PUBLIC (公開)
SAP BusinessObjects Business Intelligence プラットフォーム
ドキュメントバージョン: 4.3 Support Package 4 – 2023-12-07

Business Intelligence Platform Java SDK 開発者ガイド

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

目次

1 ドキュメント履歴 6

2 はじめに 7

2.1 SAP BusinessObjects Business Intelligence プラットフォーム Java SDK の新機能 7

JAR ファイルの変更 9

2.2 アプリケーションの移行 9

非推奨 API 9

監査の変更 12

3 SDK 基本 15

3.1 クラスの概要 15

3.2 共通のワークフロー 19

InfoObject を作成する 19

InfoObject を削除する 21

InfoObject を変更する 22

3.3 クエリー言語を使用したオブジェクトの取得 24

クエリーを実行するには、次の手順を実行します。 25

オブジェクトタイプ 26

クエリー結果内のプロパティへのアクセス 29

SELECT 句 32

FROM 句 34

WHERE 句 34

ORDER BY 句 42

クエリー文字列のベストプラクティス 43

3.4 URI クエリーを使用したオブジェクトの取得 47

予備知識 48

URI パスクエリーを実行するには、次の手順を実行します。 48

オブジェクトタイプ 50

各種のプロトコルです。 52

演算子 55

リレーションシップ 60

URI クエリのベストプラクティス 64

3.5 多言語コンテンツの操作 66

3.6 この SDK のベストプラクティス 67

未使用のセッションとリソースのクリーンアップ 67

IEnterpriseSession オブジェクトの格納 68

サポートされるインターフェイスの判別 68

SDK 例外のキャッチおよび処理. 69
一貫性ある CMS の状態の維持. 70
3.7 水平または垂直クラスタでのオブジェクトのシリアル化. 71
3.8 同じ CMS を使用したオブジェクトの読み書き. 73

4 開発環境をセットアップする. 74
4.1 コンパイルのターゲット JVM を設定する. 74
4.2 Web アプリケーションの設定. 74
Web アプリケーションを設定する. 74
web.xml ファイルの設定. 75
SAP BusinessObjects ソフトウェアのデプロイメントに必要な JAR ファイル. 82
4.3 SAP BusinessObjects Business Intelligence platform 環境の設定. 88
必要な SAP BusinessObjects Business Intelligence platform コンポーネント. 88
SAP BusinessObjects Business Intelligence platform の設定. 88
4.4 Report Application Server 環境の設定. 89
必要な Report Application Server のコンポーネント. 89
Report Application Server の場所を指定する. 91
レポートの場所. 92
水平および垂直クラスタリング. 93
4.5 SAP Crystal Reports ビューアの設定. 94
必要なビューアコンポーネント. 94
ビューア SDK のサイドバイサイドインストール. 95
4.6 ディレクトリ. 95

5 SDK の使用. 97
5.1 認証. 97
認証の基本. 98
CMS へのログオン. 99
ログオフしてライセンスを解放する. 106
5.2 拡張認証情報マッピング. 107
5.3 セキュリティ. 108
ユーザとグループ. 109
アクセス権の設定. 113
5.4 スケジューリング. 142
スケジュールのワークフロー. 142
出力先. 143
レポートをスケジュールする. 151
カスタム値を使用してレポートをスケジュールする. 153
特定のパラメータ値を含むレポートをスケジュールする. 155
カレンダーを使用してレポートをスケジュールする. 158
イベントを使用してレポートをスケジュールする. 161
5.5 イベントとアラート. 163

スケジュールイベントを作成する. 164
ファイルイベントを作成する. 165
カスタムイベントを作成する. 166
イベントのアラートを有効にする. 167
イベントを購読するには. 168
カスタムイベントを発生させるには. 170
アラート通知を表示する. 171
5.6 パブリケーション. 172
公開のワークフロー. 173
パブリケーションを作成するには. 173
パブリケーションへのドキュメントの追加. 175
パブリケーションへの受信者の追加. 180
パブリケーションドキュメントの個人用カスタマイズ. 186
出力先および出力形式の設定. 194
パブリケーション例外のトラブルシューティング. 197
パブリケーション拡張. 200
5.7 サーバー管理. 210
Server Intelligence のアーキテクチャ. 211
Server Intelligence のコンポーネント. 212
サーバーを管理する. 213
サーバーの追加. 220
サービスとサービスコンテナの設定. 227
サーバーグループの管理. 235
5.8 フェデレーション. 237
フェデレーションの設定. 238
フェデレーションに使用されるクラス. 238
レプリケーション一覧を作成する. 239
リモート接続を作成する. 241
レプリケーションジョブを作成およびスケジュールする. 243
リモートスケジュール. 245
依存タイプ. 247
5.9 モニタリング. 248
新しいプローブの追加. 248
5.10 プラットフォーム検索. 253
プラットフォーム検索に使用されるクラス. 254
検索結果を取得する 254
5.11 監査 259
監査イベントを取得する. 260
監査イベント詳細を取得する. 261
現在の監査レベルを変更する. 263
監査イベントを有効にする. 265

監査イベント詳細を有効にする. 266
カスタム監査イベントを復元する. 268
監査カテゴリを取得する. 270
サーバー監査メトリクスを取得する. 272
監査データストア接続情報を取得する. 273
監査イベントの格納期間を変更する. 275
トレーサビリティを強化してカスタムアプリケーション全体を監査する. 277
カスタムアプリケーションの CUID 値と名前. 278
5.12 ローカライズと言語パック. 279
優先表示ロケールを設定する. 280
現在の優先表示ロケールを表示する. 281
インストールされている言語パックを表示する. 281
サポートされているすべての言語を表示する. 282
5.13 暗号化キーの管理. 283
暗号化キーの状態. 284
新しい暗号化キーを作成する. 285
新しいキーでデータを再暗号化する. 286
暗号化キーを削除する. 288
5.14 Business Intelligence Archive (BIAR) 289
インポートに関する考慮事項. 291
BIAR ファイルに InfoObject をエクスポートする. 291
BIAR ファイルからインポートする. 294
live-to-live 転送を実行する. 296
コールバックを実行する. 298

6 参照. 301
6.1 処理拡張機能 API リファレンス. 301
拡張機能の作成. 301
拡張機能のロード. 301
処理拡張機能 API. 303
例. 336

1 ドキュメント履歴

この表は、最も重要なドキュメント変更の概要です。

バージョン	日付	説明
SAP BusinessObjects Business Intelligence プラットフォーム 4.3 SP2	2021 年 12 月	SAP BusinessObjects ソフトウェアのデプロイメントに必要な JAR ファイル [82 ページ]が更新されました。
SAP BusinessObjects Business Intelligence プラットフォーム 4.3	2020 年 6 月	初期リリース

2 はじめに

SAP BusinessObjects Business Intelligence プラットフォーム 4.1 Java SDK を使用すると、BI プラットフォームと直接やり取りしてユーザ認証、スケジュール、パブリケーション、サーバ管理などのタスクを実行するアプリケーションを構築できます。

SDK の使用開始

- SDK の使用 [97 ページ]
  タスクやコード例で、この SDK の一般的な使用例を理解します。
- 開発環境をセットアップする [74 ページ]
  開発環境にデプロイする JAR ファイル、および SDK を操作するための Web アプリケーションを構成する方法を確認します。

SDK に関する情報

- SAP BusinessObjects Business Intelligence プラットフォーム Java SDK の新機能 [7 ページ]
  このリリースの SDK の新機能に関する情報です。
- アプリケーションの移行 [9 ページ]
  このリリースで使用されなくなった API と、変更が既存のアプリケーションにどのように影響するかについて説明します。
- SDK 基本 [15 ページ]
  主なクラスとインタフェースの概要など、SDK の基本について説明します。

2.1 SAP BusinessObjects Business Intelligence プラットフォーム Java SDK の新機能

この節では、SAP BusinessObjects Business Intelligence プラットフォーム 4.1 Java SDK で提供される新しい機能と拡張機能の概要を示します。

サーバ監査

サーバ監査とクライアント監査の両方を統合したまったく新しいフレームワークがあります。監査イベントオブジェクトは、リポジトリで一元的インタフェースによって整理および操作できるようになりました。監査イベン

トは、それに関連付けられている詳細情報に基づいてデータを記録します。これらの監査詳細は、監査イベントとは独立して構成されます。

詳細については、監査 [259 ページ]を参照してください。

OFF から *COMPLETE* までの監査レベルで有効な監査イベントを管理できるようになりました。高い柔軟性とカスタマイズ性を提供する *CUSTOM* 監査レベルもあります。

> ⓘ 注記
>
> 既存のアプリケーションを引き続き正しく動作させるには、監査の変更 [12 ページ]を参照してください。

データセキュリティ

BI プラットフォームでは、対称暗号化キーを使用して、機密データの暗号化と解読を行います。

暗号化キーを管理するには、暗号管理者グループのメンバーであるアカウントを使用して、BI プラットフォームにログオンする必要があります。この SDK を使用して、プログラムで暗号化キーを作成、無効化、削除できます。

詳細については、暗号化キーの管理 [283 ページ]を参照してください。

アラート

アラート機能を使用すると、イベントが発生したときにユーザとグループに通知を送信できます。アラート機能を使用するには、イベントのアラートを有効にし、ユーザまたはグループをイベントの購読に登録します。イベントが発生すると、購読ユーザとグループにアラート通知メッセージが自動的に送信されます。アラート可能なイベントを購読すると、イベントが発生したときにアラート通知が自動的にユーザに送信されるため、手動でイベントの発生を確認する必要はありません。

たとえば、アラート機能を使用して、スケジュールされたジョブの正常完了、ドキュメントのアラートの発生、パフォーマンスしきい値の超過などをユーザに通知できます。

詳細は、次を参照してください。 イベントとアラート [163 ページ]

Business Intelligence Archive (BIAR)

Central Management Server (CMS) 間でオブジェクトを直接転送できる新しい API が追加されました。BIAR ファイルを使用しなくても、プログラムでオブジェクトをソース環境からエクスポートし、そのオブジェクトを目的の環境にインポートすることができます。`com.businessobjects.sdk.biar.ILiveToLivePipe` インタフェースがこの転送をサポートします。

詳細については、Business Intelligence Archive (BIAR) [289 ページ]と live-to-live 転送を実行する [296 ページ]を参照してください。

CeProgID および CeKind 定数へのアクセス

各プラグインインタフェースには、それぞれ ProgID および CeKind 値に対応する定数が含まれます。たとえば、CeKind.SERVER を呼び出す代わりに、IServer.KIND を呼び出します。同様に、CeProgID.SERVER を IServer.PROGID に代えて使用できます。CeProgID および CeKind インタフェースは非推奨となりました。

詳細については、非推奨 API [9 ページ]を参照してください。

2.1.1 JAR ファイルの変更

この節では、BI4.0 と比較してこのリリースの SDK で使用されなくなった古い JAE ファイルの一覧を示します。非推奨のアイテムは下位互換性を保つためにサポートされますが、現在のバージョンの SDK には含まれておらず、使用することはお奨めしません。

BI4.0 から BI4.1 への JAR ファイルの変更

新しい JAR ファイル	非推奨の JAR ファイル
ebus405.jar	backport-util-concurrent-2.2.jar
	stax-api-1.0.1.jar
	wxts-asl-3.2.1.jar

2.2 アプリケーションの移行

このセクションでは、この SDK の以前のバージョンからアップグレードするときに注意が必要な変更についての情報を提供します。

2.2.1 非推奨 API

このセクションでは、このリリースの SDK で使用されなくなった古い API の一覧を示します。非推奨のアイテムは下位互換性を保つためにサポートされますが、現在のバージョンの SDK で使用することはお勧めしません。

> **注記**
>
> 非推奨 API メンバーとその代替物の完全な一覧については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスを参照してください。

非推奨インターフェイス

インターフェイス名	詳細
com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditDetail	com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail に置き換えられました。
com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditDetails	com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails に置き換えられました。
com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditEvent	com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent に置き換えられました。
com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditEventInfo	com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo に置き換えられました。
com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditEvents	com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents に置き換えられました。
com.crystaldecisions.sdk.plugin.authentication.secwinnt.IsecWinNT	Windows NT 認証はサポートされなくなりました。
com.crystaldecisions.sdk.plugin.CeKind	各プラグインインターフェイスに、対応するオブジェクトタイプの SI_KIND プロパティを指定するフィールドが追加されました。以下はその例です。 • IFolder.KIND が CeKind.FOLDER に代えて使用されます。 • IServer.KIND が CeKind.SERVER に代えて使用されます。 • IUser.KIND が CeKind.USER に代えて使用されます。
com.crystaldecisions.sdk.plugin.CeProgID	各プラグインインターフェイスに、対応するオブジェクトタイプの SI_PROGID プロパティを指定するフィールドが追加されました。以下はその例です。 • IFolder.PROGID が CeProgID.FOLDER に代えて使用されます。 • IServer.PROGID が CeProgID.SERVER に代えて使用されます。 • IUser.PROGID が ProgID.USER に代えて使用されます。

非推奨メソッド

メソッド名	詳細
com.businessobjects.publisher.distribution.IDistributionCompleteContext.getAdminLog	IDistributionCompleteContext.getAdminLogAdapter を使用してください。

メソッド名	詳細
com.businessobjects.publisher.postprocessing.IPublicationPostProcessingContext.getSession	
com.businessobjects.sdk.plugin.desktop.common.IConfiguredService.getServiceStatus	代わりに、IConfiguredService.getServiceStatusEnum を使用します。
com.businessobjects.sdk.plugin.desktop.common.IConfiguredService.setServiceStatus(int)	代わりに、IConfiguredService.setServiceStatusEnum を使用します。
com.businessobjects.sdk.plugin.desktop.profile.IProfileValue.getFormula()	代わりに、IProfileValue.getFormula(String) を使用します。
com.businessobjects.sdk.plugin.desktop.profile.IProfileValue.setFormula(String)	代わりに、IProfileValue.setFormula(String, String) を使用します。
com.businessobjects.sdk.plugin.desktop.publication.IPublicationProfileTarget.setTargetID	代わりに、setSourceDocumentID を使用します。
com.businessobjects.sdk.plugin.desktop.publication.IPublicationProfileTarget.setVariable	代わりに、getVariableMappings を使用します。
com.crystaldecisions.sdk.plugin.desktop.common.IFormatInfo.getSourceDocumentKind	ソースドキュメントの種類を設定する必要はなくなりました。
com.crystaldecisions.sdk.plugin.desktop.common.IFormatInfo.setSourceDocumentKind	ソースドキュメントの種類を設定する必要はなくなりました。
com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.getEventInterface	代わりに、特定のイベントタイプインターフェイスにキャストします。
com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.getEventName	代わりに、IInfoObject.getTitle を使用します。
com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.setEventName	代わりに、IInfoObject.setTitle を使用します。
com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.getEventType	代わりに、IInfoObject.getSpecificKind または IInfoObject.getSpecificProgID を使用します。
com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.setEventType	代わりに、それぞれ特定のタイプのイベントを作成します (FileEvent オブジェクトなど)。
com.crystaldecisions.sdk.uri.PagingQueryOptions.isUriIsUnique	このオプションは使用されません。
com.crystaldecisions.sdk.uri.PagingQueryOptions.setUriIsUnique	このオプションは使用されません。

非推奨フィールド

フィールド名	詳細
`com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo.GroupChoice.PREFERED`	`ISchedulingInfo.GroupChoice.PREFERRED` に置き換えられました。
`com.crystaldecisions.sdk.uri.PagingQueryOptions.SKIP_QUERY_URI_DECODE`	URI クエリーは、自動的に URI のクエリーセクション内のデコードを回避します。

古いフィールド

フィールド名	詳細
`com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID.DOWNLOAD`	これに代わるものはありません。この権限はなくなりました。

2.2.2 監査の変更

> **注記**
>
> このリリースより前のサーバー監査で使用されていたインターフェイスの多くが使用できなくなりました。さらに、サーバー監査とクライアント監査が同じパッケージによって処理されるようになりました。これらは単一のインターフェイスとみなすことができます。このガイドの監査のセクションと *SAP BusinessObjects Business Intelligence Platform Java API* リファレンスをよく読んで、変更による既存のアプリケーションへの影響を確認してください。

このリリースでは、サーバー監査の実行方法に影響する機能強化がいくつか行われています。監査イベントは、サービスベースまたはサーバーベースでは管理されなくなりました。すべてのイベントが集中的に管理および設定されます。

監査イベントおよびその詳細

これらの変更の結果として、`com.crystaldecisions.sdk.plugin.desktop.auditeventinfo` パッケージのクラスとインターフェイスは使用できなくなり、`com.crystaldecisions.sdk.plugin.desktop.auditeventinfo2` パッケージのクラスとインターフェイスに置き換えられました。この新しいパッケージが、すべての監査イベント、関連する詳細情報、およびその編成を制御するようになりました。各サーバーで個々のサービスにアクセスし、そこから監査イベントを切り替える必要はなくなりました。イベントと詳細は、グローバルに設定されるようになりました。

監査レベル

よく使用されるカスタムイベントセットを反映した設定済みの監査レベルを設定できようになりました。新しいカスタム監査レベルがあります。個々のイベントを有効または無効にする場合は、これを設定する必要があります。

古いパッケージも、引き続きシステムに同梱されます。前バージョンのフレームワークで作成されたアプリケーションもコンパイルできますが、期待どおりには動作しません。BusinessObjects Enterprise XI 3.x フレームワー

クで開発されたアプリケーションは、使用しないことをお勧めします。監査アプリケーションは、新しい `com.crystaldecisions.sdk.plugin.desktop.auditeventinfo2` パッケージを使用するように書き直す必要があります。

カスタムアプリケーションのトレーサビリティの強化

Central Management Server（CMS）は、カスタムアプリケーションを識別するために 25 個の CUID 値を予約するようになりました。これらの CUID 値の 1 つを `IEnterpriseLogonInformation.setClientType` メソッドに渡し、`IEnterpriseLogonInformation` オブジェクトを受け取るオーバーロードログオンメソッドの 1 つを使用して、アプリケーションでユーザーを認証できます。システムが収集するすべての監査イベントにも、イベントを発生させたカスタムアプリケーションの名前と CUID が記録されます。この方法は、管理者にとってトレーサビリティの強化に役立つことがあります。アプリケーション名が提供されない場合は、特別な文字列が使用されます。

例

BusinessObjects Enterprise XI 3.x でイベントを有効にするには、特定のサーバーの特定のサービスを隔離してから、サービスのプロパティ バッグを変更する必要がありました。最初の例は、Crystal Reports Job Server の Crystal Reports スケジュールサービスで *Job Scheduling Succeeded* 監査イベント（ID 327681）を有効にする方法を示します。

> **注記**
>
> この操作は BusinessObjects Enterprise XI 3.x フレームワークで実行され、SAP BusinessObjects Business Intelligence プラットフォーム 4.1 インストールでは動作しません。これは、比較の目的で提供されています。

```
void enableEvent31() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession enterpriseSession = sessionManager.logon("username",
"password", "MyCMS:6400", "secEnterprise");
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = '" +
    CeKind.SERVER + "' AND SI_DESCRIPTION = 'Crystal Reports Job Server'";
  IServer server = (IServer) infoStore.query(serverQuery).get(0);
  String serviceQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '" +
          CeKind.SERVICE + "' AND SI_CUID = '" +
CeSecurityCUID.ServerIntelligence.CR_SCHEDULING_SERVICE + "'";
  IService service = (IService) infoStore.query(serviceQuery).get(0);

  IConfiguredServices cfgServices = server.getHostedServices();
  IConfiguredService cfgService = cfgServices.get(service.getID());
  Set enabledAuditEvents = cfgService.getEnabledAuditEvents();

  enabledAuditEvents.add(327681);
  server.save();
}
```

SAP BusinessObjects Business Intelligence プラットフォーム 4.1 でイベントを有効にするには、`IAuditEventInfo` オブジェクトを取得し、有効にされている監査イベントセットを必要に応じて調整するだけ

です。次の例は、システム全体で有効にされている監査イベントセットに *Retrieve* イベント（ID 1013）を追加します。

> ⓘ 注記
>
> この例では、このカスタムアプリケーションを "Custom Application 1" (`CUID = AZ4HLbS0loRMpE2twk442RU`) と識別する新しいログオンメソッドも使用してみます。このセッションの結果としてトリガーされるすべての SAP BusinessObjects Business Intelligence プラットフォーム 4.1 サーバイベントには、イベントの詳細と共にこの名前と CUID が記録されます。これにより、このカスタムアプリケーションでユーザーアクションによってトリガーされたイベントを識別できます。このタスクは、現在の監査レベルが既にカスタムに設定されていることを前提にしています。

```
void enableEvent40() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setClientType("AZ4HLbS0loRMpE2twk442RU");

  IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "MyCMS:6400", "secEnterprise", logonInfo);

  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_ENABLED_AUDIT_EVENTS FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);

  Set enabledAuditEvents = auditEventInfo.getEnabledEventTypeIDs();
  enabledAuditEvents.add(1013);
  infoStore.commit(infoObjects);
}
```

どちらの例も、`ISessionMgr` インターフェイスを使用して `IInfoStore` オブジェクトを取得し、操作する項目をオブジェクトに照会します。SAP BusinessObjects Business Intelligence プラットフォーム 4.1 では、公開される監査インタフェースが増えたため、1 つのクエリを実行するだけで済みます。したがって、同じ操作の実行に必要なメソッド呼び出しの回数が減ります。

関連情報

監査 [259 ページ]
現在の監査レベルを変更する [263 ページ]
トレーサビリティを強化してカスタムアプリケーション全体を監査する [277 ページ]
サーバー管理 [210 ページ]

3 SDK 基本

このセクションでは、主なクラスとインターフェイスの概要、よく使用されるワークフロー、Central Management Server（CMS）内のオブジェクトの基本操作など、この SDK の基礎について説明します。

3.1 クラスの概要

SAP BusinessObjects Business Intelligence プラットフォーム 4.1 SDK は、Central Management Server（CMS）に格納されているオブジェクトを管理します。CMS と通信および連携するための 3 つの基本インターフェイスがあります。

インターフェイス	パッケージ
`IInfoObject`	`com.crystaldecisions.sdk.occa.infostore`
`IInfoObjects`	`com.crystaldecisions.sdk.occa.infostore`
`IInfoStore`	`com.crystaldecisions.sdk.occa.infostore`

`IInfoObject` インターフェイスと `IInfoObjects` インターフェイスを使用すると、InfoObject と呼ばれる CMS オブジェクトを操作できます。`IInfoStore` インターフェイスは、InfoObject の要求や変更を CMS に伝達するために使用されます。これらのインターフェイスが CMS とどのように対話するかを理解することが SDK を使用する鍵になります。

InfoObjects

InfoObject は、CMS 内の 1 つの情報エンティティーを表す汎用モデルです。Crystal レポート、Web Intelligence ドキュメント、フォルダ、ユーザー、サーバーなど、SDK によって管理可能なシステム内のすべての項目は、それぞれ InfoObject で表されます。

各 InfoObject は、プロパティと呼ばれる多数のフィールドを持つ CMS データベース内の 1 つのレコードです。プロパティの例には、InfoObject の識別子、名前、タイプがあります。システムの状態や動作は、SDK で InfoObject のプロパティにアクセスして変更することで制御します。

個々の InfoObject を追加、取得、更新、またはスケジュールするには、`IInfoObjects` コレクションを通して `IInfostore` インターフェイスから行う必要があります。`IInfoObject` インスタンスを直接 CMS から取得したり、CMS にコミットすることはできません。

