベース名が使用されます。                                                                                                                                                                                                                                                                                                     |
| エラー数の上限値                   | 0      | エラーとして拒否される行数を制限します。エラー制限を超えると、TPump は、最後のエラーが発生したトランザクションをロールバックします。エラー制限が 0 の場合は、TPump はエラーが 1 回発生しただけで処理を中止します。                                                                                                                                                                                                                                                                          |
| チェックポイント                   | 15     | チェックポイント間の時間間隔（単位：分）。チェックポイントは、0～60 の範囲で設定する必要があります。                                                                                                                                                                                                                                                                                                                                        |
| Tenacity                   | 4      | TPump が必要なセッションにログインしようと試みる時間（単位：時間）。ログインが失敗した場合、TPump は [Sleep] 属性に指定された時間（単位：分）だけ待ってから、ログインを再試行します。TPump は、ログインが成功するか、または [Tenacity] 属性に指定された時間が経過するまで、ログインの試行を繰り返します。[Tenacity] を無効にするには、値を 0 に設定します。                                                                                                                                                                                    |
| ロードモード                     | Upsert | SQL コマンドを生成するモード。Insert、Delete、Update、Upsert、Data Driven のいずれかを選択します。<br><br>データドリブンのロードを選択すると、Integration Service は、アップデイトストラテジまたはカスタムトランスフォーメーションの指示に従って、挿入、削除、または更新の行にフラグをたてるか決定します。Integration Service はアップデイトストラテジを示すカラムをターゲットファイルまたは名前付きパイプに出力します。制御ファイルでは、これらの値を使用し、データベースに対してどのようにデータをロードするかを決定します。Integration Service では、以下の値を使用してアップデイトストラテジを指定します。<br>0 - 挿入<br>1 - 更新<br>2 - 削除 |
| Drop Error Tables          | 有効     | 次のセッションを開始する前に TPump エラーテーブルを削除します。テーブルを削除するにはこのオプションを選択します。                                                                                                                                                                                                                                                                                                                                |
| External Loader Executable | tpump  | Teradata 外部ローダ実行ファイルのファイル名と必要に応じてファイルパス。外部ローダ実行ファイルのディレクトリがシステムパスに含まれていない場合は、絶対パスを入力する必要があります。                                                                                                                                                                                                                                                                                              |
| 最大セッション数                   | 1      | TPump ジョブ 1 つあたりの TPump セッションの最大数。セッション内の各パーティションは、固有の TPump ジョブを開始します。複数の TPump セッションを実行すると、クライアントとデータベースが使用するリソースの量が増えます。そのため、この値を小さな数に設定すると、パフォーマンスが向上します。                                                                                                                                                                                                                               |
| スリープ                       | 6      | TPump がログインを再試行するまでの待ち時間（単位：分）。TPump は、ログインが成功するか、または [Tenacity] 属性に指定された時間が経過するまで、ログインの試行を繰り返します。                                                                                                                                                                                                                                                                                          |

| 属性             | デフォルト値 | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Packing Factor | 20     | 各セッションバッファが保持する行数。パッキングにより、ターゲットフラットファイルと Teradata データベースの間での送受信の回数が減るため、ネットワークおよびチャネルの効率が向上します。                                                                                                                                                                                                                                                                                                                                               |
| Statement Rate | 0      | TPump 実行ファイルが Teradata データベースに文を送信する初期最大速度（毎分）。この属性を 0 に設定した場合、速度は不定になります。                                                                                                                                                                                                                                                                                                                                                                     |
| Serialize      | 無効     | <p>特定のキーの組み合わせ（行）に対する操作が連続的に行われるかどうかを指定します。</p> <p>TPump ジョブで 1 つの行に対して複数の変更がある場合は、このオプションを有効にしてください。キー範囲が同じでフィルタ条件が異なる複数のパーティションがあるセッションでは、1 つの行に対して複数の変更が行われる場合があります。この場合（特に [Pack] 属性が 1 より大きな値に設定されている場合）には、[Serialize] を有効にして、Teradata データベースのロック競合を避けた方がよいでしょう。</p> <p>[Serialize] を有効にした場合、Integration Service は、ターゲットテーブルにキーカラムとして指定されているプライマリーキーを使用します。ターゲットテーブルにプライマリーキーがない場合には、このオプションをクリアするか、制御ファイルのデータレイアウトセクションでキーカラムを指定します。</p> |
| Robust         | 無効     | [Robust] を選択しない場合、TPump は単純なリスタートロジックを使用します。この場合、リスタートロジックにより TPump は最後のチェックポイントから処理を開始します。TPump は、チェックポイント後にロードされたデータがあれば、そのデータを再ロードします。この方法では、堅牢なロジックでの追加のデータベース書き込みによる余分なオーバーヘッドは生じません。                                                                                                                                                                                                                                                     |
| No Monitor     | 有効     | この属性を選択すると、TPump は、TPump モニタアプリケーションからの文速度の変更、または TPump モニタアプリケーションの更新ステータス情報をチェックしないようになります。                                                                                                                                                                                                                                                                                                                                                  |
| ステージング済み       | 無効     | データのロード方法。データベースへのロード前に、フラットファイルのステージング領域にデータをロードするには、[Is Staged] を選択します。それ以外の場合、データは名前付きパイプを使用してデータベースにロードされます。                                                                                                                                                                                                                                                                                                                               |
| Error Database | なし     | Error Database Name この属性を使用して、デフォルトのエラーデータベース名を上書きします。データベース名を指定しない場合、Integration Service によって、ターゲットテーブルデータベースが使用されます。                                                                                                                                                                                                                                                                                                                         |
| ログテーブルデータベース   | なし     | ログテーブルデータベースの名前。この属性を使用して、デフォルトのログテーブルデータベース名を上書きします。データベース名を指定しない場合、Integration Service によって、ターゲットテーブルデータベースが使用されます。                                                                                                                                                                                                                                                                                                                          |
| ユーザー変数         | なし     | デフォルトの制御ファイルで使用するユーザー定義変数。                                                                                                                                                                                                                                                                                                                                                                                                                     |

以下の表に、セッションプロパティで Teradata TPump 外部ローダー接続オブジェクトをオーバーライドする際に設定する属性を示します。

| 属性         | デフォルト値 | 説明                                                                                                                                 |
|------------|--------|------------------------------------------------------------------------------------------------------------------------------------|
| エラーテーブル    | なし     | エラーテーブル名。この属性を使用して、デフォルトのエラーテーブル名を上書きします。エラーテーブル名を指定しない場合、Integration Service により ET_<target_table_name><partition_number>が使用されます。 |
| ログテーブル     | なし     | ログテーブル名この属性を使用して、デフォルトのログテーブル名を上書きします。ログテーブル名を指定しない場合、Integration Service により TL_<target_table_name><partition_number>が使用されます。     |
| 制御ファイルの上書き | なし     | 制御ファイルのテキストこの属性を使用して、Integration Service が Teradata ヘロードするときの制御ファイルを上書きします。                                                        |

## Teradata FastLoad 外部ローダーの属性の設定

FastLoad 外部ローダーで作業する場合は、以下のガイドラインに従ってください。

- 各 FastLoad ジョブは、データを 1 つの Teradata データベーステーブルにロードします。FastLoad を使用して複数のテーブルにデータをロードさせるには、複数の FastLoad ジョブを作成する必要があります。
- 複数のパーティションを伴うセッションについては、ラウンドロビンのパーティションタイプを使用して、データを単一のターゲットファイルにルーティングします。
- 二次インデックスの定義されていない空のターゲットテーブルである必要があります。
- FastLoad は、ターゲットテーブルにプライマリキーが存在する場合、出力ファイルの重複行を Teradata データベースのターゲットテーブルにロードしません。
- 日付の値をターゲットテーブルにロードする場合、ターゲットテーブルのカラムに「YYYY-MM-DD」の形式で日付形式を設定する必要があります。
- FastLoad を使用してバイナリデータをロードすることはできません。
- 区切り文字としてカンマ (,)、タブ (\t)、およびパイプ (|) を使用できます。

Teradata FastLoad 外部ローダーに属性を設定するには、[接続] - [ローダー] をクリックし、[タイプ] を選択し、[編集] をクリックします。

以下の表に、Teradata FastLoad 外部ローダーに対して設定する属性を示します。

| 属性       | デフォルト値    | 説明                                              |
|----------|-----------|-------------------------------------------------|
| TDPID    | なし        | Teradata データベース ID。                             |
| データベース名  | なし        | データベース名。                                        |
| エラー数の上限値 | 1,000,000 | FastLoad がデータベーステーブルへのロードを中止するまでにリジェクトできる行の最大数。 |

| 属性                         | デフォルト値   | 説明                                                                                                                                                                                                                                                                                                                                       |
|----------------------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| チェックポイント                   | 0        | チェックポイントとチェックポイントとの間に Teradata データベースへ送信される行数。FastLoad ジョブの実行中に処理が停止した場合、最近のチェックポイントからジョブをリスタートできます。<br>「0」を入力した場合、FastLoad はチェックポイント操作を実行しません。                                                                                                                                                                                          |
| Tenacity                   | 4        | 最大数のロードジョブが Teradata データベースで既に実行されている場合に、必要な FastLoad セッションに対して FastLoad がログインの試行を繰り返す時間の長さです（単位：時間）。FastLoad は、新しいセッションへのログイン時に、最大数のロードセッションが Teradata データベースで既に実行されていることが分かった場合、ログインされた新しいセッションをすべてログオフさせ、[Sleep] 属性に指定された時間（単位：分）だけ待ってから、ログインを再度試みます。FastLoad は、必要な数のセッションにログインできるまで、または [Tenacity] 属性に指定された時間が経過するまで、ログインの試行を繰り返します。 |
| Drop Error Tables          | 有効       | 次のセッションを開始する前に FastLoad エラーテーブルを削除します。前のジョブからのデータがあるエラーテーブルが存在する場合、FastLoad は実行されません。<br>テーブルを削除するにはこのオプションを選択します。                                                                                                                                                                                                                       |
| External Loader Executable | fastload | Teradata 外部ローダ実行ファイルのファイル名と必要に応じてファイルパス。外部ローダ実行ファイルのディレクトリがシステムパスに含まれていない場合は、絶対パスを入力する必要があります。                                                                                                                                                                                                                                           |
| 最大セッション数                   | 1        | FastLoad ジョブ 1 つあたりの FastLoad セッションの最大数。[最大セッション数] は 1 以上で、さらにシステムのアクセスモジュールプロセス (AMP) の合計数以下でなければなりません。                                                                                                                                                                                                                                 |
| スリープ                       | 6        | FastLoad がログインを再試行するまでの待ち時間（単位：分）。FastLoad は、ログインが成功するか、または [Tenacity] 属性に指定された時間が経過するまで、ログインの試行を繰り返します。                                                                                                                                                                                                                                 |
| ターゲットテーブルの切り詰め             | 無効       | FastLoad ジョブを開始する前に、ターゲットデータベーステーブルを切り詰めます。FastLoad では、空でないテーブルにデータをロードすることはできません。                                                                                                                                                                                                                                                       |
| ステー징済み                     | 無効       | データのロード方法。データベースへのロード前に、フラットファイルのステー징領域にデータをロードするには、[Is Staged] を選択します。それ以外の場合、データは名前付きパイプを使用してデータベースにロードされます。                                                                                                                                                                                                                           |
| Error Database             | なし       | Error Database Name この属性を使用して、デフォルトのエラーデータベース名を上書きします。データベース名を指定しない場合、Integration Service によって、ターゲットテーブルデータベースが使用されます。                                                                                                                                                                                                                   |



以下の表に、セッションプロパティで Teradata FastLoad 外部ローダー接続オブジェクトをオーバーライドする際に設定する属性を示します。

| 属性         | デフォルト値 | 説明                                                                                                                |
|------------|--------|-------------------------------------------------------------------------------------------------------------------|
| エラーテーブル 1  | なし     | 最初のエラーテーブルのテーブル名は、デフォルトのエラーテーブル名を上書きします。エラーテーブル名を指定しない場合、Integration Service により ET_<target_table_name>が使用されます。   |
| エラーテーブル 2  | なし     | 2 番目のエラーテーブルのテーブル名は、デフォルトのエラーテーブル名を上書きします。エラーテーブル名を指定しない場合、Integration Service により、UV_<target_table_name>が使用されます。 |
| 制御ファイルの上書き | なし     | 制御ファイルのテキストこの属性を使用して、Integration Service が Teradata へロードするときの制御ファイルを上書きします。                                       |

## セッション内での外部データのロードの設定

セッション内で外部データのロードを設定する前に、Workflow Manager で外部ローダー接続を作成し、外部ローダーの属性を設定する必要があります。

外部ローダーをセッションに使用するには、次の手順を実行します。

1. リレーショナルデータベースではなく、フラットファイルに書き込むように、セッションを設定します。
2. ファイルプロパティを設定します。
3. セッションプロパティで外部ローダー接続を選択します。

### ファイルに書き込むようにセッションを設定

外部ローダを使用するには、ターゲットデータベースタイプに応じてマッピングにターゲット定義を作成します。セッションは、デフォルトでリレーショナルターゲットタイプを設定します。外部ローダ接続を選択するには、セッションがリレーショナルターゲットではなくファイルに書き込むように設定する必要があります。ファイルに書き込むようにセッションを設定するには、writer のタイプをリレーショナル writer からファイル writer に変更します。ライタのタイプは、[マッピング] タブの [Writers] 設定を使用して変更します。

ターゲットの writer のタイプを変更するには、ターゲットインスタンスを選択し、writer のタイプを [リレーショナル Writer] から [ファイル Writer] に変更します。

### ファイルプロパティの設定

ファイルに書き込むようにセッションを設定したあとは、ファイルプロパティを設定できます。出力ファイル名およびディレクトリと、拒否ファイル名およびディレクトリを指定する必要があります。これらのプロパティは、[マッピング] タブの [プロパティ] 設定で設定します。ファイルプロパティを設定するには、ターゲットインスタンスを選択します。



以下の表に、[プロパティ] 設定の属性を示します。

| 属性           | 説明                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 出力ファイルディレクトリ | 出力ファイルディレクトリの名前およびパス。このフィールドにはディレクトリ名を入力します。デフォルトでは、ディレクトリ\$PMTargetFileDir に出力ファイルが書き込まれます。<br>[出力ファイル名] フィールドに完全なディレクトリおよびファイル名を入力する場合は、このフィールドをクリアします。出力ファイルのパスにダブルスペースを使用した場合、外部ローダセッションが失敗する場合があります。                                                                                                                                                                                                                |
| 出力ファイル名      | 出力ファイルの名前。ファイル名、またはファイル名とディレクトリを入力します。デフォルトでは、ターゲットファイルには、マッピングで使用されているターゲット定義に基づいた名前が付けられます (<ターゲット名>.out)。出力ファイルのパスにダブルスペースを使用した場合、外部ローダセッションが失敗する場合があります。                                                                                                                                                                                                                                                             |
| 拒否ファイルディレクトリ | 拒否ファイルディレクトリの名前およびパス。デフォルトでは、統合サービスは拒否ファイルをすべてディレクトリ\$PMBadFileDir に書き込みます。<br>[拒否ファイル名] フィールドに完全なディレクトリおよびファイル名を入力する場合は、このフィールドをクリアします。                                                                                                                                                                                                                                                                                 |
| 拒否ファイル名      | 拒否ファイルの名前。ファイル名、またはファイル名とディレクトリを入力します。統合サービスはこのフィールドの情報を、[拒否ファイルディレクトリ] フィールドに入力された情報に追加します。たとえば、[拒否ファイルディレクトリ] フィールドに「C:/reject_file/」と入力されているときに、[拒否ファイル名] フィールドに「filename.bad」と入力すると、統合サービスはリジェクトされた行を C:/reject_file/filename.bad に書き込みます。<br>デフォルトで、統合サービスはターゲットインスタンス名に従って、「ターゲット名.bad」のように拒否ファイルに名前を付けます。<br>また、拒否ファイルセッションパラメータを入力して拒否ファイルまたは拒否ファイルとディレクトリを表すこともできます。すべての拒否ファイルパラメータには「\$BadFile 名前」のように名前を付けます。 |
| ファイルプロパティ設定  | フラットファイルプロパティの定義。外部ローダを使用する場合には、[ファイルプロパティ設定] リンクをクリックしてフラットファイルプロパティを定義する必要があります。<br>Oracle 外部ローダの場合、ターゲットフラットファイルは、固定長または区切りファイルです。<br>Sybase IQ 外部ローダの場合、ターゲットフラットファイルは、固定長または区切りファイルです。<br>Teradata 外部ローダの場合、ターゲットフラットファイルは、固定長または区切りファイルでなければなりません。<br>IBM DB2 外部ローダの場合、ターゲットフラットファイルは区切りファイルでなければなりません。                                                                                                            |

**注:** [パーティション化されたファイルの統合] を選択したり、結合ファイル名を入力したりしないでください。外部ローダを使用する場合、パーティション化された出力ファイルを結合することはできません。

## 外部ローダ接続の選択

ファイルプロパティを設定した後は、外部ローダ接続を選択できます。外部ローダ接続を選択するには、接続タイプと接続オブジェクトを選択します。接続オプションは、[マッピング] タブの [接続] 設定で設定します。

セッションに複数のパーティションが含まれていて、複数の出力ファイルからロードできるローダを選択する場合は、パーティションごとに異なる接続を選択できますが、各接続が同じタイプである必要があります。た

例えば、パーティションごとに異なる Teradata TPump 外部ローダ接続を選択できますが、1つのパーティションに Teradata TPump 接続を選択して別のパーティションに Oracle 接続を選択することはできません。

セッションに複数のパーティションが含まれていて、1つの出力ファイルからしかロードできないローダを選択する場合は、ラウンドロビンパーティション化を使用して、データを単一のターゲットファイルヘッディングします。接続ごとにローダーを選択できますが、Integration Service によってこの接続が最初のパーティションに使用されます。

外部ローダ接続を選択するには：

1. [マッピング] タブで、ナビゲータのターゲットインスタンスを選択します。
2. [ローダ] 接続タイプを選択します。
3. [値] フィールドで、[開く] ボタンをクリックします。
4. 接続オブジェクトまたは変数を選択します。
  - **オブジェクトの使用。**ローダー接続オブジェクトを選択します。[上書き] ボタンをクリックして、接続属性をオーバーライドします。オーバーライドすることのできる属性は、ローダータイプに応じて異なります。
  - **接続変数の使用。**`$LoaderConnectionName` セッションパラメータを使用して、パラメータファイル内にパラメータを定義します。パラメータファイル内の接続属性をオーバーライドします。
5. [OK] をクリックします。

## 外部データのロードのトラブルシューティング

データを外部ローダにロードするセッションを設定しようとしていますが、セッションプロパティで外部ローダ接続を選択できません。

マッピングにリレーショナルターゲットが含まれていることを確認してください。セッションの作成時に、セッションプロパティの [マッピング] タブの [Writers] 設定で [File Writer] を選択してください。次に、[接続] 設定を開いて、外部ローダ接続を選択します。

TPump を使用するセッションを実行しようとしていますが、セッションが失敗します。セッションログには、Teradata 出力ファイル名が長すぎるというエラーが表示されます。

Integration Service により、Teradata 出力ファイル名を使用して、TPump のエラーファイルとログファイルの名前、およびログテーブル名が生成されます。これらの名前を生成するために、Integration Service によって出力ファイル名に数文字のプレフィックスが追加されます。1つのパーティションを含むセッションについては3文字を追加し、複数のパーティションを含むセッションについては5文字を追加します。

Teradata では、ログテーブル名に最大 30 文字を使用できます。Integration Service によってプレフィックスが追加されるため、単一のパーティションを含むセッションを実行する場合には、ターゲット出力ファイル名を最大 27 文字（拡張子を含む）で指定します。複数のパーティションを含むセッションを実行する場合には、ターゲット出力ファイル名を最大 25 文字（拡張子を含む）で指定する必要があります。

TPump を使って Teradata にデータをロードしようとしたが、セッションが失敗しました。エラーを修正しても、やはりセッションが失敗します。

Teradata は、セッションを再実行したときにログテーブルを削除しないことがあります。Teradata データベースをチェックして、ログテーブルが存在する場合は手動で削除してください。その後で、セッションを再実行してください。

# 第 19 章

## FTP

この章では、以下の項目について説明します。

- [FTP の概要, 299 ページ](#)
- [SFTP, 300 ページ](#)
- [統合サービスの動作, 300 ページ](#)
- [セッションの FTP の設定, 302 ページ](#)

## FTP の概要

フラットファイルまたは XML ソースからの読み込みや、フラットファイルまたは XML ターゲットへの書き込みに、ファイル転送プロトコル（FTP）を使用するようにセッションを設定できます。PowerCenter 統合サービスは、メインフレームを含め、接続できるあらゆるマシンに FTP を使用してアクセスできます。ソースファイルとターゲットファイルの両方に関して、ファイルを直接転送するとき、またはファイルをローカルディレクトリにステージングするときに FTP を使用します。ソースファイルに直接アクセスするか、またはファイルリストを使用して、セッション内の間接ソースファイルにアクセスできます。

セッションで FTP ファイルソースおよびターゲットを使用するには、次のタスクを実行します。

1. Workflow Manager で FTP 接続オブジェクトを作成し、接続属性を設定します。
2. セッションで FTP 接続オブジェクトを使用するように、セッションプロパティを設定します。

## FTP の使用に関するルールおよびガイドライン

フラットファイルまたは XML のソースまたはターゲットで FTP を使用する場合、以下のガイドラインに従ってください。

- セッションプロパティで、ソースまたはターゲットの出力ディレクトリを指定できます。ディレクトリを指定しない場合、Integration Service は、UNIX では Integration Service が実行されるディレクトリ、Windows ではシステムディレクトリにファイルをステージングします。
- メインフレーム上に配置されている同じ FTP ソースファイルまたはターゲットファイルを使用する複数のセッションを並列に実行することはできません。
- メインフレームから FTP ソースまたはターゲットをステージングするセッションを含むワークフローを強制終了すると、接続がタイムアウトするまで、そのワークフローを再実行できないことがあります。
- パブリックキー認証を必要とする SFTP サーバーで FTP 接続を使用するセッションを実行するには、セッションを実行するノードでパブリックキーおよびプライベートキーファイルにアクセスできるようにする必要があります。

# SFTP

機密データをネットワーク経由で送信する場合は、Secure File Transfer Protocol (SFTP) を使用してデータを保護できます。SFTP サーバーに接続するには、SFTP を使用するように FTP 接続を設定します。SFTP を使用すると、セキュリティで保護されたデータストリームでのファイル転送を行うことができます。PowerCenter 統合サービスは、安全な接続を実現し、SFTP サーバー上のファイルへのアクセスを可能にする、SSH2 トランスポートレイヤを作成します。

SFTP は、2 台のコンピュータシステム間に暗号化されたチャネルを作成し、次の攻撃を防ぎます。

- IP スプーフィング（リモートホストが別の信頼できるホストからの送信を装ってパケットを送信すること）。
- IP ソースルーティング（ホストが IP パケットが別の信頼できるホストから送信されたと偽装する可能性があること）。
- DNS スプーフィング（攻撃者がネームサーバーレコードを偽造すること）。
- 中間ホストによるクリアテキストパスワードやその他のデータの傍受。
- 中間ホストを制御する攻撃者によるデータの操作。

SFTP は、非対称暗号方式と対称暗号方式を組み合わせ使用して、強力な暗号化と最適なパフォーマンスを実現します。ほとんどの市販サーバーや多くのオープンソースサーバーで SFTP がサポートされています。また、SFTP は暗号化する前にデータストリームを圧縮するため、大容量ファイルの送信に効果的なプロトコルです。

セッションで SFTP ファイルソースおよびターゲットを使用するには、次のタスクを実行します。

1. FTP ワークフロー接続を作成し、FTP 接続オブジェクトを SFTP 用に設定します。
2. セッションプロパティで SFTP 接続オブジェクトを選択して設定します。
3. ソースファイルプロパティを設定します。
4. ターゲットファイルプロパティを設定します。

## 統合サービスの動作

FTP または SFTP を使用した統合サービスの動作は、FTP 接続または SFTP 接続とセッションの設定方法により異なります。統合サービスは、以下の方法で、FTP または SFTP を使用してソースファイルおよびターゲットファイルにアクセスできます。

- **ソースファイル。** 統合サービスをホストするマシン上にソースファイルをステージングします。または、FTP ホストまたは SFTP ホストから直接ソースファイルにアクセスします。単一のソースファイル、または単一のソースインスタンスに対応する複数の間接ソースファイルで構成されるファイルリストを使用します。
- **ターゲットファイル。** 統合サービスをホストするマシン上にターゲットファイルをステージングします。または、FTP ホストまたは SFTP ホスト上のターゲットファイルに書き込みます。

FTP ファイルまたは SFTP ファイルをステージングすることで、ネットワーク障害による部分的な転送のリスクを減らすことができます。統合サービスをホストするマシン上でステージングされたファイルを作成します。FTP プロセスまたは SFTP プロセスがステージングされたファイルを作成した後、統合サービスは読み取り操作が開始されます。ターゲットで FTP または SFTP を使用した場合は、統合サービスがステージングされたファイルに書き込みを行った後に FTP プロセスまたは SFTP プロセスが開始されます。ステージングされたファイルが完了する前にネットワーク障害が発生した場合は、ステージングされたファイルを削除してセッションを実行し直すことができます。