InfoStore サービス

IInfoStore インターフェイスは、CMS 内の InfoObject のコントローラまたはゲートウェイとして機能します。IInfoStore インターフェイスは、アプリケーションと CMS の間でメッセージを伝達して、CMS リポジトリに格納される IInfoObject のすべてのインスタンスを (IInfoObjects コレクションを通して) 作成、取得、およびコミットするために使用されます。

InfoStore サービスへの接続を作成するには、有効なセッション (com.crystaldecisions.sdk.framework.IEnterpriseSession) にログオンする必要があります。次の例では、変数 enterpriseSession を有効なセッションであるとしています。

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

InfoStore サービスへの接続が確立されたら、query、commit、schedule などの IInfoStore インターフェイスのメソッドを使用して、CMS と通信できます。

プラグイン

さまざまなタイプの InfoObject を区別して効率よく対話するために、システムはプラグインを使用します。CMS 内の InfoObject のタイプごとに 1 つのプラグインがあり、これらのプラグインが各タイプの共通プロパティを定義します。さまざまなタイプの InfoObject の動作を定義するさまざまなプラグインがあります。

SDK は、適切なプラグインインターフェイスを通して、特定のタイプの InfoObject と対話するためのメソッドを公開しています。たとえば、インターフェイス com.crystaldecisions.sdk.plugin.desktop.user.IServer は、クラスタ内にデプロイされたサーバーと対話するためのメソッドを公開します。IInfoObjects コレクション内の個々のオブジェクトを適切なプラグインインターフェイスにキャストします。この例では、変数 infoobjects をサーバーのコレクションであるとして、get メソッドを使用して最初のサーバーを返します。

```
IServer server = (IServer) infoobjects.get(0);
```

すべてのプラグインインターフェイスと同様に IServer も、InfoObject の共通メソッドが公開される汎用 IInfoObject インターフェイスを拡張します。

以下の表では、この SDK で公開されている代表的なプラグインを紹介します。

認証プラグイン

パッケージ	説明
com.crystaldecisions.sdk.plugin.authentication.enterprise	ネイティブ BI platform ユーザーのセキュリティオプションを定義するプラグインインターフェイス IsecEnterprise が含まれます。
com.crystaldecisions.sdk.plugin.authentication.ldap	セキュリティオプションを設定し、LDAP ディレクトリサーバーのユーザーを BI platform ユーザーにマップするためのプラグインインターフェイス IsecLDAP が含まれます。

パッケージ	説明
`com.crystaldecisions.sdk.plugin.authentication.secwinad`	セキュリティオプションを設定し、Windows Active Directory サーバーのユーザーを BI platform ユーザーにマップするためのプラグインインターフェイス `IsecWinAD` が含まれます。

出力先プラグイン

パッケージ	説明
`com.crystaldecisions.sdk.plugin.destination.diskunmanaged`	ローカルまたはネットワーク ファイル システムにスケジュールされるオブジェクトの出力先オプションを定義するためのプラグインインターフェイス `IDiskUnmanaged` が含まれます。
`com.crystaldecisions.sdk.plugin.destination.ftp`	FTP サーバーにスケジュールされるオブジェクトの出力先オプションを定義するためのプラグインインターフェイス `IFTP` が含まれます。
`com.crystaldecisions.sdk.plugin.authentication.managed`	BI platform 受信ボックスにスケジュールされるオブジェクトの出力先オプションを定義するためのプラグインインターフェイス `IManaged` が含まれます。
`com.crystaldecisions.sdk.plugin.destination.smtp`	SMTP サーバーを通してリレーされる電子メールアドレスにスケジュールされるオブジェクトの出力先オプションを定義するためのプラグインインターフェイス `ISMTP` が含まれます。

デスクトッププラグイン

デスクトッププラグインは、極めて幅広い種類の InfoObject を表します。次の表では、SDK に含まれるさまざまなデスクトッププラグインの例を示します。パッケージの詳細なリストについては、*SAP BusinessObjects Business Intelligence* プラットフォーム *4.1 Java API* リファレンスを参照してください。

型	パッケージの例	説明
ドキュメント InfoObject	com.businessobjects.sdk.plugin.desktop.flash com.businessobjects.sdk.plugin.desktop.fullclient com.businessobjects.sdk.plugin.desktop.webi com.businessobjects.sdk.plugin.desktop.xcelsius com.crystaldecisions.sdk.plugin.desktop.excel com.crystaldecisions.sdk.plugin.desktop.pdf com.crystaldecisions.sdk.plugin.desktop.powerpoint com.crystaldecisions.sdk.plugin.desktop.program com.crystaldecisions.sdk.plugin.desktop.report com.crystaldecisions.sdk.plugin.desktop.rtf com.crystaldecisions.sdk.plugin.desktop.txt com.crystaldecisions.sdk.plugin.desktop.word	これらのパッケージには、Crystal レポート、Web Intelligence ドキュメント、PDF などのドキュメントと対話するためのプラグインインターフェイスが含まれます。CMS 内の InfoObject として表されているマネージドドキュメントは、File Repository Server（FRS）上に物理的に存在します。
システム InfoObject	com.businessobjects.sdk.plugin.desktop.publication com.businessobjects.sdk.plugin.desktop.profile com.crystaldecisions.sdk.plugin.desktop.server com.crystaldecisions.sdk.plugin.desktop.user	これらのパッケージには、ユーザー、サーバー、パブリケーションなどのシステムオブジェクトと対話するためのプラグインインターフェイスが含まれます。ドキュメントの InfoObject が FRS 上のファイルを表すのとは異なり、システムオブジェクトは CMS に内蔵されます。

型	パッケージの例	説明

組織 InfoObject	`com.businessobjects.sdk.plugin.desktop.category`
	`com.businessobjects.sdk.plugin.desktop.inbox`
	`com.crystaldecisions.sdk.plugin.desktop.folder`
	`com.crystaldecisions.sdk.plugin.desktop.objectpackage`
	`com.crystaldecisions.sdk.plugin.desktop.servergroup`
	`com.crystaldecisions.sdk.plugin.desktop.usergroup`
		これらのパッケージには、個々のシステム InfoObject やドキュメント InfoObject を論理的にグループ化して整理するためのプラグインインターフェイスが含まれます。たとえば、ドキュメントをグループ化するためのフォルダや、ユーザーを組織化するためのユーザーグループがあります。

3.2 共通のワークフロー

この SDK を使用して BI platform を変更するには、Central Management Server（CMS）で InfoObject とそれらのプロパティを操作します。主要なすべてのタスクに共通する次のワークフローがあります。

- 新しい InfoObject の作成
- InfoObject の削除
- InfoObject の変更
- InfoObject のスケジュール

これらの共通ワークフローは、サーバーを再起動したり、受信ボックスにレポートインスタンスを配信したり、フォルダに対するセキュリティ権限をユーザーに適用する際に使用します。

このセクションでは、InfoObject を作成、削除、および変更する方法の例を示します。InfoObject のスケジュールについては、*SDK の使用*のスケジュールセクションを参照してください。

関連情報

スケジューリング [142 ページ]

3.2.1 InfoObject を作成する

新しいユーザーまたはフォルダの作成などのタスクでは、Central Management Server（CMS）データベースに新しい InfoObject レコードを作成する必要があります。`IInfoStore.newInfoObjectCollection` ファクト

リメソッドを使用すると、新しいオブジェクトを入れる新しい IInfoObjects コレクションを要求できます。この手順の中で、CMS は、コミットされるコレクション内の新しいオブジェクトにそれぞれ一意の ID を予約します。

この例では、CMS にフォルダ InfoObject を作成します。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. IInfoStore インターフェイスの newInfoObjectCollection メソッドを呼び出して、IInfoObjects コレクションを作成します。

   ```
   IInfoObjects newFolders = infostore.newInfoObjectCollection();
   ```

3. コレクションの add メソッドを呼び出して InfoObject を作成し、結果を適切なプラグインインターフェイスにキャストします。

   add メソッドは、作成する InfoObject の種類を表す IFolder.FOLDER_KIND フィールド値を使用します。この例では、フォルダを作成し、結果を IFolder プラグインインターフェイスにキャストします。

   ```
   IFolder newFolder = (IFolder) newFolders.add(IFolder.FOLDER_KIND);
   ```

4. 名前、説明、親フォルダ情報など、新しい InfoObject のさまざまなプロパティを設定します。

   この例では、親 ID を 0 に設定します。これは、CMS のルートフォルダの最上位フォルダを示します。

   ```
   newFolder.setTitle(newInfoObjectName);
   newFolder.setDescription("This is a new folder InfoObject.");
   newFolder.setParentID(0);
   ```

5. 新しい InfoObject を含む新しい IInfoObjects コレクションを CMS にコミットして、変更を保存します。

   ```
   infostore.commit(newFolders);
   ```

例

この例では、CMS にフォルダ InfoObject を作成します。

```
void createInfoObject(IEnterpriseSession enterpriseSession, String
newInfoObjectName) throws SDKException
{
    IInfoStore infostore = (IInfoStore)
enterpriseSession.getService("InfoStore");
    IInfoObjects newFolders = infostore.newInfoObjectCollection();
    IFolder newFolder = (IFolder) newFolders.add(IFolder.FOLDER_KIND);

    newFolder.setTitle(newInfoObjectName);
    newFolder.setDescription("This is a new folder InfoObject.");
    newFolder.setParentID(0);

    infostore.commit(newFolders);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

関連情報

InfoObject を削除する [21 ページ]
InfoObject を変更する [22 ページ]
クラスの概要 [15 ページ]

3.2.2 InfoObject を削除する

この例では、CMS から既存のフォルダを削除します。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. 削除する InfoObject を含む `IInfoObjects` コレクションを CMS に照会します。

```
String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
    + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
    + "And SI_NAME='" + infoObjectName + "' And SI_INSTANCE=0";
IInfoObjects folders = infostore.query(query);
```

この例では、一般に使用される SQL（Structured Query Language）のサブセットであるクエリー言語を使用して、CMS に InfoObject を要求します。正しいオブジェクトを照会し、それらを CMS から取得することは、SDK をうまく使用するための基本です。クエリー言語の詳細については、*SDK の基本のクエリー言語を使用したオブジェクトの取得を参照してください。*

3. 削除する InfoObject をコレクションから取得します。

```
IInfoObject folder = (IInfoObject) folders.get(0);
```

> ⓘ 注記
>
> 照会によって複数の InfoObject が返されることもあります。この例では、コレクションのゼロ番目のインデックスにある最初の InfoObject を取得します。

4. `IInfoObjects` コレクションの `delete` メソッドを呼び出します。

   削除する InfoObject を引数として `delete` メソッドに渡します。

```
folders.delete(folder);
```

`delete` メソッドは、コレクションからオブジェクトを削除しません。代わりに、CMS へのコミット時に削除するためのフラグをオブジェクトに付けます。

5. 変更した `IInfoObjects` コレクションを CMS にコミットして、変更を保存します。

```
infostore.commit(folders);
```

例

この例では、CMS から既存のフォルダを削除します。

```
void deleteInfoObject(IEnterpriseSession enterpriseSession, String
infoObjectName) throws SDKException
{
    IInfoStore infostore = (IInfoStore)
enterpriseSession.getService("InfoStore");
    String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
        + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
        + "And SI_NAME='" + infoObjectName + "' And SI_INSTANCE=0";
    IInfoObjects folders = infostore.query(query);
    IInfoObject folder = (IInfoObject) folders.get(0);
    folders.delete(folder);
    infostore.commit(folders);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

関連情報

InfoObject を作成する [19 ページ]
InfoObject を変更する [22 ページ]
クラスの概要 [15 ページ]
クエリー言語を使用したオブジェクトの取得 [24 ページ]

3.2.3 InfoObject を変更する

ほとんどのタスクでは、InfoObject のプロパティを変更します。プロパティ値を変更してから InfoObject をコミットすることで、BI platform インストールの状態に影響を及ぼすことができます。

この例では、Central Management Server（CMS）内にあるフォルダの名前を変更します。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. 変更する InfoObject を含む `IInfoObjects` コレクションを CMS に照会します。

```
String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
        + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
        + "And SI_NAME='" + oldName + "' And SI_INSTANCE=0";
    IInfoObjects folders = infostore.query(query);
```

この例では、一般に使用される SQL（Structured Query Language）のサブセットであるクエリー言語を使用して、CMS に InfoObject を要求します。正しいオブジェクトを照会し、それらを CMS から取得することは、SDK をうまく使用するための基本です。クエリー言語の詳細については、*SDK の基本のクエリー言語を使用したオブジェクトの取得を参照してください。*

3. 変更する InfoObject をコレクションから取得します。

```
IInfoObject folder = (IInfoObject) folders.get(0);
```

> ⓘ 注記
>
> 照会によって複数の InfoObject が返されることもあります。この例では、コレクションのゼロ番目のインデックスにある最初の InfoObject を取得します。

4. `setTitle` メソッドを呼び出して、この InfoObject の名前を変更します。

> ⓘ 注記
>
> `setTitle` メソッドは、`IInfoObject` 基本インターフェイスを通して公開されています。フォルダオブジェクトタイプ固有のプロパティを変更するには、この基本インターフェイスを使用するのではなく、結果を `IFolder` プラグインインターフェイスにキャストします。

```
folder.setTitle(newName);
```

> ⓘ 注記
>
> InfoObject を CMS にコミットする前に、少なくとも 1 つのプロパティ値を変更する必要があります。CMS は、変更されていない InfoObject のコミットを許可しません。

5. 変更した InfoObject が含まれる `IInfoObjects` コレクションを CMS にコミットして、変更を保存します。

```
infostore.commit(folders);
```

例

この例では、CMS 内にあるフォルダの名前を変更します。

```
void modifyInfoObject(IEnterpriseSession enterpriseSession, String oldName,
String newName) throws SDKException
{
    IInfoStore infostore = (IInfoStore)
enterpriseSession.getService("InfoStore");
    String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
```

```
        + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
        + "And SI_NAME='" + oldName + "' And SI_INSTANCE=0";
    IInfoObjects folders = infostore.query(query);

    IInfoObject folder = (IInfoObject) folders.get(0);
    folder.setTitle(newName);
    infostore.commit(folders);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

関連情報

InfoObject を作成する [19 ページ]
InfoObject を削除する [21 ページ]
クラスの概要 [15 ページ]
クエリー言語を使用したオブジェクトの取得 [24 ページ]

3.3 クエリー言語を使用したオブジェクトの取得

`IInfoStore.query` メソッドとクエリー言語文字列を使用して、Central Management Server（CMS）に保存されているオブジェクトのコレクションを検索および取得します。たとえば、BI platform インストールの root レベルのフォルダを取得するには、`IInfoStore` クラスのインスタンスを取得し、次のクエリー文字列をその `query` メソッドに渡すことができます。

```
SELECT
    SI_ID, SI_NAME, SI_DESCRIPTION
FROM
    CI_SYSTEMOBJECTS
WHERE
    SI_KIND = 'Folder' AND
    SI_PARENTID = 4
```

`query` メソッドは、リポジトリからオブジェクトを取得し、コードからアクセスできるオブジェクト内に格納します。その後、プログラムでこれらのオブジェクトのプロパティを変更し、変更を保存してリポジトリに戻すことができます。

クエリー言語は、リレーショナルデータベースからデータを取得するために広く使用されている SQL のサブセットです。クエリー言語は、SQL と同じ機能を多く備えていますが、テーブル結合やネストされた `SELECT` 文などの高度な機能はサポートしていません。

このセクションでは、クエリー言語を使用して CMS リポジトリからオブジェクトを取得する方法について説明します。クエリーで最もよく使用される要素や、SDK クラスがクエリー言語とやり取りする方法について説明します。

関連情報

クエリーを実行するには、次の手順を実行します。 [25 ページ]
URI クエリーを使用したオブジェクトの取得 [47 ページ]

3.3.1 クエリーを実行するには、次の手順を実行します。

Central Management Server (CMS) に対してクエリーを実行するには、有効な `IEnterpriseSession` オブジェクトが必要です。

CMS を照会する際は、次のインターフェイスが使用されます。

- `IInfoStore`：リポジトリからオブジェクトをクエリーおよび取得するためのメソッドを提供します。
- `IInfoObjects`：クエリー結果が含まれます。
- `IInfoObject`：クエリー結果セットの個々のアイテムが含まれます。

これらのクラスは、`com.crystaldecisions.sdk.occa.infostore` パッケージに含まれます。

1. `IEntepriseSession` オブジェクトの `getService` ファクトリメソッドを呼び出し、引数を **`InfoStore`** に設定します。返されるオブジェクトを `IInfoStore` にキャストします。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. クエリー文字列を作成します。

   クエリー文字列は、次の各部で構成されます。

   - `SELECT` 句。結果セット内の各オブジェクトに含まれるオブジェクトプロパティを指定します。
   - `FROM` 句。取得するオブジェクトが含まれるリポジトリ内のテーブルを指定します。
   - （オプション）`WHERE` 句。フィルタを適用してより詳細な結果を取得できます。
   - （オプション）`ORDER BY` 句。返される結果の順序を指定できます。

   次のコードは、リポジトリ内のすべての Crystal Report オブジェクトを取得するためのクエリー文字列を作成します。

   ```
   String queryString = "SELECT SI_NAME, SI_DESCRIPTION "
           + "FROM CI_INFOOBJECTS WHERE SI_KIND = 'CrystalReport'";
   ```

   > ⓘ 注記
   >
   > クエリー言語のキーワード、プロパティ名、およびテーブル名は、大文字と小文字が区別されません。たとえば、`SELECT` と `select` は同じです。

3. `IInfoStore` オブジェクトの `query` メソッドにクエリー文字列を渡します。

   ```
   IInfoObjects infoObjects = infoStore.query(queryString);
   ```

query メソッドは、クエリーの結果が入っている IInfoObjects コレクションを返します。

4. IInfoObjects コレクションの iterator メソッドを使用して、反復子オブジェクトを取得します。

   反復子を使用して、コレクションから個々のオブジェクトを取得できます。反復子から返されるオブジェクトを IInfoObject にキャストします。

```
Iterator infoObjectsIter = infoObjects.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject folder = (IInfoObject) infoObjectsIter.next();
    ...
}
```

例: 例

次の例は、リポジトリからすべての Crystal Report オブジェクトを取得し、その名前を出力します。<enterpriseSession> は有効な IEnterpriseSession オブジェクトであることとします。

```
import com.crystaldecisions.sdk.framework.IEnterpriseSession;
import com.crystaldecisions.sdk.occa.infostore.IInfoStore;
import com.crystaldecisions.sdk.occa.infostore.IInfoObjects;
import com.crystaldecisions.sdk.occa.infostore.IInfoObject;
import java.util.Iterator;
...
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
String queryString = "SELECT SI_NAME, SI_DESCRIPTION "
        + "FROM CI_INFOOBJECTS WHERE SI_KIND = 'CrystalReport'";
IInfoObjects infoObjects = infoStore.query(queryString);
Iterator infoObjectsIter = results.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject folder = (IInfoObject) infoObjectsIter.next();
    System.out.println(folder.getTitle());
}
```

関連情報

オブジェクトタイプ [26 ページ]
クエリー結果内のプロパティへのアクセス [29 ページ]

3.3.2 オブジェクトタイプ

次の表に、CMS リポジトリの一般的なオブジェクトタイプをリストします。この表を使用して、各オブジェクトタイプの Java インターフェイス名とパッケージを確認し、クエリーで使用する SI_KIND 値とリポジトリテーブル名を決定してください。

> **注記**
>
> すべての SI_KIND 定数フィールド値の総合リストについては、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスで各プラグインインターフェイスを参照してください。たとえば、

> com.crystaldecisions.sdk.plugin.desktop.server.IServer インターフェイスには、IServer.KIND フィールドがあります。

クラス	SI_KIND 識別子	リポジトリテーブル	パッケージ
ICalendar	Calendar	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.calendar
IConnection	Connection	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.connection
ICustomRole	CustomRole	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.customrole
IDiskUnmanaged	DiskUnmanaged	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.diskunmanaged
IEvent	Event	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.event
IExcel	Excel	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.excel
IFolder	Folder	CI_INFOOBJECTS CI_SYSTEMOBJECTS CI_APPOBJECTS	com.crystaldecisions.sdk.plugin.desktop.folder
IFTP	FTP	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.ftp
IFullClient	FullClient	CI_INFOOBJECTS	com.businessobjects.sdk.plugin.desktop.fullclient
IHyperlink	Hyperlink	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.hyperlink
ILicenseKey	LicenseKey	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.licensekey
IManaged	Managed	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.managed

クラス	SI_KIND 識別子	リポジトリテーブル	パッケージ
IObjectPackage	ObjectPackage	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.objectpackage
IPDF	Pdf	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.pdf
IPowerPoint	PowerPoint	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.powerpoint
IProfile	Profile	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.profile
IProgram	Program	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.program
IPublication	Publication	CI_INFOOBJECTS	com.businessobjects.sdk.plugin.desktop.publication
IReport	CrystalReport	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.report
IRTF	Rtf	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.rtf
IsecEnterprise	secEnterprise	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.authentication.enterprise
IsecLDAP	secLDAP	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.authentication.ldap
IsecWinAD	secWinAD	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.authentication.secwinad
IServer	Server	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.server
IServerGroup	ServerGroup	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.servergroup
IService	Service	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.service

クラス	SI_KIND 識別子	リポジトリテーブル	パッケージ
IServiceContainer	ServiceContainer	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.servicecontainer
IShortcut	Shortcut	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.shortcut
ISMTP	Smtp	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.smtp
ITxt	Txt	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.txt
IUniverse	Universe	CI_APPOBJECTS	com.businessobjects.sdk.plugin.desktop.universe
IUser	User	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.user
IUserGroup	UserGroup	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.usergroup
IWebi	Webi	CI_INFOOBJECTS	com.businessobjects.sdk.plugin.desktop.webi
IWord	Word	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.word

3.3.3 クエリー結果内のプロパティへのアクセス

クエリー結果として返されるオブジェクトのプロパティにアクセスする方法は、2 とおりあります。

- オブジェクトインターフェイスのアクセッサ（get と set）メソッドを使用する。
- オブジェクトインターフェイスのプロパティ バッグを使用する。

アクセッサメソッドの使用

オブジェクトインターフェイスには、特定のオブジェクトプロパティの値の取得に使用できるアクセッサメソッドがあります。

たとえば、SELECT 句に SI_NAME プロパティが含まれている場合は、IInfoObject の getTitle メソッドを使用してプロパティを取得します。

> ⓘ 注記
>
> アクセッサメソッドを呼び出してプロパティ値を取得するときに、対応するプロパティが SELECT 句に含まれていない場合は、例外がスローされます。

次のコードは、リポジトリから指定された条件に一致するすべてのオブジェクトを取得します。クエリーの SELECT 句には、SI_NAME と SI_DESCRIPTION が含まれています。これは、それらのプロパティを結果に入れることを指定します。結果の各オブジェクトに対して、IInfoObject の getTitle および getDescription メソッドを使用してそれぞれのプロパティの値を取得します。

```
String queryString = "SELECT SI_NAME, SI_DESCRIPTION FROM CI_INFOOBJECTS "
        + "WHERE SI_NAME LIKE 'My Report%'";
IInfoObjects infoObjects = infoStore.query(queryString);
Iterator infoObjectsIter = infoObjects.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject infoObject = (IInfoObject) infoObjectsIter.next();
    out.println(infoObject.getTitle() + ": " + infoObject.getDescription() +
"<br/>");
}
```

IInfoObject インターフェイスは、CMS リポジトリ内のすべてのオブジェクトに対して使用できる基本インターフェイスです。これには、すべてのオブジェクト型に共通のプロパティに対して使用できるアクセッサメソッドだけが含まれます。オブジェクト型固有のアクセッサメソッドにアクセスするには、そのオブジェクトを適切なインターフェイスにキャストする必要があります。たとえば、フォルダオブジェクトがあり、フォルダ関連のアクセッサメソッドを使用する場合は、そのオブジェクトを IFolder にキャストします。

プロパティ バッグの使用

オブジェクトインターフェイスにはプロパティ バッグもあります。プロパティ バッグには、そのオブジェクトについて取得されたすべてのプロパティが入っています。

IInfoObject の properties メソッドを使用して、オブジェクトのプロパティ バッグを取得します。次に、IProperties の getProperty メソッドを使用して、プロパティ バッグから値を取得します。getProperty は、指定された名前のプロパティの値を含む IProperty オブジェクトを返します。

> ⓘ 注記
>
> getProperty を使用してプロパティ バッグからプロパティを取得するときは、リテラルのプロパティ名ではなく、CePropertyID に定義されている適切な定数を使用します。たとえば、リテラル文字列 SI_PARENTID ではなく、CePropertyID.SI_PARENTID を使用します。

プロパティの値を設定したり、プロパティ バッグにプロパティを追加するには、IProperties の setProperty メソッドを使用します。

オブジェクトインターフェイスに目的のプロパティのアクセッサメソッドがない場合にのみ、プロパティ バッグを使用します。たとえば、SI_CREATION_TIME プロパティには、プロパティ バッグを通じてのみアクセスできます。次のコードは、SI_CREATION_TIME プロパティの値を取得します。

```
String queryString = "SELECT SI_NAME, SI_CREATION_TIME FROM CI_INFOOBJECTS "
        + "WHERE SI_KIND='CrystalReport' AND SI_NAME = 'Accessibility'";
```

```
IInfoObjects infoObjects = infoStore.query(queryString);
IInfoObject infoObject = (IInfoObject) infoObjects.get(0);
IProperties properties = infoObject.properties();
IProperty creationTimeProperty =
properties.getProperty(CePropertyID.SI_CREATION_TIME);
String value = (String) creationTimeProperty.getValue();
```

処理情報とスケジュール情報のプロパティ

オブジェクトにスケジュール情報が入っている場合、SI_PROCESSINFO および SI_SCHEDULEINFO プロパティを使用して、そのオブジェクトの処理情報とスケジュール情報にアクセスします。

> **注記**
>
> オブジェクトがこれまでスケジュールされたことがなく、スケジュール情報で設定されたことがない場合、SI_PROCESSINFO および SI_SCHEDULEINFO プロパティは未定義になります。オブジェクトがスケジュールをサポートしているかどうかを判別する場合は、SI_IS_SCHEDULABLE プロパティを使用します。

他のプロパティとは異なり、SI_PROCESSINFO および SI_SCHEDULEINFO には、クエリーで個別に取得できるネストしたプロパティ（サブプロパティ）が含まれます。

クエリーでサブプロパティを指定する形式は次のとおりです。

```
<propertyName>.<subPropertyName>
```

ここで、<propertyName> は SI_PROCESSINFO または SI_SCHEDULEINFO のどちらか、<subPropertyName> はサブプロパティの名前です。

たとえば、次のクエリーは、処理プロパティとスケジュールプロパティから指定された値だけを取得します。

```
SELECT SI_NAME, SI_PROCESSINFO.SI_DBNEEDLOGON,
SI_SCHEDULEINFO.SI_SUBMITTER FROM CI_INFOOBJECTS
WHERE SI_NAME LIKE 'Weekly Reports%';
```

> **注記**
>
> SI_PROCESSINFO プロパティ、SI_SCHEDULEINFO プロパティ、およびこれらのサブプロパティは、WHERE 句でサポートされていません。

IInfoObject クラスには、処理情報とスケジュール情報の取得に使用される 2 つのアクセッサメソッド getProcessingInfo と getSchedulingInfo があります。

- getProcessingInfo は、処理情報を含む IProcessingInfo オブジェクトを返します。IProcessingInfo のプロパティにアクセスするには、properties メソッドを使用して取得できるプロパティ バッグを使用します。
- getSchedulingInfo は、スケジュール情報を含む ISchedulingInfo オブジェクトを返します。ISchedulingInfo のプロパティにアクセスするには、アクセッサメソッドを使用します。

関連情報

オブジェクトタイプ [26 ページ]

3.3.4 SELECT 句

SELECT 句には、クエリーから返されるオブジェクトに含めるプロパティの一覧が含まれます。

結果に含まれる各 IInfoObject には、そのオブジェクトに対するリポジトリ内の値に対応するプロパティセットが含まれます。たとえば、SELECT 句に SI_GUID プロパティと SI_DESCRIPTION プロパティを含めた場合、クエリーから返されるオブジェクトには、SI_GUID プロパティと SI_DESCRIPTION プロパティが含まれます。

```
SELECT SI_ID, SI_NAME, SI_GUID
```

次の表では、SELECT 句で最もよく使用されるプロパティについて説明し、それらのプロパティを取得するために使用できる IInfoObject のメソッドを示します。

プロパティ名	アクセッサメソッド	説明
SI_ID	IInfoObject.getID()	オブジェクトの一意の識別子を含む整数。これはプライマリキーではないことに注意してください。オブジェクトがリポジトリから削除された場合は、後でこの識別子を別のオブジェクトに割り当てることができます。
SI_NAME	IInfoObject.getTitle()	オブジェクトの名前を含む文字列。
SI_DESCRIPTION	IInfoObject.getDescription()	オブジェクトの説明を含む文字列。

SELECT 句には、ほかにも多くのプロパティを含めることができます。いくつかのプロパティは、任意のタイプのオブジェクトに適用できます。その他の多くのプロパティは、取得するオブジェクトのタイプに固有です。

> **注記**
>
> SELECT 句のプロパティ名が間違っている場合、または一致するオブジェクトでそのプロパティが定義されていない場合、クエリーパーサーはこのプロパティを無視します。SELECT 句内のすべてのプロパティ名が有効でない場合、クエリーは空の結果セットを返します。

オブジェクトのすべてのプロパティを選択する場合は、SELECT 句で個々のプロパティ名を使用するのではなく、* を使用します。たとえば、次のクエリーは、指定された範囲内の ID を持つフォルダオブジェクトのすべてのプロパティを返します。

```
SELECT * FROM CI_INFOOBJECTS WHERE SI_KIND='Folder' AND SI_ID BETWEEN 1000 AND
1200
```

> **注記**
>
> 大量の行が返されるクエリーでは、* を使用しないでください。パフォーマンスが大幅に低下する可能性があります。

3.3.4.1 関数

3.3.4.1.1 COUNT 関数

プロパティの固有の値の数を取得します。SELECT 句で COUNT 関数を使用すると、結果セットには InfoObject インスタンスが 1 つだけ含まれます。COUNT 関数の結果は、このオブジェクト内で、SI_AGGREGATE_COUNT という名前のプロパティ バッグの中に格納されます。プロパティ バッグ内の項目の名前は、値をカウントしたプロパティの名前です。

たとえば次のクエリーは、CI_INFOOBJECTS テーブル内のオブジェクトのうち、SI_ID プロパティの中に値を持っているオブジェクトの数を取得します。(すべてのオブジェクトは SI_ID プロパティ値を持っているため、このクエリーはテーブル内のオブジェクトの合計数を取得します。)

```
SELECT COUNT(SI_ID) FROM CI_INFOOBJECTS
```

次のコードは、クエリーを実行してカウントの値を取得します。

```
String queryString = "SELECT COUNT(SI_ID) FROM CI_INFOOBJECTS";
IInfoObjects infoObjects = infoStore.query(queryString);
IInfoObject infoObject = (IInfoObject) infoObjects.get(0);
IProperties counts = infoObject.properties().getProperties("SI_AGGREGATE_COUNT");
int idCount = (int)counts.getInt("SI_ID");
```

1 つのクエリーで複数のカウントを取得できます。たとえば、次のクエリーは SI_ID 値の数と SI_DESCRIPTION 値の数を取得します。SI_AGGREGATE_COUNT プロパティ バッグには、クエリー内の COUNT 関数で指定された各プロパティごとに 1 つのプロパティが入っています。

```
SELECT COUNT(SI_ID), COUNT(SI_DESCRIPTION) FROM CI_INFOOBJECTS
```

この場合、SI_AGGREGATE_COUNT プロパティ バッグには、SI_ID 項目が 1 つ、SI_DESCRIPTION 項目が 1 つ入っています。

COUNT 関数と TOP 関数は、1 つのクエリーで同時に使用しないでください。

COUNT 関数では、SI_SCHEDULEINFO プロパティと SI_PROCESSINFO プロパティの使用がサポートされていません。

> **注記**
>
> COUNT 関数は、クエリー条件に一致する行の数を返さない場合があります。一致する行の中に、指定されたプロパティが定義されていない行がある場合、それらの行はカウントに含まれません。条件に一致する行の数を返すには、COUNT(SI_ID) を使用します。SI_ID プロパティは、リポジトリ内のすべてのオブジェクトで定義されています。

3.3.4.1.2 TOP 関数

クエリーから返されるオブジェクトの最大数を指定します。たとえば、次のクエリーは、CI_INFOOBJECTS テーブルから SI_ID プロパティを持つ最初の 20 オブジェクトを検索します。

```
SELECT TOP 20 * FROM CI_INFOOBJECTS
```

3.3.5 FROM 句

FROM 句は、CMS リポジトリから取得するオブジェクトを含むテーブルを指定します。

FROM 句では、単一のテーブルまたはテーブルの一覧を指定できます。たとえば、単一のテーブルは次のように指定します。

```
FROM CI_INFOOBJECTS
```

複数のテーブルを同時に検索するには、テーブル名をカンマで区切ってリストします。たとえば、次のクエリーは、CI_INFOOBJECTS テーブルと CI_SYSTEMOBJECTS テーブルを検索します。

```
SELECT SI_NAME, SI_ID
FROM CI_INFOOBJECTS, CI_SYSTEMOBJECTS
WHERE SI_ID =123 AND SI_CREATION_TIME='2000/03/31'
```

次の表は、FROM 句で使用できるテーブルを示します。

テーブル	説明
CI_INFOOBJECTS	お気に入りフォルダやレポートなど、ユーザーデスクトップを構築するために一般に使用されるオブジェクトが含まれます。
CI_SYSTEMOBJECTS	サーバー、接続、ユーザー、ユーザーグループなど、管理デスクトップと内部システムオブジェクトを構築するために使用されるオブジェクトが含まれます。
CI_APPOBJECTS	BI platform アプリケーションを表すオブジェクトが含まれます。たとえば、このテーブルには、BI 起動パッドを表すオブジェクトが格納されています。

3.3.6 WHERE 句

オプションの WHERE 句で条件（検索条件）を指定して、クエリー結果に返される項目の数を制限できます。条件を使用すると、関連する項目の数が減るため、クエリー実行時間が削減され、結果セットを処理しやすくなります。

> ⓘ 注記
>
> SI_SCHEDULEINFO および SI_PROCESSINFO プロパティは、WHERE 句でサポートされていません。

WHERE 句の条件は、通常、プロパティ名、演算子、および値で構成されます。以下はその例です。

```
SI_ID = 9287
```

値には、別のプロパティ名またはリテラル値のどちらかを指定できます。

次の WHERE 句には、すべての CrystalReport オブジェクトを取得する条件が含まれます。

```
WHERE SI_KIND='CrystalReport'
```

> ⓘ 注記
>
> SI_PROGID プロパティは使用できなくなりました。代わりに SI_KIND プロパティを使用してください。

AND キーワードと OR キーワードを使用すると、1 つの WHERE 句で複数の条件を指定できます。次の例は、AND を使用して 2 つの条件を結合することで、1 つのスケジュールプロセスから作成され、同じ親を持つすべてのオブジェクトを取得します。

```
WHERE SI_PARENTID=231 AND SI_INSTANCE=1
```

関連情報

条件 [36 ページ]

3.3.6.1 WHERE 句で一般に使用されるプロパティ

次の例は、WHERE 句の最も一般的な使用例です。

- クラス型 IReport のすべての info オブジェクトを取得する

  ```
  "WHERE SI_KIND='CrystalReport'"
  ```

- 指定した ID の InfoObject を取得する

  ```
  "WHERE SI_ID=9287"
  ```

- 親を共有しスケジュールプロセスで作成されたすべての InfoObject を取得する

  ```
  "WHERE SI_PARENTID=231 And SI_INSTANCE=1"
  ```

多くの WHERE 句では、条件として SI_KIND、SI_ID、SI_PARENTID、および SI_INSTANCE の 4 つの列が使用されます。

> ⓘ 注記
>
> SI_INSTANCE は、項目がスケジュールプロセスの結果によるものかどうかを判別する論理値が必要な場合に使用されます。

これらのプロパティは識別子として機能します。

DB 列	ID の型	値の型	IInfoObject アクセッサメソッド	説明
SI_ID	固有	整数	getID()	データベースの各 InfoObject インスタンスを一意に識別します。これはプライマリキーではありません。インスタンスが削除された場合、後で新しいインスタンスに同じ値が指定されることがあります。
SI_KIND	InfoObject 拡張クラス型別のグループ	文字列識別子 ("CrystalReport"など)	getKind()	特定の InfoObject 拡張クラス型によって各行を識別します。
SI_PARENTID	親 InfoObject の ID 別グループ。	整数	getParentID()	現在の InfoObject の親として機能する InfoObject インスタンスを識別します。 通常、スケジュールされるように設定されるレポートは親であり、スケジュール時にコピーおよび保存される各レポートは、ソースレポートをその親と見なします。
SI_INSTANCE	論理値	論理値	isInstance()	データベース行に保存されている項目がスケジュールによって作成された InfoObject (毎日夜間に作成されるレポートなど)、つまりインスタンスであるかどうかを識別します。

3.3.6.2 条件

条件は、CMS リポジトリに対する検索を絞り込むための基準です。1 つ以上の条件を使用して、クエリーから返されるオブジェクトの数を減らすことができます。

条件は次の形式です。

```
<property> <operator> <value>
```

- `<property>` は、値が割り当てられたオブジェクトのプロパティです。

- <operator> は、=、!=、>、<、>=、<=、LIKE、IN などのサポートされている演算子の 1 つです。
- <value> は、プロパティ値と比較するリテラル値です。

たとえば、SI_KIND プロパティが "Server" に設定されているオブジェクトを検索するには、クエリーの WHERE 句に次の条件を含めます。

```
SI_KIND = 'Server'
```

この場合、SI_KIND がプロパティ、= が演算子、リテラル文字列 "Server" が値です。

条件が複数ある場合は、それらの条件を AND または OR で結合できます。

- AND は 2 つの条件を結合し、両方の条件が true のときに true として評価されます。たとえば、次のクエリーは 2 つの条件を結合して、SI_KIND が "CrystalReport" で、SI_ID が指定された範囲内にあるオブジェクトを検索します。

  ```
  SELECT SI_NAME, SI_ID, SI_DESCRIPTION
  FROM CI_INFOOBJECTS
  WHERE
      SI_KIND = 'CrystalReport'
  AND
      SI_ID BETWEEN 100 AND 800
  ```

- OR は 2 つの条件を結合し、条件の少なくとも一方が true であれば、true として評価されます。たとえば、次のクエリーは 2 つの条件を結合して、SI_ID が 1 つの値の範囲内にあるか、不連続な値の組に含まれるオブジェクトを検索します。

  ```
  SELECT SI_NAME, SI_ID, SI_DESCRIPTION
  FROM CI_INFOOBJECTS
  WHERE
      SI_ID BETWEEN 100 AND 200
  OR
      SI_ID IN(576, 578, 901)
  ```

条件を否定するには、演算子の前に NOT キーワードを使用します。たとえば、次のクエリーは、SI_ID が特定のグループに含まれないオブジェクトを検索します。

```
SELECT SI_NAME, SI_ID, SI_DESCRIPTION
FROM CI_INFOOBJECTS
WHERE SI_ID NOT IN(576, 578, 901)
```

関連情報

演算子 [37 ページ]

3.3.6.3 演算子

クエリー言語では、次の演算子がサポートされています。

- =
- !=

- >
- <
- >=
- <=
- (NOT) LIKE
- (NOT) IN
- (NOT) BETWEEN ... AND ...
- ALL

3.3.6.3.1 ALL

ALL を使用して、すべてのオブジェクトから、最小値または最大値が設定されたプロパティを保持するオブジェクトを検索します。

次の書式を使用します。ここで、<operator> は =、!=、>、<、>=、<= のいずれかです。<property1> は <property2> と同じ型のプロパティです。

property1operator ALL *property2*

例

次のクエリーは、子の最大数を返します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_CHILDREN >= ALL SI_CHILDREN
```

次のクエリーは、子の最小数を返します。

```
SELECT SI_ID, SI_NAME
 FROM CI_INFOOBJECTS WHERE SI_CHILDREN <= ALL SI_CHILDREN
```

複数のオブジェクトの子の数が同じ場合があるため、前の例は、複数のオブジェクトを返す可能性があります。ただし、場合によっては、ALL を使用して一意の値を指定することもできます。たとえば、SI_ID >= ALL SI_ID を検索するクエリーは、SI_ID が最大のオブジェクトを返します。

2 つの異なるプロパティの値も比較できます。

例

次のクエリーは、他のオブジェクトの親ではないすべてのオブジェクトを返します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_ID != ALL SI_PARENTID
```

3.3.6.3.2 BETWEEN

BETWEEN を使用して、連続する範囲内のプロパティ値を検索します。次のクエリーは、SI_ID プロパティが 100 以上 300 以下である CI_INFOOBJECTS 内のすべてのオブジェクトを取得します。

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_ID BETWEEN 100 and 300
```

このクエリーは、数値のほかに文字列値でも使用できます。次のクエリーは、SI_NAME が I から P までの文字で始まる CI_INFOOBJECTS 内のすべてのオブジェクトを取得します (R で始まる SI_NAME 値は検索されません)。

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_NAME BETWEEN 'I' and 'R'
```

3.3.6.3.3 IN

IN を使用して、特定の値セット内にあるプロパティを検索します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_ID IN (103,105,106)
```

3.3.6.3.4 IS NULL

特定のプロパティが定義されていないオブジェクトを検索するには、IS NULL を使用します。

> ⓘ 注記
>
> IS NULL に一致するプロパティの検索と空文字列 ('') に一致するプロパティの検索は異なります。IS NULL 条件に一致するプロパティは、そのオブジェクトに対して定義されていないプロパティです。空文字列 ('') に一致するプロパティは、定義されているが、その値が空文字列であるプロパティです。

例

次のクエリーは、IS NULL を使用して、SI_DESCRIPTION プロパティが定義されていないオブジェクトを検索します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION IS NULL
```

次のクエリーは、IS NOT NULL を使用して、SI_DESCRIPTION プロパティが定義されているオブジェクトを検索します。その値が空かどうかは問いません。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION IS NOT NULL
```

次のクエリーは、検索条件として空文字列を使用して、SI_DESCRIPTION プロパティの値が空でない文字列に設定されているオブジェクトを取得します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION != ''
```

> **注記**
>
> 存在しないプロパティに IS NULL 条件を使用した場合は、すべてのオブジェクトに一致します。したがって、クエリーの実行速度が低下し（クエリー内の他の条件による）、パフォーマンスが落ちる可能性があります。クエリーを実行する前に、すべてのプロパティ名のスペルが正しく記述されていることを確認してください。

3.3.6.3.5 LIKE

パターンに一致する値を検索するには、LIKE を使用します。

パターンには、次のワイルドカード文字を指定できます。

ワイルドカード文字	説明	例
%	0 個以上の文字列。	`WHERE SI_NAME LIKE '%computer%'` と指定すると、名前に "computer" という単語が含まれるオブジェクトがすべて検索されます。
_（アンダースコア）	任意の単一の文字。	`WHERE SI_USERFULLNAME LIKE '_ean %'` と指定すると、"ean" で終わる 4 文字の名前がすべて検索されます（Dean、Sean など）。
[]	指定された範囲（[a-f]）またはセット（[abcdef]）に含まれる 1 文字。	`WHERE SI_USERFULLNAME LIKE '% [C-P]arsen'` と指定すると、C ～ P の 1 文字で始まり、arsen で終わる姓を持つオブジェクトがすべて検索されます（Carsen、Larsen、Karsen など）。
[^]	指定された範囲（[^a-f]）またはセット（[^abcdef]）に含まれない 1 文字。	`WHERE SI_NAME LIKE 'de[^l]%'` と指定すると、de で始まり、次の文字が l ではない名前のオブジェクトがすべて検索されます。

ワイルドカード文字をリテラル文字として使用するには、角かっこ（[]）で囲みます。たとえば、次のクエリーは、説明の最後が "5%" というテキストであるオブジェクトをすべて検索します。

```
SELECT SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION LIKE '%5[%]'
```

通常、検索は大文字小文字が区別されません。たとえば、次のクエリーは、説明に "report" という単語が含まれるオブジェクトを大文字小文字を区別せずに検索します。

```
SELECT SI_NAME, SI_DESCRIPTION FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION LIKE
'%report%'
```

また、LIKE を使用して、大文字小文字を区別せずに検索することもできます。クエリーの中で大文字小文字を区別する場合は、検索する文字列を角かっこで囲みます。

```
SELECT SI_ID, SI_DESCRIPTION FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION LIKE '%[R]
[E][P][O][R][T]%'
```

3.3.6.4 リテラル値

3.3.6.4.1 論理値

クエリーステートメントでは、論理値を True や False でなく、1 または 0 で記述する必要があります。

例

次のクエリーは、インスタンスであるすべてのオブジェクトを返します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_INSTANCE=1
```

3.3.6.4.2 日時値。

クエリーの日時の形式には、時間、分、および秒を指定できます。次のように指定します。

```
yyyy.mm.dd.hh.mm.ss
```

右端の項目から順番に省略する限り、どの項目も省略できます。たとえば、秒、または秒および分を省略できます。

> ⓘ 注記
>
> 時刻は、24 時間制で指定する必要があります。

hh.mm.ss を指定する場合、UTC 時間（GMT、標準時間、夏時間なし）で指定する必要があります。区切り文字には任意の文字を使用できます。また、各区切り文字を異なる文字にすることもできます。次のどちらの順序でも正しく動作します。

```
yyyy/mm/dd/hh/mm/ss
yyyy.mm.dd.hh.mm.ss
yyyy/mm/dd.hh.mm.ss
yyyy/mm/dd,hh:mm:ss
```

例

次のクエリーは、2000 年 1 月 11 日の午後 6 時以降に更新されたすべてのオブジェクトを返します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_UPDATE_TS >
'2000.01.11.18:00:00'
```

次のクエリーは、2000 年 1 月中に更新されなかったすべてのオブジェクトを返します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_UPDATE_TS != '2000.01'
```

3.3.6.4.3 数値

WHERE 句内の数値は、一重引用符で囲むことができますが、必須ではありません。たとえば、次のクエリーは同等です。

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_ID BETWEEN 10 and 2000
```

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_ID BETWEEN '10' and '2000'
```

3.3.6.4.4 文字列値

WHERE 句内の文字列値は、一重引用符で囲む必要があります。以下はその例です。

```
SELECT SI_ID, SI_NAME WHERE SI_NAME = 'Jonathan'
```

文字列内に一重引用符を入れるには、一重引用符文字を 2 つ重ねて使用します。たとえば、名前に **Jonathan's** が含まれるオブジェクトをリポジトリで検索する場合は、次のように 2 つの一重引用符を使用してアポストロフィを表します。

```
SELECT SI_ID, SI_NAME WHERE SI_NAME LIKE 'Jonathan''s%'
```

3.3.7 ORDER BY 句

ORDER BY 句を使用して、クエリーの結果を並べ替えます。ORDER BY 句は、1 つ以上のプロパティで構成されます。一連のプロパティによって結果セット内のオブジェクトの順序が決定されます。