ステージングは、FTP または SFTP 接続オブジェクト内で、またはセッションの前後のシェルコマンドで設定できます。

## ソースファイルを対象とした FTP の使用

フラットファイルまたは XML ファイルソースを読み取るセッションで FTP を使用します。Integration Service をホストするマシン上にセッションのソースファイルをステージングできます。単一のソースファイル、または各ソースインスタンスのファイルリストを使用します。

ソースデータをステージングする場合、Integration Service は FTP を使用してローカルファイルを作成します。ローカルファイルはセッションのソースとして使用されます。ステージングされたファイルが完了するまで、Integration Service はデータをパイプラインに移動しません。

ソースデータをステージングしない場合、Integration Service は FTP を使用してソースファイルに直接アクセスします。ネットワーク障害が発生した場合は、セッションを再度実行します。

以下の表に、ソースファイルで FTP を使用する Integration Service の動作を示します。

| ソースタイプ   | ステージング済み | Integration Service の動作                                                                                                        |
|----------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| Direct   | はい       | Integration Service は、セッションの開始後に、FTP ホストから Integration Service のホストマシンにファイルをコピーします。                                            |
| Direct   | いいえ      | Integration Service は、FTP を使用して、ソースファイルに直接アクセスします。                                                                             |
| Indirect | はい       | Integration Service は、セッションの開始後にファイルリストを読み取り、そのファイルリストとソースファイルを Integration Service のホストマシンにコピーします。                            |
| Indirect | いいえ      | Integration Service は、セッションの開始後に、Integration Service のホストマシンにファイルリストをコピーします。Integration Service は、FTP を使用して、ソースファイルに直接アクセスします。 |

## ターゲットファイルでの FTP の使用

フラットファイルまたは XML ファイルターゲットに書き込むセッションで FTP を使用します。ターゲットファイルを FTP ホストにコピーする前に、Integration Service をホストするマシンでターゲットファイルをステージングできます。

ターゲットデータをステージングする際、Integration Service は、ローカルでターゲットファイルを作成し、セッションの完了後にそのファイルを FTP ホストに転送します。ターゲットファイルをステージングしない場合、Integration Service は、FTP ホスト上のターゲットファイルに直接書き込みます。ネットワーク障害が発生した場合は、セッションを再度実行します。

パーティション化オプションを使用している場合は、複数のターゲットパーティションインスタンスに FTP を使用します。Integration Service または FTP ホスト上で、複数のターゲットファイルまたは統合ファイルに書き込むことができます。

## セッションの FTP の設定

FTP を使用するようにセッションを設定する前に、Workflow Manager で FTP 接続オブジェクトを作成する必要があります。Integration Service は、FTP 接続属性を使用して、FTP サーバーに接続します。

Workflow Manager で FTP 接続オブジェクトを作成したら、その後、FTP を使用するセッションを設定できます。安全な接続を使用するには、SFTP に設定された FTP 接続オブジェクトを選択します。フラットファイル、XML ソース、または XML ターゲットで任意のセッションを使用します。

セッションを設定するには、FTP 接続を必要とするソースおよびターゲットごとに次のタスクを実行します。

- FTP 接続を選択します。
- ソースファイルプロパティを設定します。
- ターゲットファイルプロパティを設定します。

Integration Service マシンでソースファイルまたはターゲットファイルをステージングするには、セッションプロパティで FTP 接続を編集して、一時ファイルのディレクトリとファイル名を設定します。

## セッションの SFTP の設定

パブリックキー認証を必要とする SFTP 接続オブジェクトを使用してセッションを実行するには、セッションを実行するノードでパブリックキーおよびプライベートキーファイルにアクセスできるようにする必要があります。

統合サービスがプライマリとバックアップのノードで実行するように設定されている場合、統合サービスプロセスを実行するように設定された各ノード上で、これらのキーファイルにアクセスできるようにしてください。

統合サービスがグリッド上で実行するように設定されている場合、グリッド上で実行するように設定されている各ノードでこれらのキーファイルにアクセスできるようにしてください。グリッド内で各ノードにこれらのファイルを置くことができない場合、ドメインにリソースを作成して、ファイルを置く各ノードに割り当てます。セッションを作成するときに、リソースを使用するようにセッションを設定します。

例えば、「SFTP」という名前のカスタムリソースを作成するとします。セッションを作成するときに、SFTP リソースを使用するためにセッションが必要になる場合があります。ロードバランサは、これらのキーファイルにアクセス可能なノードにセッションをディスパッチするだけです。

## FTP 接続の選択

FTP を使用するようにセッションを設定するには、接続タイプと接続オブジェクトを選択します。FTP 接続を使用するソースおよびターゲットごとに FTP 接続オブジェクトを選択します。SFTP を使用するには、SFTP に設定された FTP 接続オブジェクトを選択します。接続オプションは、[マッピング] タブの [接続] 設定で設定します。

ソースインスタンスまたはターゲットインスタンスに FTP 接続を選択するには：

1. [マッピング] タブの [トランスフォーメーション] ビューでソースインスタンスとターゲットインスタンスを選択します。
2. FTP 接続タイプを選択します。
3. [値] フィールドで、[開く] ボタンをクリックします。
4. 接続オブジェクトまたは変数を選択します。
  - **オブジェクトの使用。** FTP 接続オブジェクトを選択します。 [上書き] ボタンをクリックして、接続属性をオーバーライドします。



- **接続変数の使用。** \$FTPConnectionName セッションパラメータを使用して、パラメータファイル内にパラメータを定義します。パラメータファイル内の接続属性をオーバーライドします。

以下の属性をオーバーライドできます。

| 属性                     | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| リモートファイル名              | <p>ソースまたはターゲットのリモートファイル名。間接ソースファイルを使用する場合には、間接ソースファイル名を入力します。</p> <p>このファイル名には 7 ビット ASCII 文字を使用する必要があります。Unicode 文字のリモートファイル名を使用すると、セッションは失敗します。</p> <p>ソースファイル名に完全修飾名を入力すると、Integration Service により、[デフォルトリモートディレクトリ] フィールドに入力されたパスが無視されます。完全修飾ファイル名を一重引用符または二重引用符で囲むと、セッションは失敗します。</p> <p>リモートファイル名には、パラメータまたは変数を使用できます。パラメータファイルで定義可能なパラメータまたは変数タイプを使用します。たとえば、ソースまたはターゲットリモートファイル名としてセッションパラメータ \$ParamMyRemoteFile を使用し、パラメータファイルでそのファイル名に \$ParamMyRemoteFile を設定できます。</p> |
| ステージング済み               | Integration Service 上のソースファイルまたはターゲットファイルをステージングします。デフォルトでは、ステージングされません。                                                                                                                                                                                                                                                                                                                                                                                                          |
| Is Transfer Mode ASCII | <p>転送モードを変更します。有効時には、ASCII 転送モードが Integration Service で使用されます。Windows マシン上でのファイル転送時には、ASCII モードを使用すると、テキストファイル内の行末文字を確実に正しく変換することができます。無効時には、バイナリ転送モードが Integration Service で使用されます。UNIX マシン上では、ファイル転送時にバイナリ転送モードを使用してください。デフォルトでは無効になっています。</p>                                                                                                                                                                                                                                  |

5. [OK] をクリックします。

## ソースファイルプロパティの設定

FTP または SFTP でソースファイルにアクセスする場合、ソースインスタンスに FTP または SFTP 接続オブジェクトを選択した後、ソースファイルプロパティを設定します。ソースファイルプロパティで、ソースファイルタイプと一時領域を特定します。ソースファイルプロパティは [マッピング] タブの [プロパティ] で設定できます。

ソースファイルをステージングする場合は、ソースファイル名、ディレクトリ、およびファイルタイプを選択します。

ソースファイルをステージングしない場合は、ソースファイルタイプを指定します。PowerCenter 統合サービスでは、FTP 接続オブジェクトのリモートファイル名とディレクトリが使用され、ソースファイル名とディレクトリは無視されます。

1. Workflow Manager で、Task Developer を開き、[タスク] > [作成] をクリックします。
2. タスクタイプに [セッション] を選択します。
3. セッションタスクの名前を入力します。タスク名にピリオド (.) を使用しないでください。Workflow Manager では、タスク名にピリオドを使用できません。
4. [作成] をクリックします。  
セッションタスクが作成されます。
5. [マッピング] ダイアログボックスで、セッションタスクで使用するマッピングを選択し、[OK] をクリックします。

6. **【完了】** をクリックします。
7. ワークスペースでアイコンをダブルクリックして、セッションプロパティを開きます。  
**【タスクの編集】** ダイアログボックスが表示されます。
8. **【マッピング】** タブの **【ソース】** ノードの **【プロパティ】** 設定で、次のソースファイルプロパティを設定します。

| 属性            | 説明                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ソースファイルタイプ    | ソースファイルにソースデータと、同じファイルプロパティを持つファイルのリストとのどちらが格納されているかを表示します。ソースファイルにソースデータが含まれる場合には、 <b>[Direct]</b> を選択します。ソースファイルにファイルのリストが含まれる場合には、 <b>[Indirect]</b> を選択します。                                                                                                                                                                                                                         |
| ソースファイルディレクトリ | <p>ソースデータのステージングに使用するローカルソースファイルディレクトリの名前とパス。デフォルトでは、PowerCenter 統合サービスは、サービスのプロセス変数ディレクトリ「\$PMSourceFileDir」をファイルソースとして使用します。PowerCenter 統合サービスは、セッションの実行時に、このフィールドと <b>【ソースファイル名】</b> フィールドを連結します。</p> <p>ソースファイルをステージングしない場合、PowerCenter 統合サービスは、SFTP 用に設定された FTP 接続オブジェクトのファイル名とディレクトリを使用します。</p> <p><b>【ソースファイル名】</b> フィールドに完全修飾ファイル名を入力した場合、PowerCenter 統合サービスはこのフィールドを無視します。</p> |
| ソースファイル名      | <p>ソースデータのステージングに使用するローカルソースファイルの名前。ファイル名、またはファイル名とパスを入力できます。完全修飾ファイル名を入力した場合、PowerCenter 統合サービスは <b>【ソースファイルのディレクトリ】</b> フィールドを無視します。</p> <p>ソースファイルをステージングしない場合、PowerCenter 統合サービスは、SFTP 用に設定された FTP 接続オブジェクトのリモートファイル名とデフォルトディレクトリを使用します。</p>                                                                                                                                        |

9. **【適用】** をクリックします。
10. **【タスクの編集】** ダイアログボックスを閉じるには、**【OK】** をクリックします。

## ターゲットファイルプロパティの設定

FTP または SFTP でターゲットファイルに書き込む場合、ターゲットインスタンスに FTP または SFTP 接続オブジェクトを指定した後、ターゲットファイルプロパティを指定します。ターゲットファイルのプロパティにより、拒否ファイルとディレクトリ、および一時領域が特定されます。**【マッピング】** タブの **【プロパティ】** 設定でターゲットファイルプロパティを指定します。

ターゲットファイルをステージングする場合は、ターゲットファイル名とディレクトリ、および拒否ファイル名とディレクトリを設定します。ターゲットファイルをステージングしない場合は、拒否ファイルとディレクトリを設定します。PowerCenter 統合サービスでは、FTP 接続オブジェクトのリモートファイル名とディレクトリが使用されます。

パーティション化オプションを使用する場合は、統合ファイルプロパティも選択できます。

1. Workflow Manager で、Task Developer を開き、**【タスク】** > **【作成】** をクリックします。
2. タスクタイプに **【セッション】** を選択します。
3. セッションタスクの名前を入力します。タスク名にピリオド (.) を使用しないでください。Workflow Manager では、タスク名にピリオドを使用できません。



4. **【作成】** をクリックします。  
セッションタスクが作成されます。
5. **【マッピング】** ダイアログボックスで、セッションタスクで使用するマッピングを選択し、**【OK】** をクリックします。
6. **【完了】** をクリックします。
7. ワークスペースでアイコンをダブルクリックして、セッションプロパティを開きます。  
**【タスクの編集】** ダイアログボックスが表示されます。
8. **【マッピング】** タブの **【ターゲット】** ノードの **【プロパティ】** 設定で、次のターゲットファイルプロパティを設定します。

| 属性           | 説明                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 出力ファイルディレクトリ | <p>ターゲットデータのステージングに使用するローカルターゲットファイルディレクトリの名前とパス。デフォルトでは、PowerCenter 統合サービスは、サービスプロセスの変数ディレクトリ「\$PMTargetFileDir」を使用します。</p> <p>PowerCenter 統合サービスは、セッションの実行時に、このフィールドと <b>【出力ファイル名】</b> フィールドを連結します。</p> <p>ターゲットファイルをステージングしない場合、PowerCenter 統合サービスは、FTP 接続オブジェクトのファイル名とディレクトリを使用します。</p> <p><b>【出力ファイル名】</b> フィールドに完全修飾ファイル名を入力した場合、PowerCenter 統合サービスはこのフィールドを無視します。</p> |
| 出力ファイル名      | <p>ターゲットデータのステージングに使用するローカルターゲットファイルの名前。ファイル名、またはファイル名とパスを入力できます。完全修飾ファイル名を入力した場合、PowerCenter 統合サービスは <b>【出力ファイルディレクトリ】</b> フィールドを無視します。</p> <p>ソースファイルをステージングしない場合、PowerCenter 統合サービスは、FTP 接続オブジェクトのリモートファイル名とデフォルトディレクトリを使用します。</p>                                                                                                                                        |

9. **【適用】** をクリックします。
10. **【タスクの編集】** ダイアログボックスを閉じるには、**【OK】** をクリックします。
11. ターゲットファイルプロパティの変更を保存するには、**【リポジトリ】** > **【保存】** をクリックします。

## FTP ファイルターゲットのパーティション化

セッションのパーティション化したターゲットに対して FTP 接続タイプを選択する場合、ターゲットパーティションの FTP 設定を行います。

パーティションごとにターゲットファイルまたは個々のターゲットファイルを統合できます。

ターゲットパーティションに対して FTP 設定を行う場合には、以下の規則とガイドラインに従ってください。

- ターゲットパーティションごとに FTP 接続を使用する必要があります。
- ターゲットパーティションに接続オブジェクトを選択すると、ファイルのステージングを選択できます。シケンシャル統合を使用するにはファイルをステージングする必要があります。
- ターゲットパーティションの FTP 接続に、リモートファイル名以外の設定が指定されている場合、Integration Service は、統合ファイルを作成しません。

次の表に、パーティション化された FTP ファイルターゲットに対する Integration Service のアクションを示します。

| Merge Type       | Integration Service の動作                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| No Merge         | Integration Service は、パーティションごとにターゲットファイルを 1 つ生成します。ファイルをステージングする場合、セッションの終わりにターゲットファイルがリモートロケーションに転送されます。ファイルをステージングしない場合、リモートロケーションでターゲットファイルが 1 つ生成されます。                                                                                                                                                                                                                                                                                                                                                                                           |
| シーケンシャル統合        | Integration Service は、接続オブジェクトの <b>【ステージングの有無】</b> オプションを有効にします。パーティションごとに出力ファイルを 1 つ作成します。セッションの終わりに次のアクションを実行します。 <ol style="list-style-type: none"> <li>1. 個々の出力ファイルを統合ファイルに統合します。</li> <li>2. 個々の出力ファイルを削除します。</li> <li>3. 統合ファイルをリモートロケーションに転送します。</li> </ol>                                                                                                                                                                                                                                                                                    |
| ファイルリスト          | <p>ファイルをステージングする場合、Integration Service は、以下のファイルを作成します。</p> <ul style="list-style-type: none"> <li>- パーティションごとの出力ファイル</li> <li>- ローカルファイルの名前とパスが含まれているファイルリスト</li> <li>- リモートファイルの名前とパスが含まれているファイルリスト</li> </ul> <p>セッションの終わりに、Integration Service は、これらのファイルをリモートロケーションに転送します。個々のターゲットファイルが Merge File Directory 内に配置されている場合、ファイルリストには、相対パスが指定されています。上記以外の場合には、ファイルリストには絶対パスが指定されています。</p> <p>ファイルをステージングしない場合、Integration Service は、リモートロケーションでパーティションごとにデータを書き込み、個々のターゲットファイルのリストを含むリモートファイルリストを作成します。</p> <p>ファイルリストは別のマッピングでソースファイルとして使用します。</p> |
| Concurrent Merge | <p>ファイルをステージングする場合、Integration Service は、すべてのターゲットパーティションのデータを、ローカルマージファイルに同時に書き込みます。セッションの終わりに、Integration Service は、このマージファイルをリモートロケーションに転送します。Integration Service は、中間出力ファイルに書き込むことはしません。</p> <p>ファイルをステージングしない場合、Integration Service は、すべてのパーティションのターゲットデータを、リモートロケーションのマージファイルに同時に書き込みます。</p>                                                                                                                                                                                                                                                    |

## 第 20 章

# セッションのキャッシュ

この章では、以下の項目について説明します。

- [セッションのキャッシュの概要, 307](#) ページ
- [キャッシュメモリ, 308](#) ページ
- [キャッシュファイル, 309](#) ページ
- [キャッシュサイズの設定, 312](#) ページ
- [キャッシュのパーティション化, 315](#) ページ
- [アグリゲータキャッシュ, 316](#) ページ
- [ジョイナキャッシュ, 317](#) ページ
- [ルックアップキャッシュ, 321](#) ページ
- [ランクキャッシュ, 322](#) ページ
- [ソータキャッシュ, 324](#) ページ
- [XML ターゲットキャッシュ, 324](#) ページ
- [キャッシュサイズの最適化, 325](#) ページ

## セッションのキャッシュの概要

Integration Service では、XML ターゲット、アグリゲータ、ジョイナ、ルックアップ、ランク、およびソータの各トランスフォーメーションに対して、キャッシュメモリが割り当てられます。Integration Service では、XML ターゲット、アグリゲータ、ジョイナ、ルックアップ、およびランクの各トランスフォーメーションに対して、インデックスキャッシュおよびキャッシュメモリが作成されます。Integration Service ではインデックスキャッシュにキー値が格納され、データキャッシュに出力値が格納されます。Integration Service では、ソータトランスフォーメーションに対して、ソートキーおよびソートするデータを保存する 1 つのキャッシュが作成されます。

セッションプロパティで、キャッシュのメモリパラメータを設定します。キャッシュサイズを最初に設定する場合、トランスフォーメーションの処理に必要なメモリ容量を計算するか、または実行時にメモリ要件が自動的に設定されるように Integration Service を設定できます。

セッションの実行後に、そのセッション内のトランスフォーメーションのキャッシュサイズを調整できます。トランスフォーメーション統計を分析すると、最適なセッションパフォーマンスに必要なキャッシュサイズを決定し、設定したキャッシュサイズを更新できます。

設定よりも多いメモリを Integration Service が必要とする場合は、オーバーフローした値がキャッシュファイルに格納されます。セッションの終了時に、Integration Service によってキャッシュメモリが解放されます。ほとんどの場合、これらのキャッシュファイルは削除されます。

セッションに複数のパーティションが含まれる場合は、Integration Service によって、パーティションごとに1つのメモリキャッシュが作成されます。特別な状況では、Integration Service はキャッシュのパーティション化を使用し、パーティションごとに個別のキャッシュが作成されます。

以下の表に、Integration Service で各キャッシュに保存される情報のタイプを示します。

| マッピングオブジェクト | キャッシュタイプと説明                                                                                                                                 |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| アグリゲータ      | <ul style="list-style-type: none"> <li>- インデックス。GroupBy ポートの設定に従ってグループ値を格納します。</li> <li>- データ。GroupBy ポートに基づいて計算結果を格納します。</li> </ul>        |
| ジョイナ        | <ul style="list-style-type: none"> <li>- インデックス。結合条件に、一意キーを持つすべてのマスター行を格納します。</li> <li>- データ。マスターソースの行を格納します。</li> </ul>                    |
| ルックアップ      | <ul style="list-style-type: none"> <li>- インデックス。ルックアップ条件の情報を格納します。</li> <li>- データ。インデックスキャッシュに格納されないルックアップデータを格納します。</li> </ul>             |
| ランク         | <ul style="list-style-type: none"> <li>- インデックス。GroupBy ポートの設定に従ってグループ値を格納します。</li> <li>- データ。GroupBy ポートに基づいてランク情報を格納します。</li> </ul>       |
| ソータ         | <ul style="list-style-type: none"> <li>- ソータ。ソートキーおよびデータを格納します。</li> </ul>                                                                  |
| XML ターゲット   | <ul style="list-style-type: none"> <li>- インデックス。別々のキャッシュに、プライマリキーおよび外部キーの情報を格納します。</li> <li>- データ。XML ターゲットの生成中に、XML 行データを格納します。</li> </ul> |

## キャッシュメモリ

Integration Service では、設定したキャッシュサイズに基づいて、各メモリキャッシュが作成されます。セッションの作成時に、セッションプロパティで各トランスフォーメーションインスタンスにキャッシュサイズを設定できます。

Integration Service では、以下のいずれかの理由により、設定したキャッシュサイズを増やすことがあります。

- **設定されたキャッシュサイズが、操作を処理するために必要な最小キャッシュサイズよりも小さい場合。** Integration Service では、各セッションを初期化するために最小限のメモリが必要です。設定したキャッシュサイズが必要な最小キャッシュサイズよりも小さい場合は、最小要件を満たすように、Integration Service によって設定済みのキャッシュサイズが増加されます。Integration Service が必要な最小メモリを割り当てることができない場合、セッションは失敗します。
- **設定されたキャッシュサイズが、キャッシュページサイズの倍数ではない場合。** Integration Service は、キャッシュデータをキャッシュページに格納します。キャッシュされたページは、キャッシュに均等に収まる必要があります。したがって、たとえばキャッシュサイズを 10MB (1,048,576 バイト) に設定し、キャッシュページサイズが 10,000 バイトである場合、Integration Service では、設定したキャッシュサイズが 10,000 バイトのページサイズの倍数になるように 1,050,000 バイトまで増やされます。

Integration Service によって、設定したキャッシュサイズが増やされると、セッションの実行が継続され、以下のようなメッセージがセッションログに書き込まれます。

```
MAPPING> TE_7212 Increasing [Index Cache] size for transformation <transformation name> from <configured index cache size> to <new index cache size>.
```

セッションログで、最小要件を満たすのに十分なメモリが割り当てられているかどうかを確認してください。

最適なパフォーマンスを達成するには、キャッシュサイズをトランスフォーメーションの処理に必要な総メモリ量に設定します。トランスフォーメーションを処理するのに十分なキャッシュメモリがない場合、Integration Service では、メモリ内でトランスフォーメーションの一部が処理され、残りを処理するために情報がディスクにページングされます。

以下の情報を使用して、32 ビットのマシンと 64 ビットのマシンとでメモリキャッシュの処理方法が異なることを理解してください。

- 32 ビットのマシン上で実行される Integration Service プロセスは、設定されたすべてのセッションのキャッシュの合計が 2GB を超える場合、セッションを実行できません。グリッドでセッションを実行する場合、単一ノードで実行するすべてのセッションスレッドの合計キャッシュサイズが、2GB を超えないようにする必要があります。
- グリッドに 32 ビットと 64 ビットの Integration Service プロセスがあり、セッションが 2GB のメモリを超えている場合、64 ビットマシンで Integration Service を実行するようにセッションを設定する必要があります。

## キャッシュファイル

セッションを実行すると、Integration Service によって各トランスフォーメーションに対して最低でも 1 つのキャッシュファイルが作成されます。Integration Service のメモリでトランスフォーメーションを処理できない場合、オーバーフローした値がキャッシュファイルに書き込まれます。

以下の表に、Integration Service でさまざまなマッピングオブジェクトに対して作成されるキャッシュファイルのタイプを示します。

| マッピングオブジェクト                           | キャッシュファイル                                                                                                                                                                                                                                    |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| アグリゲータ、ジョイナ、ルックアップ、およびランクトランスフォーメーション | Integration Service では、以下のタイプのキャッシュファイルが作成されます。 <ul style="list-style-type: none"><li>- インデックスキャッシュおよびデータキャッシュごとに 1 つのヘッダファイル</li><li>- インデックスキャッシュおよびデータキャッシュごとに 1 つのデータファイル</li></ul>                                                      |
| ソータトランスフォーメーション                       | Integration Service では、1 つのソータキャッシュファイルが作成されます。                                                                                                                                                                                              |
| XML ターゲット                             | Integration Service では、以下のタイプのキャッシュファイルが作成されます。 <ul style="list-style-type: none"><li>- XML ターゲットグループごとに 1 つのデータキャッシュファイル</li><li>- XML ターゲットグループごとに 1 つのプライマリキーインデックスキャッシュファイル</li><li>- XML ターゲットグループごとに 1 つの外部キーインデックスキャッシュファイル</li></ul> |

Integration Service は、Integration Service コードページに基づいてキャッシュファイルを作成します。

セッションを実行すると、Integration Service は、セッションログにキャッシュファイル名とトランスフォーメーション名を示すメッセージを書き込みます。セッションが完了すると、Integration Service はキャッシュメモリを解放し、通常はキャッシュファイルを削除します。以下の状況では、インデックスキャッシュファイルおよびデータキャッシュファイルは、キャッシュディレクトリに保持されます。