プロパティ値に基づいて、クエリー結果を昇順 (ASC) または降順 (DESC) で並べ替えることができます。並べ替え順序が指定されない場合、デフォルトは ASC です。

例

次の例は、SI_NAME プロパティに基づいて昇順で並べ替えられたオブジェクトのセットを返します。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS ORDER BY SI_NAME
```

次の例は、2 つのプロパティに基づいて並べ替えられたオブジェクトのセットを返します。最初に、SI_ID プロパティに基づいて、結果が昇順で並べ替えられます。次に、SI_NAME プロパティに基づいて、結果が降順で並べ替えられます。

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS ORDER BY SI_ID ASC, SI_NAME DESC
```

3.3.8 クエリー文字列のベストプラクティス

以下の推奨事項に従って、クエリー文字列のスケーラビリティを改善します。効率のよいクエリーによって、システムのパフォーマンスが改善されます。

3.3.8.1 クエリーに必要なプロパティだけを記述

"SELECT *"、"SELECT DYNAMIC"、"SELECT STATIC"などの汎用クエリーは非常に負荷が大きく、アプリケーションのスケーラビリティを大きく阻害します。

最も効率のよいクエリーは、必要なプロパティだけを取得するクエリーです。

一部のプロパティは動的です。これらのプロパティは、基盤となるデータベースの特定の値にマップされません。クエリー文字列は、CMS リポジトリに渡されるときに動的に計算されます。

動的なプロパティにより、システムに対する負荷は大きくなります。前のセクションの静的なプロパティとは異なり、動的なプロパティには処理が必要です。したがって、動的な列は必要なときだけ呼び出し、スケーラビリティが問題になる場合は使用しないでください。

次の一覧に、最も簡単で高速のクエリーから最も負荷の高いクエリーまでを示します。

- ID プロパティだけを選択します。

  ```
  "SELECT SI_ID"
  ```

  これはセキュリティ権限の設定に役立ちます。
- ID プロパティと Name プロパティを選択します。

  ```
  "SELECT SI_ID, SI_NAME"
  ```

  これはレポートやユーザーなどアイテムの一覧を表示する場合に役立ちます。
- 必要なすべてのプロパティを選択しますが、動的なプロパティは含めないでください。

  ```
  "SELECT SI_ID, SI_NAME, SI_DESCRIPTION, SI_SCHEDULEINFO.DESTINATION, ..."
  ```

  詳細な情報も含めてクラスを取得するときに最も一般的な選択のタイプです。Select 句には多数のプロパティの一覧が含まれますが、特定のクラスに必要なプロパティだけが取得されます。インデックス付きプロパティはクエリーの先頭部分にあります。
- 必要なすべてのプロパティを選択し、必要に応じて動的なプロパティも含めます。

  ```
  "SELECT SI_ID, SI_NAME, SI_DESCRIPTION, SI_SCHEDULEINFO.DESTINATION,
  SI_PATH, ..."
  ```

  前と同じクエリーですが、必要とされる動的なプロパティが追加されています。動的なプロパティによってクエリーの負荷は高くなりますが、指定された動的なプロパティだけが影響します。
- すべての静的なプロパティを選択します。

  ```
  "SELECT STATIC"
  ```

  これは、静的なプロパティをすべて収集しますが、動的なプロパティは収集しません。これは検証のためのクエリーにだけ使用し、本稼動アプリケーションでは使用しないでください。

> ⓘ 注記
>
> 動的なプロパティを使用しなかった以前のバージョンでは、このクエリーは "SELECT *" と同等です。

- すべての動的なプロパティを選択します。

  ```
  "SELECT DYNAMIC"
  ```

  これは、計算を必要とするすべてのプロパティが返されるため、プロセスに大きな負荷がかかります。検証のためのクエリーにだけ使用してください。
- 静的なプロパティと動的なプロパティのどちらであるかには関係なく、すべてのプロパティを選択します。

  ```
  "SELECT *"
  ```

  これは最も遅いクエリーです。これは、計算を必要とするすべての静的なプロパティとすべての動的なプロパティが返されるため、プロセスに大きな負荷がかかります。検証のためのクエリーにだけ使用してください。

3.3.8.2 TOP N 関数の使用

サーバーを照会するときは、対象をできるだけ限定してください。たとえば、SELECT ステートメントの中でアスタリスク演算子（*）を使用するときは、必ず TOP N 関数か WHERE 句を使用して、選択対象を少数のオブジェクトに限定してください。すべてのオブジェクトを選択するように CMS を設定すると、システムがリクエストによって過負荷の状態になるおそれがあります。CMS は、特定のオブジェクトに関するすべてのプロパティを選択する場合、データの取得に何倍もの時間を要することがあります。

少数のオブジェクトを取得する場合以外は、TOP N を指定せずに * を使用しないでください。たとえば、`SELECT * FROM CI_INFOOBJECTS WHERE SI_ID=128` とすると、オブジェクトを 1 つだけ取得できます。

- `SELECT *` が返す上限は 1000 オブジェクトです。TOP N 関数を使用すると、1000 個を超えるオブジェクトを要求できます。
- IInfoObjects コレクションの getResultSize() メソッドには、特定のクエリーを満たす、CMS 上のオブジェクトの個数が含まれます。その数を、CMS から返された実際のオブジェクト数と比較できます。

例: 効率の悪い使用例

ここでは、CMS リポジトリのすべてのレポートから、レポートのさまざまな詳細情報を取得する必要がある例を考えてみましょう。

```
Select * From CI_INFOOBJECTS Where SI_KIND = 'CrystalReport'
```

このクエリーは、すべてのレポートオブジェクト（上限は 1000 オブジェクト）のすべてのプロパティを返します。

例: 効率のよい使用例

多数のレポートが存在する場合は、次のように、アスタリスクに TOP N 演算子を付けたステートメントを使用すると、処理速度が大幅に向上します。

```
Select TOP 100 * From CI_INFOOBJECTS Where SI_KIND = 'CrystalReport'
```

プロパティのリストに TOP N を付加した次のようなステートメントを使用することもできます。

```
Select TOP 100 SI_NAME, SI_ID, SI_CHILDREN, SI_DESCRIPTION, SI_HASTHUMBNAIL _
From CI_INFOOBJECTS Where SI_KIND = 'CrystalReport'
```

このクエリーは大きくて扱いにくいように思われるかもしれませんが、アスタリスクを TOP N 演算子と求めるプロパティに置き換えることで、検索時間が分単位から秒単位になる場合があります。

どうしても SELECT * を使用する場合は、TOP N を使用すれば、クエリーで返すレコードの数を制限できます。たとえば、次のようなことができます。

```
Select TOP 20 * From CI_SYSTEMOBJECTS Where SI_KIND = 'User' ORDER BY SI_NAME
```

このコードは、最初の 20 人のユーザーを文字順に返します（たとえば、"Anna"から"Joanna"）。その後、次の 20 人のユーザーを取得できます。

```
Select TOP 20 * From CI_SYSTEMOBJECTS Where SI_KIND = 'User' AND SI_NAME > 'Joe'
ORDER BY SI_NAME
```

3.3.8.3 インデックス付きプロパティの使用

InfoStore クエリーの結果セットをフィルタしたり、オブジェクトを迅速に検索するには、インデックス付きプロパティを使用します。

たとえば、クエリーの結果セットに 10,000 個の Report オブジェクトが含まれるとします。このようなクエリーのパフォーマンスを向上させるには、結果セットをフィルタして絞り込みます。SI_PROGID などのインデックス付きプロパティを使用して、小さな結果セットを返す効率のよいクエリーを作成することで、検索時間を短縮できます。

次に、すべてのインデックス付きプロパティを示します。

列/プロパティ
SI_CUID
SI_GUID
SI_HIDDEN_OBJECT
SI_ID
SI_INSTANCE_OBJECT

列/プロパティ
SI_KIND
SI_NAME
SI_NAMEDUSER
SI_NEXTRUNTIME
SI_OWNERID
SI_PARENTID
SI_PLUGIN_OBJECT
SI_PROGID
SI_RECURRING
SI_RUID
SI_RUNNABLE_OBJECT
SI_SCHEDULE_STATUS
SI_UPDATE_TS

> **注記**
>
> SI_NAME プロパティは、インスタンスを含むすべてのオブジェクトに対してインデックスが付けられます。

3.3.8.4 WHERE 句の作成で、インデックス付きでないプロパティの使用を回避

Central Management Server（CMS）は WHERE 句を検査し、プロパティがインデックス付きかどうかを判別します。CMS は、この情報を使用して、最適化されたクエリーを作成します。

SI_NAME='testName' などのインデックス付きプロパティは、データベースのリレーショナルデータに格納されます。インデックス付きプロパティを含むクエリーは、直接データベースを照会し、検索条件（testName）に一致するすべてのオブジェクト ID を取得します。

SI_SERVER_ID='myServer' などのインデックス付きでないプロパティは、データベースのバイナリラージオブジェクト（BLOB）データに格納されます。BLOB は、サイズと場所によってのみ判別され、データベース管理システムによって構造は解釈されません。CMS は、データベース内の各オブジェクトの BLOB を取得および検査してから、検索条件（myServer）とオブジェクト（BLOB の SI_SERVER_ID の値）の一致があるかないかを判定します。

たとえば、WHERE 句は、インデックス付きのプロパティとインデックスなしのプロパティの両方を含むことがあります。

このクエリーを最適化するために、CMS は検索を次の 2 つのカテゴリに分けます。

1. CMS は、名前が testObject であるすべてのオブジェクトの Object ID を迅速に取得します。

2. CMS は、BLOB を取得し（名前が testObject であるオブジェクトだけから）、SI_SERVER_ID が一致するかどうかをチェックします。

クエリーのパフォーマンスを向上させるには、できる限りインデックス付きのプロパティを使用し、インデックス付きでないプロパティは制限して、WHERE 句を作成します。

- ISchedulingInfo プロパティや IProcessingInfo プロパティを直接 WHERE 句で使用することは可能ですが、これらのプロパティはインデックス付きでないため、クエリープロセスが遅くなります。このようなプロパティは、結果セットが小さい場合に使用してください。結果を小さくするには、WHERE 句に、インデックス付きでないプロパティと共に適切なインデックス付きプロパティのフィルタを設定します。
- WHERE 句の中では、論理値と日時の値を特定の形式で指定する必要があります。

3.3.8.5 WHERE 句での LIKE の使用を禁止

WHERE 句では LIKE を使用しないでください。

LIKE 演算子は、“=”演算子より処理に時間がかかります。LIKE を「=」に置き換えると、パフォーマンスが大幅に向上します。

3.4 URI クエリーを使用したオブジェクトの取得

`IInfoStore.getPagingQuery` メソッドと URI クエリー文字列を使用して、Central Management Server (CMS) に保存されているオブジェクトを含む一連のページ化された結果を検索および取得します。たとえば、インストールのルートフォルダにあるすべての子フォルダを取得するには、次の URI パスクエリーを実行します。

```
path://InfoObjects/Root Folder/*[SI_KIND='Folder']
```

URI クエリー言語によって提供されるパスベースのリポジトリビューは、BI 起動パッドなどのユーザーアプリケーションのビューによく対応しています。これにより、言語が直感的にわかりやすくなり、InfoObject をフォルダベースの構造のように使用できます。

URI クエリー言語を使用して、次の操作を実行できます。

- パスによってリソースに直接アクセスする。
- リポジトリフォルダ階層から特定の InfoObject を再帰的に検索する。
- 特定の条件またはプロパティに一致する InfoObject を検索する。

このセクションでは、URI クエリー言語を使用して CMS リポジトリからオブジェクトを取得する方法について説明します。URI クエリーで最もよく使用される要素や、SDK クラスがパスクエリー言語とやり取りする方法について説明します。

関連情報

URI パスクエリーを実行するには、次の手順を実行します。 [48 ページ]
クエリー言語を使用したオブジェクトの取得 [24 ページ]

3.4.1 予備知識

URI クエリー言語を使用するには、いくつかの重要な SQL クエリー言語の概念を理解しておく必要があります。

次に示す概念については、基本的な理解が必要です。

- SI_NAME、SI_CUID などのクエリー言語プロパティ。
- WHERE 句、ORDER BY 句などのクエリー言語の構文。形式が異なりますが、これらの句は URI クエリー言語でも使用されます。
- システムパフォーマンスを向上させるために、パスクエリーで必要なプロパティだけを選択する方法。
- インデックス付きプロパティによってクエリーを最適化する方法。

関連情報

インデックス付きプロパティの使用 [64 ページ]

3.4.2 URI パスクエリーを実行するには、次の手順を実行します。

CMS リポジトリに対して URI パスクエリーを実行するには、有効な `IEnterpriseSession` オブジェクトが必要です。

URI クエリーは、ページ単位の結果を生成することを最終目的として、まず `IInfoStore.getPagingQuery` メソッドによって処理されます。このメソッドは、この URI クエリーにページング操作を実行し、InfoObjects のページごとに一連のサブクエリーを返します。次に、これらのページベースのサブクエリーを反復処理し、それぞれを有効なクエリー言語（SQL）クエリーに変換して、各ページの結果を処理できます。

1. `IEntepriseSession` オブジェクトの `getService` ファクトリメソッドを呼び出し、引数を **`InfoStore`** に設定します。返されるオブジェクトを `IInfoStore` にキャストします。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. クエリー文字列を作成します。
   クエリー文字列の形式は、次のとおりです。

   ```
   <protocol>://<query>
   ```

   - `<protocol>` は、実行する URI パスクエリーのタイプを指定します。次の 4 つのプロトコルがサポートされています。

    - path：オブジェクトまたはオブジェクトのグループへの URI パスを指定できます。このパスにフィルタを含めて、結果を絞り込むことができます。
    - cuid：1 つ以上のオブジェクト CUID を指定できます。
    - search：検索条件を指定できます。
    - query：SQL 形式のクエリー言語を使用できます。
  - <query> は、プロトコル固有のクエリー文字列です。

次のコードは、リポジトリ内の ID が 678 ～ 700 の範囲のすべてのシステムオブジェクトを取得するクエリー文字列です。

```
String queryString =
      "path://SystemObjects/**/[SI_ID BETWEEN 678 AND 700]";
```

> ⓘ 注記
>
> クエリー言語のキーワード、プロパティ名、およびテーブル名は、大文字と小文字が区別されません。たとえば、プロトコルを指定する場合、path は PATH と同じです。

3. PagingQueryOptions オブジェクトを作成します。コンストラクタの呼び出しで、1 ページに表示できる項目の最大数を渡します。

```
int maxItemsPerPage = 20;
PagingQueryOptions options =
        new PagingQueryOptions(maxItemsPerPage);
```

4. クエリー文字列と PagingQueryOptions オブジェクトを IInfoStore オブジェクトの getPagingQuery メソッドに渡します。

```
IPageResult pageResult =
        infoStore.getPagingQuery(queryString, options);
```

getPagingQuery メソッドは、クエリーと PagingQueryOptions オブジェクトで指定された 1 ページあたりの最大項目数を検証します。このメソッドは、この情報を使用して必要なページ数を決定し、次に、サブクエリーのセットを生成します。各サブクエリーを使用して、特定のページの項目を取得できます。getPagingQuery は、サブクエリーを含む IPageResult コレクションを返します。

5. IPageResult コレクションの iterator メソッドを使用して反復子オブジェクトを取得し、コレクションから個別のクエリーを取得します。反復子から返されるオブジェクトを IStatelessPageInfo にキャストします。

```
Iterator pageResultIter = pageResult.iterator();
while(pageResultIter.hasNext()) {
    IStatelessPageInfo pageInfo = (IStatelessPageInfo) pageResultIter.next();
    ...
}
```

6. IStatelessPageInfo オブジェクトの getPageSQL を呼び出します。

```
String pageQuery = pageInfo.getPageSQL();
```

IInfoStore の query メソッドに渡すことができるクエリー文字列が返されます。

7. IInfoStore オブジェクトの query メソッドにクエリー文字列を渡します。

```
IInfoObjects pageInfoObjects = infoStore.query(pageQuery);
```

query メソッドは、クエリーの結果が入っている IInfoObjects コレクションを返します。

8. IInfoObjects コレクションの iterator メソッドを使用して、反復子オブジェクトを取得します。コレクションから個別のオブジェクトを取得します。反復子から返されるオブジェクトを IInfoObject にキャストします。

```
Iterator infoObjectsIter = pageInfoObjects.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject infoObject = (IInfoObject) infoObjectsIter.next();
    ...
}
```

3.4.3 オブジェクトタイプ

URI パスクエリーを使用して InfoObject を取得するには、CMS リポジトリ内のパスがわかっている必要があります。InfoObject は、タイプごとにルートテーブルとベースフォルダが異なります。

InfoObject タイプに基づくパス

InfoObject タイプ	ルートテーブル	ベースフォルダ
Calendar	SystemObjects	Calendars
Category	InfoObjects	Categories
Connection	SystemObjects	Logon Sessions
CrystalReport	InfoObjects	Root Folder
CrystalEnterprise.DiskUnmanaged	SystemObjects	Plugins/Destination Plugins/
Event	SystemObjects	Events
Excel	InfoObjects	Root Folder
FavoritesFolder	InfoObjects	User Folders
Flash	InfoObjects	Root Folder
Folder	InfoObjects	Root Folder
CrystalEnterprise.Ftp	SystemObjects	Plugins/Destination Plugins/
FullClient	InfoObjects	Root Folder
Hyperlink	InfoObjects	Root Folder

InfoObject タイプ	ルートテーブル	ベースフォルダ
Inbox	InfoObjects	Inboxes
LicenseKey	SystemObjects	License Keys
CrystalEnterprise.Managed	SystemObjects	Plugins/Destination Plugins/
ObjectPackage	InfoObjects	Root Folder
Overload	AppObjects	Application Folder
Pdf	InfoObjects	Root Folder
PersonalCategory	InfoObjects	Personal Categories
PowerPoint	InfoObjects	Root Folder
Profile	SystemObjects	Profiles
Program	InfoObjects	Root Folder
Publication	InfoObjects	Root Folder
RTF	InfoObjects	Root Folder
secEnterprise	SystemObjects	Plugins/Auth Plugins/
secLDAP	SystemObjects	Plugins/Auth Plugins/
secWinAD	SystemObjects	Plugins/Auth Plugins/
Server	SystemObjects	Servers
ServerGroup	SystemObjects	Server Groups
Shortcut	InfoObjects	Root Folder
CrystalEnterprise.Smtp	SystemObjects	Plugins/Destination Plugins/
Txt	InfoObjects	Root Folder
Universe	AppObjects	Application Folder
User	SystemObjects	Users
UserGroup	SystemObjects	User Groups
Webi	InfoObjects	Root Folder

InfoObject タイプ	ルートテーブル	ベースフォルダ
Word	`InfoObjects`	`Root Folder`
Xcelsius	`InfoObjects`	`Root Folder`

関連情報

パス演算子：/ [56 ページ]

3.4.4 各種のプロトコルです。

URI クエリーは、以下の 4 種類のプロトコルをサポートします。

- `path://`
  このプロトコルでは、URI パスを使用して InfoObject を検索します。InfoObject の検索にはワイルドカード文字を使用できます。
- `cuid://`
  このプロトコルでは、CUID を使用して InfoObject を検索します。
- `search://`
  このプロトコルでは、ワイルドカード文字より具体的な検索条件を使用して InfoObject を検索します。
- `query://`
  このプロトコルでは、従来の SQL クエリー言語を使用して InfoObject を検索します。

> **注記**
>
> すべてのプロトコルで同じ結果が生成され、CMS から 1 つ以上の InfoObject リソースが返されます。したがって、どのクエリーを使用してもまったくかまいません。

3.4.4.1 パスプロトコル

パスクエリー言語の構文は次のとおりです。

```
path://[Root table]/[Base folder]/folder1/folder2/.../[Target resource SI_NAME]
```

パスは、次の 3 つのルートテーブルのいずれかで始まります。

- InfoObjects
- SystemObjects
- AppObjects

ルートテーブルからターゲットリソースまでのフォルダの階層パスをスラッシュ文字で区切って指定します。

ルートテーブルとベースフォルダは、InfoObject のタイプによって異なります。

クエリーとパスクエリー演算子を組み合わせることができます。

サンプル：**Crystal** レポートの取得

この例では、CMS から `My Report` という名前のレポートを取得します。このレポートは、`Feature Samples` フォルダに存在するとします。

> ⓘ 注記
>
> InfoObject のフルネームまたは場所がわからない場合は、パスクエリー言語を使用して検索を実行できます。

`CrystalReport` InfoObject タイプの場合、ルートテーブルは `InfoObjects`、ベースフォルダは `Root Folder` です。したがって、パスクエリーは、`path://InfoObjects/Root Folder` で始める必要があります。

`Feature Samples` フォルダはベースフォルダ内にあるため、次のようにパスクエリーを終えます。

```
path://InfoObjects/Root Folder/Feature Samples/My Report
```

例：ユーザーの取得

この例では、CMS から `Administrator` ユーザーオブジェクトを取得します。

`User` InfoObject タイプの場合、ルートテーブルは `SystemObjects`、ベースフォルダは `Users` です。したがって、パスクエリーは、`path://SystemObjects/Users` で始める必要があります。

`Administrator` ユーザーはベースフォルダ内にあるため、次のようにパスクエリーを終えます。

```
path://SystemObjects/Users/Administrator
```

関連情報

オブジェクトタイプ [50 ページ]
演算子 [55 ページ]

3.4.4.1.1 ワイルドカード文字

パスクエリーでは、フォルダとターゲットリソース名にワイルドカード文字を使用できます。ワイルドカード文字は次の 2 種類あります。

- 1 つのアスタリスク（*）。
  クエリーで任意の 1 要素に一致します。

- 2 つのアスタリスク（**）。
  任意の要素に再帰的に一致します。

> ⓘ 注記
>
> 各ワイルドカード文字に対して CMS への呼び出しが 1 往復増えるため、システムのパフォーマンスは低下します。

サンプル：すべてのテーブル（**InfoObjects**、**AppObjects**、**SystemObjects**）内の名前**"My Report"**に一致するすべての **InfoObject** を返します

```
path://*/**/My Report
```

例：文字 **Y** で始まるすべてのユーザーを返します

```
path://SystemObjects/Users/Y*
```

例：ルートフォルダから **2** レベル下のすべての **InfoObject** の検索

```
path://InfoObjects/Root Folder/*/*/*
```

例：文字 **B** で始まるフォルダ内の文字 **C** で始まるすべての **InfoObject** の検索

```
path://InfoObjects/Root Folder/B*/C*
```

3.4.4.2 CUID プロトコル

CUID プロトコルの構文は次のとおりです。

```
cuid://<CUID_1, CUID_2,..., CUID_N>
```

クエリーと演算子を組み合わせることができます。

関連情報

演算子 [55 ページ]

3.4.4.3 検索プロトコル

検索プロトコルの構文は次のとおりです。

```
search://{search_term_1 search_term_2 ... search_term_N}?search_option_1=[true |
false]&...search_option_n=[true | false]
```

検索語の形式は文字列です。検索クエリーは、少なくとも 1 つの検索語に一致するすべての InfoObject を返します。一重引用符で文字列全体を囲むと、文字列にスペースを挿入できます。たとえば、文字列 `'My World'` は 1 語ですが、`My World` は 2 語になります。

各検索オプションの値は `true` または `false` です。

例

次のクエリーは、`SI_KEYWORDS` または `SI_NAME` が文字列 "Financial" または "My World" のいずれかに完全一致するすべての InfoObject を返します。

```
search://{Financial 'My World'}?MatchExact=true&CaseSensitive=true
```

関連情報

パラメータ演算子：? [58 ページ]

3.4.4.4 SQL プロトコル

SQL プロトコルの構文は次のとおりです。

```
query://{SELECT [Properties] FROM [Table] WHERE [Conditions] ORDER BY [Properties]}
```

関連情報

クエリー言語を使用したオブジェクトの取得 [24 ページ]

3.4.5 演算子

パスクエリー演算子は、パスクエリーに追加できるオプションの演算子です。1 つのクエリーで、複数の演算子を組み合わせることができます。

3.4.5.1 パス演算子：/

関係を介してターゲットリソースにリンクされている InfoObject を返すには、パスクエリーの最後でパス演算子 (/) を使用します。

この演算子の後に関係キーワードを指定できます。関係キーワードを指定しない場合は、その InfoObject のデフォルトの関係が使用されます。

サンプル：パス演算子なし

このパスクエリーは、`Report Samples` フォルダを返します。

```
path://InfoObjects/Root Folder/Report Samples
```

例：パス演算子

クエリーの最後にパス演算子を追加した場合は、デフォルト関係を介してこのフォルダにリンクされている InfoObject が返されます。フォルダのデフォルトの関係は、その子オブジェクトです。したがって、このパスクエリーは、`Report Samples` フォルダ内にあるすべての InfoObject を返します。

```
path://InfoObjects/Root Folder/Report Samples/
```

関連情報

リレーションシップ [60 ページ]

3.4.5.2 親包含演算子：+

クエリー結果に親 InfoObject を含めるには、パス演算子 (/) と一緒に親包含演算子 (+) を使用します。

サンプル：**"My Report"**オブジェクトと、**"My Report"**を親として参照するすべてのインスタンスを返します

```
path://InfoObjects/Root Folder/Report Samples/My Report+/*
```

3.4.5.3 属性演算子：[ゾクセイエンザンシ：]

クエリーで返す InfoObject プロパティを指定するには、属性演算子 (@) を使用します。この演算子を指定しない場合は、InfoObject のすべてのプロパティが返されます。

サンプル："My Report"オブジェクトの SI_CUID および SI_NAME プロパティを返します

```
path://InfoObjects/Root Folder/Report Samples/My Report@SI_CUID, SI_NAME
```

3.4.5.4 条件演算子：[ ]

クエリーを条件でフィルタするには、条件演算子 ([ ]) を使用します。条件演算子には、任意の有効な SQL WHERE 句を入れることができます。

サンプル："Report Samples"フォルダ内の CUID が 123 より大きいすべてのオブジェクトを返します

```
path://InfoObjects/Root Folder/Report Samples/[SI_CUID > 123]
```

例：「Report Samples」フォルダ内の文字 P で始まるすべてのオブジェクトを返します

```
path://InfoObjects/Root Folder/Report Samples/[SI_NAME LIKE 'P%']
```

これは次のクエリーと同じです。

```
path://InfoObjects/Root Folder/Report Samples/P*
```

関連情報

WHERE 句 [34 ページ]

3.4.5.5 パラメータ演算子：?