- セッションが差分集計を実行している場合。

- 永続キャッシュを使用してルックアップトランスフォーメーションを設定した場合。
- セッションが正常終了しなかった場合。次回セッションを実行すると、Integration Service によって既存のキャッシュファイルが削除され、新しいキャッシュファイルが作成されます。

**注:** キャッシュファイルへの書き込みによってセッションのパフォーマンスが低下する可能性があるため、メモリでトランスフォーメーションが処理されるようにキャッシュサイズを設定します。

## キャッシュファイルの命名規則

では、インデックス、データ、およびソータキャッシュファイルに対して、さまざまな命名規則が使用されます。

以下の表に、各キャッシュファイルタイプの命名規則を示します。

| キャッシュファイル  | 命名規則                                                                                         |
|------------|----------------------------------------------------------------------------------------------|
| データおよびソーター | [<名前の接頭語> <接頭語><セッション ID>_<トランスフォーメーション ID>]_[パーティションインデックス]_[OS][BIT].<接尾語> [オーバーフローインデックス] |
| インデックス     | <接頭語><セッション ID>_<トランスフォーメーション ID>_<グループ ID>_<キータイプ>.<接尾語><オーバーフロー>                           |

以下の表に、キャッシュファイル名のコンポーネントを示します。

| ファイル名の要素          | 説明                                                                                                                                                                                                                                                                                       |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 名前の接頭語            | Lookup トランスフォーメーションで設定されたキャッシュファイル名の接頭語。<br>Lookup トランスフォーメーションキャッシュファイル用です。                                                                                                                                                                                                             |
| Prefix            | トランスフォーメーションのタイプを表します。<br><ul style="list-style-type: none"> <li>- アグリゲータトランスフォーメーション - PMAGG</li> <li>- ジョイナトランスフォーメーション - PMJNR</li> <li>- ルックアップトランスフォーメーション - PMLKUP</li> <li>- ランクトランスフォーメーション - PMAGG</li> <li>- ソータトランスフォーメーション - PMSORT</li> <li>- XML ターゲット - PMXML.</li> </ul> |
| Session ID        | セッションインスタンスの ID 番号。                                                                                                                                                                                                                                                                      |
| Transformation ID | トランスフォーメーションインスタンスの ID 番号。                                                                                                                                                                                                                                                               |
| Group ID          | 階層型の XML ターゲットでのグループごとの ID。Integration Service では、グループごとに 1 つのインデックスキャッシュが作成されます。XML ターゲットキャッシュファイル用です。                                                                                                                                                                                  |
| Key Type          | キーのタイプ。外部キーまたはプライマリキーを使用できます。XML ターゲットキャッシュファイル用です。                                                                                                                                                                                                                                      |

| ファイル名の要素        | 説明                                                                                                                                                                                                                                                                                                                                          |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Partition Index | セッションに複数のパーティションが含まれる場合、この ID によってパーティション番号が識別されます。パーティションインデックスはゼロベースになります。したがって、先頭パーティションにはパーティションインデックスはありません。パーティションインデックス 2 は、3 つ目のパーティションに作成されたキャッシュファイルという意味になります。                                                                                                                                                                   |
| OS              | Integration Service プロセスを実行するマシンのオペレーティングシステムを識別します。<br><ul style="list-style-type: none"> <li>- W - Windows</li> <li>- S - Solaris</li> <li>- A - AIX</li> <li>- L - Linux</li> <li>- M - Mainframe</li> </ul> Lookup トランスフォーメーションキャッシュファイル用です。                                                                                            |
| BIT             | Integration Service プロセスを実行するマシンのビットプラットフォームを識別します (32 ビットまたは 64 ビット)。Lookup トランスフォーメーションキャッシュファイル用です。                                                                                                                                                                                                                                      |
| Suffix          | キャッシュファイルのタイプを識別します。<br><ul style="list-style-type: none"> <li>- インデックスキャッシュファイルは、ヘッダファイルの場合は.idx0、データファイルの場合は.idx<i>n</i> です。</li> <li>- データキャッシュファイルは、ヘッダファイルの場合は.dat0、データファイルの場合は.dat<i>n</i> です。</li> <li>- ソータキャッシュファイルは、.PMSORT() です。</li> </ul>                                                                                      |
| Overflow Index  | キャッシュファイルが 2GB を超えるデータを扱う場合、Integration Service では、より多くのキャッシュファイルが作成されます。これらのファイルを作成するときに Integration Service はファイル名にオーバーフローインデックスを付加し、PMAGG*.idx2 や PMAGG*.idx3 のようにします。キャッシュファイルの数は、キャッシュディレクトリのディスクの空きによって制限されます。<br><b>注:</b> ソータトランスフォーメーションのキャッシュファイルが 2GB を超えるデータを処理する場合、PowerCenter Integration Service がさらにキャッシュファイルを作成することはありません。 |

例えば、インデックスキャッシュのデータファイル名は PMLKUP748\_2\_5S32.idx1 とします。PMLKUP は、トランスフォーメーションタイプがルックアップであることがわかり、748 はセッション ID、2 はトランスフォーメーション ID、5 はパーティションインデックス、S (Solaris) はオペレーティングシステム、32 はビットプラットフォームです。

## キャッシュファイルディレクトリ

Integration Service では、キャッシュファイルはデフォルトで \$PMCacheDir ディレクトリに作成されます。Integration Service プロセスがディレクトリを見つけれない場合は、セッションは失敗し、キャッシュファイルを作成するまたは開くことができなかったことを示すメッセージがセッションログに書き込まれます。

Integration Service で、複数のキャッシュファイルが作成される場合もあります。キャッシュファイルの数は、キャッシュディレクトリのディスクの空きによって制限されます。

グリッドで Integration Service を実行する際に、一部の Integration Service ノードのみが共有キャッシュファイルディレクトリに高速アクセスしている場合は、ディレクトリへのアクセスを高速なノードで実行するた



めに、大きいキャッシュを持つ各セッションを設定します。ディレクトリへのアクセスが高速なノードで実行するようにセッションを設定するには、以下の手順を実行します。

1. PowerCenter リソースを作成します。
2. ディレクトリへのアクセスが高速なノードで、そのリソースを使用できるようにします。
3. そのリソースをセッションに割り当てます。

グリッドのすべての Integration Service プロセスがキャッシュファイルに低速でアクセスしている場合は、Integration Service プロセスごとに別々のローカルキャッシュファイルディレクトリを設定します。キャッシュディレクトリを含むマシンと同一のマシンで Integration Service プロセスを実行する場合は、キャッシュファイルへのアクセスが高速になることもあります。

## キャッシュサイズの設定

セッションプロパティでキャッシュのメモリ容量を設定します。セッションプロパティで指定したキャッシュサイズによって、トランスフォーメーションプロパティで設定した値はオーバーライドされます。

設定するメモリ容量は、使用するメモリキャッシュおよびディスクキャッシュのサイズによって異なります。設定したキャッシュサイズが、メモリでトランスフォーメーションを処理するのに十分な大きさではない場合は、Integration Service ではトランスフォーメーションの一部はメモリで実行され、残りのトランスフォーメーションを処理するためにキャッシュファイルに情報がページングされます。セッションのパフォーマンスを最適にするには、Integration Service のメモリですべてのデータを処理できるように、キャッシュサイズを設定します。

セッションが再利用可能な場合、セッションのすべてのインスタンスで、再利用可能なセッションプロパティで設定したキャッシュサイズが使用されます。セッションインスタンスのキャッシュサイズはオーバーライドできません。

キャッシュサイズを設定するには、以下のいずれかの方法を使用します。

- **キャッシュの計算。**この計算を使用して、トランスフォーメーションの処理に必要な総メモリ容量を計算します。
- **自動キャッシュメモリ。**自動メモリを使用して、トランスフォーメーションの処理に割り当てるキャッシュサイズに上限を指定します。Integration Service プロセスを実行しているマシンのキャッシュメモリが制限されている場合に、この方法を使用します。
- **数値。**キャッシュサイズに特定の値を設定します。キャッシュサイズをチューニングするときに、特定の値を設定します。

Integration Service によってキャッシュのパーティション化が使用されている場合、メモリ要件は別の方法で設定します。Integration Service によってキャッシュのパーティション化が使用されている場合、各パーティションには設定されたキャッシュサイズが割り当てられます。キャッシュのパーティション化を使用するトランスフォーメーションのメモリ要件を設定するには、トランスフォーメーションに必要な合計サイズを計算し、それをパーティション数で割ります。

トランスフォーメーションへの入力の変更されると、トランスフォーメーションのキャッシュファイル要件も変更される場合があります。定期的にセッションログでキャッシュサイズを監視すると、キャッシュサイズの調整に役立ちます。

## キャッシュサイズの計算

キャッシュの計算を使用して、トランスフォーメーションの処理に必要な総メモリ容量を見積もります。キャッシュサイズを計算するには、入力を行う必要があります。この入力は、トランスフォーメーションのタイプ



によって異なります。たとえば、アグリゲータトランスフォーメーションのキャッシュサイズを計算するには、グループの数を入力します。

キャッシュの計算では、以下のいずれかのモードを選択できます。

- **自動。** [設定オブジェクト] タブで設定した最大メモリに基づいて、Integration Service の実行中にキャッシュサイズを決定する場合に、自動モードを選択します。
- **計算。** 入力値に基づいてトランスフォーメーションに必要なサイズの合計を計算する場合に選択します。キャッシュの計算には、トランスフォーメーションごとに別々の入力が必要となります。計算したキャッシュサイズを適用するには、適切なキャッシュタイプを選択する必要があります。たとえば、計算したキャッシュサイズをインデックスサイズではなくデータキャッシュに適用するには、[データキャッシュサイズ] オプションのみを選択します。

キャッシュの計算では、入力値に基づいて最適なセッションパフォーマンスに必要なキャッシュサイズを見積もります。キャッシュサイズを設定し、セッションを実行した後に、セッションログでトランスフォーメーションの統計を確認して、設定したキャッシュサイズを調整できます。

**注:** キャッシュの計算を使用して、XML ターゲットのキャッシュサイズを概算することはできません。

## 自動キャッシュサイズ

デフォルトでは、トランスフォーメーションのメモリキャッシュは自動モードに設定されます。Integration Service では、キャッシュが自動モードに設定されているすべてのトランスフォーメーションに自動的にキャッシュメモリが割り当てられます。Integration Service でトランスフォーメーションに割り当てることができる最大キャッシュメモリ量を設定することができます。

自動キャッシュモードのトランスフォーメーションの最大キャッシュメモリを設定するには、次のセッションプロパティを設定します。

### 自動メモリ属性で利用できる最大メモリ

セッションキャッシュに割り当てる最大メモリ量です。キャッシュメモリが自動に設定されたすべてのトランスフォーメーションに、セッションキャッシュからメモリが割り当てられます。デフォルトの単位はバイトです。他の単位を指定するには、値に KB、MB、または GB を追加します。例えば、「1048576」、「1024KB」、「1MB」のように指定します。

### 自動メモリ属性で利用できる合計メモリの最大割合 (%)

セッションキャッシュに割り当てるマシンのメモリの割合です。キャッシュメモリが自動に設定されたすべてのトランスフォーメーションに、セッションキャッシュからメモリが割り当てられます。

セッションの最大キャッシュサイズを設定すると、メモリの最大割合が計算され、指定した最大メモリ量と比較されます。その後、自動キャッシュモードのトランスフォーメーションに、どちらか少ない方のメモリが割り当てられます。自動キャッシュモードのトランスフォーメーションが複数ある場合は、自動キャッシュモードのすべてのトランスフォーメーションにメモリが割り当てられます。

例えば、Integration Service をホストするマシンのメモリが 1GB であるとします。[自動メモリ属性で利用できる最大メモリ] プロパティを 800MB に設定し、さらに [自動メモリ属性で利用できる合計メモリの最大割合 (%) ] プロパティを 10% に設定します。この場合、セッションキャッシュに 102.4MB のメモリが割り当てられ、自動キャッシュモードのすべてのトランスフォーメーションにそのキャッシュメモリが分割されます。

最大セッションキャッシュサイズの設定は、キャッシュモードが自動に設定されたトランスフォーメーションだけに適用されます。特定のキャッシュサイズを設定したトランスフォーメーションには、個別にメモリが割り当てられます。

キャッシュが必要なトランスフォーメーションがセッションに複数ある場合、トランスフォーメーションごとに、キャッシュモードを自動に設定したり、キャッシュサイズを指定したりできます。このようにすると、キャッシュサイズの数値が設定されたトランスフォーメーションのメモリに加え、自動キャッシュモードのトランスフォーメーションに指定されたメモリも割り当てられます。

例えば、キャッシュが必要なトランスフォーメーションがセッションに3つあるとします。そのうちの2つのトランスフォーメーションを自動キャッシュモードに設定し、セッションの最大メモリキャッシュサイズを800MBと指定します。さらに、3つ目のトランスフォーメーションのキャッシュサイズを500MBと指定します。Integration Service では、合計 1,300MB のメモリが割り当てられます。

キャッシュのパーティション化が使用される場合、Integration Service ではセッション内のすべてのトランスフォーメーションに、自動キャッシュメモリに指定した最大キャッシュサイズが割り当てられ、そのパーティションのすべてに、各トランスフォーメーションのキャッシュメモリが分割されます。

## 数値のキャッシュサイズの設定

キャッシュサイズに特定の値を設定できます。キャッシュサイズを調整するときに、特定の値を設定します。キャッシュサイズを最初に設定するときに、キャッシュの計算または自動キャッシュメモリを使用できます。キャッシュサイズを設定し、セッションを実行した後、セッションログでトランスフォーメーションの統計を分析して、キャッシュサイズを調整できます。セッションログには、ディスクにページングすることなくメモリでトランスフォーメーションを処理するのに必要となるキャッシュサイズが表示されます。セッションのパフォーマンスを最適にするには、セッションログに指定されたキャッシュサイズを使用します。

## キャッシュサイズを設定する手順

セッションプロパティで、トランスフォーメーションのキャッシュサイズを設定できます。Integration Service でキャッシュのパーティション化が使用されない場合に限り、キャッシュサイズを設定するときに、トランスフォーメーションが必要とする合計サイズを指定します。

Integration Services でキャッシュのパーティション化が使用される場合は、別のキャッシュサイズを設定します。Integration Services でキャッシュのパーティション化が使用される場合にキャッシュサイズを計算するには、トランスフォーメーションに必要な合計サイズを計算して、パーティションの数で割ります。

セッションのキャッシュサイズを設定するには、以下の手順を実行します。

1. Workflow Manager でセッションを開きます。
2. [マッピング] タブをクリックします。
3. 左ペインでマッピングオブジェクトを選択します。

[マッピング] タブの右ペインに、キャッシュサイズを設定できるオブジェクトプロパティが表示されます。

4. キャッシュサイズを設定するには、以下のいずれかの方法を使用します。

キャッシュサイズの値を入力し、[OK] をクリックして、手順 8 に進みます。デフォルトでは、値はすべてバイト単位で入力します。ただし、値を入力するときに、KB、MB、GB のいずれかの単位を指定できます。単位を入力する場合は、値と単位の間にスペースを入力しないでください。たとえば、350000KB、200MB、1GB と入力します。

-または-

キャッシュサイズに「自動」と入力し、[OK] をクリックして、手順 8 に進みます。

-または-

[開く] ボタンをクリックして、キャッシュの計算を開きます。

5. モードを選択します。

トランスフォーメーションに割り当てられるキャッシュの合計を制限するには、[自動] モードを選択します。手順 8 に進みます。

-または-

トランスフォーメーションに必要なメモリの合計を計算するには、[計算] モードを選択します。

6. トランスフォーメーションタイプに基づいて入力値を指定し、[計算] をクリックします。

**注:** 入力値が大きすぎてキャッシュの計算に値を入力できない場合は、自動メモリキャッシュを使用します。

キャッシュの計算によって、キロバイト単位でキャッシュサイズが計算されます。

7. トランスフォーメーションにデータキャッシュおよびインデックスキャッシュがある場合は、[データキャッシュサイズ] か [インデックスキャッシュサイズ]、またはその両方を選択します。
8. [OK] をクリックして、手順 7 で選択したキャッシュサイズに、計算した値を適用します。

## キャッシュのパーティション化

複数のパーティションを持つセッションを作成した場合、Integration Service ではアグリゲータ、ジョイナ、ルックアップ、ランク、ソータの各トランスフォーメーションについて、キャッシュのパーティション化が使用される場合があります。キャッシュをパーティション化する場合、Integration Service ではパーティションごとに個別のキャッシュが作成され、各パーティションに設定済みのキャッシュサイズが割り当てられます。Integration Service では各キャッシュに異なるデータが格納されます。この場合、各キャッシュには、そのパーティションが必要とする行のみが格納されます。したがって、Integration Service には各パーティションのキャッシュメモリ全体の一部が必要です。

Integration Service でキャッシュのパーティション化が使用されている場合、各パーティションのキャッシュへのアクセスは並列的に行われます。キャッシュのパーティション化が使用されていない場合、各パーティションのキャッシュへのアクセスは直列的に行われます。

以下の表に、適用可能なそれぞれのトランスフォーメーションに対して Integration Service が使用するキャッシュのパーティション化を示します。

| トランスフォーメーション       | 説明                                                                                       |
|--------------------|------------------------------------------------------------------------------------------|
| アグリゲータトランスフォーメーション | アグリゲータトランスフォーメーションを含むセッションに複数のパーティションを作成します。アグリゲータトランスフォーメーションにパーティションポイントを設定する必要はありません。 |
| ジョイナトランスフォーメーション   | ジョイナトランスフォーメーションにパーティションポイントを作成します。                                                      |
| ルックアップトランスフォーメーション | ルックアップトランスフォーメーションに自動ハッシュキーパーティションポイントを作成します。                                            |
| ランクトランスフォーメーション    | ランクトランスフォーメーションを含むセッションに複数のパーティションを作成できます。ランクトランスフォーメーションにパーティションポイントを設定する必要はありません。      |
| ソータトランスフォーメーション    | ソータトランスフォーメーションを含むセッションに複数のパーティションを作成できます。ソータトランスフォーメーションにパーティションポイントを設定する必要はありません。      |

## キャッシュのパーティション化用のキャッシュサイズの設定

Integration Service によってキャッシュのパーティション化が使用されている場合、メモリ要件は別の方法で設定します。Integration Service によってキャッシュのパーティション化が使用されている場合、各パーティションには設定されたキャッシュサイズが割り当てられます。キャッシュのパーティション化を使用するトラ

ンスフォーメーションのメモリ要件を設定するには、トランスフォーメーションに必要な合計サイズを計算し、それをパーティション数で割ります。

例えば、アグリゲータトランスフォーメーションで1つのセッションにパーティションを4つ作成します。アグリゲータトランスフォーメーションに400MBのデータキャッシュが必要になるとします。アグリゲータトランスフォーメーションのデータキャッシュサイズに、100MBを設定します。セッションを実行する際に、Integration Service は、アグリゲータトランスフォーメーションに合計400MBを使用して、各パーティションに100MBを割り当てます。

キャッシュの計算を使用して、トランスフォーメーションに必要な合計サイズを計算します。動的パーティション化を使用する場合、動的パーティション化の方法に基づいて、パーティションの数を決定できます。グリッドのノードに基づいた動的パーティション化を使用する場合は、Integration Service によってノードごとに1つのパーティションが作成されます。ソースのパーティション化に基づいた動的パーティション化を使用する場合は、ソースデータベースのパーティション数を使用します。

## アグリゲータキャッシュ

Integration Service ではキャッシュメモリを使用して、入力が未ソートのアグリゲータトランスフォーメーションを処理します。セッションを実行すると、Integration Service では集計の計算が完了するまで、メモリにデータが格納されます。

Integration Service では、アグリゲータトランスフォーメーションに対して以下のキャッシュが作成されます。

- **インデックスキャッシュ。** GroupBy ポートの設定に従ってグループ値を格納します。
- **データキャッシュ。** GroupBy ポートに基づいて計算結果を格納します。

デフォルトでは、Integration Service によって、トランスフォーメーションのデータとインデックスの両方に対して、メモリキャッシュとディスクキャッシュが1つずつ作成されます。

アグリゲータトランスフォーメーションで1つのセッションに複数のパーティションを作成すると、Integration Service ではキャッシュのパーティション化が使用されます。すべてのパーティションにディスクキャッシュが1つ、各パーティションに別々のメモリキャッシュが作成されます。

## 差分集計

差分集計セッションを初めて実行するときに、Integration Service ではソース全体が処理されます。セッションの最後で、Integration Service によって、集計したデータが2つのキャッシュファイル（インデックスキャッシュファイルとデータキャッシュファイル）に格納されます。Integration Service ではキャッシュファイルがキャッシュファイルディレクトリに格納されます。次回セッションを実行するときに、Integration Service ではキャッシュファイルにキャッシュされた集計値を使用して、新しい行が集計されます。

差分アグリゲータトランスフォーメーションを持つセッションを実行した場合、Integration Service によりセッションの開始時に\$PMCacheDir にアグリゲータキャッシュファイルのバックアップが作成されます。Integration Service では、セッションリカバリの実行開始時に、バックアップキャッシュが初期キャッシュへプロモートされます。セッションが強制終了した場合、Integration Service はバックアップキャッシュファイルをリストアできません。

差分集計を使用したセッションで複数のパーティションを作成すると、Integration Service では各パーティションに1つのキャッシュファイルが作成されます。

## アグリゲータトランスフォーメーションのキャッシュサイズの設定

未ソートのポートを使用したアグリゲータトランスフォーメーションのキャッシュサイズを設定します。

ソート済みポートを使用するアグリゲータトランスフォーメーションでは、キャッシュメモリを設定する必要はありません。Integration Service ではシステムメモリを使用して、ソート済みのポートを使用したアグリゲータトランスフォーメーションが処理されます。

以下の表に、アグリゲータのキャッシュサイズを計算する場合に入力する値の説明を示します。

| オプション名   | 説明                                                                                                                                                                                                                            |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| グループ数    | グループの数です。アグリゲータトランスフォーメーションでは、グループ別にデータが集計されます。ポート別のグループを使用して、グループ数を計算します。たとえば、Store ID と Item ID 別にグループ化したときに、Store が 5 個、Item が 25 個あり、各 Store に 25 個の Item がすべて含まれる場合は、以下のようにグループ数を計算します。<br>$5 * 25 = 125 \text{ groups}$ |
| データ移動モード | Integration Service のデータ移動モードです。キャッシュ要件はデータ移動モードによって異なります。各 ASCII 文字には、1 バイトが使用されます。各 Unicode 文字には、2 バイトが使用されます。                                                                                                              |

入力値を入力して [計算] をクリックすると、データおよびソータキャッシュサイズが計算されます。[データキャッシュサイズ] フィールドおよび [インデックスキャッシュサイズ] フィールドに、計算値が表示されます。

## アグリゲータキャッシュのトラブルシューティング

このセッションの情報を使用すると、アグリゲータトランスフォーメーションのキャッシュのトラブルシューティングに役立ちます。

キャッシュの計算によって、アグリゲータトランスフォーメーションのキャッシュサイズを計算するときに、以下の警告が表示されます。

CMN\_2019 Warning: The estimated data cache size assumes the number of aggregate functions equals the number of connected output-only ports. If there are more aggregate functions, increase the cache size to cache all data in memory.