クエリーに並べ替え順と検索オプションを追加するには、パラメータ演算子 (?) を使用します。この演算子は、次の 1 つ以上のキーワードと一緒に使用します。

> ⓘ 注記
>
> アンパサンド（&）文字を使用して、複数の検索オプションを組み合わせることができます。

OrderBy

OrderBy パラメータは、クエリー結果の並べ替え方法を指定します。

```
path://[Path query]?OrderBy=[property_1 [ASC | DESC]], [property_2 [ASC | DESC]],
```

ASC は昇順を指定し、DESC は降順を指定します。デフォルトの並べ替え順は昇順です。

デフォルトのオプションは OrderBy=SI_ID です。SI_CUID による並べ替えはサポートされていません。

> ⓘ 注記
>
> クエリー結果は、並べ替え順を指定しない場合でも、デフォルトでページングされます。クエリー結果をページングするには、OrderBy 句の各 InfoObject プロパティをインデックス付きプロパティにする必要があります。ページングが行われない場合、結果は 1 ページで返されます。大量の結果が 1 ページで返されると、パフォーマンスが大幅に低下します。

次の例は、ルートフォルダ内のすべての InfoObject を返し、SI_NAME の降順でオブジェクトを並べ替えます。

```
path://InfoObjects/Root Folder/**/*?OrderBy=SI_NAME DESC
```

次の例は、ルートフォルダ内のすべての InfoObject を返し、SI_PARENTCUID の降順で並べ替え、次に SI_NAME の昇順で並べ替えます。

```
path://InfoObjects/Root Folder/**/*?OrderBy=SI_PARENTCUID DESC, SI_NAME ASC
```

検索プロトコルでは、次の検索オプションが使用されます。

SearchName

InfoObject の SI_NAME プロパティを検索するかどうかを指定します。

デフォルト値は True です。

SearchKeywords

InfoObject の `SI_KEYWORDS` プロパティを検索するかどうかを指定します。

デフォルト値は True です。

CaseSensitive

検索語の大文字小文字と InfoObject の大文字小文字が一致する必要があるかどうかを指定します。

デフォルト値は false です。

MatchAllWords

すべての検索語が InfoObject に一致する必要があるかどうかを指定します。

> **注記**
>
> 複数の検索用語を使用し、`MatchAllWords` および `MatchExact` がどちらも `true` に設定されている場合、結果は何も返されません。複数の検索語を使用し、すべての検索語が同時に一致する必要がある場合、完全一致は行われません。

デフォルト値は false です。

FindWithoutWords

どの検索語にも一致しない InfoObject を検索するかどうかを指定します。

たとえば、`FindWithoutWords` を `true` に設定し、InfoObject 名 `Kitty` を検索すると、`Kitty` 以外の名前のすべての InfoObject が返されます。

デフォルト値は false です。

MatchExact

検索語が InfoObject に完全に一致にする必要があるかどうかを指定します。部分文字列の一致は無視されます。

デフォルト値は false です。

IncludeInstances

検索結果に InfoObject のインスタンスを含めるかどうかを指定します。

デフォルト値は false です。

3.4.6 リレーションシップ

最も一般的な関係の 1 つは、親フォルダ InfoObject とその子の関係です。この場合、子は親フォルダより 1 レベル低いため、この関係は階層関係です。

1 つの階層構造内に共存しない InfoObject 間にも関係は存在します。階層的に関連していない InfoObject を返す場合は、クエリーで関係キーワードを使用する必要があります。

関係言語の構文は次のとおりです。

```
path://[Path query]/[Target resource SI_NAME]/[keyword-property combination]
```

キーワード/プロパティの組み合わせは、`keyword[property]` という形式で指定します。`keyword` には関係キーワードを指定し、`property` には関係の適用先の InfoObject プロパティを指定します。

このセクションでは、クエリーに挿入できる関係キーワードを一覧します。

3.4.6.1 Ancestors（先祖）関係

構文

`ancestors[property]`

階層内でターゲットリソースより上にあるすべての InfoObject（親を含む）を返します。

例

Administrator ユーザーを含むすべてのユーザーグループを返します。Administrator を直接含むユーザーグループのユーザーグループも含まれます。

`path://SystemObjects/Users/Administrator/ancestors[SI_USERGROUPS]`

3.4.6.2 Descendants（子孫）関係

構文

```
descendants[property]
```

階層内でターゲットリソースより下にあるすべての InfoObject（子を含む）を返します。

例

Administrators ユーザーグループ内のすべてのユーザーグループを返します。

```
path://SystemObjects/User Groups/Administrators/descendants[SI_GROUP_MEMBERS]
```

3.4.6.3 Connected components（接続コンポーネント）関係

構文

```
connected components[property]
```

ターゲットリソースへの接続コンポーネントであるすべての InfoObject を返します。

例

My Report オブジェクトに関連付けられているすべての接続コンポーネントを返します。

```
path://InfoObjects/Root Folder/Report Samples/My Report/connected
components[SI_BUSINESSVIEWS]
```

3.4.6.4 Parents（親）関係

構文

```
parents[property]
```

ターゲットリソースの親であるすべての InfoObject を返します。

例

Administrator ユーザーを含むユーザーグループを返します。

```
path://SystemObjects/Users/Administrator/parents[SI_USERGROUPS]
```

3.4.6.5 Children（子）関係

構文

```
children[property]
```

ターゲットリソースの子であるすべての InfoObject を返します。

例

Administrators ユーザーグループの直下のユーザーグループを返します。ただし、そのサブグループは返しません。

```
path://SystemObjects/User Groups/Administrators/children[SI_GROUP_MEMBERS]
```

3.4.6.6 Members（メンバ）関係

構文

```
members[property]
```

ターゲットリソースに含まれる InfoObject メンバを返します。

例

Administrators ユーザーグループのユーザーグループメンバーを返します。

```
path://SystemObjects/User Groups/Administrators/members[SI_GROUP_MEMBERS]
```

3.4.6.7 Groups（グループ）関係

構文

```
groups[property]
```

ターゲットリソースを含むすべてのユーザーグループを返します。グループ関係は、メンバー関係の反対です。

例

Administrator ユーザーを含むすべてのユーザーグループを返します。

```
path://SystemObjects/Users/Administrator/groups[SI_USERGROUPS]
```

3.4.6.8 デフォルトのリレーションシップ

パス演算子の後に関係キーワードを指定しない場合は、その InfoObject のデフォルトの関係が使用されます。ほとんどの場合、デフォルト関係は子関係になります。

たとえば、CrystalReport InfoObject のデフォルト関係は子関係です。したがって、次の 2 つのクエリーは同じです。

```
path://InfoObjects/Root Folder/Feature Samples/My Report/
```

```
path://InfoObjects/Root Folder/Feature Samples/My Report/children[SI_PARENTID]
```

どちらのクエリーでも CrystalReport InfoObject に属するインスタンスが返されます。

この InfoObject に子がない場合は、何も返されません。

InfoObject タイプに基づくデフォルト関係

InfoObject タイプ	デフォルト関係
User	`groups[SI_USERGROUPS]`
UserGroup	`members[SI_GROUP_MEMBERS]`
その他の InfoObject タイプ	`children[SI_PARENTID]`

3.4.7 URI クエリのベストプラクティス

3.4.7.1 クエリーに必要なプロパティだけを記述

最も効率のよいクエリーは、必要なプロパティだけを取得するクエリーです。使用する予定がある InfoObject プロパティだけを返すには、属性演算子を使用します。

関連情報

属性演算子：[ゾクセイエンザンシ：] [57 ページ]

3.4.7.2 インデックス付きプロパティの使用

クエリーの結果セットをフィルタしたり、オブジェクトを迅速に検索するには、インデックス付きプロパティを使用します。

たとえば、クエリーの結果セットに 10,000 個の CrystalReport オブジェクトが含まれるとします。このようなクエリーのパフォーマンスを向上させるには、結果セットをフィルタして絞り込みます。SI_NAME などのインデックス付きプロパティを使用して、小さな結果セットを返す効率のよいクエリーを作成することで、検索時間を短縮できます。

次に、すべてのインデックス付きプロパティを示します。

プロパティ
SI_CUID
SI_GUID
SI_HIDDEN_OBJECT
SI_INSTANCE_OBJECT
SI_KIND
SI_NAME
SI_NAMEDUSER
SI_NEXTRUNTIME
SI_OWNERID
SI_PARENTID

プロパティ
SI_PLUGIN_OBJECT
SI_PROGID
SI_RECURRING
SI_RUID
SI_RUNNABLE_OBJECT
SI_SCHEDULE_STATUS
SI_UPDATE_TS

> ⓘ 注記
>
> SI_CUID による並べ替えはサポートされていません。SI_NAME プロパティは、インスタンスを含むすべてのオブジェクトに対してインデックスが付けられます。

3.4.7.3 URIParserException 例外のキャッチおよび処理

`URIParserException` は、CMS に対してページ化された URI クエリーを実行したときに発生するエラーを区分けするために使用される `SDKException` の特別なサブクラスです。このようなエラーには、クエリー内の不正なクエリー式や無効な URI パスがあります。`URIParserException` には、このようなエラーが発生したときにスローされる多数のサブクラスが含まれます。アプリケーションの柔軟性を高めるには、`URIParserException` サブクラス例外を最初にキャッチして処理します。

たとえば、ドキュメントのクエリー時にフォルダパス内のエラーをチェックできます。

```
try {
    ...
    String URIquery = "path://InfoObjects/Root Folders/Report Samples/
Demonstration/" + reportName;
    int maxItemsPerPage = 20;
    PagingQueryOptions options = new PagingQueryOptions(maxItemsPerPage);
    IPageResult pageResult = iStore.getPagingQuery(URIquery, options);
}
catch (URIParserException.InvalidPathNode e) {
    ...
}
catch (URIParserException e){
    ...
}
```

すべての SDK URI クエリーの例外の一覧については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスで `com.crystaldecisions.sdk.exception.URIParserException` クラスを参照してください。

3.5 多言語コンテンツの操作

BI platform Java SDK で多言語コンテンツを作成または変更することはできません。ただし、この SDK を使用して、BI platform インストール内の多言語コンテンツを取得および表示することはできます。多言語コンテンツを作成および管理するには、SAP BusinessObjects Business Intelligence Platform Client Tools インストーラでインストールできる トランスレーションマネジメントツールを使用します。

多言語 **InfoObject** プロパティ

InfoObject には、多言語情報を含むプロパティがいくつかあります。

- `SI_ML_NAME` - ローカライズされた InfoObject の名前を含むプロパティ バッグ。
- `SI_ML_DESCRIPTION` - ローカライズされた InfoObject の説明を含むプロパティ バッグ。
- `SI_FILES` - InfoObject のローカライズされたファイルプロパティを含むプロパティ バッグ。

> **注記**
>
> CMS に照会する場合は、`SI_CUID` プロパティを使用して、検索する InfoObject を指定します。各 InfoObject は、`SI_CUID` プロパティによって一意に特定できます。`SI_NAME` プロパティと `SI_ML_NAME` プロパティは一意性が保証されないため、照会の際に InfoObject を特定するためには使用しないでください。

新しい多言語 **API**

この SDK の一部の API は、ローカライズされたコンテンツを返す多言語オーバーロードメソッドを備えています。これらのオーバーロードメソッドは、要求されたロケールをパラメータとして受け取り、要求されたロケールが存在しない場合は、フォールバックメカニズムを使用して、別のロケールでコンテンツを返します。

`IInfoObject` インターフェイスは、一部のメソッドに対して多言語オーバーロードメソッドを定義しています。

- `getTitle` - 要求されたロケールの `SI_ML_NAME` プロパティを返します。
- `getDescription` - 要求されたロケールの `SI_ML_DESCRIPTION` プロパティを返します。
- `getFiles` - 要求されたロケールの `SI_FILES` プロパティを返します。

`IInfoObject` インターフェイスは、使用できるロケールを判定するためのメソッドも定義しています。

- `getTitleLocales` - `SI_ML_NAME` プロパティに対して使用できるロケールを返します。
- `getDescriptionLocales` - `SI_ML_DESCRIPTION` プロパティに対して使用できるロケールを返します。
- `getFileLocales` - `SI_FILES` プロパティに対して使用できるロケールを返します。

InfoObject の多言語プロパティを設定するための API はありません。これは、翻訳状態がトランスレーションマネジメントツールによって管理されるためです。このツールは、翻訳する必要があるコンテンツにフラグを付け、他の翻訳管理タスクを実行します。この SDK を使用して、多言語環境内の InfoObject のソース言語プロパティを変更する場合は、トランスレーションマネジメントツールを再実行して InfoObject の多言語プロパティを更新する必要があります。

ロケールフォールバックメカニズム

特定の言語ロケール (en_US、en_UK、en_CA など) の情報を要求すると、システムは、そのロケールでローカライズされたコンテンツを提供しようとします。ただし、要求されたロケールで使用可能なコンテンツがない場合、システムは、フォールバックアルゴリズムの決定に従い、要求されたコンテンツを別のロケール (en_UK ではなく en_US など) またはフォールバック言語で返すか、元のコンテンツを返します。

各 InfoObject には、コンテンツのロケールを決定する方法を指定するための LocaleOption 値が定義されています。LocaleOption 値は、次のいずれかです。

- DEFAULT - 要求されたロケールでコンテンツが返されます。要求されたロケールで使用可能なコンテンツがない場合は、null 値が返されます。
- FALLBACK - 要求されたロケールまたはそれに最も近い使用可能なロケールがフォールバックアルゴリズムによって決定されて、コンテンツが返されます。

各プラグインには、要求された言語でコンテンツがない場合に使用するフォールバック言語を定義する SI_FALLBACK_LOCALE プロパティが含まれます。LocaleOption 値を FALLBACK に設定した場合、システムは次の順序でローカライズされたコンテンツを検索します。

1. 要求された言語とロケール、または要求された言語の別のロケール。
2. フォールバック言語とロケール、またはフォールバック言語の別のロケール。
3. 翻訳されていない元のコンテンツ。

たとえば、システムに de_DE の元のコンテンツがあり、それが ja_JP に翻訳されているとします。SI_FALLBACK_LOCALE プロパティが en_US に設定されており、ユーザーが fr_FR のコンテンツを要求した場合は、以下の順序でコンテンツが検索されます。

- 要求されたロケール fr_FR のコンテンツを検索しますが、見つかりません。
- 要求された言語の任意のロケール fr_* のコンテンツを検索しますが、見つかりません。
- フォールバックロケール en_US のコンテンツを検索しますが、見つかりません。
- フォールバック言語の任意のロケール en_* のコンテンツを検索しますが、見つかりません。
- 元の翻訳されていない de_DE のコンテンツを返します。

3.6 この SDK のベストプラクティス

このセクションでは、アプリケーションの効率を向上させるコーディング方法について説明します。

3.6.1 未使用のセッションとリソースのクリーンアップ

常に最大数のライセンスを活用できるように、セッションが不要になったユーザーはログオフするようにします。アクティブなセッションには、それぞれライセンスが必要です。使用できるライセンスの最大数を超えると、それ以上のユーザーは BI platform にログオンできなくなります。

そのうえ、Session に保存されている変数すべてを破棄してサーバーの資源を解放するのは有益なことです。IInfoStore オブジェクトを含む、アプリケーションサーバーのセッションオブジェクト内のすべての変数を破棄する Invalidate メソッドを使用します。

この例では、既にログオンし、IEnterpriseSession オブジェクトが保存されているとします。

例

```
IEnterpriseSession enterpriseSession =
(IEnterpriseSession)session.getAttribute("MyCESession");
session.removeAttribute("MyCESession");
enterpriseSession.logoff();
session.invalidate();
```

3.6.2 IEnterpriseSession オブジェクトの格納

レポートを表示する場合は、ビューアに渡される IEnterpriseSession オブジェクトをセッションオブジェクトに保存する必要があります。その後のレポートビューアの要求では、保存されている IEnterpriseSession オブジェクトを使用します。そうしないと、ビューアの要求があるたびに新しい印刷ジョブが作成され、システム全体のパフォーマンスに影響を及ぼします。

3.6.3 サポートされるインターフェイスの判別

オブジェクトのサポートされるインターフェイスは、リフレクションによって判別できます。通常は、開発段階でリフレクションを使用して、サポートされるインターフェイスを判別します。この方法では、呼び出すメソッドが常に同じタイプのオブジェクトを返す場合に、どの IInfoObject インスタンスまたは Object インスタンスにキャストする必要があるかを判別できます。

リフレクション

この SDK には、戻り値のタイプが Object オブジェクトまたは IInfoObject オブジェクトであるメソッドが数多くあります。そのオブジェクトがサポートするインターフェイスを特定するには、リフレクションを使用する必要があります。リフレクションを使用すると、オブジェクトがサポートするインターフェイスを判別できます。これにより、特定されたタイプにオブジェクトを手動でキャストできます。戻り値のタイプが常に同じ場合は、リフレクションを使用することをお勧めします。これは、主に特定の ProgId、SI_ID、または SI_NAME を使用して照会する場合です。

次の例は、リフレクションを使用して特定のタイプの IInfoObject インスタンス (この場合は IUserGroup オブジェクト) を判別する方法を示しています。

```
import java.lang.reflect.*;
.
.
.
IInfoObjects objs = iStore.query("SELECT SI_NAME FROM CI_SYSTEMOBJECTS "
    + "WHERE SI_KIND='UserGroup'");
IInfoObject obj = (IInfoObject) objs.get(0);
```

```
Class[] interfaces = obj.getClass().getInterfaces();
for( int i = 0; i < interfaces.length; i++ )
{
    out.println( interfaces[i].getName() );
}
```

3.6.4 SDK 例外のキャッチおよび処理

エラーが発生すると、SDK はさまざまな例外をスローします。これらの例外をアプリケーションでチェックして、ユーザーのためにさまざまな状況に対処してください。

SDKException クラス

SDKException は、SDK によってスローされたすべての例外をカプセル化する基本クラスです。SDKException には、SDK がエラーに遭遇したときにスローされる数多くのサブクラスが含まれます。アプリケーションの柔軟性を高めるには、サブクラス例外を最初にキャッチして処理します。

たとえば、ドキュメントをスケジュールできるアプリケーションがあったとして、ユーザーがこのようなアクションの実行に必要な権限を持っていない場合があります。この操作中に、SDK によって SDKException.NoRight 例外がスローされます。この例外をキャッチして処理することができます。

```
try {
    ...
    infoStore.schedule(infoObjects);
}
catch (SDKException.NoRight e) {
    ...
}
catch (SDKException e){
    ...
}
```

この例では、SDKException.NoRight がスローされたかどうかをチェックした後で、最終的により一般的な SDKException クラスをキャッチします。

すべての SDK 例外の完全な一覧については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスで com.crystaldecisions.sdk.exception.SDKException クラスを参照してください。

SDKServerException クラス

SDKServerException は、Central Management Server（CMS）への要求時に発生するエラーを区分けするために使用される、SDKException の特別なサブクラスです。このようなエラーには、IInfoStore インターフェイスを使用して BI platform リポジトリ内のオブジェクトと対話する際に発生する問題、認証ワークフローで発生するセキュリティエラーなどがあります。

SDKServerException クラスは、発生した特定のサーバーエラーに関するさまざまな情報をキャプチャします。この情報には、getErrorCodeString メソッドを使用して取得できるエラーコード文字列が含まれます。このエラーコードを使用して、サーバーエラーをアプリケーションで適切に処理できます。

たとえば、シングルサインオンソリューションとして「信頼できる認証」を使用するアプリケーションがあるとします。Web サーバー上の共有シークレットの有効期限が切れると、CMS では、エラーコード `FWB 00022` のエラーが発生し、SDK は `SDKServerException` 例外をスローします。例外をキャッチしたら、より一般的な例外処理ロジックを呼び出す前に、適切な方法でエラー `FWB 00022` を処理できます。

```
try {
    ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
    ITrustedPrincipal trustedPrincipal =
sessionMgr.createTrustedPrincipal(username, cms);
    IEnterpriseSession enterpriseSession = sessionMgr.logon(trustedPrincipal);
}
catch (SDKServerException e) {
    if (e.getErrorCodeString() == "FWB 00022")
    {
        ...
    }
    else
    {
        ...
    }
}
```

すべての BI platform サーバーエラーコードの完全な一覧については、エラーメッセージの説明ガイドの「“BI Platform サーバー（FWB）のエラーメッセージ”」のセクションを参照してください。

SDKRuntimeException クラス

`SDKRuntimeException` クラスは、SDK で発生したランタイムエラーをキャプチャします。通常、これらの例外をアプリケーションでキャッチして処理する必要はありません。

すべての SDK ランタイム例外の完全な一覧については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスで `com.crystaldecisions.sdk.exception.SDKRuntimeException` クラスを参照してください。

3.6.5 一貫性ある CMS の状態の維持

ベスト プラクティスとして、複数の CMS クエリーが必要な操作は、CMS の一貫性を保つために必ず 1 つのトランザクションで実行します。

1 つのトランザクションで複数の InfoObject を CMS にコミットするには、以下の手順を実行します。

1. 新しい InfoObject をメモリ内に作成します。
2. InfoObject にファイルを追加します。
3. InfoObject をコミットします。