アグリゲータトランスフォーメーションでは、1 つ以上の集計関数を使用できます。出力が 1 つの集計関数に基づいている場合、キャッシュの計算によってキャッシュサイズの見積もりが行われます。複数の集約関数を使用して、1 つの出力ポートの値を決定する場合は、キャッシュサイズを増加する必要があります。

セッションログでトランスフォーメーションの統計を確認して、セッション内のアグリゲータトランスフォーメーションのキャッシュサイズを調整します。

## ジョイナキャッシュ

Integration Service ではキャッシュメモリを使用して、ジョイナトランスフォーメーションが処理されます。セッションを実行するときに、Integration Service によってマスターソースと明細ソースから同時に行が読み込まれ、マスター行に基づいてインデックスキャッシュおよびデータキャッシュが構築されます。Integration Service では、明細ソースデータおよびキャッシュされたマスターデータに基づいて結合が実行されます。



Integration Service では、ジョイナトランスフォーメーションのタイプに基づいて、さまざまな数の行が格納されます。

以下の表に、Integration Service でさまざまなタイプのジョイナトランスフォーメーションに対してキャッシュに格納される情報を示します。

| ジョイナトランスフォーメーションのタイプ | インデックスキャッシュ                                                                                                                                                                                                                                                                                             | データキャッシュ                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 未ソートの入力              | ジョイン条件に、一意のインデックスキーを持つすべてのマスター行を格納します。                                                                                                                                                                                                                                                                  | すべてのマスター行を格納します。                                                                                                                                             |
| 別のソースを使用したソート済み入力    | ジョイン条件に、一意のインデックスキーを持つマスター行を 100 行格納します。                                                                                                                                                                                                                                                                | インデックスキャッシュに格納された行に対応するマスター行を格納します。マスターデータに同じキーを持つ複数の行が含まれる場合、Integration Service によって 100 件以上の行がデータキャッシュに格納されます。                                            |
| 同じソースを使用したソート済み入力    | ジョイン条件に、一意のインデックスキーを持つすべてのマスター行または明細行を格納します。Integration Service で、マスターパイプラインよりも高速な明細パイプラインを処理する場合、明細行を格納します。それ以外の場合は、マスター行を格納します。格納される行数は、マスターパイプラインと明細パイプラインの処理の速さによって異なります。1 つのパイプラインによる行の処理が他のパイプラインよりも速い場合、Integration Service では、既に処理されたすべての行がキャッシュに格納され、他のパイプラインによる行の処理が終了するまでキャッシュに格納されたままになります。 | インデックスキャッシュに格納された行に、データを格納します。インデックスキャッシュにマスターパイプラインのキーが格納されると、データキャッシュにはマスターパイプラインのデータが格納されます。インデックスキャッシュに明細パイプラインのキーが格納されると、データキャッシュには明細パイプラインのデータが格納されます。 |

データが格納されると、Integration Service によって、すべてのパーティションに 1 つのディスクキャッシュ、各パーティションに別々のメモリキャッシュが作成されます。行のデータが結合された後に、キャッシュから各行が開放されます。

データが格納されず、ジョイナトランスフォーメーションにパーティションがない場合は、Integration Service によって各パーティションに 1 つのディスクキャッシュおよび別々のメモリキャッシュが作成されます。データが格納されず、ジョイナトランスフォーメーションにパーティションがある場合は、Integration Service によって各パーティションに別々のディスクキャッシュおよびメモリキャッシュが作成されます。データが格納されない場合、Integration Service では、すべてのデータが結合されるまでキャッシュにすべてのマスターデータが保持されます。

セッションに複数のパーティションを作成する場合、1:n のパーティション化または n:n のパーティション化を使用できます。Integration Service では、1:n のパーティション化を使用するときと、n:n パーティション化を使用するときではそれぞれ異なる方法でジョイナトランスフォーメーションが処理されます。

## 1:n のパーティション化

ソート済み入力を使用したジョイナトランスフォーメーションでは、1:n のパーティション化を使用できます。1:n のパーティション化を使用するときは、マスターパイプラインに 1 つのパーティション、明細パイプラインに複数のパーティションを作成します。Integration Service で結合処理を行うときに、明細パーティションの行とマスターソースの行が比較されます。アウタージョインでマスターデータおよび明細データを処理す

るときには、すべての明細パーティションが処理された後に、Integration Service によって一致しないマスタ行が出力されます。

## n:n のパーティション化

ソート済みまたは未ソートの入力を使用したジョイナトランスフォーメーションでは、 $n:n$  のパーティション化を使用できます。ジョイナトランスフォーメーションで  $n:n$  のパーティション化を使用するときは、マスターパイプラインおよび明細パイプラインに  $n$  個のパーティションを作成します。Integration Service で結合処理を行うときに、明細パーティションの行と対応するマスターパーティションの行とが比較され、その他のマスターパーティションの行は無視されます。アウタージョインでマスタデータおよび明細データを処理するときには、各明細キャッシュのパーティションが処理された後に、Integration Service によって一致しないマスタ行が出力されます。

**ヒント:** マスタソースに数多くの行がある場合は、セッションのパフォーマンスをより良くするために、 $n:n$  のパーティション化を使用します。

$n:n$  のパーティション化を使用するには、セッションに複数のパーティションを作成し、ジョイナトランスフォーメーションにパーティションポイントを作成する必要があります。ジョイナトランスフォーメーションにパーティションポイントを作成して、ジョイナトランスフォーメーションのマスタソースと明細ソースの両方に、複数のパーティションを作成します。

ジョイナトランスフォーメーションにパーティションポイントを作成する場合、Integration Service ではキャッシュのパーティション化が使用されます。各パーティションにメモリキャッシュが 1 つ作成されます。各パーティションのメモリキャッシュには、そのパーティションに必要な行のみが含まれます。したがって、Integration Service には各パーティションのキャッシュメモリ全体の一部が必要です。

## ジョイナトランスフォーメーションのキャッシュサイズの設定

ジョイナトランスフォーメーションのセッションプロパティに、インデックスキャッシュおよびデータキャッシュのサイズを設定できます。

1:n のパーティション化を使用すると、Integration Service によって各パーティションにメモリキャッシュが複製されます。各パーティションには、トランスフォーメーションに必要な合計と同じ容量のメモリが必要です。1:n のパーティション化を使用してジョイナトランスフォーメーションのキャッシュサイズを設定するときは、トランスフォーメーションに必要な合計にキャッシュサイズを設定します。

$n:n$  のパーティション化を使用するときは、各パーティションにはトランスフォーメーションの処理に必要な合計メモリの一部が必要です。 $n:n$  のパーティション化を使用してジョイナトランスフォーメーションのキャッシュサイズを設定するときは、トランスフォーメーションに必要な合計を計算して、それをパーティション数で割ります。

キャッシュの計算を使用して、トランスフォーメーションの処理に必要なキャッシュサイズを決定できます。たとえば、キャッシュの計算を使用して、ジョイナトランスフォーメーションではインデックスキャッシュ用に 2,000,000 バイトのメモリ、データキャッシュ用に 4,000,000 バイトのメモリが必要であることを決定します。また、パイプラインに対して 4 つのパーティションを作成したとします。1:n のパーティション化を使用する場合は、インデックスキャッシュに 2,000,000 バイト、データキャッシュに 4,000,000 バイトを設定します。 $n:n$  のパーティション化を使用する場合は、インデックスキャッシュに 500,000 バイト、データキャッシュに 1,000,000 バイトを設定します。

以下のテーブルで、ジョイナのキャッシュサイズを計算するために入力する値について説明します。

| 入力       | 説明                                                                                                                                                                         |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| マスタ行の数   | マスターソース内の行数。未ソートの入力を使用したジョイナトランスフォーメーションに適用します。マスター行数は、ソート済みのジョイナトランスフォーメーションのキャッシュサイズには影響しません。<br><b>注:</b> マスタソースの行が一意的キーを共有する場合は、キャッシュの計算によってインデックスキャッシュのサイズが過大に概算されます。 |
| データ移動モード | Integration Service のデータ移動モードです。キャッシュ要件はデータ移動モードによって異なります。ASCII 文字は 1 バイトを使用します。Unicode 文字は 2 バイトを使用します。                                                                   |

入力値を入力して [計算] をクリックすると、データおよびソータキャッシュサイズが計算されます。[データキャッシュサイズ] フィールドおよび [インデックスキャッシュサイズ] フィールドに、計算値が表示されます。

## ジョイナキャッシュのトラブルシューティング

このセッションの情報を使用すると、ジョイナトランスフォーメーションのキャッシュのトラブルシューティングに役立ちます。

**ソート済み入力を使用するジョイナトランスフォーメーションのキャッシュサイズを計算するときに、キャッシュの計算によって以下の警告が表示されます。**

CMN\_2020 Warning: If the master and detail pipelines of a sorted Joiner transformation are from the same source, the Integration Service cannot determine how fast it will process the rows in each pipeline. As a result, the cache size estimate may be inaccurate.

マスターパイプラインと明細パイプラインとで、同時に行が処理されます。同じソースのデータを結合する場合、パイプラインでは異なる速さで行が処理されることがあります。1 つのパイプラインによる行の処理が他のパイプラインよりも速い場合、Integration Service では、既に処理されたすべての行がキャッシュに格納され、他のパイプラインによる行の処理が終了するまでキャッシュに格納されたままになります。キャッシュされた行の合計は、2 つのパイプライン間での処理の速さの違いによって異なります。

最適なセッションパフォーマンスを実現するために、キャッシュされた行をすべて格納するのに十分なキャッシュサイズが必要です。キャッシュサイズが十分に大きくない場合は、サイズを増加します。

**注:** 異なるソースのデータを結合するときにこのメッセージが表示される場合でも、同じソースのデータを結合すると表示されます。

**ソート済み入力を使用するジョイナトランスフォーメーションのキャッシュサイズを計算するときに、キャッシュの計算によって以下の警告が表示されます。**

CMN\_2021 Warning: Increase the data cache size if the sorted Joiner transformation processes master rows that share the same key. To determine the new cache size, divide the existing cache size by 2.5 and multiply the result by the average number of master rows per key.

ソート済み入力を使用したジョイナトランスフォーメーションのキャッシュサイズを計算する場合、キャッシュの計算では、一意キーごとに平均 2.5 のマスター行という要件に基づいてキャッシュが見積もられます。一意キーごとのマスター行の平均数が 2.5 よりも大きい場合、それに応じてキャッシュサイズを増加します。たとえば、一意キーごとのマスター行の平均数が 5 (2.5 の 2 倍のサイズ) の場合、キャッシュの計算によって計算されたキャッシュサイズを 2 倍にします。



# ルックアップキャッシュ

ルックアップトランスフォーメーションでキャッシュを有効にした場合、Integration Service ではルックアップデータを格納するためのキャッシュがメモリに構築されます。Integration Service によってメモリにルックアップキャッシュが構築されると、トランスフォーメーションの最初のデータ行が処理され、トランスフォーメーションを入力する各行に対してキャッシュのクエリが実行されます。キャッシュが有効でない場合は、Integration Service では各入力行に対してルックアップソースのクエリが実行されます。

ルックアップソースをキャッシュに格納するかどうかに関わらず、ルックアップクエリの結果および処理は同じです。ただし、ルックアップキャッシュを使用するとセッションのパフォーマンスを向上させることができます。ソースが大きい場合は、ルックアップソースをキャッシュすることによってパフォーマンスを最適化できます。

セッションとセッションの間にルックアップが変更されない場合は、永続ルックアップキャッシュを使用するようにトランスフォーメーションを設定できます。セッションの実行時、キャッシュファイルが存在しない場合やキャッシュファイルが無効である場合、Integration Service は永続キャッシュを構築します。

Integration Service では、ルックアップトランスフォーメーションに対して以下のキャッシュが作成されます。

- **データキャッシュ。**ルックアップトランスフォーメーションが接続されている場合、接続された出力ポートで、ルックアップ条件で使用されるポート以外のデータを格納します。ルックアップトランスフォーメーションが接続されていない場合、戻りポートのデータを格納します。
- **インデックスキャッシュ。**ルックアップ条件で使用されるカラムのデータを格納します。

Integration Service では、ルックアップキャッシュおよびパーティション化の情報に基づいて、ディスクおよびメモリキャッシュが作成されます。

以下のテーブルでは、キャッシュおよびパーティション化の情報に基づいて Integration Service で作成されるキャッシュについて説明します。

| ルックアップ条件                                      | ディスクキャッシュ                    | メモリキャッシュ                    |
|-----------------------------------------------|------------------------------|-----------------------------|
| - 静的キャッシュ<br>- 自動ハッシュキーパーティションなし              | すべてのパーティションに対して1つのディスクキャッシュ。 | パーティションごとに1つのメモリキャッシュ。      |
| - 動的キャッシュ<br>- 自動ハッシュキーパーティションなし              | すべてのパーティションに対して1つのディスクキャッシュ。 | すべてのパーティションに対して1つのメモリキャッシュ。 |
| - 静的キャッシュまたは動的キャッシュ<br>- 自動ハッシュキーのパーティションポイント | パーティションごとに1つのディスクキャッシュ。      | パーティションごとに1つのメモリキャッシュ。      |

ルックアップトランスフォーメーションを使用してセッションに複数のパーティションを作成し、ルックアップトランスフォーメーションに自動ハッシュキーのパーティションポイントを作成すると、Integration Service ではキャッシュのパーティション化が使用されます。

Integration Service でキャッシュのパーティション化が使用されると、任意のパーティションの最初の行がルックアップトランスフォーメーションに到達したときに、ルックアップトランスフォーメーションのキャッシュが作成されます。コンカレントキャッシュ用にルックアップトランスフォーメーションを設定した場合、Integration Service によってパーティションのすべてのキャッシュが同時に構築されます。

## キャッシュの共有

Integration Service では、キャッシュが静的か動的かどうかによって、共有されたルックアップキャッシュの処理が異なります。

- **静的キャッシュ。** 2つのルックアップトランスフォーメーションで1つの静的キャッシュを共有する場合、Integration Service では、同じパイプラインステージで共有されたトランスフォーメーションに対して追加メモリの割り当てを行いません。異なるパイプラインステージで共有されるトランスフォーメーションについては、Integration Service によって追加メモリが割り当てられます。

同じデータまたはデータのサブセットを使ってディスクキャッシュを作成する静的なルックアップトランスフォーメーションであれば、複数のルックアップトランスフォーメーションがディスクキャッシュを共有できます。ただし、ルックアップキーは異なる場合があるため、各トランスフォーメーションは別々のメモリキャッシュを持つ必要があります。

- **動的キャッシュ。** ルックアップトランスフォーメーションが1つの動的キャッシュを共有する場合、Integration Service はメモリキャッシュとディスクキャッシュを更新します。キャッシュの同期を保つために、Integration Service はディスクキャッシュとそれに対応するメモリキャッシュをトランスフォーメーション間で共有する必要があります。

## ルックアップトランスフォーメーションのキャッシュサイズの設定

セッションプロパティで、ルックアップトランスフォーメーションのキャッシュサイズを設定できます。

以下のテーブルで、ルックアップのキャッシュサイズを計算するために入力する値について説明します。

| 入力               | 説明                                                                                                       |
|------------------|----------------------------------------------------------------------------------------------------------|
| 一意のルックアップキーを持つ行数 | 一意のルックアップキーを持つルックアップソースの行数です。                                                                            |
| データ移動モード         | Integration Service のデータ移動モードです。キャッシュ要件はデータ移動モードによって異なります。ASCII 文字は 1 バイトを使用します。Unicode 文字は 2 バイトを使用します。 |

入力値を入力して [計算] をクリックすると、データおよびソータキャッシュサイズが計算されます。[データキャッシュサイズ] フィールドおよび [インデックスキャッシュサイズ] フィールドに、計算値が表示されます。

## ランクキャッシュ

Integration Service ではキャッシュメモリを使用して、ランクトランスフォーメーションが処理されます。ランク付けが完了するまで、ランクメモリにデータが格納されます。

Integration Service は、ランクトランスフォーメーションでセッションを実行するとき、入力行とデータキャッシュの行を比較します。格納されている行よりも入力行の方がランクが高くなった場合に、Integration Service は格納されている行を入力行に置き換えます。

例えば、売り上げ高の上位 3 位までを検索するランクトランスフォーメーションを設定します。Integration Service は、以下の入力データを読み込みます。

**SALES**  
10,000  
12,210

5,000  
2,455  
6,324

Integration Service は最初の 3 つの行（10,000、12,210、および 5,000）をキャッシュに格納します。次に Integration Service はその次の行（2,455）を読み込んでキャッシュ内の値と比較します。この行はキャッシュに格納されている行よりもランクが低いため、サーバーは 2,455 の値を持つこの行を削除します。しかしその次の行（6,324）はキャッシュ内の行の 1 つよりも高いランクを持ちます。したがって、Integration Service はキャッシュされた行をより上位のランクを持つ入力行に置き換えます。

複数のグループにまたがってランク付けを行うようにランクトランスフォーメーションが設定されている場合、Integration Service は検索したそれぞれのグループに対して同じやり方でランク付けを行い、ランクの数字は増加していきます。

Integration Service では、ランクトランスフォーメーションに対して以下のキャッシュが作成されます。

- **データキャッシュ。**グループ化ポートに基づいてランキング情報を格納します。
- **インデックスキャッシュ。**GroupBy ポートの設定に従ってグループ値を格納します。

デフォルトでは、Integration Service はすべてのパーティションに対して 1 つのメモリキャッシュとディスクキャッシュを作成します。

セッションに複数のパーティションを作成した場合、Integration Service ではキャッシュのパーティション化が使用されます。ランクトランスフォーメーションに 1 つのディスクキャッシュ、各パーティションに 1 つのメモリキャッシュが作成され、トランスフォーメーションのグループキー値に基づいてパーティションから別のパーティションにデータがルーティングされます。

## ランクトランスフォーメーションのキャッシュサイズの設定

セッションプロパティで、ランクトランスフォーメーションのキャッシュサイズを設定できます。

以下の表に、ランクのキャッシュサイズを計算するために入力する値を示します。

| 入力       | 説明                                                                                                                                                                                                                            |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| グループ数    | グループの数です。ランクトランスフォーメーションでは、グループ別にデータをランク付けします。ポート別のグループを使用して、グループの数を決定します。たとえば、Store ID と Item ID 別にグループ化したときに、Store が 5 個、Item が 25 個あり、各 Store に 25 個の Item すべてが含まれる場合、以下のようにグループの数を計算します。<br>$5 * 25 = 125 \text{ groups}$ |
| ランク数     | ランク付けの項目数です。たとえば、上位 10 位の売り上げをランク付けする場合は、10 個のランクがあります。キャッシュの計算では、ランクトランスフォーメーションで設定した値に基づいて、この値が入力されます。                                                                                                                      |
| データ移動モード | Integration Service のデータ移動モードです。キャッシュ要件はデータ移動モードによって異なります。ASCII 文字は 1 バイトを使用します。Unicode 文字は 2 バイトを使用します。                                                                                                                      |

入力値を入力して [計算] をクリックすると、データおよびソータキャッシュサイズが計算されます。[データキャッシュサイズ] フィールドおよび [インデックスキャッシュサイズ] フィールドに、計算値が表示されます。

## ソータキャッシュ

Integration Service ではキャッシュメモリを使用して、ソータトランスフォーメーションが処理されます。Integration Service は、ソート操作を実行する前に、すべての受信データをソータトランスフォーメーションに渡します。

Integration Service ではデータのソート中に、ソートキーおよびデータを格納するためのソータキャッシュが作成されます。デフォルトでは、Integration Service はすべてのパーティションに対して 1 つのメモリキャッシュとディスクキャッシュを作成します。

セッションに複数のパーティションを作成した場合、Integration Service ではキャッシュのパーティション化が使用されます。ソータトランスフォーメーションに 1 つのディスクキャッシュ、各パーティションに 1 つのメモリキャッシュが作成されます。Integration Service によって、各パーティションに別々のキャッシュが作成され、各パーティションが別々にソートされます。

メモリ内のすべてのデータをソートするようにキャッシュサイズを設定していない場合、Integration Service によってソースデータに複数のパスが作成されたことを示す警告が、セッションログに表示されます。ソートを完了するためにディスクに情報をページングする必要がある場合、Integration Service によってデータに複数のパスが作成されます。Integration Service でデータが 1 回読み込まれ、ディスクにページングしないでメモリでソートが実行される場合に、このメッセージには単一パスに必要なバイト数が表示されます。セッションのパフォーマンスを向上させるには、Integration Service によってデータに 1 つのパスが作成されるように、キャッシュサイズを設定します。

## ソータトランスフォーメーションのキャッシュサイズの設定

セッションプロパティで、ソータトランスフォーメーションのソータキャッシュを設定できます。

以下の表に、ソータのキャッシュサイズを計算するために入力する値を示します。

| 入力       | 説明                                                                                                       |
|----------|----------------------------------------------------------------------------------------------------------|
| 行数       | 行数。                                                                                                      |
| データ移動モード | Integration Service のデータ移動モードです。キャッシュ要件はデータ移動モードによって異なります。ASCII 文字は 1 バイトを使用します。Unicode 文字は 2 バイトを使用します。 |

ソータキャッシュサイズを計算するには、入力値を入力して、[計算] をクリックします。計算された値が、[ソータキャッシュサイズ] フィールドに表示されます。

## XML ターゲットキャッシュ

Integration Service ではキャッシュメモリを使用して、XML ターゲットが作成されます。Integration Service では、XML ターゲットの生成中に、データおよび XML 階層がキャッシュメモリに格納されます。

Integration Service によって、XML ターゲットに対して以下のタイプのキャッシュが作成されます。

- **データキャッシュ。**XML ターゲットドキュメントの生成中に、XML 行データを格納します。すべてのグループに対して 1 つのデータキャッシュを格納します。
- **インデックスキャッシュ。**プライマリキーまたは外部キーを格納します。各グループにプライマリキーインデックスキャッシュおよび外部キーインデックスキャッシュを作成します。

## XML ターゲットのキャッシュサイズの設定

ターゲットプロパティまたはセッションプロパティで、XML ターゲットのキャッシュサイズを設定します。デフォルトでは、キャッシュサイズが「自動」に設定されます。必要なキャッシュメモリ容量は、Integration Service のランタイムに決定されます。

また、キャッシュサイズを設定して、キャッシュメモリ容量をバイト単位で指定することもできます。キャッシュサイズを計算するには、次の手順を完了します。

1. 各グループ内の行数を見積もります。
2. 次の計算式を使用して、各グループのキャッシュサイズを計算します。  
$$\text{Group cache size} = \text{Data cache size} + \text{Primary key index cache size} + \text{Foreign key index cache size}$$
3. 次の計算式を使用して合計キャッシュサイズを計算します：  
$$\text{合計キャッシュサイズ} = \text{Sum (すべてのグループのキャッシュサイズ)}$$

グループのデータキャッシュのサイズを計算する式を次に示します。

$(\text{グループ内の行数}) \times (\text{グループの行サイズ})$

グループのプライマリーキーツリーのサイズを計算する式を次に示します。

$(\text{グループ内の行数}) \times (\text{プライマリーキーインデックスのキャッシュサイズ})$

グループの外部キーツリーのサイズを計算する式を次に示します。

$\text{Sum} ((\text{親グループ内の行数}) \times (\text{外部キーインデックスのキャッシュサイズ}))$

**注:** キャッシュの計算を使用して、XML ターゲットのキャッシュサイズを設定することはできません。

## キャッシュサイズの最適化

セッションパフォーマンスを最適にするには、ディスクにページングしないでメモリでトランスフォーメーションが処理されるように、Integration Service のキャッシュサイズを設定します。Integration Service でディスクにページングされると、セッションパフォーマンスが減少します。

キャッシュの計算を使用してキャッシュサイズを計算すると、キャッシュの計算では、入力値に基づいて最適なセッションパフォーマンスに必要なキャッシュサイズが見積もられます。セッションログに指定されたキャッシュサイズを使用して、その見積もりを調整できます。セッションの実行後に、セッションログのトランスフォーメーション統計を確認して、キャッシュサイズを所得します。

たとえば、AGGTRANS というアグリゲータトランスフォーメーションを実行します。セッションログには、以下のテキストが含まれます。

```
MAPPING> TT_11031 Transformation [AGGTRANS]:
MAPPING> TT_11114 [AGGTRANS]: Input Group Index = [0], Input Row Count [110264]
MAPPING> TT_11034 [SQ_V_PETL]: Input - 110264
MAPPING> TT_11115 [AGGTRANS]: Output Group Index = [0]
MAPPING> TT_11037 [FILTRANS]: Output - 1098,Dropped - 0
MAPPING> CMN_1791 The index cache size that would hold [1098] aggregate groups of input rows for [AGGTRANS],
in memory, is [286720] bytes
MAPPING> CMN_1790 The data cache size that would hold [1098] aggregate groups of input rows for [AGGTRANS], in
memory, is [1774368] bytes
```

このログは、ディスクにページングしないでメモリでトランスフォーメーションを処理するには、インデックスキャッシュには 286,720 バイト、データキャッシュには 1,774,368 バイトが必要なことを示しています。

キャッシュサイズは、セッションまたはソースデータへの変更に応じて異なる場合があります。後続セッションの実行後にセッションログを確認して、キャッシュサイズへの変更を監視します。

Integration Service でトランスフォーメーション統計のセッションログへの書き込みを有効にするには、セッションプロパティのトレースレベルを [Verbose Initialization] に設定する必要があります。

**注:** セッションログには、ソータ、ソート済み入力を使用したジョイナトランスフォーメーション、ソート済み入力を使用したアグリゲータトランスフォーメーション、または XML ターゲットのトランスフォーメーション統計は含まれません。

## 第 21 章

# 差分集計

この章では、以下の項目について説明します。

- [差分集計の概要, 327 ページ](#)
- [差分集計のための Integration Service の処理, 328 ページ](#)
- [集計ファイルの再初期化, 328 ページ](#)
- [集計ファイルの移動と削除, 329 ページ](#)
- [差分集計を使用したパーティション化のガイドライン, 330 ページ](#)
- [差分集計のための準備, 330 ページ](#)

## 差分集計の概要

差分集計では、ソース内から取得した変更をセッションで集計計算に適用します。ソースの変更が差分であり、その変更を取得できる場合は、その変更を処理するようにセッションを設定できます。この結果、Integration Service は、セッションを実行するたびにソース全体を処理して同じデータを再計算する代わりに、ターゲットの差分だけを更新することができます。

たとえば、新しいデータを毎日受け取るソースを使用するセッションがあるとします。既存のデータをデータフローから取り除くフィルタ条件をマッピングに追加してあり、それによって差分変更を取得することができます。そして、差分集計を有効にします。

このセッションを、差分集計を有効にして 3 月 1 日に初めて実行するときには、ソース全体を使用します。この結果、Integration Service は、必要な集計データを読み込んで格納できます。3 月 2 日にセッションを再び実行すると、3 月 2 日のタイムスタンプが付いたレコード以外はすべてフィルタによって除外されます。Integration Service は、新しいデータを処理し、それに応じてターゲットを更新します。

次の状況での差分集計の使用を検討します。

- **新しいソースデータがキャプチャ可能な場合。**セッションを実行するたびに新しいソースデータをキャプチャできる場合、差分集計を使用します。ストアドプロシージャトランスフォーメーションまたはフィルタトランスフォーメーションを使用して、新しいデータを処理します。
- **差分変更がターゲットを大幅に変更しない場合。**変更がターゲットを大幅に変更しない場合、差分集計を使用します。差分変更のソースによって既存のターゲットの半分以上が変更される場合、セッションで差分集計を使用することで得られる利点がなくなります。この場合には、テーブルを削除してから、完全なソースデータを使ってターゲットを再作成してください。

**注:** マッピングに Percentile 関数または Median 関数が含まれている場合、差分集計は使用しないでください。Integration Service は、これらの関数を処理するときに、セッションプロパティで設定したキャッシュメモリだけでなく、システムメモリも使用します。この結果、Integration Service はディスクキャッシュに、Percentile 関数および Median 関数の差分集計の値を格納しません。



# 差分集計のための Integration Service の処理

差分集計セッションを初めて実行するときに、Integration Service ではソース全体が処理されます。セッションの最後に、Integration Service はセッションの実行によって得た集計データを 2 つのファイル（インデックスファイルとデータファイル）に格納します。Integration Service は、アグリゲータトランスフォーメーションのプロパティで指定したキャッシュディレクトリでこれらのファイルを作成します。

差分集計を含むセッションを以降に実行するたびに、ソースの差分変更をセッションで使用します。入力レコードごとに、Integration Service はインデックスファイル内の履歴情報をチェックして対応するグループがあるかどうかを調べます。対応するグループが見つかった場合、Integration Service はそのグループの集計データを使って差分集計処理を行い、差分変更を保存します。対応するグループが見つからない場合、Integration Service は新しいグループを作成してレコードデータを保存します。

ターゲットに書き込むときには、Integration Service は既存のターゲットに変更を適用します。変更された集計データをインデックスファイルとデータファイルに保存し、次にセッションを実行するときに履歴データとして使用できるようにします。

ソースに大幅な変更があり、かつ Integration Service に将来の差分変更に対応して集計データの保存を継続させたい場合は、既存の集計データを新しい集計データで上書きするように Integration Service を設定します。

差分集計を含むセッションを以降に実行するたびに、Integration Service では、差分集計ファイルのバックアップが作成されます。アグリゲータトランスフォーメーションのキャッシュディレクトリには、これらの 2 つのファイルのセットを格納するのに十分なディスク容量が必要です。

差分集計を使用するセッションをパーティション化する場合、Integration Service では、パーティションごとにキャッシュファイルが 1 セット作成されます。

次のいずれかのタスクを実行した場合、Integration Service では履歴データが使用されるのではなく、新しい集計データが作成されます。

- マッピングの新しいバージョンを保存する。
- 集計キャッシュを再初期化するようにセッションを設定する。
- セッションプロパティでファイルに設定されているパスまたはディレクトリを修正せずに、集計ファイルを移動する。
- 集計ファイルを新しい場所に移動せずに、集計ファイルに対して設定されているパスまたはディレクトリを変更する。
- キャッシュファイルを削除する。
- パーティション数を減らす。

Integration Service によって差分集計ファイルが再構築された場合、前回のファイルに含まれていたデータは失われます。

**注:** ファイルの破損やディスク障害から差分集計ファイルを保護するために、これらのファイルを定期的にバックアップします。

## 集計ファイルの再初期化

ソーステーブルに大幅な変更があった場合、履歴データを使用するのではなく、Integration Service で集計データを新たに作成できます。Integration Service で集計データを新たに作成するには、集計キャッシュを再初期化するようにセッションを設定します。

たとえば、セッションで使うソースが毎日差分で変更され、月に 1 回は完全に変更されるという場合、集計キャッシュの再初期化を行うことができます。その月の新しいソースデータを受け取ったときに、集計キャッシ



ュを再初期化し、既存のターゲットを切り詰め、セッション中に新しいソーステーブルを使用するようにセッションを設定するとします。

集計キャッシュを再初期化するセッションを実行した後は、セッションプロパティを編集して、[集計キャッシュの再初期化] オプションを無効にしてください。[集計キャッシュの再初期化] の選択を取り消さない場合、Integration Service ではセッションを実行するたびに集計キャッシュが上書きされます。

**注:** Windows から UNIX へ移行する場合は、キャッシュを再初期化する必要があります。したがって、コードページの互換性の有無にかかわらず、Latin1 から MSLatin1 へのコードページの変更はできません。

## 集計ファイルの移動と削除

差分集計セッションを実行した後は、集計情報の履歴が格納されているインデックスファイルやデータファイルの移動や変更を避けます。

これらのファイルを別のディレクトリに移動し、Integration Service で集計ファイルを使用する場合は、セッションプロパティでこれらのファイルのパスも変更する必要があります。また、ファイルのパスを変更してもファイルが移動していない場合、次回セッションを実行するときに、Integration Service によってファイルが再構築されます。

特定のセッションまたは Integration Service のプロパティを変更した場合、Integration Service で差分集計ファイルが使用できず、セッションは失敗します。次のいずれかの作業を行う際には、セッションの失敗を回避するために、既存の差分集計ファイルを削除します。

- Integration Service のデータ移動モードを ASCII から Unicode、または Unicode から ASCII に変更する。
- Integration Service のコードページを互換性のないコードページに変更する。
- Integration Service が Unicode モードで動作しているときにセッションのソート順を変更する。
- [高精度 10 進演算を有効にする] セッションオプションを変更する。

## インデックスファイルとデータファイルの検索

デフォルトでは、Integration Service によってインデックスファイルとデータファイルは、Workflow Manager でプロセス変数 \$PMCacheDir に入力されたディレクトリに格納されます。Integration Service は、インデックスファイルに「PMAGG\*.idx\*」という名前を付けます。そして、Integration Service はデータファイルに「PMAGG\*.dat\*」という名前を付けます。

セッションの実行時、Integration Service は、ファイル名をセッションログに書き込みます。ファイルの場所を特定するには、以前のセッションログを調べて、キャッシュファイルの名前とディレクトリを示す SM\_7034 および SM\_7035 の各メッセージを探します。たとえば、セッションログのエントリには、次のようなメッセージが出力されます。

```
MAPPING> SM_7034 Aggregate Information: Index file is [C:\Informatica\PowerCenter8.0\server\infa_shared\Cache\PMAGG8_4_2.idx2]
MAPPING> SM_7035 Aggregate Information: Data file is [C:\Informatica\PowerCenter8.0\server\infa_shared\Cache\PMAGG8_4_2.dat2]
```

# 差分集計を使用したパーティション化のガイドライン

複数のパーティションを含むセッションで差分集計を使用する場合、Integration Service は、パーティションごとにキャッシュファイルを 1 セット作成します。

パーティション数またはキャッシュディレクトリを変更する場合は、次のガイドラインを使用します。

- **パーティション用のキャッシュディレクトリの変更。**パーティション用のディレクトリを変更し、Integration Service でキャッシュファイルを再利用する場合は、変更したディレクトリに関連付けられたパーティション用のキャッシュファイルを移動する必要があります。
  - 最初のパーティション用のディレクトリを変更し、キャッシュファイルを移動しない場合、Integration Service ではすべてのパーティション用にキャッシュファイルが再構築されます。
  - パーティション 2- $n$ 用のディレクトリを変更し、キャッシュファイルを移動しない場合、Integration Service では見つからなかったキャッシュファイルが再構築されます。
- **パーティション数を減らす。**パーティションを削除し、Integration Service でキャッシュファイルを再利用する場合は、削除されたパーティション用のキャッシュファイルを、最初のパーティション用に設定されたディレクトリに移動する必要があります。最初のパーティションのディレクトリにファイルを移動しなかった場合、Integration Service では見つからなかったキャッシュファイルが再構築されます。

**注:** パーティションの数を増やした場合、次回セッションを実行するときに、Integration Service によって、インデックスとデータのキャッシュファイルが再調整されます。これらのファイルを再構築する必要はありません。
- **キャッシュファイルの移動。**パーティション用のキャッシュファイルを移動し、Integration Service でファイルを再利用する場合は、パーティションのディレクトリも変更する必要があります。ディレクトリを変更しなかった場合、次回セッションを実行するときに Integration Service によりファイルが再構築されます。
- **キャッシュファイルを削除する。**キャッシュファイルを削除した場合、次回セッションを実行するときに Integration Service によりファイルが再構築されます。

パーティション数とキャッシュディレクトリを変更した場合、その両方について、キャッシュファイルを移動する必要があります。たとえば、パーティション 1 のキャッシュディレクトリを変更し、パーティションの数を減らした場合、削除したパーティションのキャッシュファイル、および変更したディレクトリに関連付けられたパーティションのキャッシュファイルを移動する必要があります。

## 差分集計のための準備

差分集計を使用する場合、マッピングのプロパティとセッションのプロパティを両方とも設定する必要があります。

- 既存のデータを除外するマッピングロジックまたはフィルタを実装します。
- 差分集計用にセッションを設定し、ファイルディレクトリに集計ファイルを格納できるだけの十分なディスク容量があることを確認します。

## マッピングの設定

差分集計を有効にする前に、ソースデータ内の変更を取得しなければなりません。マッピングでフィルタまたはストアドプロシージャトランスフォーメーションを使用すれば、セッション中に既存のソースデータを削除できます。

## セッションの設定

差分集計用にセッションを設定する場合は、以下のガイドラインを使用します。

- **集計ファイルを格納する場所の確認。** インデックスファイルとデータファイルは、ソースデータに比例して大きくなります。セッションの履歴データを格納するのに十分なディスクスペースがあることを確認してください。

複数の差分集計セッションを実行する場合は、ファイルの格納先を決めてください。その後、Workflow Manager で、プロセス変数 \$PMCacheDir に適切なディレクトリを入力してください。インデックスファイルとデータファイル用に、セッションごとに固有のディレクトリを入力できます。ただし、差分集計を使うすべてのセッションにプロセス変数を使用すれば、\$PMCacheDir を変更するだけで、必要なときにキャッシュディレクトリを簡単に変更できます。

ファイルを移動しないで、キャッシュディレクトリを変更した場合、Integration Service により集計キャッシュが再初期化され、新しい集計データが収集されます。

グリッドでは、Integration Service によって、見つからなかった差分集計ファイルが再構築されます。Integration Service によって差分集計ファイルが再構築された場合、集計の履歴は失われます。

- **セッションプロパティでの差分集計設定の確認。** [プロパティ] タブの [パフォーマンス] 設定で、差分集計のセッションを設定できます。

集計キャッシュを再初期化するように、セッションを設定することもできます。キャッシュの再初期化を選択した場合、Workflow Manager は、Integration Service が既存のキャッシュを上書きすることを示す警告を表示し、セッションの実行後にこのオプションの選択を取り消すように注意を促します。

**注:** [Transaction] をトランスフォーメーション範囲とするアグリゲータトランスフォーメーションがマッピングに含まれる場合、差分集計を使用することはできません。そのようなセッションは、Workflow Manager によって無効のマークが付けられます。

## 第 22 章

# セッションログインタフェース

この章では、以下の項目について説明します。

- [セッションログインタフェースの概要, 332 ページ](#)
- [セッションログインタフェースの実装, 332 ページ](#)
- [セッションログインタフェースの関数, 333 ページ](#)
- [セッションログインタフェースの例, 337 ページ](#)

## セッションログインタフェースの概要

デフォルトでは、Integration Service により、セッションイベントはサービスプロセスが実行されるノード上のバイナリログファイルに書き込みます。さらに、Integration Service により、セッションイベント情報が外部ライブラリに渡されます。外部共有ライブラリ内に、Integration Service がログイベントを書き込む方法の手順を提供することができます。

PowerCenter では、セッションログインタフェースを通してセッションイベント情報へのアクセスが提供されます。共有ライブラリを作成する場合、セッションログインタフェースで提供された関数を実装します。

セッションイベントを書き込む際、Integration Service により、セッションログインタフェースで指定された関数が呼び出されます。作成した共有ライブラリ内の関数は、セッションログインタフェースで定義された関数宣言に一致しなければなりません。

## セッションログインタフェースの実装

セッションイベント情報の処理にカスタム手順を使用するように Integration Service を設定するには、以下の手順を実行します。

1. セッションログインタフェースを実装する共有ライブラリを作成します。
2. Administrator ツールで Integration Service プロパティを設定する際、作成する共有ライブラリの名前に ExportSessionLogLibName プロパティを設定します。

## Integration Service とセッションログインタフェース

Integration Service の ExportSessionLogLibName プロパティを共有ライブラリの名前に設定した場合、Integration Service では、イベントログファイルの作成に加えて、共有ライブラリに定義された手順が実行されます。

Integration Service では、以下の方法で共有ライブラリが使用されます。

1. Integration Service は、共有ライブラリをロードし、INFA\_InitSessionLog()関数を呼び出してから、セッション内の最初のイベントをログに記録します。
2. セッションログファイルにイベントを記録するたびに、Integration Service は INFA\_OutputSessionLog()関数を呼び出し、メッセージ、コード、およびセッション情報を共有ライブラリに渡します。
3. セッションが完了し、最後のイベントが記録されると、Integration Service は INFA\_EndSessionLog()を呼び出し、共有ライブラリをアンロードします。

共有ライブラリが Integration Service によって正しく呼び出されるようにするには、次のガイドラインに従って共有ライブラリを記述します。

## セッションログインタフェースの実装に関するルールおよびガイドライン

セッションログインタフェースを実装するためにコードを書き込む場合は、以下のルールおよびガイドラインを使用します。

- セッションログインタフェース内のすべての関数を実装する必要があります。
- 異常終了を除き、Integration Service からセッションログインタフェース内の関数へのすべての呼び出しはシリアライズされます。Integration Service により、イベントをセッションログに記録する場合に、これらの関数への呼び出しが実行されます。そのため、セッションログインタフェース内の関数を実装する際に、ミューテックスオブジェクトを使用して、一度に 1 つのスレッドしかコードセクションを実行できないようにする必要はありません。
- セッションログインタフェースを UNIX で実装する場合、関数内でシグナル処理を実行しないようにします。これは、関数により PowerCenter が信号を処理する方法を妨害されることがないようにするためです。シグナルハンドラを登録しないか、登録を解除します。
- Integration Service はマルチスレッドプロセスであるため、共有ライブラリをマルチスレッドライブラリとしてコンパイルし、共有ライブラリが正しくロードされるようにする必要があります。

## セッションログインタフェースの関数

セッションログインタフェースの関数からは値は返されません。そのため、セッションログインタフェースの関数に対する Integration Service 呼び出しが原因でセッションが失敗することはありません。

以下の表に、セッションログインタフェースの関数を示します。

| 関数                            | 説明                                                 |
|-------------------------------|----------------------------------------------------|
| INFA_InitSessionLog           | Integration Service がイベントログを書き込むセッションに関する情報を提供します。 |
| INFA_OutputSessionLogMsg      | イベントが記録されるたびに呼び出されます。イベントに関する情報を渡します。              |
| INFA_OutputSessionLogFatalMsg | 異常終了する前に最後のイベントが記録されたとき、呼び出されます。                   |

| 関数                              | 説明                                           |
|---------------------------------|----------------------------------------------|
| INFA_EndSessionLog              | 最後のメッセージがセッションログに送られ、セッションが正常に終了した後に呼び出されます。 |
| INFA_AbnormalSessionTermination | 最後のメッセージがセッションログに送られ、セッションが異常終了した後に呼び出されます。  |

この節に記述された関数は、標準 C ヘッダファイルの *time.h* で宣言された時間構造体を使用します。また、この関数は、以下の宣言を前提としています。

```
typedef int INFA_INT32;
typedef unsigned int INFA_UINT32;
typedef unsigned short INFA_UNICHAR;
typedef char INFA_MBCSCHAR;
typedef int INFA_MBCS_CODEPAGE_ID;
```

## INFA\_InitSessionLog

```
void INFA_InitSessionLog(void ** dllContext,
 const INFA_UNICHAR * sServerName,
 const INFA_UNICHAR * sFolderName,
 const INFA_UNICHAR * sWorkflowName,
 const INFA_UNICHAR * sessionHierName[]);
```

Integration Service は、セッションログイベントを書き込む前に INFA\_InitSessionLog 関数を呼び出します。この関数に渡されるパラメータは、Integration Service がイベントログを書き込むセッションに関する情報を提供します。

INFA\_InitSessionLog には、以下のパラメータがあります。

| パラメータ       | データタイプ        | 説明                                                                                                                                                                            |
|-------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dllContext  | 未指定           | 共有ライブラリに固有のユーザー定義情報。このパラメータは、後続する関数呼び出しのすべての関数に渡されます。このパラメータは、ネットワーク接続に関連する情報を格納したり、セッションログ出力の処理中に必要となるメモリを割り当てる際に使用できます。共有ライブラリは、このパラメータに関連付けられたメモリの割り当ておよび割り当て解除を行う必要があります。 |
| sServerName | 符号なしの short 型 | セッションを実行している Integration Service の名前。                                                                                                                                         |
| sFolderName | 符号なしの short 型 | セッションを含むフォルダ名。                                                                                                                                                                |

| パラメータ             | データタイプ           | 説明                                                                                                        |
|-------------------|------------------|-----------------------------------------------------------------------------------------------------------|
| sWorkflowName     | 符号なしの short 型    | セッションに関連付けられたワークフローの名前。                                                                                   |
| sessionHierName[] | 符号なしの short 型の配列 | セッション階層を含む配列。配列には、セッションが属しているリポジトリ、ワークフロー、およびワークレット（指定している場合）が含まれます。ポインタのサイズで除算された配列のサイズは、配列要素の数と等しくなります。 |

## INFA\_OutputSessionLogMsg

```
void INFA_OutputSessionLogMsg(
 void * dllContext,
 time_t curTime,
 INFA_UINT32 severity,
 const INFA_UNICHAR * msgCategoryName,
 INFA_UINT32 msgCode,
 const INFA_UNICHAR * msg,
 const INFA_UNICHAR * threadDescription);
```

Integration Service は、イベントを記録するたびに、この関数を呼び出します。この関数に渡されるパラメータには、ログイベントメッセージのさまざまな要素が含まれます。このパラメータは、ログ出力の形式をカスタマイズしたり、メッセージをフィルタリングする際に使用できます。

INFA\_OutputSessionLogMsg には、以下のパラメータがあります。

| パラメータ           | データ型                 | 説明                                                                                                                                                                   |
|-----------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dllContext      | 未指定                  | 共有ライブラリに固有のユーザ定義情報。このパラメータは、ネットワーク接続に関連する情報を格納したり、セッションログ出力の処理中に必要となるメモリを割り当てる際に使用できます。共有ライブラリは、このパラメータに関連付けられたメモリの割り当ておよび割り当て解除を行う必要があります。                          |
| curTime         | time_t               | Integration Service によりイベントが記録される時刻。                                                                                                                                 |
| severity        | 符号なしの int 型          | ログイベントメッセージのタイプを示すコード。イベントログでは、以下の重要度コードが使用されます。<br>32：デバッグメッセージ<br>8：情報メッセージ<br>2：エラーメッセージ<br>4：警告メッセージ                                                             |
| msgCategoryName | 定数<br>unsigned short | ログイベントメッセージのカテゴリを示すコードのプレフィックス。<br>次のメッセージ例で、文字列 <i>BLKR</i> は msgCategoryName パラメータに渡される値です。<br><br>READER_1_1_1> BLKR_16003 Initialization completed successfully. |



| パラメータ             | データ型                 | 説明                                                                                                                                                                           |
|-------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| msgCode           | 符号なしの int 型          | ログイベントメッセージを識別する番号。<br>次の例で、文字列 <i>16003</i> は msgCode パラメータに渡される値です。<br><i>READER_1_1_1&gt; BLKR_16003 Initialization completed successfully.</i>                           |
| msg               | 定数<br>unsigned short | ログイベントメッセージのテキスト。<br>次の例で、文字列 <i>Initialization completed successfully</i> は msg パラメータに渡される値です。<br><i>READER_1_1_1&gt; BLKR_16003 Initialization completed successfully.</i> |
| threadDescription | 定数<br>unsigned short | イベントログを生成しているスレッドを示すコード。<br>次の例で、文字列 <i>READER_1_1_1</i> は threadDescription パラメータに渡される値です。<br><i>READER_1_1_1&gt; BLKR_16003 Initialization completed successfully.</i>     |

## INFA\_OutputSessionLogFatalMsg

```
void INFA_OutputSessionLogFatalMsg(void * dllContext, const char * msg);
```

Integration Service は、異常終了の前に、この関数を呼び出して最後のイベントを記録します。パラメータ msg は、Integration Service コードページ内の MBSC 文字です。

UNIX でこの関数を実装する場合には、この関数から非同期シグナルセーフ関数のみを呼び出すようにしてください。

INFA\_OutputSessionLogFatalMsg には、以下のパラメータがあります。

| パラメータ      | データタイプ  | 説明                                                                                                                                           |
|------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------|
| dllContext | 未指定     | 共有ライブラリに固有のユーザー定義情報。このパラメータは、ネットワーク接続に関連する情報を格納したり、セッションログ出力の処理中に必要となるメモリを割り当てる際に使用できます。共有ライブラリは、このパラメータに関連付けられたメモリの割り当ておよび割り当て解除を行う必要があります。 |
| msg        | キャラクタ定数 | エラーメッセージのテキスト。一般に、これらのメッセージはアサーションエラーメッセージか、オペレーティングシステムエラーメッセージです。                                                                          |

## INFA\_EndSessionLog

```
void INFA_EndSessionLog(void * dllContext);
```

Integration Service は、最後のメッセージがセッションログに送られ、セッションが正常に終了した後に、この関数を呼び出します。この関数は、クリーンアップ操作の実行、およびメモリとリソースの解放を行う際に使用できます。

INFA\_EndSessionLog には、以下のパラメータがあります。

| パラメータ      | データタイプ | 説明                                                                                                                                           |
|------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------|
| dllContext | 未指定    | 共有ライブラリに固有のユーザー定義情報。このパラメータは、ネットワーク接続に関連する情報を格納したり、セッションログ出力の処理中に必要となるメモリを割り当てる際に使用できます。共有ライブラリは、このパラメータに関連付けられたメモリの割り当ておよび割り当て解除を行う必要があります。 |

## INFA\_AbnormalSessionTermination

```
void INFA_AbnormalSessionTermination(void * dllContext);
```

Integration Service は、最後のメッセージがセッションログに送られ、セッションが異常終了した後に、この関数を呼び出します。Integration Service は、INFA\_OutputSessionLogFatalMsg 関数を呼び出した後に、この関数を呼び出します。Integration Service は、この関数を呼び出した後、INFA\_EndSessionLog を呼び出しません。

例えば、DTM が強制終了またはタイムアウトになった場合に、Integration Service はこの関数を呼び出します。UNIX では、シグナル例外が発生した場合に、Integration Service はこの関数を呼び出します。

この関数を実装する場合、最小限の機能のみが含まれます。UNIX では、この関数から非同期シグナルセーフ関数のみを呼び出すようにしてください

INFA\_AbnormalSessionTermination には、以下のパラメータがあります。

| パラメータ      | データタイプ | 説明                                                                                                                                           |
|------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------|
| dllContext | 未指定    | 共有ライブラリに固有のユーザー定義情報。このパラメータは、ネットワーク接続に関連する情報を格納したり、セッションログ出力の処理中に必要となるメモリを割り当てる際に使用できます。共有ライブラリは、このパラメータに関連付けられたメモリの割り当ておよび割り当て解除を行う必要があります。 |

## セッションログインタフェースの例

Informatica では、セッションログインタフェースを使用するサンプルプログラムが用意されています。このサンプルプログラムでは、*sesslog.log* という名前のテキストファイルにセッションログイベントが送信されます。このサンプルプログラムを表示し、セッションログインタフェースを使用した、各自の要件に基づくセッションログイベントの処理方法について理解を深めることができます。また、サンプルプログラムをコンパイルして、テキストファイルにセッションログイベントを送信する外部ライブラリを構築することもできます。

セッションログサンプルプログラムは、Informatica Development Platform インストーラから PowerCenter SDK ファイルをインストールすると利用できるようになります。デフォルトでは、セッションログサンプルプログラムは、以下のディレクトリにインストールされます。

<SDKInstallationDir>/SessionLog\_API/samples

## 外部セッションログライブラリの構築

外部ライブラリを構築するには、サンプルプログラム demo\_sesslog.cpp に付属の make ファイルを使用します。ライブラリをコンパイルするためのコマンドは、使用するプラットフォームによって異なります。

### UNIX でのライブラリの構築

以下の表に、各種プラットフォームでライブラリを構築するためのコマンドを示します。

| プラットフォーム     | コンパイラ | コマンド                   |
|--------------|-------|------------------------|
| Solaris      | CC    | make -f makefile_sol   |
| HP/UX        | aCC   | make -f makefile_hpx   |
| HP-UX 64 ビット | aCC   | make -f makefile_hpx64 |
| AIX          | xCL_r | make -f makefile_aix   |
| AIX 64 ビット   | xCL_r | make -f makefile_aix64 |
| Linux        | g++   | make -f makefile_linux |