次の例は、InfoObject に 2 つのフォルダに追加し、それを CMS にコミットします。

```
void createInfoObject(IEnterpriseSession enterpriseSession, String newIn
foObjectName) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoS
tore");
  IInfoObjects newFolders = infostore.newInfoObjectCollection();
```

```
  IFolder newFolder1 = (IFolder) newFolders.add(IFolder1.FOLDER_KIND);
  IFolder newFolder2 = (IFolder) newFolders.add(IFolder2.FOLDER_KIND);
  infostore.commit(newFolders);
}
```

複数のデータベーストランザクションが必要な場合は、次のように、CMS の外部にある一時ストレージを使用することで一貫性を維持できます。

1. ユーザーからアクセスできないプライベートリポジトリ領域にある一時ストレージに InfoObject を作成します。
2. データベーストランザクションを実行します。たとえば、Crystal レポートのパラメータを更新します。
3. 一時ストレージ領域から CMS 内のアクセス可能なフォルダに InfoObject をコピーします。

関連情報

InfoObject を作成する [19 ページ]

3.7 水平または垂直クラスタでのオブジェクトのシリアル化

水平クラスタ

水平クラスタは、1 台の仮想サーバーとしてブラウザクライアントに公開される複数のサーバーから成るクラスタです。水平クラスタは、アプリケーションのスケーラビリティ、パフォーマンス、信頼度の向上に役立ちます。

垂直クラスタ

垂直クラスタは、水平クラスタに似ていますが、相互に接続された複数のサーバーでなく、複数の CPU が組み込まれた 1 台のマシンを使用します。垂直クラスタは、作業を複数のプロセスに分配するため、マルチプロセッサコンピュータのスケーラビリティの向上に役立ちます。各プロセスは異なる CPU 上で実行されます。

シリアル化の要件

水平および垂直クラスタは、ユーザーがグループ内のどのマシンにもアクセスできるように設定されることがよくあります。この設定では、ユーザーセッション情報をすべてのマシンで共有する必要があります。

ユーザーセッション情報を共有するために、アウトプロセスセッション状態サーバーに情報が転送されます。アウトプロセスセッション状態サーバーに格納されるオブジェクトは、シリアル化可能である必要があります。ただし、すべてのオブジェクトがシリアル化可能である必要はありません。オブジェクトがシリアル化可能でない場合は、別の方法でオブジェクトの永続性を管理する必要があります。

次のクラスがシリアル化可能です。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects

状態管理

状態管理とは、Web ページの状態情報を保持する方法です。状態管理の主なアプローチは、Session とシリアル化された文字列です。

セッション

Session を使用すると、Web サーバーのメモリにオブジェクトを保存することで、ユーザーのセッション時間中のあらゆるオブジェクトを永続できます。

一般に Session は、次のことを実行するために使用されます。

- ユーザーのセッション時間全体で、状態の永続が必要な情報 (ユーザーが Web アプリケーションを移動するのに必要なログオン情報など) を保存します。
- ページの再ロード、または機能的にグループ化されたページのセットなど、状態の永続が必要なオブジェクトを保存します。

永続化のためにセッション変数を使用する利点は、アクセスのある Web サーバーに関するユーザーの状態情報を、どの時点のどのページからでも永続化できることです。

次のコードは、IInfoObject を取得し、それを HttpSession の属性に保存します。

```
String reportQuery = "SELECT * FROM CI_INFOOBJECTS WHERE SI_ID=" + reportID;
IInfoObjects reports = infoStore.query(reportQuery);
IInfoObject report = reports.get(0);
session.setAttribute("MyReport", report);
```

次に、別の Web ページで、HttpSession からオブジェクトを取得します。

```
IInfoObject report = (IInfoObject) session.getAttribute("MyReport");
```

このメソッドは、IInfoObject、IInfoObjects、IInfoStore、および IIEnterpriseSession オブジェクトを永続化するために使用できます。

シリアル化された文字列

IEnterpriseSession オブジェクトは、シリアル化されたセッション文字列から再構築することもできます。有効なセッションがある場合は、シリアル化されたセッション文字列を次のように取得できます。

```
String serializedSession = enterpriseSession.getSerializedSession();
```

次に、シリアル化された文字列を別の Web ページに渡し（Cookie やフォーム属性などを使用して）、それからセッションを再構築できます。

```
IEnterpriseSession enterpriseSession = sessionMgr.getSession(serializedSession);
```

3.8 同じ CMS を使用したオブジェクトの読み書き

デプロイメントで Central Management Server（CMS）の複数のインスタンスを使用している場合は、同じオブジェクトの読み書きには同じ CMS インスタンスが使用されることを確認してください。

ある CMS インスタンスを使用してオブジェクトを変更した場合は、その CMS インスタンスでそのオブジェクトを更新してからでないと、同じ CMS インスタンスで同じオブジェクトを再度読み取ったり変更することはできません。変更は、システムランドスケープ内の他の CMS インスタンスに伝播されます。一方、1 つのオブジェクトをある CMS インスタンスを使用して変更し、同時に別の CMS インスタンスを使用して読み取る場合は、変更が適用される前に読み取ることができます。

これは、以下のすべての条件が満たされる場合に起こり得ます。

- BI platform デプロイメントで、複数の CMS インスタンスが使用される。
- アプリケーションで複数のセッションが使用される。
- 異なる CMS インスタンスを使用する複数のセッションを使用して、同じオブジェクトに同時またはほとんど同時にアクセスする。

> ⓘ 注記
>
> これは、マルチスレッドアプリケーションでよく発生するシナリオです。

このシナリオを回避するには、同じオブジェクトの読み書きには同じ CMS インスタンスが使用されることを確認してください。特定のオブジェクトには 1 つのセッションを使用してアクセスするか、同じ CMS インスタンスを使用して複数のセッションを使用するかのいずれかの方法があります。

セッションで使用する CMS の検出

セッションから CMS 名を取得することで、セッションで使用する CMS インスタンスを検出できます。たとえば、BI platform Java SDK を使用している場合は、`IEnterpriseSession.getCMSName` メソッドを使用して、セッション名を取得します。BI platform .NET SDK を使用している場合は、`EnterpriseSession.CMSName` プロパティを読み取ります。

> ⓘ 注記
>
> セッションがアクティブである場合でも、セッションの CMS を別の CMS に切り替えることができます。これは、メンテナンスのために CMS を停止する場合などに役立ちます。CMS にアクセスする直前に、セッションがどの CMS を使用するかを確認することをお勧めします。

4 開発環境をセットアップする

SAP BusinessObjects Business Intelligence platform SDK、Report Application Server SDK で開発している場合、または Crystal レポートビューアを使用してレポート表示をアプリケーションに組み込む場合は、インストールおよび設定が必要なコンポーネントが多数あります。SDK 設定を利用するように `web.xml` ファイルを設定し、SDK を利用するように Web アプリケーションで特定の JAR ファイルをデプロイして、前提条件となる特定のソフトウェアをインストールする必要があります。

Java Platform Enterprise Edition (Java EE) Web コンポーネントの開発およびデプロイメントについて、基本的な知識が必要です。

4.1 コンパイルのターゲット JVM を設定する

この SDK を使用する SAP BusinessObjects Business Intelligence Suite 4.x アプリケーションをコンパイルする場合は、JVM 1.5 をターゲットにする必要があります。たとえば、`-target` Java コンパイラオプション（`javac -target 1.5`）を使用するか、JSP コンパイラが JVM 1.5 をターゲットにしていることを確認してください。

> **注記**
>
> Java アプリケーションをコンパイルする際のターゲット JVM バージョンは、製品出荷マトリックスにあるサポートされるアプリケーションサーバーに対してリストされている JVM バージョンとは異なります。

4.2 Web アプリケーションの設定

Web アプリケーションが正しく機能するには、それを正しく設定する必要があります。Web アプリケーションの設定には、ディレクトリ構造の設定、`web.xml` ファイルの作成、必要なサポートファイルおよびライブラリファイルの Web アプリケーションへのコピーが含まれます。

以下の手順では、Web アプリケーションを最初から作成すると仮定します。また、以下の手順を使用して、既存の Web アプリケーションを変更し、ビューア機能を追加することもできます。

4.2.1 Web アプリケーションを設定する

この SDK を使用する Web アプリケーションのデプロイに必要なライブラリとクラスを使用するように Web アプリケーションを設定できます。

1. Web アプリケーションサーバーの Web アプリケーションディレクトリに次のディレクトリ構造を作成します。ここで、`web_application_name` は、Web アプリケーションの名前です。

```
web_application_name
```

```
      WEB-INF
         lib
         classes
      crystalreportviewers
```

> ⓘ 注記
>
> Web アプリケーションディレクトリ (web_application_name) は、Web アプリケーションサーバーのメイン Web アプリケーションディレクトリに置かれている必要があります。Tomcat の場合、通常、このディレクトリの名前は webapps です。これは Web アプリケーションのルートディレクトリであり、Web アプリケーションに必要なすべてのファイルが含まれます。

ディレクトリの内容：

- WEB-INF ディレクトリには、Web アプリケーションに必要なサポートファイルが含まれます。
- WEB-INF¥lib ディレクトリには、Web アプリケーションに必要な JAR ファイルが含まれます。
- crystalreportviewers ディレクトリには、ビューアに必要なサポートファイルが含まれます。

2. Web アプリケーションに必要なサポートファイルとライブラリ JAR ファイルを WEB-INF¥lib ディレクトリにコピーします。
   デプロイする JAR の詳細については、SAP BusinessObjects ソフトウェアのデプロイメントに必要な JAR ファイル [82 ページ]を参照してください。

   > ⓘ 注記
   >
   > java¥lib¥external にある依存 jar ファイルを WEB-INF¥lib にコピーします。WEB-INF¥lib¥external サブディレクトリを作成する必要はありません。

3. Web アプリケーションの web.xml ファイルを作成し、WEB-INF ディレクトリに保存します。または、既存の Web アプリケーションに表示機能を追加する場合は、既存の web.xml ファイルを変更します。
   web.xml ファイルの設定方法については、web.xml ファイルの設定 [75 ページ]を参照してください。

4.2.2 web.xml ファイルの設定

web.xml ファイルは、Java Web アプリケーションで使用される設定を定義します。web.xml ファイルは、Web アプリケーションの WEB-INF ディレクトリに配置する必要があります。このファイルのルート要素は、<web-app> にする必要があります。次の設定を web.xml ファイルに追加できます。

Web.xml 設定参照テーブル

設定値	説明
crystal_document_view [76 ページ]	レポートビューペインのビュータイプを設定します。
crystal_exception_info [77 ページ]	例外情報を表示できます。
crystal_exception_log_file [77 ページ]	例外情報をログに記録できます。

設定値	説明
crystal_image_uri [77 ページ]	Web アプリケーションの crystalreportsviewers ディレクトリの場所を設定します。
crystal_image_use_relative [78 ページ]	Web ページ、アプリケーション、またはサーバーに相対的に crystal_image_uri を解釈するように設定します。
crystal_max_number_parameter_default_values [78 ページ]	リストとして返される値のデフォルトの個数を設定します。
crystal_processing_indicator_delay [78 ページ]	レポート処理指標を有効にします。
crystal_processing_indicator_text [79 ページ]	レポート処理指標によって表示されるテキストを変更します。
crystal_use_asynchronous_requests [79 ページ]	非同期リクエストを有効にします。
CrystalReportViewerServlet [79 ページ]	Web アプリケーションにパラメータパネルを表示し、レポートのエクスポート、インタラクティブパラメータ、画像、およびチャートを処理します。
Faces サーブレット [80 ページ]	Web アプリケーションで JavaServer Faces（JSF）コンポーネントを使用できるようにします。

crystal_document_view

レポートビューペインのビュータイプを設定します。crystal_document_view パラメータは、次のいずれかの値に設定できます。

- printlayout - レポート ページの影を付けてレポートに灰色の背景を表示します。印刷時にレポートのページレイアウトを表示します。
- weblayout - レポートビューペイン全体にレポートを表示します。灰色の背景はありません。

> ⓘ 注記
>
> crystal_document_view パラメータが指定されない場合は、デフォルトで printlayout に設定されます。

```
<context-param>
        <param-name>crystal_document_view</param-name>
        <param-value>weblayout</param-value>
</context-param>
```

crystal_exception_info

例外情報を表示できます。例外情報を表示しない場合でも、crystal_exception_log_file setting を使用すると、例外情報をログに記録できます。crystal_exception_info パラメータは、次のいずれかの値に設定できます。

- short - 付随するスタックトレースなしで例外情報を表示します。
- long - 付随するスタックトレースと共に例外情報を表示します。
- disable - 例外情報を表示しません。ユーザー側で例外を処理する必要があります。

> ⓘ 注記
>
> crystal_exception_info パラメータが指定されない場合は、デフォルトで short に設定されます。

```
<context-param>
        <param-name>crystal_exception_info</param-name>
        <param-value>long</param-value>
</context-param>
```

crystal_exception_log_file

例外情報をログに記録できます。 パラメータ値は、ログファイルの場所を指定します。crystal_exception_info パラメータの設定に関係なく、ログファイルに出力される例外情報の形式は long です。

> ⓘ 注記
>
> デフォルトでは、例外のログは記録されません。

```
<context-param>
        <param-name>crystal_exception_log_file</param-name>
        <param-value>c:¥temp¥webreportingexception.log</param-value>
</context-param>
```

crystal_image_uri

Web アプリケーションの crystalreportsviewers ディレクトリの場所を設定します。Web アプリケーションでレポートビューアを使用している場合は、この設定を web.xml ファイルに入れる必要があります。crystalreportviewers ディレクトリには、ビューアに必要なすべてのファイルが含まれます。

```
<context-param>
         <param-name>crystal_image_uri</param-name>
         <param-value>/web_application_name/crystalreportviewers</param-value>
</context-param>
```

crystal_image_use_relative

Web ページ、アプリケーション、またはサーバーに相対的に crystal_image_uri を解釈するように設定します。 パラメータは、次のいずれかの値に設定できます。

- webapp - Web アプリケーションディレクトリに対して相対的として URI を設定します。
- server - サーバールートに対して相対的として URI を設定します。
- Page - crystalreportviewers ディレクトリにあるリソースをリクエストするページに対して相対的として URI を設定します。

```
<context-param>
        <param-name>crystal_image_use_relative</param-name>
        <param-value>webapp</param-value>
</context-param>
```

crystal_max_number_parameter_default_values

Java DHTML ビューアは、デフォルトでは一度に 1 つのリストあたり最大 200 個の値を返します。 crystal_max_number_parameter_default_values パラメータ値は、このデフォルトサイズを上書きします。リストの内容がデフォルトサイズの 200 またはこのパラメータで設定された数を超える場合は、リストに含まれなかった値へのリンクが表示されます。

```
<context-param>
         <param-name>crystal_max_number_parameter_default_values</param-name>
                  <param-value>100</param-value>
</context-param>
```

crystal_processing_indicator_delay

レポート処理指標は、一般の処理状態と印刷の処理状態の両方に関するフィードバックを示します。デフォルトでは、500 ミリ秒を超えるポストバックアクションに対してレポート処理指標が表示されます。 crystal_processing_indicator_delay パラメータ値は、デフォルトの遅延時間（500 ミリ秒）を上書きします。ミリ秒単位で 0 以上の数値を指定できます。このパラメータ値が 0 に設定された場合は、レポート処理指標が無効になります。

> **注記**
>
> この機能は、Java DHTML ビューアの使用時にのみ動作します。この機能は、現在 Internet Explorer ブラウザにのみ実装されています。

```
<context-param>
        <param-name>crystal_processing_indicator_delay</param-name>
        <param-value>0</param-value>
</context-param>
<context-param>
        <param-name>crystal_processing_indicator_text</param-name>
           <param-value>some text</param-value>
</context-param>
```

crystal_processing_indicator_text

レポート処理指標は、一般の処理状態と印刷の処理状態の両方に関するフィードバックを示します。`crystal_processing_indicator_text` パラメータ値は、レポート処理指標によって表示されるテキストを変更します。

> ⓘ 注記
>
> この機能は、Java DHTML ビューアの使用時にのみ動作します。この機能は、現在 Internet Explorer ブラウザにのみ実装されています。

```
<context-param>
        <param-name>crystal_processing_indicator_delay</param-name>
        <param-value>0</param-value>
</context-param>
<context-param>
        <param-name>crystal_processing_indicator_text</param-name>
          <param-value>some text</param-value>
</context-param>
```

crystal_use_asynchronous_requests

デフォルトでは、DHTML ビューア内で非同期リクエストが許可されます。ユーザーのブラウザの［戻る］ボタンが DHTML ビューアで動作するようにする場合は、このパラメータを `false` に設定して、非同期リクエストを無効にする必要があります。

> ⓘ 注記
>
> `crystal_use_asynchronous_requests` パラメータを `false` に設定しないことをお勧めします。こうすると、ビューアのパフォーマンスと表示を強化する機能の一部が無効になります。

```
<context-param>
    <param-name>crystal_use_asynchronous_requests</param-name>
    <param-value>false</param-value>
</context-param>
```

CrystalReportViewerServlet

CrystalReportViewerServlet を使用すると、Web アプリケーションにパラメータパネルを表示できます。このサーブレットを使用して、他のコンテンツに埋め込まれている Java DHTML ビューアで、エクスポートおよび印刷を行うことができます。さらに、インタラクティブパラメータ、画像、およびチャートも処理します。Web アプリケーションでレポートビューアを使用している場合は、この設定を web.xml ファイルに入れる必要があります。

```
<servlet>
        <servlet-name>CrystalReportViewerServlet</servlet-name>
        <servlet-
class>com.crystaldecisions.report.web.viewer.CrystalReportViewerServlet</servlet-
class>
```

```
</servlet>
<servlet-mapping>
        <servlet-name>CrystalReportViewerServlet</servlet-name>
        <url-pattern>/CrystalReportViewerHandler</url-pattern>
</servlet-mapping>
```

Faces サーブレット

Web アプリケーションで JavaServer Faces（JSF）コンポーネントを使用できるようにします。Web アプリケーションで JSF コンポーネントまたは Bean を使用している場合は、この設定を web.xml ファイルに入れる必要があります。

- `load-on-startup` - この値が 0 に設定されている場合、サーブレットは起動時にロードされません。この値が 1 に設定されている場合、サーブレットは起動時にロードされます。
- `url-pattern` – `Faces` サーブレットのパスを設定します。
- `javax.faces.application.CONFIG_FILES` パラメータ値 – faces-config.xml ファイルの場所を設定します。JSF プロジェクトの設定時に、このファイルを作成する必要があります。詳細は、以下を参照してください。

```
<listener>
     <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
<servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup> 1 </load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.faces</url-pattern>
</servlet-mapping>
<context-param>
        <param-name>javax.faces.application.CONFIG_FILES</param-name>
<param-value>/WEB-INF/faces-config.xml</param-value>
</context-param>
```

例: 基本的な web.xml ファイル

次のパラメータを変更する必要があります。

- `display-name` - Web アプリケーションの名前。
- `crystal_image_uri` パラメータ値 - `crystalreportviewers` フォルダの場所。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
        PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<display-name>web_application_name</display-name>
        <context-param>
                        <param-name>crystal_image_uri</param-name>
                        <param-value>/web_application_name/crystalreportviewers</
param-value>
```

```
            </context-param>
        <context-param>
                    <param-name>crystal_servlet_uri</param-name>
                     <param-value>/CrystalReportViewerHandler</param-value>
        </context-param>
        <servlet>
                       <servlet-name>CrystalReportViewerServlet</servlet-name>
                    <servlet-
class>com.crystaldecisions.report.web.viewer.CrystalReportViewerServlet</servlet-
class>
        </servlet>
        <servlet-mapping>
                    <servlet-name>CrystalReportViewerServlet</servlet-name>
                    <url-pattern>/CrystalReportViewerHandler</url-pattern>
        </servlet-mapping>
</web-app>
```

例: JSF を使用する Web.xml ファイル

次のパラメータを変更する必要があります。

- `display-name` - Web アプリケーションの名前。
- `crystal_image_uri` パラメータ値 - `crystalreportviewers` フォルダの場所。

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
        <display-name>web_application_name</display-name>
        <listener>
                <listener-class>com.sun.faces.config.ConfigureListener</listener-
class>
        </listener>
        <servlet>
                <servlet-name>Faces Servlet</servlet-name>
                <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
                <load-on-startup> 1 </load-on-startup>
        </servlet>
        <servlet-mapping>
                <servlet-name>Faces Servlet</servlet-name>
                <url-pattern>*.faces</url-pattern>
        </servlet-mapping>
        <context-param>
                <param-name>javax.faces.application.CONFIG_FILES</param-name>
                <param-value>/WEB-INF/faces-config.xml</param-value>
        </context-param>
        <context-param>
                <param-name>crystal_image_uri</param-name>
                <param-value>/web_application_name/crystalreportviewers</param-
value>
        </context-param>
        <context-param>
                     <param-name>crystal_servlet_uri</param-name>
                     <param-value>/crystalreportviewerservlet</param-value>
        </context-param>
        <servlet>
                       <servlet-name>CrystalReportViewerServlet</servlet-name>
                     <servlet-
class>com.crystaldecisions.report.web.viewer.CrystalReportViewerServlet</servlet-
class>
        </servlet>
```

```
        <servlet-mapping>
                    <servlet-name>CrystalReportViewerServlet</servlet-name>
                    <url-pattern>/CrystalReportViewerHandler</url-pattern>
        </servlet-mapping>
</web-app>
```

4.2.3 SAP BusinessObjects ソフトウェアのデプロイメントに必要な JAR ファイル

SAP BusinessObjects Business Intelligence プラットフォーム SDK を使用してカスタムアプリケーションを作成したり、用意されているサンプルアプリケーションをデプロイする場合は、使用する JAR ファイルが分かっている必要があります。これらの JAR ファイルを取得するには、SAP BusinessObjects Business Intelligence プラットフォームクライアントツール 4.1 インストーラを実行し、アプリケーションで使用する SDK の開発者用コンポーネントを選択します。すべてのコア JAR ファイルおよび言語リソース JAR ファイルは、以下のディレクトリにインストールされます。

```
C:¥Program Files (x86)¥SAP Business Objects¥SAP BusinessObjects Enterprise XI
4.0¥java¥lib
```

依存 JAR ファイルは、以下のディレクトリにインストールされます。

```
C:¥Program Files (x86)¥SAP Business Objects¥SAP BusinessObjects Enterprise XI
4.0¥java¥lib¥external
```

Web サービス JAR ファイルは、以下のディレクトリに格納されます。

```
C:¥Program Files (x86)¥SAP Business Objects¥ SAP BusinessObjects Enterprise XI
4.0¥dataAccess¥connectionServer¥xml¥drivers¥webservices¥
```

各 SDK の詳細については、http://help.sap.com を参照してください。

多言語アプリケーション開発では、言語リソースを含む特定の JAR ファイルをインストールする必要があります。コマンドの説明やエラーメッセージは、ユーザのデフォルトのシステムロケールに基づいて適切な言語で表示されます。メッセージが正しく表示されるようにするには、アプリケーションに適切な JAR ファイルを含める必要があります。

次の表に、使用可能なさまざまな言語に対応する拡張子をリストします。言語によっては、多言語デプロイメントのためにその他の JAR ファイルが必要になる場合があります。<xx> 拡張子は、次の表に示す適切な拡張子に置き換えてください。

言語の拡張子

英語	en	オランダ語	nl
フランス語	fr	ドイツ語	de
イタリア語	it	スペイン語	es
日本語	ja	簡体字中国語	zh_CN
ロシア語	ru	トルコ語	tr