### Windows でのライブラリの構築

Windows では、Microsoft Visual C++ 6.0 を使用してセッションログライブラリのサンプルを構築します。Visual C++ 6.0 でサンプルプログラム demo\_sesslog.dsw を開き、プロジェクトを構築します。

## 外部セッションログライブラリの使用

構築したライブラリを使用して、セッションログの出力をファイルに書き込むことができます。

サンプルの外部セッションログライブラリを使用するには、以下の手順を実行します。

1. Administrator ツールにログインし、セッションログファイルを設定する Integration Service を選択します。
2. Integration Service の [プロパティ] タブで、設定プロパティを編集します。
3. ExportSessionLogLibName プロパティを、セッションログサンプルプログラムから作成したセッションログライブラリのパスおよびファイル名に設定します。

## 第 23 章

# バッファメモリについて

この章では、以下の項目について説明します。

- [バッファメモリの概要について, 339 ページ](#)
- [自動バッファメモリ設定, 340 ページ](#)
- [バッファメモリの設定, 340 ページ](#)
- [セッションキャッシュメモリの設定, 341 ページ](#)

## バッファメモリの概要について

セッションを実行すると、Integration Service プロセスは、Data Transformation Manager (DTM) を開始します。DTM は、セッションプロパティの [DTM バッファサイズ] 設定に基づいて、実行時にバッファメモリをセッションに割り当てます。

DTM は、セッションのプロパティの [デフォルトのバッファブロックサイズ] 設定に従って、メモリを複数のバッファブロックに分割します。reader スレッド、トランスフォーメーションスレッド、および writer スレッドは、バッファブロックを使用してソースからターゲットにデータを移動します。バッファブロックサイズは、ソースまたはターゲット内のデータの最大行の精度より大きくなっている必要があります。

Integration Service では、パーティションのソースおよびターゲットごとに少なくとも 2 つのバッファブロックを割り当てます。XML のソースおよびターゲットについては、ソースおよびターゲットに含まれるグループ数の 2 倍以上のバッファブロックが必要です。非正規化カラムを含む XML リーダーや循環参照を含む XML スキーマでは、追加のバッファブロックが必要になる場合があります。

以下のセッションプロパティを調整して、バッファメモリ設定を行います。

### DTM バッファサイズ

DTM バッファサイズには、DTM でセッションを処理するときに Integration Service が使用するバッファメモリの量を指定します。セッションのプロパティの [プロパティ] タブで DTM バッファサイズを設定します。

### デフォルトのバッファブロックサイズ

バッファブロックサイズには、ソースからターゲットにデータのブロックを移動するのに使用するバッファメモリの量を指定します。セッションのプロパティの [設定オブジェクト] タブでバッファブロックサイズを設定します。

Integration Service により、バッファメモリとバッファブロックの最小メモリ割り当てが計算されます。デフォルトでは、ブロックあたり 64,000 バイトか、マッピングのソースまたはターゲットの最大行のサイズのうち、どちらか大きい方が割り当てられます。

DTM が設定された量のバッファメモリをセッションに割り当てられなかった場合、そのセッションは初期化できません。通常、バッファメモリに 1 GB を超えるメモリ量は必要ありません。

バッファサイズの値は手動で設定できます。また、必要なバッファメモリサイズを Integration Service で特定するようにセッションを設定することもできます。

## 自動バッファメモリ設定

DTM の初期化時に、セッションの実行時に使用するバッファメモリが割り当てられます。バッファメモリサイズを Integration Service で自動的に割り当てるように設定することも、メモリバッファとブロックのサイズを手動で設定することもできます。

デフォルトでは、PowerCenter Integration Service によって、トランスフォーメーションの要件とマッピングのソースおよびターゲットに基づいて、セッションに必要なバッファメモリが自動的に計算されます。この計算では、ホストマシンのメモリの量や、そのうちの使用可能なメモリの量は考慮されません。そのため、場合によっては、使用可能なメモリのごく一部しかセッションに割り当てられないこともあります。

PowerCenter Integration Service によって自動的に計算されたバッファメモリではセッションに必要なメモリフットプリントを確保できない可能性がある場合は、PowerCenter Integration Service でセッションに割り当てるバッファメモリのサイズとブロックのサイズを手動で指定することができます。

関連項目：

- [「セッションのキャッシュの概要」 \(ページ 307\)](#)

## セッション設定オブジェクトを使用したメモリの設定

セッション設定オブジェクトを使用して、複数のセッションのメモリ設定を設定することができます。セッション設定オブジェクトごとに異なるメモリ設定を設定できます。

リポジトリ内の各フォルダには、コミット/ロード設定、ログオプション、エラー処理設定などのセッションプロパティを含むデフォルトのセッション設定オブジェクトがあります。セッションを作成すると、Workflow Manager によってデフォルトの設定オブジェクトの設定がセッションに適用されます。セッションで使用する設定オブジェクトを選択することもできます。

複数のセッションに異なる設定を適用する場合、複数の設定オブジェクトを作成できます。例えば、テスト環境からプロダクション環境に移行する場合や、異なる自動メモリ要件を持つ複数のセッションがある場合に、セッション設定オブジェクトでメモリ設定を設定できます。

## バッファメモリの設定

Integration Service でセッションに必要なメモリを自動的に計算することも、DTM のバッファサイズとデフォルトのブロックサイズを手動で設定することもできます。

バッファメモリ設定は、セッションプロパティで設定できます。

1. セッションを開き、[設定オブジェクト] タブをクリックします。
2. デフォルトバッファブロックサイズに値を入力します。

自動または数値を指定できます。

デフォルトの単位はバイトです。他の単位を指定するには、値に KB、MB、または GB を追加します。例えば、「1048576」、「1024KB」、「1MB」のように指定します。

3. [プロパティ] タブをクリックします。

4. DTM バッファサイズの値を入力します。

自動または数値を指定できます。セッションで DTM バッファサイズに設定した値より多くのメモリが必要な場合、セッションパフォーマンスが下がり、セッションが失敗することがあります。

デフォルトの単位はバイトです。他の単位を指定するには、値に KB、MB、または GB を追加します。例えば、「1048576」、「1024KB」、「1MB」のように指定します。

## セッションキャッシュメモリの設定

Integration Service は、次のセッションのキャッシュに必要なメモリを判定することができます。

- ルックアップトランスフォーメーションのインデックスとデータキャッシュ
- アグリゲータトランスフォーメーションのインデックスとデータキャッシュ
- ランクトランスフォーメーションのインデックスとデータキャッシュ
- ジョイナトランスフォーメーションのインデックスとデータキャッシュ
- ソータトランスフォーメーションのキャッシュ
- XML ターゲットキャッシュ

トランスフォーメーションのプロパティまたはセッションプロパティの [マッピング] タブでインデックスとデータのキャッシュサイズを自動的に設定することができます。

## セッションキャッシュの制限

Integration Service は、メモリキャッシュが自動モードに設定されているトランスフォーメーションに、セッションキャッシュを使用してメモリを割り当てます。セッションキャッシュのメモリの量は制限することができます。セッションキャッシュを制限する場合、他のプロセス用に一部のメモリが残るよう、Integration Service がセッションに使用できるメモリの量を制限します。

キャッシュメモリの制限を指定するときは、数値と合計メモリに対する割合の両方を指定する必要があります。割合は、Integration Service が実行されているマシンの物理メモリの合計に基づきます。

Integration Service では、数値と割合の値を比較してどちらが少ないかを特定し、少ない方の値をセッションキャッシュの合計メモリとして割り当てます。

セッションに割り当てられるメモリキャッシュに制限を設定するには、セッション設定オブジェクトの次の属性を使用します。

### 自動メモリ属性で利用できる最大メモリ

セッションキャッシュに割り当てるメモリ量です。自動キャッシュメモリの合計がこのプロパティの値を超えないように制限されます。これは、使用されるマシンメモリの割合が [自動メモリ属性で利用できる合計メモリの最大割合 (%) ] プロパティの値より低くなる場合でも同様です。物理メモリが多いマシンでセッションを実行する場合に、この状況になることがあります。

### 自動メモリ属性で利用できる合計メモリの最大割合 (%)

セッションキャッシュに割り当てるマシンのメモリの割合です。自動キャッシュメモリの合計は、このプロパティの値を超えないように制限されます。これは、[自動メモリ属性で利用できる最大メモリ] プロパ

ティの値の方が高い場合でも同様です。物理メモリが非常に少ないマシンでセッションを実行する場合に、この状況になることがあります。

キャッシュメモリが自動に設定されたすべてのトランスフォーメーションに、セッションキャッシュからメモリが割り当てられます。そのメモリがすべてのトランスフォーメーションのキャッシュで分割されます。

例えば、セッションの3つのルックアップトランスフォーメーションに自動キャッシュを設定するとします。セッションのメモリキャッシュの制限は500MBに設定します。セッションを実行すると、500MBの割り当てメモリが、3つすべてのルックアップトランスフォーメーションのインデックスキャッシュとデータキャッシュで分割されます。セッションのメモリキャッシュの制限は、自動キャッシュの対象として設定していないトランスフォーメーションには適用されません。

セッションキャッシュを自動割り当ての対象として設定している場合、自動キャッシュ割り当ての対象に設定された各トランスフォーメーションに対して、インデックスキャッシュ用に1MB以上、データキャッシュ用に2MB以上のメモリが割り当てられます。セッションキャッシュの制限でインデックスキャッシュとデータキャッシュに割り当てるメモリを確保できない場合、キャッシュの制限は無効になり、インデックスキャッシュとデータキャッシュにそれぞれ最小メモリが割り当てられます。

例えば、セッションキャッシュのメモリを4MBに制限し、自動キャッシュ割り当ての対象として2つのトランスフォーメーションを設定しているとします。この場合、セッションキャッシュの制限は無効になり、自動キャッシュ割り当ての対象に設定された各トランスフォーメーションに対して、インデックスキャッシュ用に1MB、データキャッシュ用に2MBの最小メモリが割り当てられます。そのため、トランスフォーメーションのキャッシュに割り当てられるメモリは合計で6MBになります。

グリッドでセッションを実行する場合、[自動メモリ属性で利用できる最大メモリ]を設定すると、割り当てたメモリキャッシュがグリッドのすべてのノード間で分割されます。[自動メモリ属性で利用できる合計メモリの最大割合(%)]を設定すると、指定した割合のメモリキャッシュがグリッドの各ノードに割り当てられます。

## セッションキャッシュの自動メモリ設定

セッションキャッシュの自動メモリ設定をするには：

1. Transformation Developer、またはセッションのプロパティの[マッピング]タブでトランスフォーメーションを開きます。
2. トランスフォーメーションのプロパティで、以下の各キャッシュサイズ設定でAutoを選択するか入力します
  - インデックスおよびデータキャッシュ
  - ソータキャッシュ
  - XML キャッシュ
3. Task Developer または Workflow Designer でセッションを開いて、[設定オブジェクト] タブをクリックします。
4. [自動メモリ属性で利用できる最大メモリ] に値を入力します。  
この値は、セッションキャッシュに使用する最大メモリ量を指定します。  
デフォルトの単位はバイトです。他の単位を指定するには、値に KB、MB、または GB を追加します。例えば、「1048576」、「1024KB」、「1MB」のように指定します。
5. [自動メモリ属性で利用できる合計メモリの最大割合(%)] に値を入力します。  
この値は、セッションキャッシュが利用できる合計メモリの最大パーセンテージを指定します。



## 第 24 章

# 高精度データ

この章では、以下の項目について説明します。

- [高精度データの概要, 343 ページ](#)
- [Bigint, 343 ページ](#)
- [Decimal, 344 ページ](#)

## 高精度データの概要

高精度データは、桁数の多い数値をより正確に表す方法を決定します。数字の桁数には、小数点桁数が含まれます。例えば、値 11.47 は、4 桁の精度と 2 桁の小数点桁数を持ちます。桁数の多い数値は、オーバーフローが発生する計算で使用される場合、丸め処理が行われることにより精度が損なわれる場合があります。高精度データを切り捨てる際にエラーが発生すると、正しい演算結果が得られなくなる可能性があります。

高精度データの値は、より精度が高い値です。精度の高い値が必要な場合は、高精度を有効にします。

高精度はセッションの [プロパティ] タブで有効にします。Integration Service は、Bigint 値および 10 進値に対して別々に高精度データ処理を行います。

## Bigint

結果が 10 進値となる計算では、Integration Service は Bigint 値を Double タイプまたは Decimal タイプとして処理します。セッションに結果が 10 進値となる計算が含まれ、かつ実行に高精度を必要としない場合、Integration Service は計算を実行する前に Bigint 値を Double タイプに変換します。トランスフォーメーションの Double データタイプは最大 15 桁の精度をサポートし、Bigint データタイプは最大 19 桁の精度をサポートします。このため、精度が 15 桁より大きい Bigint 値となる計算では、精度の損失が発生することがあります。

例えば、式トランスフォーメーションには以下の計算が含まれています。

```
POWER(BIGINTVAL, EXPVAL)
```

Integration Service では、計算を開始する前に、POWER 関数への入力を Double 値に変換します。

BIGINTVAL ポートに Bigint 値 9223372036854775807 が含まれている場合、Integration Service はこの値を 9.22337203685478e+18 に変換し、最後 4 桁の精度が失われます。EXPVAL ポートに値 1.0 が含まれており、結果ポートが Bigint の場合、計算結果の 9223372036854780000 が最大 bigint 値を超えているため、行エラーが作成されます。

結果が 10 進値となる計算で Bigint 値を使用し、高精度でセッションを実行する場合、Integration Service は Bigint 値を 10 進値に変換します。トランスフォーメーションの Decimal データタイプは最大 28 桁までの精度をサポートします。このため、計算結果が、精度が 28 桁より大きい値にならない限り、精度の損失は発生しません。この場合、Integration Service は結果を Double 値として保存します。

## Decimal

高精度を有効にせずにセッションを実行すると、Integration Service は 10 進値を Double 値に変換します。トランスフォーメーションの Decimal データタイプは 28 桁までの精度をサポートしますが、Double データタイプは 15 桁までの精度をサポートします。したがって、15 桁を超える精度を持つ 10 進値の場合、精度の損失が発生します。

例えば、Decimal (20,0) を使用して数値 40012030304957666903 を渡すマッピングがあるとします。高精度でセッションを実行しない場合、Integration Service は 10 進値を Double 値に変換し、 $4.00120303049577 \times 10^{19}$  を渡します。

最大 28 桁の精度を確保するためには、Decimal データタイプを使用して、セッションプロパティで高精度を有効にします。高精度でセッションを実行すると、Integration Service は 10 進値を Decimal として処理します。計算結果が、精度が 28 桁より大きい値にならない限り、精度の損失は発生しません。この場合、Integration Service は結果を Double 値として保存します。

# 索引

## 記号

{Teradata でのプッシュダウンの最適化

派生テーブル [107](#)

\$\$PushdownConfig

説明 [110](#)

\$AppConnection

using [238](#)

\$BadFile

using [238](#)

命名規則 [238](#)

\$DynamicPartitionCount

説明 [238](#)

\$InputFile

using [238](#)

命名規則 [238](#)

\$LoaderConnection

using [238](#)

\$LookupFile

using [238](#)

命名規則 [238](#)

\$OutputFile

using [238](#)

命名規則 [238](#)

\$PMSessionLogFile

using [238](#)

\$PMStorageDirPMStorageDir

ワークフローのセッションの状態 [183](#)

\$PMStorageDir

ワークフローの操作の状態 [183](#)

\$PMWorkflowRunId

コンカレントワークフロー [208](#)

\$PMWorkflowRunInstanceName

コンカレントワークフロー [208](#)

\$QueueConnection

using [238](#)

\$Source 接続値

パラメータおよび変数のタイプ [249](#)

\$Target 接続値

パラメータおよび変数のタイプ [249](#)

## 数字

85 以前のタイムスタンプの互換性オプション、Netezza でのプッシュダウンの最適化 [81](#)

## A

ABORT 関数

セッションの失敗 [202](#)

## B

bigint

高精度処理 [343](#)

## C

ConnectionParam.prm ファイル

使用 [257](#)

CPU の数

動的パーティションでの設定 [23](#)

CPU の数を基準

設定 [23](#)

CUME 関数

パーティション化の制限 [60](#)

## D

\$DBConnection

使用 [238](#)

命名規則 [238](#)

decimal

高精度処理 [343](#)

DTM (データ変換マネージャ)

バッファサイズ [340](#)

DTM バッファサイズ要件

構成 [340](#)

## F

FTP

SFTP [300](#)

概要 [299](#), [300](#)

セッションの作成 [302](#)

ソースファイルのアクセス [302](#)

ターゲットのパーティション化 [305](#)

ターゲットファイルのアクセス [302](#)

ファイルターゲットへの接続 [44](#)

リモートディレクトリ、パラメータおよび変数のタイプ [249](#)

リモートファイル名、パラメータおよび変数のタイプ [249](#)

\$FTPConnection

使用 [238](#)

FTP 接続

セッションパラメータ [238](#)

パスワード、パラメータのタイプ [249](#)

パラメータタイプ [249](#)

ユーザー名、パラメータのタイプ [249](#)

## H

HTTP トランスフォーメーション

スレッド [48](#)

HTTP トランスフォーメーション (続く)  
パイプラインのパーティション化 [47](#)

## I

IBM DB2  
データベースパーティション化。 [65](#)  
データベースパーティション化 [61](#), [68](#)

IBM DB2 EE  
外部データのロード [276](#)  
空白スペースのロード [280](#)  
属性 [279](#)

IBM DB2 EEE  
外部データのロード [276](#)  
属性 [281](#)

INFA\_AbnormalSessionTermination  
セッションログインタフェース [337](#)

INFA\_EndSessionLog  
セッションログインタフェース [336](#)

INFA\_InitSessionLog  
セッションログインタフェース [334](#)

INFA\_OutputSessionLogFatalMsg  
セッションログインタフェース [336](#)

INFA\_OutputSessionLogMsg  
セッションログインタフェース [335](#)

Informix  
行レベルのロック [44](#)

Integration Service  
外部ローダーのサポート [274](#)  
グリッド上での動作 [219](#)  
グリッドへの割り当て [219](#)  
コミット間隔の概要 [154](#)  
セッションログインタフェースでの関数の呼び出し [332](#)

統合サービス  
グリッド上でのセッションの実行 [216](#)  
グリッドの概要 [214](#)

Integration Service コードページ  
差分集計への影響 [329](#)

## J

Java トランスフォーメーション  
スレッド [48](#)  
パイプラインのパーティション化 [47](#)

JMS 接続先  
パラメータおよび変数のタイプ [249](#)

## L

Logtable 名  
FastExport 属性 [269](#)

## M

Microsoft Access  
パイプラインのパーティション化 [44](#)

Microsoft Azure SQL Data Warehouse 接続  
プッシュダウンの最適化、ルールおよびガイドライン [83](#)

MOVINGAVG 関数  
パーティション化の制限 [60](#)

MOVINGSUM 関数  
パーティション化の制限 [60](#)

## N

Netezza 接続  
プッシュダウンの最適化、ルールおよびガイドライン [81](#)

## O

Oracle  
データベースパーティション化。 [65](#)  
データベースパーティション化 [61](#)

Oracle 外部ローダー  
外部ローダーのサポート [274](#), [283](#)  
拒否ファイル [283](#)  
区切りフラットファイルターゲット [283](#)  
固定幅フラットファイルターゲット [283](#)  
属性 [284](#)  
データ精度 [283](#)  
パーティション化されたターゲットファイル [284](#)  
マルチバイトデータ [283](#)

## P

PM\_TGT\_RUN\_ID  
形式 [184](#)  
手動での作成 [186](#)  
説明 [184](#)

PMError\_MSG テーブル  
スキーマ [174](#)

PMError\_ROWDATA テーブル  
スキーマ [172](#)

PMError\_Session テーブル  
スキーマ [175](#)

PM\_RECOVERY テーブル  
形式 [184](#)  
手動での作成 [186](#)  
説明 [184](#)  
デッドロックリトライ [184](#)

PM\_REC\_STATE テーブル  
手動での作成 [186](#)  
説明 [184](#)  
リアルタイムセッション [145](#)

PostgreSQL 接続  
プッシュダウンの最適化、ルールおよびガイドライン [81](#)

PowerCenter リアルタイム製品  
概要 [152](#)

PowerExchange Client for PowerCenter  
リアルタイム変更データ [135](#)

## R

reader  
Teradata FastExport に対する選択 [271](#)

Reader 時間制限  
構成 [139](#)

## S

SDK ソース  
リカバリ [194](#)

SetID  
PeopleSoft、パラメータおよび変数のタイプ [249](#)

SFTP  
キーファイルの場所 [302](#)  
グリッド上でのセッションの実行 [302](#)

SFTP (続く)

セッションの作成 [302](#)

説明 [300](#)

SQL

パーティション化されたパイプラインでのクエリ [37](#)

プッシュダウンの最適化に対する生成 [77](#), [78](#)

SQL オーバーライド

プッシュダウンの最適化 [107](#)

SQL クエリ

パラメータおよび変数のタイプ [249](#)

Sybase IQ

パーティション化の制限 [44](#)

Sybase IQ 外部ローダー

概要 [284](#)

区切りフラットファイルターゲット [285](#)

固定幅フラットファイルターゲット [285](#)

サポート [274](#)

データ精度 [285](#)

マルチバイトデータ [285](#)

属性 [285](#)

## T

TDPID

説明 [269](#)

Tenacity

FastExport 属性 [269](#)

Teradata FastExport

fexp コマンド [269](#)

reader の選択 [271](#)

TDPID 属性 [269](#)

一時ファイル、変数のタイプ [249](#)

使用するための手順 [268](#)

ステージングデータ [271](#)

制御ファイルのオーバーライド [272](#)

セッション属性の説明 [271](#)

接続属性 [269](#)

接続の作成 [269](#)

説明 [268](#)

ソース接続の変更 [271](#)

マルチバイト文字の読み取り [269](#)

ルールおよびガイドライン [273](#)

Teradata 外部ローダー

FastLoad の属性 [294](#)

MultiLoad の属性 [289](#)

TPump の属性 [291](#)

コードページ [287](#)

サポート [274](#)

制御ファイルコンテンツのオーバーライド、パラメータおよび変数

のタイプ [249](#)

制御ファイルのオーバーライド [287](#)

日付形式 [287](#)

TIB/リポジトリ

TIB/Adapter SDK リポジトリ URL、変数のタイプ [249](#)

## U

UNIX システム

外部ローダーの動作 [275](#)

## V

Vertica 接続

プッシュダウンの最適化、ルールおよびガイドライン [82](#)

## W

Web Services Hub

コンカレントワークフローの実行 [207](#)

Web サービスメッセージ

リアルタイムデータ [135](#)

Windows システム

外部ローダーの動作 [275](#)

Workflow Manager

グリッド上でのセッションの実行 [214](#)

グリッド上でのワークフローの実行 [214](#)

Workflow Advanced の開始

コンカレントワークフローの開始 [209](#)

Workflow Monitor

コンカレントワークフローの表示 [211](#)

WriterWaitTimeOut

ターゲットベースのコミット [155](#)

## X

XML ジェネレータートランスフォーメーション

パーティション化の制限 [59](#)

XML ターゲット

キャッシュ [324](#)

キャッシュの設定 [325](#)

ターゲットベースのコミット [155](#)

パーティション化の制限 [59](#)

XML ターゲットキャッシュ

説明 [324](#)

変数のタイプ [249](#)

## あ

アイドル時間

構成 [138](#)

アイドル状態のデータベース

説明 [79](#)

アクティブソース

コミットの生成 [156](#)

ソースベースのコミット [155](#), [156](#)

アグリゲータートランスフォーメーション

キャッシュ [316](#)

キャッシュの計算の入力 [317](#)

キャッシュの設定 [317](#)

キャッシュのパーティション化 [315](#), [316](#)

コンカレントワークフローへの追加 [212](#)

ソート済みポート [317](#)

パーティションポイントの使用 [34](#)

プッシュダウンの最適化 [119](#)

アグリゲータキャッシュ

概要 [316](#)

説明 [316](#)

アップデートストラテジトランスフォーメーション

プッシュダウンの最適化 [132](#)

アプリケーション接続

セッションパラメータ [238](#)

パスワード、パラメータのタイプ [249](#)

パラメータタイプ [249](#)

ユーザー名、パラメータのタイプ [249](#)

## い

一時ファイル

Teradata FastExport 属性 [271](#)

イベント待ちタスク  
トリガファイル名、変数のタイプ [249](#)  
インスタンス  
ワークフローインスタンスの説明 [205](#)  
インデックス  
検索 [329](#)  
ディレクトリの作成 [331](#)  
インデックスキャッシュ  
差分集計 [329](#)  
命名規則 [310](#)

## え

永続変数  
定義 [232](#)  
ワークレット [234](#)  
エクスターナルプロシージャトランスフォーメーション  
初期化プロパティ、変数のタイプ [249](#)  
パーティション化のガイドライン [59](#)  
エラー  
しきい値 [202](#)  
重大 [202](#)  
エラーしきい値  
エラー時の停止 [202](#)  
パイプラインのパーティション化 [202](#)  
変数のタイプ [249](#)  
エラー処理  
PMEError\_MSG テーブルスキーマ [174](#)  
PMEError\_ROWDATA テーブルスキーマ [172](#)  
PMEError\_Session テーブルスキーマ [175](#)  
エラーログファイル [177](#)  
オプション [180](#)  
概要 [203](#)  
トランザクション制御 [160](#)  
ブッシュダウンの最適化 [104](#)  
エラーメッセージ  
外部ローダー [275](#)  
エラーログ  
オプション [180](#)  
概要 [170](#)  
セッションエラー [203](#)  
エラーログテーブル  
概要 [171](#)  
作成 [171](#)  
エラーログファイル  
概要 [177](#)  
ディレクトリ、パラメータおよび変数のタイプ [249](#)  
テーブル名プレフィックス長の制限 [264](#)  
名前、パラメータおよび変数のタイプ [249](#)  
エンドポイント URL  
Web サービス、パラメータおよび変数のタイプ [249](#)  
パラメータおよび変数のタイプ [249](#)

## お

オーバーライド  
Teradata ローダー制御ファイル [287](#)  
オープントランザクション  
定義 [162](#)

## か

外部ローダー  
DB2 [276](#)  
Integration Service のサポート [274](#)

外部ローダー (続く)  
Oracle [283](#)  
Sybase IQ [284](#)  
Teradata [287](#)  
Windows システム上 [275](#)  
Workflow Manager の設定 [296](#)  
エラーメッセージ [275](#)  
概要 [274](#)  
コードページ [274](#)  
サブ秒の処理 [275](#)  
動作 [275](#)  
パーティション化されたパイプラインでの使用 [44](#)  
マルチバイトデータのロード [283](#), [285](#)  
リソースとしての設定 [274](#)  
外部ローダー接続  
セッションパラメータ [238](#)  
パスワード、パラメータのタイプ [249](#)  
パラメータタイプ [249](#)  
ユーザー名、パラメータのタイプ [249](#)  
カスタムトランスフォーメーション  
スレッド [48](#)  
パイプラインのパーティション化 [47](#)  
パーティション化のガイドライン [59](#)  
環境 SQL  
パラメータおよび変数のタイプ [249](#)  
完全なブッシュダウンの最適化  
説明 [78](#)

## き

キー範囲パーティション化  
キー範囲の追加 [72](#)  
説明 [22](#), [61](#)  
追加 [70](#)  
パーティションキーの追加 [71](#)  
パーティションビュー [28](#)  
パフォーマンス [71](#)  
ブッシュダウンの最適化 [114](#)  
キャッシュ  
XML ターゲット [324](#)  
XML ターゲットの設定 [325](#)  
アグリゲータトランスフォーメーション [316](#)  
アグリゲータトランスフォーメーションの設定 [317](#)  
アグリゲータトランスフォーメーションのソート済み入力用 [317](#)  
永続ルックアップ [321](#)  
オーバーライド [312](#)  
概要 [307](#)  
キャッシュの計算 [312](#), [314](#)  
グリッド上のインデックスキャッシュ [218](#)  
グリッド上のデータキャッシュ [218](#)  
構成 [314](#)  
最大メモリ量の設定 [341](#)  
最適化 [325](#)  
再利用可能なセッション [312](#)  
再利用不可能なセッション [312](#)  
自動メモリ [313](#)  
ジョイナトランスフォーメーション [317](#)  
ジョイナトランスフォーメーションの設定 [319](#), [324](#)  
数値 [314](#)  
セッションキャッシュファイル [307](#)  
設定方法 [312](#)  
ソータトランスフォーメーション [324](#)  
トランスフォーメーション [307](#)  
パーティション化 [25](#)  
メモリ [308](#)  
リンクトランスフォーメーション [322](#)  
リンクトランスフォーメーションの設定 [323](#)

キャッシュ (続く)  
リアルタイムセッションによるリセット [163](#)  
ルックアップトランスフォーメーション [321](#)  
ルックアップトランスフォーメーションの設定 [322](#)  
キャッシュサイズ  
構成 [312](#)  
最適化 [325](#)  
セッションメモリ要件、設定 [341](#)  
キャッシュディレクトリ  
共有 [311](#)  
最適、選択 [311](#)  
変数のタイプ [249](#)  
キャッシュの計算  
アグリゲータトランスフォーメーションの入力 [317](#)  
ジョイナトランスフォーメーションの入力 [319](#)  
使用 [314](#)  
説明 [312](#)  
ソータトランスフォーメーションの入力 [324](#)  
ランクトランスフォーメーションの入力 [323](#)  
ルックアップトランスフォーメーションの入力 [322](#)  
キャッシュのパーティション化  
アグリゲータトランスフォーメーション [315](#), [316](#)  
キャッシュサイズの設定 [315](#)  
差分集計 [316](#)  
ジョイナトランスフォーメーション [315](#), [319](#)  
説明 [25](#)  
ソータトランスフォーメーション [315](#), [324](#)  
トランスフォーメーション [315](#)  
パフォーマンス [25](#)  
ランクトランスフォーメーション [315](#), [322](#)  
ルックアップトランスフォーメーション [56](#), [315](#), [321](#)  
キャッシュファイル  
場所 [329](#)  
命名規則 [310](#)  
キュー接続  
セッションパラメータ [238](#)  
パラメータタイプ [249](#)  
強制終了  
Integration Service の処理 [201](#)  
セッション [204](#)  
タスク [204](#)  
ワークフロー [203](#)  
共有ライブラリ  
セッションログインタフェースの実装 [333](#)  
拒否ファイル  
Oracle 外部ローダー [283](#)  
セッションパラメータ [238](#)  
トランザクション制御 [160](#)  
パラメータおよび変数のタイプ [249](#)  
拒否ファイルディレクトリ  
ターゲットファイルプロパティ [45](#)  
パラメータおよび変数のタイプ [249](#)  
拒否ファイル名  
説明 [45](#)

## <

グリッド  
Integration Service の動作 [219](#)  
Integration Service のプロパティ設定 [219](#)  
概要 [214](#)  
キャッシュ要件 [218](#)  
最大メモリ量の指定 [341](#)  
セッションの実行 [216](#)  
セッションの分散 [216](#), [219](#)  
セッションのリカバリ [219](#)  
セッションプロパティの設定 [219](#)

グリッド (続く)  
パイプラインのパーティション化 [217](#)  
要件 [220](#)  
リソースの設定 [219](#)  
ワークフローの分散 [215](#), [219](#)  
ワークフローのリカバリ [219](#)  
ワークフロープロパティの設定 [219](#)  
グリッド上のセッション  
シーケンスジェネレータトランスフォーメーションのパーティション化 [58](#)  
説明 [216](#)  
グリッドのノード数  
動的パーティション化での設定 [23](#)

## け

継続サブスクリプション名  
JMS 用の変数のタイプ [249](#)  
検証  
セッションリカバリ [194](#)

## こ

更新  
差分 [330](#)  
高精度  
Bigint データタイプ [343](#)  
Decimal データタイプ [343](#)  
処理 [343](#)  
コードページ  
外部ローダーファイル [274](#)  
コマンド  
パーティション化されたソース [38](#)  
パーティション化されたターゲット [46](#)  
コマンドタスク  
変数のタイプ [249](#)  
リソースの割り当て [223](#)  
コマンドのタイプ  
ファイルソースのパーティション化 [39](#)  
コマンドプロパティ  
パーティション化されたターゲットの設定 [45](#)  
ファイルソースのパーティション化 [39](#)  
コミット間隔  
構成 [168](#)  
説明 [154](#)  
ソースベースおよびターゲットベース [154](#)  
コミットソース  
ソースベースのコミット [156](#)  
コミットタイプ  
構成 [140](#)  
リアルタイムセッション [140](#)  
ワールドスタート  
リアルタイムセッション [148](#)  
コンカレントワークフロー  
pmcmd を使用したワークフローインスタンスの作成 [210](#)  
Web サービスワークフローの実行 [207](#)  
[Workflow Advanced の開始] オプション [209](#)  
Workflow Monitor での表示 [211](#)  
一意のインスタンスの設定 [206](#)  
インスタンス名の追加 [209](#)  
開始および停止 [209](#)  
異なるセッションパラメータファイルの使用 [262](#)  
コマンドラインからの開始 [210](#)  
コマンドラインからの停止 [210](#)  
スケジューリング [212](#)  
設定手順 [209](#)



コンカレントワークフロー (続く)

説明 [205](#)

同一名による実行の設定 [206](#)

トランスフォーメーション制限 [212](#)

パラメータの使用 [208](#)

ルールおよびガイドライン [212](#)

ログの表示 [211](#)

[ワークフローの開始] オプション [210](#)

コンカレントワークレット

説明 [212](#)

## さ

再開リカバリ戦略

再現可能なデータの使用 [194](#)

リカバリターゲットテーブルの使用 [184](#)

再現可能なデータ

ソース [194](#)

トランスフォーメーション [194](#)

ワークフローのリカバリ [194](#)

最後のチェックポイントから再開

リカバリ戦略 [190](#), [192](#)

再初期化

集計キャッシュ [328](#)

最大セッション数

FastExport 属性 [269](#)

最大メモリ量

キャッシュでの設定 [341](#)

グリッド上のセッション [341](#)

再利用可能なセッション

キャッシュ [312](#)

再利用不可能なセッション

キャッシュ [312](#)

作成

FTP セッション [302](#)

インデックスディレクトリ [331](#)

エラーログテーブル [171](#)

データファイルディレクトリ [331](#)

パーティション化されたソースのファイルリスト [39](#)

ワークフロー変数 [233](#)

サスペンド時のメール

変数のタイプ [249](#)

サスペンド状態

ステータス [188](#)

サスペンド中

ステータス [188](#)

電子メール [188](#)

動作 [188](#)

ワークフロー [188](#)

サービスプロセス変数

パラメータファイル [248](#)

サービス変数

パラメータファイル [248](#)

サービスレベル

タスクへの割り当て [222](#)

サブ秒

外部データのロード [275](#)

差分集計

Integration Service のデータ移動モード [329](#)

概要 [327](#)

キャッシュの再初期化 [328](#)

キャッシュのパーティション化 [316](#)

処理 [328](#)

セッションの設定 [331](#)

セッションのソート順の変更 [329](#)

データのパーティション化 [330](#)

ファイルの移動 [329](#)

差分集計 (続く)

ファイルの削除 [329](#)

有効化の準備 [330](#)

差分変更

キャプチャ [330](#)

差分リカバリ

説明 [192](#)

## し

シーケンスジェネレータートランスフォーメーション

コンカレントワークフローへの追加 [212](#)

パーティション化 [58](#)

パーティション化のガイドライン [35](#), [59](#)

プッシュダウンの最適化 [126](#)

シェルコマンド

パラメータおよび変数のタイプ [249](#)

式

パラメータおよび変数のタイプ [249](#)

プッシュダウンの最適化 [87](#)

シーケンシャル統合

ファイルターゲット [47](#)

実行可能名

FastExport 属性 [269](#)

実行後のシェルコマンド

パラメータおよび変数のタイプ [249](#)

実行時位置

変数のタイプ [249](#)

実行時のパーティション化

セッションプロパティでの設定 [23](#)

自動タスクリカバリ

構成 [192](#)

自動メモリ設定

構成 [340](#)

集計キャッシュ

再初期化 [328](#)

集計ファイル

移動 [329](#)

再初期化 [328](#)

削除 [329](#)

終了条件

構成 [138](#)

出力が確定的 (プロパティ)

概要 [195](#)

出力が再現可能 (プロパティ)

概要 [195](#)

出力タイププロパティ

ファイルターゲットのパーティション化 [45](#)

出力ファイルディレクトリプロパティ

ターゲットファイルのパーティション化 [45](#)

パラメータおよび変数のタイプ [249](#)

出力ファイル名プロパティ

ターゲットファイルのパーティション化 [45](#)

パラメータおよび変数のタイプ [249](#)

ジョイナキャッシュ

説明 [317](#)

ジョイナトランスフォーメーション

キャッシュ [317](#)

キャッシュの計算の入力 [319](#)

キャッシュの設定 [319](#), [324](#)

キャッシュのパーティション化 [315](#), [319](#)

ソート済みフラットファイルの結合 [51](#)

ソート済みリレショナルデータの結合 [53](#)

パーティション化 [317](#)

パーティション化のガイドライン [59](#)

プッシュダウンの最適化 [121](#)

小数秒の精度

Teradata FastExport 属性 [271](#)

## す

スクリプトファイル

パラメータおよび変数のタイプ [249](#)

ステージング済み

FastExport セッション属性 [271](#)

ステージングファイル

SAP ファイル名およびディレクトリ、変数のタイプ [249](#)

ステータス

サスペンド状態 [188](#)

サスペンド中 [188](#)

ストアドプロシージャトランスフォーメーション

接続情報、パラメータおよび変数のタイプ [249](#)

呼び出しテキスト、パラメータおよび変数のタイプ [249](#)

スリープ

FastExport 属性 [269](#)

スループットの最適化

セッションのプロパティ [43](#)

スレッド

HTTP トランスフォーメーション [48](#)

Java トランスフォーメーション [48](#)

カスタムトランスフォーメーション [48](#)

パーティション [20](#)

## せ

制御タスク

ワークフローの停止または強制終了 [203](#)

制御値設定

PeopleSoft、パラメータおよび変数のタイプ [249](#)

制御ファイルのオーバーライド

Teradata FastExport 文の設定 [272](#)

Teradata FastExport をオーバーライドするための手順 [272](#)

Teradata のロード [287](#)

説明 [271](#)

生成

ソーススペースのコミットによるコミット [156](#)

セッション

FTP の使用 [302](#)

SFTP の使用 [302](#)

外部データのロード [274](#), [296](#)

完全なプッシュダウンの最適化 [78](#)

強制終了 [201](#), [204](#)

グリッド上での実行 [216](#)

グリッド上での分散 [216](#), [219](#)

グリッド上でのリカバリ [219](#)

結合処理のパフォーマンスを最適化するための設定 [50](#)

失敗 [27](#), [202](#)

情報の受け渡し [245](#)

情報の受け渡し、例 [245](#)

セッション実行前および実行後の変数の割り当て [245](#)

操作の状態 [183](#)

ソース側でのプッシュダウンの最適化 [77](#)

ターゲット側でのプッシュダウンの最適化 [77](#)

停止 [201](#), [204](#)

パラメータ [238](#)

プッシュダウンの最適化 [77](#)

プッシュダウンの最適化のための設定 [111](#)

リソースの割り当て [223](#)

セッションエラー

処理 [203](#)

セッション実行後の電子メール

パラメータおよび変数のタイプ [249](#)

セッション実行後の変数割り当て

失敗時に実行 [245](#)

成功時に実行 [245](#)

セッション実行前および実行後の SQL

コマンド、パラメータおよび変数のタイプ [249](#)

セッション実行前の変数割り当て

実行 [245](#)

セッションのプロパティ

FastExport ソース [271](#)

ソート順 [329](#)

ターゲットベースのコミット [168](#)

セッションパラメータ

FTP 接続パラメータ [238](#)

アプリケーション接続パラメータ [238](#)

外部ローダー接続パラメータ [238](#)

概要 [238](#)

キュー接続パラメータ [238](#)

拒否ファイルパラメータ [238](#)

組み込み [238](#)

セッション間での値の受け渡し [245](#)

セッションログパラメータ [238](#)

ソースファイルパラメータ [238](#)

ターゲットファイルパラメータ [238](#)

データベース接続パラメータ [238](#)

パーティションの数 [238](#)

パラメータファイル [248](#)

ファイル名、変数のタイプ [249](#), [262](#)

命名規則 [238](#)

ユーザー定義 [238](#)

リソースとしての設定 [243](#)

セッションパラメータファイル名

変数のタイプ [249](#), [262](#)

セッションリカバリデータのフラッシュ

メッセージリカバリ [144](#)

セッションリカバリデータのフラッシュ（プロパティ）

Integration Service [144](#)

セッションログ

外部ライブラリへの受け渡し [332](#)

外部ローダーのエラーメッセージ [275](#)

セッションパラメータ [238](#)

ディレクトリ、変数のタイプ [249](#)

ファイル名、パラメータのタイプ [249](#)

ワークフローリカバリ [199](#)

セッションログインタフェース

INFA\_AbnormalSessionTermination [337](#)

INFA\_EndSessionLog [336](#)

INFA\_InitSessionLog [334](#)

INFA\_OutputSessionLogFatalMsg [336](#)

INFA\_OutputSessionLogMsg [335](#)

Integration Service の呼び出し [332](#)

ガイドライン [333](#)

関数 [333](#)

実装 [332](#)

説明 [332](#)

セッションログカウンタ数

変数のタイプ [249](#)

接続

Teradata FastExport 接続の変更 [271](#)

Teradata FastExport 文の作成 [269](#)

パラメータファイルのテンプレート [257](#)

接続環境 SQL

パラメータおよび変数のタイプ [249](#)

接続されていないトランスフォーメーション

パーティション化の制限 [35](#)

絶対入力行の順序の維持

セッションのプロパティ [43](#)

## そ

### 操作の状態

- セッションリカバリ [183](#)
- チェックポイント [183](#), [192](#)
- ワークフローリカバリ [183](#)

### 相対入力行の順序の維持

- セッションのプロパティ [43](#)
- ソース修飾子トランスフォーメーション
- パーティションポイントの使用 [34](#)
- プッシュダウンの最適化 [128](#)
- プッシュダウンの最適化、SQL オーバーライド [107](#)

### ソーターキャッシュ

- 命名規則 [310](#)
- ソータートランスフォーメーション
- 作業用ディレクトリ、変数のタイプ [249](#)
- プッシュダウンの最適化 [127](#)

### 属性

- パーティションレベル [25](#)

### ソース

- コマンド [38](#)
- セッションのプロパティ [39](#)
- 同時読み込み [39](#)
- 入力行のソート順の維持 [43](#)
- パーティション化 [38](#)

### ソース側でのプッシュダウンの最適化

- 説明 [77](#)

### ソースデータ

- 集計のための変更のキャプチャ [327](#)

### ソースデータベース

- データベース接続セッションパラメータ [238](#)

### ソーステーブル

- パラメータおよび変数のタイプ [249](#)

### ソースの場所

- セッションのプロパティ [39](#)

### ソースパイプライン

- 説明 [18](#), [33](#), [61](#)

### ソースファイル

- FTP 経由でのアクセス [299](#), [300](#), [302](#)
- セッションのプロパティ [39](#)
- セッションパラメータ [238](#)
- パラメータの使用 [238](#)

### ソースファイルタイプ

- 説明 [39](#)

### ソースファイル名

- 説明 [39](#)

### ソースベースのコミット

- アクティブソース [156](#)
- 構成 [140](#)
- 説明 [155](#)
- リアルタイムセッション [140](#)

### ソータキャッシュ

- 説明 [324](#)

### ソータートランスフォーメーション

- キャッシュ [324](#)
- キャッシュの計算の入力 [324](#)
- キャッシュのパーティション化 [315](#), [324](#)
- 最適化された結合処理のパフォーマンスのパーティション化 [55](#)
- パーティション化 [58](#)

### ソート順

- 差分集計への影響 [329](#)
- 入力行について維持 [43](#)

### ソート済みフラットファイル

- 最適化された結合処理のパフォーマンスのパーティション化 [51](#)

### ソート済みポート

- キャッシュ要件 [317](#)

### ソート済みリレーショナルデータ

- 最適化された結合処理のパフォーマンスのパーティション化 [53](#)

### 存在する場合追加

- フラットファイルターゲットのプロパティ [45](#)

## た

### ターゲット

- FTP 経由でのアクセス [299](#), [300](#), [302](#)
- 出力ファイルの統合 [44](#), [45](#)
- パーティションポイントの削除 [34](#)
- パーティション化 [43](#), [44](#)
- プッシュダウンの最適化の使用 [114](#)
- プッシュダウンの最適化 [129](#)

### ターゲットテーブル

- パラメータおよび変数のタイプ [249](#)

### タイマータスク

- 変数 [226](#)

### タイマタスク

- 変数 [249](#)

### ターゲット側でのプッシュダウンの最適化

- 説明 [77](#)

### ターゲットコマンド

- ターゲット [46](#)
- パーティションでの使用 [46](#)

### ターゲット接続グループ

- データのコミット [155](#)
- トランザクションコントロールトランスフォーメーション [166](#)

### ターゲットデータベース

- データベース接続セッションパラメータ [238](#)

### ターゲットの更新

- パラメータおよび変数のタイプ [249](#)

### ターゲットファイル

- 追加 [つか] [45](#)
- セッションパラメータ [238](#)

### ターゲットファイルの統合

- FTP [44](#)
- FTP ファイルターゲット [305](#)
- コマンド [46](#)
- シーケンシャル統合 [47](#)
- セッションのプロパティ [45](#)
- 同時統合 [47](#)
- ファイルリスト [47](#)
- ローカル接続 [44](#), [45](#)

### ターゲットベースのコミット

- WriterWaitTimeOut [155](#)
- 構成 [140](#)
- リアルタイムセッション [140](#)

### ターゲットベースのコミット間隔

- 説明 [155](#)

### ターゲットリカバリテーブル

- 手動による作成 [186](#)

- 説明 [184](#)

### タスク

- 強制終了 [204](#)
- 自動リカバリ [192](#)
- 停止 [204](#)
- リカバリ戦略 [190](#)
- リソースの割り当て [223](#)
- ロードバランスの設定 [222](#)

### タスクリカバリ戦略の失敗

- 説明 [190](#), [192](#)

### タスクリカバリ戦略のリスタート

- 説明 [190](#), [192](#)

## ち

チェックポイント  
セッションリカバリ [192](#)  
操作のセッション状態 [183](#), [192](#)  
致命的なエラー  
セッションの失敗 [202](#)  
抽出日  
PeopleSoft、パラメータおよび変数のタイプ [249](#)

## つ

ツリー  
PeopleSoft、パラメータおよび変数のタイプ [249](#)

## て

停止  
Integration Service の処理 [201](#)  
エラーしきい値 [202](#)  
セッション [204](#)  
タスク [204](#)  
ワークフロー [203](#)  
ディンジョンタスク  
変数 [226](#)  
[ディンジョン] タスク  
変数のタイプ [249](#)  
ディレクトリ  
共有キャッシュ [311](#)  
履歴集計データ用 [331](#)  
データ暗号化  
FastExport 属性 [269](#)  
データベース接続  
セッションパラメータ [238](#)  
パスワード、パラメータのタイプ [249](#)  
パラメータタイプ [249](#)  
パラメータ [242](#)  
プッシュダウン互換性 [83](#)  
ユーザー名、パラメータのタイプ [249](#)  
データベースパーティション化。  
パフォーマンス [65](#)  
データベースビュー  
孤立したビューの削除 [108](#)  
トラブルシューティング [108](#)  
プッシュダウンの最適化 [108](#)  
プッシュダウンの最適化による作成 [107](#)  
リカバリ中の削除 [108](#)  
テーブル名  
アイドル状態のデータベースの構文 [85](#)  
プッシュダウン互換性の修飾 [85](#)  
データ  
ソースの差分変更のキャプチャ [327](#), [330](#)  
データ移動モード  
差分集計への影響 [329](#)  
データキャッシュ  
差分集計 [329](#)  
命名規則 [310](#)  
データのコミット  
ターゲット接続グループ [155](#)  
トランザクション制御 [159](#)  
データのロールバック  
トランザクション制御 [159](#)  
データファイル  
検索 [329](#)  
ディレクトリの作成 [331](#)

データベースシーケンス  
孤立シーケンスの削除 [108](#)  
トラブルシューティング [108](#)  
プッシュダウンの最適化 [108](#)  
リカバリ中の削除 [108](#)  
データベースパーティション化  
1つのソース [65](#)  
Integration Service に関するルールおよびガイドライン [67](#)  
Integration Service の処理 [67](#)  
説明 [22](#), [61](#)  
ソースに関するルールおよびガイドライン [67](#)  
ターゲット [68](#)  
ターゲットに関するルールおよびガイドライン [68](#)  
パフォーマンス [68](#)  
複数のソース [66](#)  
テーブル所有者名  
パラメータおよび変数のタイプ [249](#)  
テーブル名のプレフィックス  
ターゲット、パラメータおよび変数のタイプ [249](#)  
リレシヨナルエラーログ、長さ制限 [264](#)  
リレシヨナルエラーログ、パラメータおよび変数のタイプ [249](#)  
電子メール  
サスペンド時、変数のタイプ [249](#)  
セッション実行後、パラメータおよび変数のタイプ [249](#)  
電子メールタスク  
サスペンド時のメール [188](#)  
変数のタイプ [249](#)

## と

統合コマンド  
説明 [45](#)  
パラメータおよび変数のタイプ [249](#)  
統合タイプ  
説明 [45](#)  
統合ファイルディレクトリ  
説明 [45](#)  
パラメータおよび変数のタイプ [249](#)  
統合ファイル名  
説明 [45](#)  
パラメータおよび変数のタイプ [249](#)  
動作可能なデータベース  
説明 [79](#)  
同時接続  
パーティション化されたパイプライン [44](#)  
同時統合  
ファイルターゲット [47](#)  
同時読み込みのパーティション化  
セッションのプロパティ [39](#)  
動的パーティション  
CPU の数を基準 [23](#)  
グリッドのノード数を基準 [23](#)  
説明 [23](#)  
ソースパーティションの使用 [23](#)  
パーティションタイプでの使用 [24](#)  
パーティションの数、パラメータのタイプ [249](#)  
パーティションの数を基準 [23](#)  
パフォーマンス [23](#)  
無効 [23](#)  
ルールおよびガイドライン [24](#)  
トランザクション  
定義 [162](#)  
トランザクション環境 SQL  
パラメータおよび変数のタイプ [249](#)  
トランザクション境界  
削除 [162](#)  
トランザクション制御 [162](#)