> **注記**
>
> 各クライアントマシンのロケール設定と一致する言語でメッセージを表示するには、システムロケール設定を現在のユーザアカウントとデフォルトのユーザプロファイルの両方に適用する必要があります。

en、de、es、fr、it、ja、nl、ru、tr、および zh_CN 以外の言語を使用してデプロイする場合は、さらに多くのリソースをアプリケーションに含める必要があります。その他の言語を使用したデプロイメントの場合は、「その他の言語のリソース」、「コア JAR ファイル」、「依存 JAR ファイル」、および「言語リソース」としてリストされているすべての JAR ファイルを含めてください。

BI プラットフォームの管理

> **注記**
>
> SAP BusinessObjects Business Intelligence プラットフォーム Java Server Faces コンポーネントはリリース XI 4.1 で非推奨となり、それ以降のリリースでは使用できなくなります。SAP BusinessObjects Business Intelligence プラットフォーム Java アプリケーションを開発するには、SAP BusinessObjects Business Intelligence プラットフォーム Java SDK または SAP BusinessObjects Business Intelligence プラットフォーム Web サービスコンシューマ Java SDK を使用してください。

	en、de、es、fr、it、ja、nl、ru、tr、zh_CN でのデプロイメント		
SDK	コア **JAR** ファイル	依存 **JAR** ファイル	その他の言語のリソース
SAP BusinessObjects Business Intelligence プラットフォーム Java SDK	• bcm.jar • biarengine.jar • ceaspect.jar • cecore.jar • celib.jar • ceplugins_core.jar • cesession.jar • corbaidl.jar • ebus405.jar • logging.jar • TraceLog.jar	• activation.jar • aspectjrt.jar • axiom-api-1.2.21.jar • axiom-impl-1.2.21.jar • axis2-adb-1.7.9.jar • axis2-kernel-1.7.9.jar • axis2-saaj-1.7.9.jar • com.sap.js.passport.api.jar • commons-logging-1.1.1.jar • derby.jar • freessl201.jar • log4j.jar • sapjce.jar **注記** 4.3 SP2 より、certjFIPS.jar、cryptojFIPS.jar、ssljFIPS.jar、jcmFIPS.jar、および cryptojce.jar が sapjce.jar に置き換わります。追加の DLL と設定 (sapcrypto.dll、slcryptokernel.dll、および slcryptokernel.dll.sha256) が必要になるシナリオがあります。各種のシナリオに対する SAP JCE の設定方法については、SAP ノート 3101582 を参照してください。 • wsdl4j-1.6.2.jar • xmlSchema-core2.2.1.jar **注記** アスタリスクが付いている項目は、Web Tier とバ	• ceresprops_<xx>.jar • cecore_<xx>.jar • celib_<xx>.jar **注記** 前の列でリストした JAR ファイルに加えて、これらのファイルも含めます。「<xx>」を適切な言語コードに置き換えてください。

SDK	en、de、es、fr、it、ja、nl、ru、tr、zh_CN でのデプロイメント		その他の言語のリソース
	コア JAR ファイル	依存 JAR ファイル	
		ックエンドサーバの間、およびバックエンドサーバ間で SSL を使用する際に必要な JAR ファイルを示します。	

SAP Crystal Reports

> **注記**
>
> SAP Crystal Reports デプロイメントには依存 JAR ファイルが既に含まれているため、それらを別途含める必要はありません。

SDK	en、de、es、fr、it、ja、nl、ru、tr、および zh_CN でのデプロイメント	
	コア JAR ファイル	その他の言語のリソース
Report Application Server Java SDK	• CrystalReportsSDK.jar • cereports.jar* **注記** アスタリスクが付いている項目は、Report Application Server Java SDK と BI プラットフォームを統合する際に使用されます。	• CrystalReportsSDK_xx.jar • cereports_xx.jar* **注記** 「<xx>」を適切な言語コードに置き換えてください。アスタリスクが付いている項目は、Report Application Server Java SDK と BI プラットフォームを統合する際に使用されます。

SDK	en、de、es、fr、it、ja、nl、ru、tr、および zh_CN でのデプロイメント コア JAR ファイル	その他の言語のリソース
Viewers Java SDK	• CrystalReportsSDK.jar • log4j.jar • logging.jar • TraceLog.jar • webreporting.jar • sap.com~tc~sec~csi.jar **注記** BI プラットフォームに格納されているレポートを表示するアプリケーションを再配布する場合には、次の jar ファイルを含める必要があります。 • aspectjrt.jar • bcm.jar • ceaspect.jar • cecore.jar • celib.jar • ceplugins_core.jar • cereports.jar • cesession.jar • corbaidl.jar • sapjce.jar **注記** 4.3 SP2 より、certjFIPS.jar、cryptojFIPS.jar、ssljFIPS.jar、jcmFIPS.jar、および cryptojce.jar が sapjce.jar に置き換わります。追加の DLL と設定 (sapcrypto.dll、slcryptokernel.dll、および slcryptokernel.dll.sha256) が必要になるシナリオがあります。各種のシナリオに対する SAP JCE の設定方法については、SAP ノート 3101582 を参照してください。	• webreporting_<xx>.jar **注記** 「<xx>」を適切な言語コードに置き換えてください。

SDK	en、de、es、fr、it、ja、nl、ru、tr、および zh_CN でのデプロイメント コア JAR ファイル	その他の言語のリソース
	• CrystalReportsSDK.jar • ebus405.jar • icu4j.jar • log4j.jar • logging.jar • ras21sdk.jar • SL_plugins.jar • TraceLog.jar • webreporting.jar • sap.com~tc~sec~csi.jar	

Web サービス

Web サービスに必要な JAR ファイルは、ディレクトリ

```
C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI
4.0¥web services¥en¥dsws_consumer¥data¥dswsjavaApi
```

SDK	en、de、es、fr、it、ja、nl、ru、tr、zh_CN でのデプロイメント コア JAR ファイル	依存 JAR ファイル	その他の言語のリソース
Web サービス Java SDK	• dsws-bicatalog.jar • dsws-biplatform.jar • dsws-common.jar • dsws-common-util.jar • dsws-publish.jar • dsws-reportengine.jar • dsws-session.jar • TraceLog.jar	• axiom-api-1.2.21.jar • axiom-impl-1.2.21.jar • axis2-kernel-1.7.9.jar • axis2-xmlbeans-1.7.9.jar • commons-codec.jar • commons-httpclient-3.1.jar • commons-logging-1.1.1.jar • log4j.jar • mail-1.4.jar • wsdl4j-1.6.2.jar • xmlbeans-2.6.0.jar • xmlSchema-core2.2.1.jar	`

4.3 SAP BusinessObjects Business Intelligence platform 環境の設定

SAP BusinessObjects Business Intelligence platform SDK を使用して、BI platform インストールでレポート、ドキュメント、およびユーザーを管理できます。この SDK を開発およびテストするには、いくつかの必須コンポーネントおよび設定が必要です。

4.3.1 必要な SAP BusinessObjects Business Intelligence platform コンポーネント

インストールが必要なコンポーネント

SAP BusinessObjects Business Intelligence platform SDK を使用するには、開発マシンまたは開発マシンと同じネットワークに SAP BusinessObjects Business Intelligence platform がインストールされている必要があります。BI platform のインストール方法の詳細については、*SAP BusinessObjects Business Intelligence Platform* インストールガイドを参照してください。

実行が必要なコンポーネント

BI platform Web アプリケーションを実行するには、すべてのクライアントコンポーネントとサーバーコンポーネント、つまり Web サーバー、アプリケーションサーバー、および Central Management Server（CMS）が実行されている必要があります。Windows システムで実行する場合、BI platform をデフォルト設定でインストールすると、サーバーコンポーネントはコンピュータサービスとして自動的に実行されます。このサービスを手動でオフにしない限り、コンピュータを起動するたびに、サービスは開始されます。Unix システムでは、インストールが完了すると、サーバーが自動的に起動します。ただし、サーバーはサービスとして実行されません。また、次回のコンピュータへの電源投入時や再起動時には、自動的に起動されません。サーバーを起動および停止する方法については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

> **注記**
>
> さまざまなサーバーコンポーネントを異なるマシンにインストールできます。たとえば、アプリケーションサーバーを Web サーバーとは異なるマシンにインストールできます。この場合、Web アプリケーションを使用する前に、ほかのマシンにあるサーバーコンポーネントが動作していることを確認してください。

4.3.2 SAP BusinessObjects Business Intelligence platform の設定

SAP BusinessObjects Business Intelligence platform Java SDK がデフォルトのディレクトリにインストールされていない場合は、システムが適切なファイルを見つけることができるように、さまざまなパスを指定する必要

があります。パスを指定するには、CeEnterpriseContext クラスを使用します。デフォルトのインストールディレクトリを使用しない場合は、このクラスの 2 つのプロパティを設定する必要があります。次の表に、これらのプロパティおよびその使用方法を示します。

- BOBJ_ENTERPRISE_HOME プロパティは、IReport.refreshProperties() および IServer.manageServer() メソッドを使用する場合にだけ設定する必要があります。
- BOBJ_COMMON プロパティは、IServer.manageServer() メソッドを使用するが、デフォルトのディレクトリに共通ファイルがインストールされていない場合にのみ設定する必要があります。
- Windows 環境を使用する場合、UNIX_BSHELL プロパティを設定する必要はありません。
- これらのプロパティは、ユーザーがログオンする前に設定された場合にのみ有効です。

プロパティ名	説明
BOBJ_ENTERPRISE_HOME	このプロパティを使用して、インストール時に作成された BI platform ディレクトリのパスを設定します。これにより、システムは、Windows ベースのサーバーの開始および停止に使用される servicemanagerjni.dll を検索できます。また、このプロパティは、レポートされた実行可能ファイルも検出し、アプリケーションサーバーが新しいプロセスを作成してレポートオブジェクトを追加またはリフレッシュできるようにします。
BOBJ_COMMON	このプロパティは、共通ファイルがデフォルトのディレクトリにインストールされていない場合にのみ必要です。
UNIX_BSHELL	このプロパティは、Bourne シェルがデフォルトディレクトリにインストールされていない場合にのみ必要です（UNIX のみ）。

4.4 Report Application Server 環境の設定

Report Application Server SDK を使用して、プログラムで Crystal Report ファイルを作成および変更できます。この SDK を開発してテストするにはいくつかのコンポーネントが必要です。また、レポートの保存およびアクセス方法に応じた考慮も必要です。

4.4.1 必要な Report Application Server のコンポーネント

この節では、Report Application Server SDK を使用する Web アプリケーションを実行および開発するために必要なコンポーネントについて説明します。

インストールが必要なコンポーネント

Report Application Server SDK を使用するには、開発マシンまたは開発マシンと同じネットワークに SAP BusinessObjects Business Intelligence プラットフォーム 4.1 がインストールされている必要があります。
Report Application Server SDK 開発者コンポーネントをインストールするには、SAP BusinessObjects Business Intelligence クライアントツール 4.1 インストーラを使用します。BI プラットフォームのインストール方法の詳細については、*SAP BusinessObjects Business Intelligence* プラットフォームインストールガイドを参照してください。

特定の JAR ファイルを Web アプリケーションにデプロイする必要があります。製品に必要な JAR ファイルの詳細については、SAP BusinessObjects ソフトウェアのデプロイメントに必要な JAR ファイル [82 ページ]を参照してください。

> **注記**
>
> Report Application Server SDK を使用してレポートを作成または変更するには、レポートの表示、最新表示、編集、エクスポートを行う権限のほかに、製品の適切なアクティベーションキーコードが必要です。詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム管理者ガイドを参照してください。

実行が必要なコンポーネント

実行する必要があるコンポーネントは使用のシナリオによって異なります。

- ファイルパスにあるアンマネージドレポートにアクセスする Web アプリケーションを実行するには、Report Application Server や Web サーバなど、クライアントおよびサーバのすべてのコンポーネントが実行されている必要があります。また、Report Application Server と BI プラットフォームを使用して、プログラムからアンマネージドレポートを開くには、いくつかの手順を実行する必要があります。
  - guest アカウントを有効にします。バージョン XI 3.0 では、guest アカウントはデフォルトで無効になっています。`ReportClientDocument.Open` メソッドを呼び出すと、ユーザアカウントが無効であるというエラーメッセージが表示されます。
  - セントラル管理コンソール (CMC) で Report Application Server のコマンドラインパラメータを使用して、`ipport` スイッチを設定し、空いているポート番号を指定します。たとえば、デフォルトのポート 1566 を指定する場合は、`ipport` スイッチを `-ipport "1566"` と指定します。詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム管理者ガイドを参照してください。
  - また、CMC で Report Application Server のコマンドラインパラメータを使用して、表示するレポートを含むディレクトリのフルパスを指定します。たとえば、レポートディレクトリパスを設定する場合は、コマンドラインの先頭に `-reportdirectory "C:¥My Documents¥Reports"` を追加します。このオプションを指定しない場合、Report Application Server は、以下のデフォルトディレクトリでレポートを探します。`C:¥Program Files (x86)¥SAP Business Objects¥SAP BusinessObjects Enterprise XI 4.0¥Samples¥en¥Reports`。
- BI プラットフォームリポジトリにあるマネージドレポートにアクセスする Web アプリケーションを実行するには、Report Application Server、Web サーバ、Web Component Server、CMS など、クライアントおよびサーバのすべてのコンポーネントが実行されている必要があります。
  BI プラットフォームシステムをデフォルト設定でインストールすると、サーバコンポーネントはコンピュータサービスとして自動的に実行されます。このサービスを手動でオフにしない限り、コンピュータの起動時にこれらのサーバが開始されます。サーバを起動および停止する方法については、*SAP BusinessObjects Business Intelligence* プラットフォーム管理者ガイドの「サーバを管理する」を参照してください。

> ⓘ 注記
>
> マシンごとに異なるサーバコンポーネントをインストールできます。たとえば、Web Component Server を Web サーバとは異なるマシンにインストールできます。この場合、他のマシンにあるサーバコンポーネントが動作していることを確認してから、Web アプリケーションを起動します。

サーバを安全に起動および停止する方法については、製品の *SAP BusinessObjects Business Intelligence* プラットフォーム管理者ガイドを参照してください。

4.4.2 Report Application Server の場所を指定する

Report Application Server の場所は、`clientSDKOptions.xml` ファイルの Server 属性で指定されます。Report Application Server SDK を使用してプログラム的にアクセスすることもできます。実行時の Server 属性を設定するには、`ReportAppSession` オブジェクトの `setReportAppServer` メソッドを使用します。

Report Application Server のインストール中に、`clientSDKOptions.xml` ファイルが SDK JAR ファイルのインストールディレクトリに作成されます (詳細については、ディレクトリ [95 ページ]を参照)。接続する Report Application Server が別のコンピュータにインストールされている場合は、このファイルを修正する必要があります。

`clientSDKOptions.xml` ファイルの場所は、このファイルを参照する classpath を設定して静的に指定するか、このファイルの場所を指定して動的に指定することができます。複数の Report Application Server がある場合、負荷分散を有効にするには、`clientSDKOptions.xml` ファイルにネットワーク上のすべての Report Application Server を指定します。詳細は *Windows での Report Application Server インストールを参照してください。*

clientSDKOptions.xml ファイルの場所の静的な指定

`clientSDKOptions.xml` ファイルの場所を静的に指定するには、そのファイルパスを Web アプリケーションサーバーの `CLASSPATH` 環境変数に追加する必要があります。ローカルシステムと Web サーバーの `CLASSPATH` にファイルパスを追加することが必要な場合があります。classpath の追加方法については、使用している Web サーバーのマニュアルを参照してください。

clientSDKOptions.xml ファイルの場所の動的な指定

実行時に `clientSDKOptions.xml` ファイルの場所を指定できます。JSP または Java ファイルで、System クラスの Java メソッド `setProperty` を使用します。`ras.config` キーで示されるシステムプロパティを、次のように指定したディレクトリに設定します。

```
system.setProperty("ras.config","c:¥temp")
```

この関数呼び出しでは、`c:¥temp` にある `clientSDKOptions.xml` ファイルを使用して Report Application Server を見つけるように指示しています。

プログラム全体で clientSDKOptions.xml ファイルの場所のハードコーディングを避けるためには、web.xml ファイル (デフォルトの場所は Web アプリケーションの ¥WEB-INF¥ ディレクトリ) で clientSDKOptions.xml の場所を指定します。

4.4.3 レポートの場所

Report Application Server は、アンマネージドレポートファイルとマネージドレポートファイルの両方にアクセスできます。マネージドアクセスでは、ユーザーは BI platform InfoStore によって管理されるレポートオブジェクトを変更できますが、アンマネージドアクセスではローカル ファイル システムに保存されているレポートファイル (*.rpt) を変更します。レポートを開いて保存する場合、開くレポートと、保存先のフォルダを指定する必要があります。

レポートとフォルダは次のように識別されます。

- BI platform で管理されているレポートオブジェクトを識別するには、InfoObject への参照を指定します。
- Report Application Server を使用してレポートにアクセスするとき、レポートやフォルダを識別するには、ファイルパスを指定します。

Report Application Server にはすべての Report Application Server 配布用にアプリケーションプログラミングインターフェイスが用意されているため、両方のオブジェクト識別方法がサポートされます。たとえば、ReportClientDocument クラスの saveAs メソッドのメソッド署名は、次のとおりです。

```
public void saveAs(java.lang.String displayName, java.lang.Object parentFolder,
int options)
```

2 番目のパラメータはレポートを保存するフォルダを指定します。このパラメータは次のいずれかになります。

- ファイルパスを指定する String オブジェクト。
- BI platform のフォルダを表す InfoObject。

レポートファイルのパスの指定

Report Application Server からファイルパスに基づいてアクセスする Crystal レポート (.rpt) ファイルは、Report Application Server のレポートディレクトリとして指定されているフォルダにある必要があります。Report Application Server から開いたり保存することができるレポートは、このフォルダおよびそのサブフォルダにあるもののみです。これ以外のフォルダに対する書き込みを試みると、アクセス拒否エラーが発生します。

セントラル設定マネージャを使用してレポートディレクトリを設定します。 デフォルトのインストールプロセス時には、サンプルのレポートフォルダがレポートディレクトリとして設定されます。

パフォーマンスを向上させるために、レポートディレクトリとして Report Application Server マシンのローカルフォルダを使用することを強くお勧めします。Report Application Server マシン外にレポートを保存すると、そのレポートにアクセスするたびに、レポートをシリアル化して Report Application Server に送信する処理を行う必要があるため、パフォーマンスが低下します。

> **注記**
>
> ［Report Directory］フィールドにアスタリスク（*）を入力すると、ローカルマシンのすべてのレポートにアクセスできます。

レポートはいくつかの方法で指定できます。

- ras プレフィックスを使用する
  ras プレフィックスは ras://c:¥directory¥reportname.rpt のように使用できます。この文字列は、Report Application Server マシン上の reportname.rpt の場所を示します。

  > ⓘ 注記
  >
  > ras プレフィックスは、デフォルトで付くことが前提となっているので不要です。c:¥reportname.rpt は、Report Application Server マシンへの相対パスと見なされています。

- rassdk プレフィックスを使用する
  rassdk プレフィックスは rassdk://c:¥directory¥reportname.rpt のように使用できます。この文字列は、Report Application Server SDK が実行されているマシン上で reportname.rpt という名前のレポートが保存されている場所を示します。通常このマシンは Web アプリケーションサーバーです。

  > ⓘ 注記
  >
  > パフォーマンス上の理由により、Report Application Server SDK が実行されているマシン、または Report Application Server 以外のマシンにレポートを保存することはお勧めできません。

- プレフィックスを使用しない
  プレフィックスを使用せずにレポートを指定できます。この場合、レポートは Report Application Server のローカルフォルダ上にあるものとみなされます。たとえば、c:¥directory¥reportname.rpt は Report Application Server の C ドライブ上の reportname.rpt の場所です。この形式は、ras:// c:¥directory¥reportname.rpt の指定と同じです。

4.4.4 水平および垂直クラスタリング

Report Application Server は、水平および垂直クラスタリングをサポートします。水平クラスタリングでは、複数の物理マシン上で複数の Java アプリケーションサーバーが実行されている必要があります。一方、垂直クラスタリングは、単一の物理マシン上の複数の Java アプリケーションサーバーから成ります。

ReportClientDocument は、Java アプリケーションサーバーがセッションに格納できるシリアライズ可能なオブジェクトです。1 つの Java アプリケーションサーバーが失敗した場合、他のサーバーがセッションに格納されたオブジェクトを使用して処理を続行できます。ユーザーがこの動作を認識することはありません。Java アプリケーションサーバーの水平または垂直クラスタリングを設定でき、Report Application Server がそのサーバーをサポートしている場合、Report Application Server はクラスタ環境で実行されます。

> ⓘ 注記
>
> 注記：Report Application Server を水平または垂直クラスタ環境で実行するには、ReportClientDocument オブジェクトがセッションに格納されている必要があります。

プラットフォーム

Report Application Server は、複数の Java アプリケーションサーバーをサポートします。詳細については、¥Platforms¥platforms.txt ファイルを参照してください。このファイルは、製品と一緒に配布されます。

環境設定

サーバークラスタリングの設定は、Java アプリケーションサーバーごとに異なります。デプロイメントの手順については、Java アプリケーションサーバーのマニュアルを参照してください。

4.5 SAP Crystal Reports ビューアの設定

SAP Crystal Reports ビューアを使用すると、Java Server Pages（JSP）内でシンクライアントレポート表示機能をカスタマイズして開発およびデプロイできます。これにより、SAP Crystal Reports がインストールされていないクライアントマシンでレポートファイルを表示したりエクスポートすることができます。レポートを取得および処理するためにビューアがやり取りする必要があるコンポーネントがいくつかあります。

4.5.1 必要なビューアコンポーネント

ビューアは、ほかの BI platform ソフトウェアコンポーネントに依存します。これらのコンポーネントについては、ここで説明します。

インストールが必要なコンポーネント

Crystal Reports 2020 Processing Server または Crystal Reports 2020 Report Application Server とともにビューアを使用する場合は、開発マシンまたは開発マシンと同じネットワークのどちらかに SAP BusinessObjects Business Intelligence プラットフォームがインストールされており、Web アプリケーションで適切な JAR ファイルを使用できる必要があります。各製品のインストール方法については、それぞれのインストールガイドを参照してください。

> ⓘ 注記
>
> いずれかのビューアを使用してレポートを表示するには、レポートへの適切なアクセス権限に加えて、製品の適切なアクティベーションキーコードが必要です。詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

実行が必要なコンポーネント

- **SAP BusinessObjects Business Intelligence** プラットフォーム環境
  BI プラットフォーム環境には、Crystal Reports 2020 Processing Server と Crystal Reports 2020 Report Application Server の少なくとも一方がインストールされて実行されている必要があります。必要なサーバーは、実際のデプロイメントと、Web アプリケーションで使用しているビューアによって異なります。
- **Web** アプリケーションサーバー

Web アプリケーションサーバーと Web サーバーも正しく設定され、実行されている必要があります。インストールと設定の詳細については、Web アプリケーションサーバーと Web サーバーに付属のマニュアルを参照してください。

4.5.2 ビューア SDK のサイドバイサイドインストール

1 台のマシンに 2 つの異なるバージョンのビューア SDK をインストールすることができます。このようなインストールを、サイドバイサイドインストールといいます。サイドバイサイドインストールを実現するには、Web サーバーの `WEB-INF¥lib` ディレクトリを使用して、ビューア SDK に必要な JAR ファイルを格納する方法をお勧めします。詳細については、SAP BusinessObjects ソフトウェアのデプロイメントに必要な JAR ファイル [82 ページ]を参照してください。新しいバージョンのビューア SDK をインストールしても、自動的にインストールされる新しい JAR ファイルによって `WEB-INF¥lib` ディレクトリの JAR ファイルが上書きされることはありません。古いバージョンの SDK を維持しておくと、新しいバージョンに移行する準備が整うまでは、Web アプリケーションで引き続き古いライブラリを使用できます。