トランザクションコントロールトランスフォーメーション

ターゲット接続グループ [166](#)

パーティション化のガイドライン [63](#)

トランザクションジェネレータ

トランザクション制御ポイント [162](#)

トランザクション制御

Integration Service の処理 [159](#)

オーブントランザクション [162](#)

概要 [162](#)

拒否ファイル [160](#)

トランスフォーメーションエラー [160](#)

トランスフォーメーション範囲 [163](#)

バルクロード [159](#)

ファイルの末尾 [160](#)

ポイント [162](#)

ユーザー定義コミット [159](#)

リアルタイムセッション [162](#)

ルールおよびガイドライン [167](#)

トランザクション制御単位

説明 [166](#)

トランスフォーメーション

キャッシュ [307](#)

再現可能なデータの生成 [194](#)

差分アグリゲータによるセッションのリカバリ [183](#)

パーティション化の制限 [59](#)

ブッシュダウンの最適化の設定 [117](#)

リアルタイムセッション [149](#)

トランスフォーメーション式

パラメータおよび変数のタイプ [249](#)

トランスフォーメーション範囲

説明 [163](#)

トランスフォーメーション [163](#)

リアルタイム処理 [163](#)

## に

入力タイプ

ファイルソースのパーティション化のプロパティ [39](#)

## の

ノーマライズトランスフォーメーション

パーティションポイントの使用 [34](#)

## は

パーティション

XML ジェネレータ [59](#)

計算 [23](#)

削除 [29](#)

セッションのプロパティ [45](#)

説明 [20](#)

説明の入力 [29](#)

ターゲットデータの統合 [46](#)

追加 [29](#)

ブッシュダウンの最適化のための統合 [114](#)

パイプライン

説明 [18, 33, 61](#)

パイプラインステージ

説明 [18](#)

パイプラインのパーティション化

FTP ファイルターゲット [305](#)

HTTP トランスフォーメーション [47](#)

Informix へのロード [44](#)

Java トランスフォーメーション [47](#)

パイプラインのパーティション化 (続く)

SQL クエリ [37](#)

エラーしきい値 [202](#)

オブジェクトの検証 [27](#)

ガイドライン [38](#)

外部ローダー [44, 276](#)

カスタムトランスフォーメーション [47](#)

間接ファイルのパーティション化 [39](#)

キー範囲 [70](#)

キー範囲の追加 [72](#)

キャッシュ [25](#)

グリッド上 [217](#)

結合処理のパフォーマンスを最適化するための設定 [50](#)

シーケンスジェネレータトランスフォーメーション [58](#)

ジョイナトランスフォーメーション [50](#)

使用例 [62](#)

数値関数の制限 [60](#)

スレッドとパーティション [20](#)

セッションの設定 [28](#)

説明 [18, 33, 61](#)

ソータトランスフォーメーション [55, 58](#)

ソート済みデータの設定 [50](#)

ソート済みフラットファイル [51](#)

ソート済みリレーショナルデータ [53](#)

ターゲットファイルの統合 [44, 45](#)

データベースの互換性 [44](#)

同時接続 [44](#)

動的パーティション [23](#)

トランザクションコントロールトランスフォーメーション [63](#)

パイプラインステージ [18](#)

パススルーパーティションタイプタイプ [73](#)

ハッシュキーの追加 [69](#)

ハッシュ自動キーのパーティション化 [69](#)

ハッシュユーザーキーのパーティション化 [69](#)

パーティションキー [69, 71](#)

パーティションポイントの編集 [28](#)

パフォーマンス [69, 75](#)

パフォーマンス [71](#)

ファイルソース [38](#)

ファイルターゲット [44](#)

ファイルリスト [39](#)

フィルタ条件 [37](#)

複数グループトランスフォーメーション [21](#)

ブッシュダウンの最適化の設定 [112](#)

マッピング変数 [26](#)

メッセージキュー [44](#)

有効なパーティションタイプ [63](#)

ラウンドロビンパーティション化 [75](#)

リカバリ [202](#)

リレーショナルターゲット [43](#)

ルール [27](#)

パイプラインルックアップ

ソーステーブルのパーティション化 [57](#)

パススルーパーティションタイプ

概要 [61](#)

処理 [73](#)

説明 [22](#)

パフォーマンス [73](#)

ブッシュダウンの最適化 [114](#)

ハッシュ自動キーのパーティション化

概要 [69](#)

説明 [22](#)

ハッシュパーティション化

説明 [61](#)

ハッシュキーの追加 [69](#)

ハッシュユーザーキー

説明 [22](#)

## ハッシュユーザーキーのパーティション化

概要 [69](#)

パフォーマンス [69](#)

## バッファブロックサイズ

構成 [339](#)

## バッファメモリ

構成 [339](#)

バッファブロック [339](#)

割り当て [339](#)

## パーティション化

差分集計 [330](#)

ジョイナトランスフォーメーション [317](#)

パイプラインルックアップソーステーブル [57](#)

パフォーマンス [75](#)

複数のターゲットでの FTP の使用 [301](#)

## パーティションカウント

セッションパラメータ [238](#)

## パーティション化の制限

Informix [44](#)

Sybase IQ [44](#)

XML ジェネレータ [59](#)

XML ターゲット [59](#)

数値関数 [60](#)

接続されていないトランスフォーメーション [35](#)

トランスフォーメーション [59](#)

パーティションの数 [27](#)

リレーショナルターゲット [44](#)

## パーティションキー

NULL 値を含む行 [72](#)

キー範囲の追加 [72](#)

追加 [69](#), [71](#)

ルールおよびガイドライン [73](#)

## パーティショングループ

ステージ [216](#)

説明 [216](#)

## パーティションタイプ

概要 [22](#)

キー範囲 [70](#)

設定 [63](#)

説明 [61](#)

デフォルト [63](#)

パススルー [73](#)

パーティションポイントでの使用 [63](#)

パフォーマンス [62](#)

変更 [29](#)

ラウンドロビン [75](#)

## パーティションの数

概要 [20](#)

セッションパラメータ [238](#)

動的パーティションでの設定 [23](#)

パフォーマンス [20](#)

## パーティションの数を基準

設定 [23](#)

## パーティションポイント

HTTP トランスフォーメーション [47](#), [48](#)

Java トランスフォーメーション [47](#), [48](#)

概要 [19](#)

カスタムトランスフォーメーション [47](#), [48](#)

ジョイナトランスフォーメーション [50](#)

追加および削除 [33](#)

追加、手順 [29](#)

編集 [28](#)

ルックアップトランスフォーメーション [56](#)

## パーティション名

設定 [29](#)

## パーティションレベルの属性

説明 [25](#)

## パフォーマンス

キャッシュ設定 [312](#)

コミット間隔 [155](#)

## パラメータ

セッション [238](#)

タイプの概要 [248](#)

データベース接続 [242](#)

パラメータファイルでの定義 [249](#)

パラメータを受け入れる入力フィールド [249](#)

## パラメータファイル

NULL 値、入力 [260](#)

pmcmd での使用 [263](#)

概要 [247](#)

構造体 [258](#)

コメント、追加 [260](#)

コンカレントワークフローインスタンスの設定 [208](#)

作成ガイドライン [258](#), [264](#)

作成のヒント [266](#)

サンプルのパラメータファイル [260](#)

使用するファイルを指定 [247](#)

使用例 [263](#)

セクション [259](#)

セッションでの使用 [261](#)

セッションパラメータファイル名、変数のタイプ [249](#), [262](#)

接続属性のオーバーライド [257](#)

説明 [247](#)

テンプレートファイル [257](#)

トラブルシューティング [265](#)

名前、設定 [261](#)

日時形式 [264](#)

場所、設定 [261](#)

パラメータおよび変数のタイプ [248](#)

パラメータおよび変数の範囲 [258](#)

パラメータおよび変数を受け入れる入力フィールド [249](#)

プロパティの定義 [249](#)

変数を使用した指定 [262](#)

見出し [259](#)

優先 [263](#)

ワークフローでの使用 [261](#)

## バルクロード

ユーザー定義コミットの使用 [159](#)

## ひ

### 非永続変数

定義 [232](#)

## ふ

### ファイルソース

コードページ、パラメータおよび変数のタイプ [249](#)

ディレクトリ、パラメータおよび変数のタイプ [249](#)

名前、パラメータおよび変数のタイプ [249](#)

入力ファイルコマンド、パラメータおよび変数のタイプ [249](#)

パーティション化 [38](#)

### ファイルターゲット

コードページ、パラメータおよび変数のタイプ [249](#)

パーティション化 [44](#)

### ファイルの末尾

トランザクション制御 [160](#)

### ファイルリスト

ターゲットファイルの統合 [47](#)

パーティション化されたソースの作成 [39](#)

### フィルタ条件

WebSphere MQ、パラメータおよび変数のタイプ [249](#)

追加 [72](#)



フィルタ条件 (続く)  
パーティション化されたパイプライン [37](#)  
パラメータおよび変数のタイプ [249](#)  
フィルタトランスフォーメーション  
プッシュダウンの最適化 [120](#)  
複数グループトランスフォーメーション  
パーティション化 [21](#)  
プッシュダウングループ  
説明 [115](#)  
表示 [115](#)  
プッシュダウンの最適化ビューア、使用 [115](#)  
プッシュダウン互換性  
互換性がないデータベースユーザー [85](#)  
説明 [83](#)  
要求条件 [83](#)  
プッシュダウンの最適化  
    \$\$PushdownConfig パラメータ[ぶっしゅだうんのさいてきか  
        pushdown config] [110](#)  
SQL と ANSI SQL [79](#)  
アグリゲータトランスフォーメーション [119](#)  
アップデートストラテジトランスフォーメーション [132](#)  
一時シーケンス [106](#)  
エラー処理 [104](#)  
概要 [76](#)  
完全なプッシュダウンの最適化 [78](#)  
キー範囲パーティション化、使用 [114](#)  
シーケンスジェネレータトランスフォーメーション [126](#)  
式 [87](#)  
ジョイナトランスフォーメーション [121](#)  
生成される SQL [77](#), [78](#)  
セッション [77](#)  
セッションの設定 [111](#)  
ソース修飾子トランスフォーメーション [128](#)  
ソートトランスフォーメーション [127](#)  
ソース側での最適化 [77](#)  
ソースデータベースのパーティション化 [67](#)  
ターゲット [129](#)  
ターゲット側での最適化 [77](#)  
ターゲットにロード [114](#)  
データベースビューの作成 [107](#)  
データベースシーケンス [108](#)  
データベースビュー [108](#)  
トランスフォーメーション [117](#)  
ネイティブデータベースドライバ [79](#)  
パススルーパーティションタイプ [114](#)  
パーティション化の設定 [112](#)  
パーティションの統合 [114](#)  
パフォーマンスの問題 [78](#)  
パラメータタイプ [249](#)  
フィルタトランスフォーメーション [120](#)  
マッピングへのトランスフォーメーションの追加 [115](#)  
マッピング変数 [88](#)  
リカバリ [104](#)  
ルータトランスフォーメーション [125](#)  
ルールおよびガイドライン [115](#)  
ロギング [104](#)  
一時ビュー [107](#)  
演算子 [87](#)  
関数 [89](#)  
共有体トランスフォーメーション [131](#)  
式トランスフォーメーション [120](#)  
プッシュダウンの最適化ビューア  
    プッシュダウングループの表示 [115](#)  
プッシュダウン最適化  
    AWS Redshift [80](#)  
    Azure DW [80](#)  
    Google Big Query [80](#)  
    Greenplum [80](#)

プッシュダウン最適化 (続く)  
    ODBC 接続 [80](#)  
    PostgreSQL [80](#)  
    Snowflake [80](#)  
フッタ  
    パラメータおよび変数のタイプ [249](#)  
    ファイルターゲットでの作成 [45](#)  
フッタのコマンド  
    フラットファイルターゲット [45](#)  
フラッシュ待ち時間  
    構成 [139](#)  
    説明 [139](#)  
フラットファイル  
    出力ファイルセッションパラメータ [238](#)  
    ソースファイルセッションパラメータ [238](#)  
    入力行の順序の維持 [43](#)  
    フッタのコマンドプロパティ [45](#)  
    ヘッダのオプションプロパティ [45](#)  
    ヘッダのコマンドプロパティ [45](#)  
フラットファイル  
    リカバリの設定 [194](#)  
フラットファイルのロギング  
    エラーログタイプ、設定 [180](#)  
    エラーログファイルディレクトリ、設定 [180](#)  
    エラーログファイル名、設定 [180](#)  
フルリカバリ  
    説明 [192](#)  
ブロックサイズ  
    FastExport 属性 [269](#)



並列実行の設定  
    ワークフローインスタンスの設定 [209](#)  
ベース URL  
    パラメータおよび変数のタイプ [249](#)  
ヘッダ  
    パラメータおよび変数のタイプ [249](#)  
    ファイルターゲットでの作成 [45](#)  
ヘッダのオプション  
    フラットファイルターゲット [45](#)  
ヘッダのコマンド  
    フラットファイルターゲット [45](#)  
変更データ  
    PowerExchange リアルタイム変更データキャプチャ [135](#)  
変数  
    \$PMWorkflowRunId [208](#)  
    \$PMWorkflowRunInstanceName [208](#)  
    タイプの概要 [248](#)  
    パラメータファイルでの定義 [249](#)  
    変数を受け入れる入力フィールド [249](#)  
    ワークフロー [226](#)  
変数値  
    パーティション全体での計算 [26](#)

## ま

待ち時間  
    説明 [134](#)  
マッピング  
    パーティションからのセッションの失敗 [27](#)  
マッピングパラメータ  
    \$\$PushdownConfig  
        pushdown config] [110](#)  
オーバーライド [244](#)  
セッション間での値の受け渡し [245](#)



マッピングパラメータ (続く)  
セッションプロパティ内 [244](#)  
パラメータファイル [248](#)  
マッピング変数  
セッション間での値の受け渡し [245](#)  
データベース内で使用可能 [88](#)  
パーティション化されたパイプライン [26](#)  
パラメータファイル [248](#)  
プッシュダウンの最適化 [88](#)  
マルチバイトデータ  
Oracle 外部ローダー [283](#)  
Sybase IQ 外部ローダー [285](#)  
Teradata FastExport [269](#)

## め

命名規則  
セッションパラメータ [238](#)  
メッセージの処理  
リアルタイムセッション [145](#), [146](#)  
リカバリキュー [146](#)  
リカバリテーブル [145](#)  
リカバリトリック [146](#)  
ルールおよびガイドライン [149](#)  
メッセージカウント  
構成 [139](#)  
メッセージキュー  
パーティション化されたパイプラインでの使用 [44](#)  
リアルタイムデータの処理 [135](#)  
メッセージとメッセージキュー  
リアルタイムデータ [135](#)  
メッセージリカバリ  
セッションリカバリデータのフラッシュ [144](#)  
説明 [140](#)  
有効化 [141](#)  
リアルタイムセッション [140](#), [144](#), [146](#)  
リカバリキュー [140](#), [146](#)  
リカバリテーブル [140](#), [146](#)  
リカバリトリック [140](#), [146](#)  
リカバリファイル [140](#), [144](#)  
ルールおよびガイドライン [150](#)  
前提条件 [141](#)  
メモリ  
キャッシュ [308](#)  
自動設定の設定 [340](#)  
複数のセッションに対する設定 [340](#)  
メモリ設定  
複数のセッションでの設定 [340](#)  
メモリ要件  
DTM バッファサイズ [340](#)  
セッションキャッシュサイズ [341](#)

## ゆ

有効日  
PeopleSoft、パラメータおよび変数のタイプ [249](#)  
優先度  
タスクへの割り当て [222](#)  
ユーザー定義結合  
パラメータおよび変数のタイプ [249](#)  
ユーザー定義コミット  
バルクロード [159](#)

## ら

ラウンドロビンパーティション化  
説明 [22](#), [61](#), [75](#)  
リンクキャッシュ  
説明 [322](#)  
ランクトランスフォーメーション  
キャッシュ [322](#)  
キャッシュの計算の入力 [323](#)  
キャッシュの設定 [323](#)  
キャッシュのパーティション化 [315](#), [322](#)  
パーティションポイントの使用 [34](#)

## り

リアルタイム処理  
サンプルマッピング [150](#)  
説明 [134](#)  
リアルタイムセッション  
PM\_REC\_STATE テーブル [145](#)  
Reader 時間制限、設定 [139](#)  
アイドル時間、設定 [138](#)  
概要 [134](#)  
強制終了 [147](#)  
構成 [138](#)  
コミットタイプ、設定 [140](#)  
コールドスタート [148](#)  
再開 [148](#)  
再起動 [148](#)  
サポートされる製品 [152](#)  
サンプルマッピング [150](#)  
終了条件、設定 [138](#)  
説明 [134](#)  
停止 [147](#)  
トランスフォーメーション [149](#)  
トランスフォーメーション範囲 [163](#)  
フラッシュ待ち時間、設定 [139](#)  
メッセージカウント、設定 [139](#)  
メッセージの処理 [145](#), [146](#)  
メッセージリカバリ [144](#), [146](#)  
リカバリ [148](#)  
ルールおよびガイドライン [149](#)  
レジリエンス [149](#)  
リアルタイムデータ  
PowerExchange ソースからの変更データ [135](#)  
Web サービスメッセージ [135](#)  
概要 [135](#)  
サポートされる製品 [152](#)  
メッセージ、メッセージキュー、および変更データキャプチャ [135](#)  
リアルタイムフラッシュ待ち時間  
構成 [139](#)  
リカバリ  
PM\_RECOVERY テーブル形式 [184](#)  
PM\_TGT\_RUN\_ID テーブル形式 [184](#)  
SDK ソース [194](#)  
インスタンス名によるリカバリ [206](#)  
概要 [182](#)  
グリッド上のセッション [219](#)  
グリッド上のワークフロー [219](#)  
最後のチェックポイントから再開 [190](#), [192](#)  
差分 [192](#)  
差分アグリゲータを含むセッション [183](#)  
実行 ID によるワークフローのリカバリ [207](#)  
セッション内の再現可能なデータ [194](#)  
セッションの検証 [194](#)  
戦略 [190](#)  
操作のセッション状態 [183](#)

## リカバリ (続く)

- ターゲットリカバリテーブル [184](#)
- タスクからのワークフローのリカバリ [199](#)
- タスクのリカバリ [198](#)
- チェックポイントからのセッション [192](#)
- データベースシーケンスの削除 [108](#)
- データベースビューの削除 [108](#)
- パイプラインのパーティション化 [202](#)
- プッシュダウンの最適化 [104](#)
- フラットファイル [194](#)
- フルリカバリ [192](#)
- リアルタイムセッション [140](#)
- リカバリ不可能なセッションの完了 [200](#)
- ルールおよびガイドライン [199](#)
- ワークフローの操作の状態 [183](#)
- リカバリ可能なタスク
  - 説明 [190](#)
- リカバリキャッシュフォルダー
  - JMS 用の変数のタイプ [249](#)
  - TIBCO 用の変数のタイプ [249](#)
  - webMethods 用の変数のタイプ [249](#)
  - WebSphere MQ 用の変数のタイプ [249](#)
- リカバリキュー
  - メッセージの処理 [146](#)
  - メッセージリカバリ [140, 146](#)
- リカバリ戦略
  - 最後のチェックポイントから再開 [190, 192](#)
  - タスクのリスタート [190, 192](#)
  - タスクを失敗してワークフローを続行 [190, 192](#)
- リカバリテーブル
  - スクリプトからの手動での作成 [186](#)
  - 説明 [184](#)
  - メッセージの処理 [145](#)
  - メッセージリカバリ [140, 146](#)
- リカバリトピック
  - メッセージの処理 [146](#)
  - メッセージリカバリ [140, 146](#)
- リカバリファイル
  - メッセージリカバリ [140, 144](#)
- リソース
  - 外部ローダーの割り当て [274](#)
  - タスクへの割り当て [223](#)
- リレーショナルターゲット
  - パーティション化 [43](#)
  - パーティション化の制限 [44](#)
- リレーショナルデータベースのロギング
  - エラーログタイプ、設定 [180](#)
- リンク
  - 変数 [226](#)
  - 変数のタイプ [249](#)

## る

- ルータトランスフォーメーション
  - プッシュダウンの最適化 [125](#)
- ルックアップ
  - 永続キャッシュ [321](#)
- [ルックアップ SQL オーバーライド] オプション
  - パラメータおよび変数のタイプ [249](#)
- ルックアップキャッシュ
  - 説明 [321](#)
  - ファイル名のプレフィックス、パラメータおよび変数のタイプ [249](#)
- ルックアップソースファイル
  - パラメータの使用 [238](#)
- ルックアップデータベース
  - データベース接続セッションパラメータ [238](#)

## ルックアップトランスフォーメーション

- キャッシュ [321](#)
- キャッシュの計算の入力 [322](#)
- キャッシュの設定 [322](#)
- キャッシュのパーティション化 [56, 315, 321](#)
- コンカレントワークフローへの追加 [212](#)
- 接続情報、パラメータおよび変数のタイプ [249](#)
- ソースファイル、パラメータおよび変数のタイプ [249](#)
- プッシュダウンの最適化 [122](#)
- ルックアップファイル
  - ルックアップファイルセッションパラメータ [238](#)

## れ

- レジリエンス
  - リアルタイムセッション [149](#)

## ろ

- ロギング
  - プッシュダウンの最適化 [104](#)
- ロードバランサ
  - タスクへの優先度の割り当て [222](#)
  - タスクへのリソースの割り当て [223](#)
  - ワークフローの設定 [222](#)

## わ

- ワークフロー
  - pmcmd を使用したコンカレントワークフローの開始 [210](#)
  - 一意のインスタンスの設定 [206](#)
  - インスタンス名の設定 [209](#)
  - 強制終了 [203](#)
  - グリッド上での実行 [215](#)
  - グリッド上での分散 [215, 219](#)
  - グリッド上でのリカバリ [219](#)
  - コンカレントインスタンス [205](#)
  - コンカレントワークフローのスケジューリング [212](#)
  - サスペンド中 [188](#)
  - サービスレベル [222](#)
  - ステータス [188](#)
  - 操作の状態 [183](#)
  - タスクのディスパッチ [222](#)
  - 停止 [203](#)
  - 同一名のコンカレントの設定 [206](#)
  - パラメータファイル [232](#)
  - 変数 [226](#)
- ワークフローインスタンス
  - \$PMWorkflowRunInstanceName 変数の使用 [208](#)
  - Workflow Monitor での表示 [211](#)
  - 開始および停止 [209](#)
  - コマンドラインからの開始 [210](#)
  - 説明 [205](#)
  - 動的に作成 [210](#)
  - ワークフローインスタンスの追加 [209](#)
- ワークフローの実行 ID
  - 説明 [206](#)
  - ワークフローログに表示 [212](#)
- ワークフローのスケジューリング
  - コンカレントワークフロー [212](#)
- ワークフローのリカバリ
  - インスタンス名によるワークフローのリカバリ [206](#)
  - 実行 ID によるインスタンスのリカバリ [207](#)
- ワークフロープロパティ
  - サービスレベル [222](#)

## ワークフロー変数

永続変数 [232](#)

キーワード [226](#)

組み込み変数 [227](#)

作成 [233](#)

式での使用 [230](#)

使用 [226](#)

初期値とカレント値 [232](#)

セッション間での値の受け渡し [245](#)

定義済み [227](#)

データー型 [233](#)

データ型 [227](#)

デフォルト値 [227](#), [232](#), [233](#)

日時形式 [233](#)

パラメータファイル [248](#)

非永続変数 [232](#)

命名規則 [233](#)

ユーザー定義 [231](#)

ワークレット間での値の受け渡し [235](#)

## ワークフローログ

ファイル名およびディレクトリ、変数のタイプ [249](#)

ワークフローログカウンタ数、変数のタイプ [249](#)

## ワークフローログファイル

コンカレントワークフローの表示 [211](#)

## ワークレット

永続変数 [234](#)

## ワークレット (続く)

永続変数の例 [234](#)

コンカレントワークフローへの追加 [212](#)

情報の受け渡し [235](#)

情報の受け渡し、例 [236](#)

[パラメータ] タブ [235](#)

変数 [234](#)

変数値のオーバーライド [235](#)

ワークレット実行前および実行後の変数の割り当て、手順 [236](#), [246](#)

ワークレット実行前および実行後の変数割り当て [235](#)

ワークレット実行後の変数割り当て

実行 [235](#)

ワークレット実行前の変数割り当て

実行 [235](#)

ワークレット変数

セッション間での値の受け渡し [245](#)

パラメータファイル [248](#)

ワークレット間での値の受け渡し [235](#)

割り当てタスク

変数 [226](#), [249](#)