新しいバージョンに移行するには、Web アプリケーションを適切に変更した後で、新しい JAR ファイルを `WEB-INF¥lib` ディレクトリにコピーします。

4.6 ディレクトリ

さまざまなコンポーネントが次のデフォルトのディレクトリ構造にインストールされます。

製品/コンポーネント	インストールディレクトリ
SAP BusinessObjects Business Intelligence platform メインインストールディレクトリ（Windows）	`C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥`
SAP BusinessObjects Business Intelligence platform メインインストールディレクトリ（Unix）	インストール中に定義されます。この表では `<businessobjects_install_folder>` です。
SAP Crystal Reports ビューアのリソース ファイル	`C:¥Program Files (x86)¥SAP BusinessObjects¥Crystal Reports 14.0¥crystalreportviewers¥`
SAP Crystal Reports ビューアのリソース ファイル	`<businessobjects_install_folder>/crystalreports_xi40/crystalreportviewers/`
SDK JAR ファイル（Windows）	`C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥`

製品/コンポーネント	インストールディレクトリ
SDK JAR ファイル（Unix）	<businessobjects_install_folder>/enterprise_xi40/java/lib/
処理拡張機能（Windows）	C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥win32_x86¥ProcessExt¥
処理拡張機能（Unix）	<businessobjects_install_folder>/enterprise_xi40/processext/

5 SDK の使用

このセクションでは、`InfoObject` モデルの API を使用して、プログラムで Central Management Server を設定し、`InfoObject` を取得および操作する方法について説明します。このセクションに含まれるトピックでは、基本的な考え方を説明し、これらのシナリオをアプリケーションに実装する方法をコード例を使用して具体的に説明します。

5.1 認証

BI platform 認証とは、ユーザーを検証し、Central Management Server（CMS）へのアクセスを許可または拒否する機能です。SDK は、ユーザー認証を実装するための 1 つの認証フレームワークとさまざまなセキュリティプロトコルを備えています。ログイン時に数種類のセキュリティプロトコル（Enterprise、LDAP、WinAD）を使用するように BI platform を設定できます。シングルサインオン（SSO）認証と信頼できる認証を有効にすることもできます。

ユーザー認証時に使用されるクラス

- `com.crystaldecisions.sdk.plugin.authentication.enterprise.IsecEnterprise`
  ネイティブ BI platform ユーザーアカウントのグローバルセキュリティオプションを設定します。
- `com.crystaldecisions.sdk.plugin.authentication.ldap.IsecLDAP`
  LDAP プリンシパル（ユーザーとグループ）を BI platform にマップするためのプロパティとメソッドを提供します。
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
  `IEnterpriseSession` は、最初に取得されるオブジェクトです。他のどの SDK オブジェクトより先に作成されます。このオブジェクトは、ネームサービスとクライアント側セキュリティオブジェクトを結合します。
- `com.crystaldecisions.sdk.framework.ISessionMgr`
  CMS へのアクセスポイントを表します。
- `com.crystaldecisions.sdk.framework.ITrustedPrincipal`
  信頼できる認証を使用して CMS にログオンするために使用されます。ユーザー名とサーバーの共有シークレットが必要です。
- `com.crystaldecisions.sdk.occa.security.IEnterpriseLogonInformation`
  報告されて解決されたホスト名と IP を含む追加のログオン情報を格納します。
- `com.crystaldecisions.sdk.occa.security.ILogonTokenMgr`
  カスタムログオントークンを作成したり、デフォルトのログオントークンを取得するために使用されます。

5.1.1 認証の基本

フレームワーク

次の 2 つの主要なフレームワーククラスが認証を処理します。

- `ISessionMgr` クラス
  `ISessionMgr` クラスは、ユーザー認証情報を受け取り、`IEnterpriseSession` オブジェクトを返す認証時コントローラとして機能します。
- `IEnterpriseSession` クラス
  `IEnterpriseSession` クラスは、ユーザーの現在のセッションを識別する主要な要素です。通常は、このクラスを使用することで、`IEnterpriseSession` インターフェイスに渡される文字列の識別子に基づいて `IInfoStore` インスタンスを取得します。

> **注記**
>
> 他のほとんどのインターフェイスとは異なり、`ISessionMgr` と `IEnterpriseSession` は、`IInfoObject` インターフェイスを継承していません。

クラス	パッケージ
`ISessionMgr`	`com.crystaldecisions.sdk.framework`
`IEnterpriseSession`	`com.crystaldecisions.sdk.framework`

セキュリティプロトコル

ログイン時に次のセキュリティプロトコルを使用できるように BI platform を設定できます。

- `IsecEnterprise` クラス
  このクラスを使用すると、ネイティブの Enterprise アカウントを使用して BI platform にログオンするユーザーについて、パスワード設定とその他のセキュリティオプションを指定できます。
- `IsecLDAP` クラス
  LDAP プロトコルは、LDAP サーバーからのユーザーとグループにアクセスします。
- `IsecWinAD` クラス
  Windows AD プロトコルは、Windows Active Directory サーバーからのユーザーとグループにアクセスします。

これらのクラスを使用すると、グループを外部システムから BI platform にマップできます。これにより、そのグループに属するユーザーは、サードパーティエイリアスを使用して BI platform にログオンできます。また、シングルサインオン（SSO）認証も使用できるようになります。SSO 認証の詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

これらのクラスのそれぞれが `IInfoObject` クラスから継承します。

クラス	パッケージ
IsecEnterprise	com.crystaldecisions.sdk.plugin.authentication.enterprise
IsecLDAP	com.crystaldecisions.sdk.plugin.authentication.ldap
IsecWinAD	com.crystaldecisions.sdk.plugin.authentication.secwinad

5.1.2 CMS へのログオン

この SDK で作業する際は、最初に Central Management Server（CMS）にログオンし、IEnterpriseSession オブジェクトを取得します。SDK と CMS の間のすべての通信は、ユーザーセッションのコンテキストで行われます。これを実行するには、次のような方法があります。

- 基本ログオン
  ユーザー名とパスワードを使用して、ユーザーを CMS にログオンさせます。
- 信頼できる認証
  パスワードを提供せずに、信頼できる Web アプリケーションからユーザーを CMS にログオンさせます。CMS と Web サーバーで共有シークレットを設定する必要があります。
- ログオントークン
  ログオントークンを使用して、ユーザー名またはパスワードなしでユーザーを CMS にログオンさせます。ログオントークンは、既存のユーザーセッションから実行時に作成され、他の Web アプリケーションに渡すことができます。
- シリアライズセッション
  ユーザー名またはパスワードを提供せずに、既存のユーザーセッションにアクセスします。シリアライズセッションは、既存のユーザーセッションから実行時に作成され、他の Web アプリケーションに渡すことができます。
- 監査情報を使用したログオン
  ログオントークンと基本ログオンを使用する方法です。このとき、報告されて解決されたホスト名と IP を含む監査情報を格納する IEnterpriseLogonInformation オブジェクトを渡します。この情報は、CMS に接続しているユーザーのホスト名と IP を示すために監査データベースに格納されます。

5.1.2.1 基本ログオン

Central Management Server（CMS）にログオンする最も基本的な方法は、ISessionMgr.logon メソッドを使用します。このメソッドを使用するには、サーバー情報、ユーザー名、パスワード、および認証の種類を指定する必要があります。このメソッドは、IEnterpriseSession オブジェクトを取得します。

例

```
IEnterpriseSession basicLogon() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession enterpriseSession = sessionManager.logon("username",
"password", "<cms>:<port>", "secEnterprise");
  return enterpriseSession;
}
```

> ⓘ 注記
>
> - **<cms>** を CMS マシンの名前に、**<port>** を CMS のポート番号に置き換えます。
> - **username** を有効なユーザー名に、**password** を対応するパスワードに置き換えます。
> - `ISessionMgr.getInstalledAuthIDs` メソッドを使用して、使用可能なすべての認証タイプのリストを取得します。標準の値には `secEnterprise`、`secLDAP`、`secWinAD` があります。

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.exception.SDKException`

5.1.2.2 ログオントークンの使用

ログオントークンを使用すると、ユーザーに認証情報を要求せずにユーザーセッションを作成できます。たとえば、ユーザーが CMS に正しくログオンした場合は、そのユーザーのセッションからログオントークンを作成して、別の Web アプリケーションに渡すことができます。これにより、ユーザーはユーザー名またはパスワードを入力せずに、その Web アプリケーションにアクセスできます。

ログオントークンには、デフォルトのログオントークンとカスタムのログオントークンの 2 つのタイプがあります。カスタムのログオントークンの場合は、ホスト名、有効期間、およびログオン数を変更できます。デフォルトのログオントークンを変更することはできません。どちらのログオントークンでも新しいユーザーセッションが作成されます。これにより、同時接続ユーザーライセンスシステムではライセンス数が増加します。1 つのユーザーセッションがログオフされても、他のセッションは有効なままです。

> ⓘ 注記
>
> URL を使用してトークンを別の Web アプリケーションに渡す場合は、そのトークンを URL エンコードすることを忘れないでください。

例

デフォルトのログオントークンを作成する。

デフォルトのログオントークンを作成し、そのトークンを使用して 2 つ目のユーザーセッションを作成する方法を示します。enterpriseSesssion2 は、enterpriseSession1 がログオフされた後も有効です。

```
void defaultToken() throws SDKException
{
  IEnterpriseSession enterpriseSession1 =
CrystalEnterprise.getSessionMgr().logon("username", "password", "<cms>:<port>",
"secEnterprise");
  ILogonTokenMgr tokenMgr = enterpriseSession1.getLogonTokenMgr();
  String defaultLogonToken = tokenMgr.getDefaultToken();
  IEnterpriseSession enterpriseSession2 =
CrystalEnterprise.getSessionMgr().logonWithToken(defaultLogonToken);
  enterpriseSession1.logoff();
  String CMSName = enterpriseSession2.getCMSName();
}
```

例

カスタムのログオントークンを作成する。

カスタムのログオントークンを作成し、そのトークンを使用して 2 つ目のユーザーセッションを作成する方法を示します。このログオントークンは、マシン Host1 からのみ使用することができます。120 分間有効になり、10 のログオンで使用できます。enterpriseSesssion2 は、enterpriseSession1 がログオフされた後も有効です。

```
void customToken() throws SDKException
{
  IEnterpriseSession enterpriseSession1 =
CrystalEnterprise.getSessionMgr().logon("username", "password", "<cms>:<port>",
"secEnterprise");
  ILogonTokenMgr tokenMgr = enterpriseSession1.getLogonTokenMgr();
  String customLogonToken = tokenMgr.createLogonToken("Host1",120,10);
  IEnterpriseSession enterpriseSession2 =
CrystalEnterprise.getSessionMgr().logonWithToken(customLogonToken);
  enterpriseSession1.logoff();
  String CMSName = enterpriseSession2.getCMSName();
}
```

ⓘ 注記

- **<cms>** を CMS マシンの名前に、**<port>** を CMS のポート番号に置き換えます。
- **username** を有効なユーザー名に、**password** を対応するパスワードに置き換えます。
- ISessionMgr.getInstalledAuthIDs メソッドを使用して、使用可能な認証タイプのリストを取得します。標準の値には、secEnterprise、secLDAP、および secWinAD があります
- トークンが不要になったら、必ず ILogonTokenMgr.releaseToken メソッドを呼び出します。このメソッドは、ログオントークンの解放に失敗すると SDKException エラーをスローします。このメソッドは、次のように使用できます。

  ```
  tokenMgr.releaseToken(customLogonToken);
  ```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.framework.CrystalEnterprise

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.security.ILogonTokenMgr

関連情報

シリアライズセッションの使用 [102 ページ]

5.1.2.3 シリアライズセッションの使用

シリアライズセッションは、CMS 内のユーザーセッションを表す文字列です。これは、ユーザーに認証情報を要求せずに既存のユーザーセッションにアクセスするために使用されます。ログオントークンとは異なり、シリアライズセッションを使用してもライセンス数は増加しません。ユーザーがシリアライズセッションから作成されたセッションからログオフした場合は、元のセッションもログオフされます。

> **注記**
>
> URL を使用してシリアライズセッションを別の Web アプリケーションに渡す場合は、そのシリアライズセッションを URL エンコードすることを忘れないでください。

例

シリアライズセッションの取得。

既存のユーザーセッションからシリアライズセッションを作成する方法を示します。このシリアライズセッションを別のアプリケーションに渡すことができます。enterpriseSession2 と enterpriseSession1 は同じユーザーセッションを参照するため、enterpriseSession1.logoff() が呼び出されると、enterpriseSession2 はログオフされます。

```
void defaultToken() throws SDKException
{
  IEnterpriseSession enterpriseSession1 =
CrystalEnterprise.getSessionMgr().logon("username", "password", "<cms>:<port>",
"secEnterprise");
  String serializedSession = enterpriseSession1.getSerializedSession();
  IEnterpriseSession enterpriseSession2 =
CrystalEnterprise.getSessionMgr().getSession(serializedSession);
  enterpriseSession1.logoff();
}
```

関連情報

ログオントークンの使用 [100 ページ]

5.1.2.4 信用できる認証の使用

一度システムにログオンしたら、そのセッションの間に何回もパスワードを入力する必要がない方が好まれます。信頼できる認証は、BI platform 認証ソリューションとサードパーティの認証ソリューションを統合する方法として、シングルサインオンソリューション（SSO）を提供します。Central Management Server（CMS）と信頼を確立したアプリケーションは、信頼できる認証を使用して、パスワードを提供せずにログオンできます。

信頼できる認証が既にセントラル管理コンソール（CMC）にある場合は、`ISessionMgr` インターフェイスを使用して、信用できるユーザー名と共有シークレットを CMS に渡してログオンできます。

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
ITrustedPrincipal trustedPrincipal =
sessionMgr.createTrustedPrincipal("userName", "<cms>:<port>", "sharedSecret");
IEnterpriseSession enterpriseSession = sessionMgr.logon(trustedPrincipal);
```

> ⓘ 注記
>
> - **`username`** を有効なユーザー名に置き換えます。また、**`sharedSecret`** を CMS で設定されている信頼できる認証の共有シークレットに置き換えます。
> - **`<cms>`** を CMS マシンの名前に、**`<port>`** を CMS のポート番号に置き換えます。

この一覧には、サンプルコードで使用されるクラスが含まれます。

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.ITrustedPrincipal`

> ⓘ 注記
>
> セキュリティ上の理由から、信用できる認証を HTTPS なしで有効化しないでください。信用できる認証を https なしで有効にすると、URL が認証されていないユーザに公開されるため、セキュリティ侵害とみなされます。セキュリティ侵害を防ぐために、有効な証明書を使用してユーザーの情報を検証できます。詳細については、1388240 を参照してください。

5.1.2.4.1 信用できる認証を設定する

この例では、セントラル管理コンソール（CMC）にログオンできる管理者権限を持つアカウントが必要となります。

この例は、カスタムアプリケーションで信用を確立するために使用できる共有シークレットを CMS で設定する方法を示します。

> ⓘ 注記
>
> セキュリティ上の理由から、信用できる認証を HTTPS なしで有効化しないでください。信用できる認証を https なしで有効にすると、URL が認証されていないユーザに公開されるため、セキュリティ侵害とみなされます。セキュリティ侵害を防ぐために、有効な証明書を使用してユーザーの情報を検証できます。詳細については、1388240 を参照してください。

1. CMC に管理者権限でログオンします。
2. [認証] 管理エリアに移動します。
3. [*Enterprise*] オプションをクリックします。
   [*Enterprise*] ダイアログボックスが開きます。
4. 信頼できる認証が表示されるまで下にスクロールします。
   a. [信頼できる認証を有効にする] をクリックします。
   b. [新規共有シークレット] をクリックします。
      次のメッセージが表示されます。

      共有シークレットキーが生成され、ダウンロードが可能になります。
   c. [共有シークレットのダウンロード] をクリックします。

      > ⓘ 注記
      >
      > 共有シークレットは、信用を確立するために Web サーバーと CMS によって使用されます。

      [ファイルのダウンロード] ダイアログが開きます。
   d. [保存] をクリックし、次のディレクトリを指定して `TrustedPrincipal.conf` ファイルを保存します。

      <INSTALLDIR>¥SAP BusinessObjects Enterprise XI 4.0¥win32_x86
   e. 共有シークレットの有効日数を指定するには、[共有シークレット有効期間] フィールドに値を入力します。
   f. 信頼できる認証の要求のタイムアウト値を指定します。

      > ⓘ 注記
      >
      > タイムアウト値は、クライアントのクロックと CMS のクロックの許容できる最大時間差（ミリ秒）です。0 を入力すると、2 つのクロックの時間差は無制限になります。脆弱性が増して攻撃が繰り返される可能性があるため、この値を 0 に設定することはお奨めしません。
5. [更新] をクリックして、共有シークレットをコミットします。

   > ⓘ 注記
   >
   > SAP BusinessObjects Business Intelligence プラットフォームは、信頼できる認証のパラメータに対する変更のすべてを監査しません。信頼できる認証の情報はすべて手動でバックアップすることをお奨めします。

これで、カスタムアプリケーションで API 呼び出しを使用して、信頼できるユーザー名と共有シークレットを CMS に渡して認証を行うことができます。

5.1.2.5 非 Enterprise ユーザのパスワードの変更

BI プラットフォーム Java SDK から、BW ユーザなどの非 Enterprise ユーザのパスワードを変更する API が提供されます。

> ⓘ 注記
>
> パスワード変更が成功すると、ユーザはすべての現在セッションからログアウトします。

パスワードを変更するには、開発環境で次の手順を実行します。

1. 次の構文を使用して、IsessionMgr クラスオブジェクトインスタンスを作成します。ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
2. 示した引数とともに、IsessionMgr オブジェクトに changePassword メソッドを使用します。sessionMgr.changePassword(“Username”, “Authentication”, “cms”, “oldpassword”, “newPassword”);

5.1.2.6 監査情報を使用したログオン

追加監査情報を渡すには、ISessionMgr.logonEx と ISessionMgr.logonWithTokenEx のいずれかのメソッドを使用します。これらのメソッドは、追加パラメータとして IEnterpriseLogonInformation オブジェクトを受け取ること以外は、ISessionMgr.logon および ISessionMgr.logonWithToken と同じです。このオブジェクトは、報告されて解決されたホスト名と IP を含む監査情報を格納します。これらのメソッドを実行すると、SDK によってログオン監査イベントがトリガーされ、Central Management Server（CMS）に接続しているユーザーのホスト名と IP が監査データベースに格納されます。

例

logonEx を使用するログオン。

```
IEnterpriseSession logonEx() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setReportedHostname("computerName");
  logonInfo.setReportedIP("computerIP");
  IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "<cms>:<port>", "secEnterprise", logonInfo);
  return enterpriseSession;
}
```

例

logonWithTokenEx を使用するログオン。

```
IEnterpriseSession logonWithcustomTokenEx() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession enterpriseSession = sessionManager.logon ("username",
"password", "<cms>:<port>", "secEnterprise");
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setReportedHostname("computerName");
  logonInfo.setReportedIP("computerIP");
  String customLogonToken =
enterpriseSession.getLogonTokenMgr().createLogonToken("",120,100);
  enterpriseSession = sessionManager.logonWithTokenEx(customLogonToken,
logonInfo);
  return enterpriseSession;
}
```

> ⓘ 注記
>
> - **computerName** を接続先コンピュータの名前に、**computerIP** を接続先コンピュータの IP に置き換えます。
> - **<cms>** を CMS マシンの名前に、**<port>** を CMS のポート番号に置き換えます。
> - **username** を有効なユーザー名に、**password** を対応するパスワードに置き換えます。
> - ISessionMgr.getInstalledAuthIDs メソッドを使用して、使用可能なすべての認証タイプのリストを取得します。標準の値には、secEnterprise、secLDAP、および secWinAD があります

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.framework.CrystalEnterprise
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.ISessionMgr
- com.crystaldecisions.sdk.occa.security.IEnterpriseLogonInformation
- com.crystaldecisions.sdk.occa.security.ILogonTokenMgr

関連情報

基本ログオン [99 ページ]
ログオントークンの使用 [100 ページ]

5.1.3 ログオフしてライセンスを解放する

アクティブなセッションには、それぞれライセンスが必要です。使用できるライセンスの最大数に達すると、それ以上のユーザーは BI platform にログオンできなくなります。使用可能な数のライセンスを最大限に活用するには、セッションを必要としないユーザーを常にログオフします。

例

この例では、既にログオンし、IEnterpriseSession オブジェクトが保存されているとします。

```
IEnterpriseSession enterpriseSession =
(IEnterpriseSession)session.getAttribute("MySession");
  session.removeAttribute("MySession");
  enterpriseSession.logoff();
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.framework.CrystalEnterprise

- `com.crystaldecisions.sdk.exception.SDKException`

5.2 拡張認証情報マッピング

BI 4.2.X 以前のリリースでは、管理者は CMC でユーザごとにデータベース認証情報のセットを 1 つのみ保存できました。

この機能を利用するには、管理者がすべての異なるデータベースに対して同じ認証情報を維持する必要があります。BI 4.3 では、データソース参照を使用して、各ユーザに対して複数のデータベース認証情報のセットを保存できます。

データソース参照

管理者は、BI プラットフォームでデータソース参照を作成します。このデータソース参照は、管理者がデータベース認証情報のセットを 1 つ定義するユーザプロパティで使用されます。このデータソース参照は、接続で使用可能な認証のモードである認証情報マッピングの一部として使用されます。認証モードとして認証情報マッピングが選択された場合、管理者には、好みのデータソース参照を選択するオプションが提供されます。同様に、管理者は、BI プラットフォームに接続している複数のデータベースがある場合、複数のデータソース参照を作成し、各ユーザに対して一意の認証情報を定義できます。

データソース参照の詳細については、『Business Intelligence プラットフォーム管理者ガイド』の拡張認証情報マッピングの節を参照してください。

API 使用の例

新しい InfoObject タイプ (=DatasourceReference) が CMC 固有のページで作成され、プロパティは Title および Description になります。

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionManager.logon(CMS_USER, CMS_PAWD,
CMS_HOST, CMS_AUTH);
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

IInfoObjects infoobjects = infostore.newInfoObjectCollection();

IDatasourceReference dr =
(IDatasourceReference)infoobjects.add(IDatasourceReference.KIND);
final String title = "MyNewDatasourceReference";
dr.setTitle(title);
dr.setDescription("My New Datasource Reference");
infostore.commit(infoobjects);
```

特定のデータソースリファレンスのユーザアカウントにユーザ名とパスワードを追加します。

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionManager.logon(CMS_USER, CMS_PAWD,
CMS_HOST, CMS_AUTH);
```

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
IInfoObjects res = infostore.query(SELECT STATIC, RELATIONSHIPS FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='User' AND SI_NAME='user'");
IUser user = (IUser)res.get(0);
res = infostore.query("SELECT STATIC, RELATIONSHIPS FROM CI_SYSTEMOBJECTS WHERE
SI_KIND='DatasourceReference' AND SI_NAME='credMapping'");
IDatasourceReference dr = (IDatasourceReference)res.get(0);
final UserDatasourceReference mapping = new UserDatasourceReference("username",
"password");
user.addDatasourceReference(dr.getID(), dr.getCUID(), mapping);
user.save();
```

UserInfo インスタンスから、データソース参照のユーザ名とパスワードを取得します。

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionManager.logon(CMS_USER, CMS_PAWD,
CMS_HOST, CMS_AUTH);
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
IInfoObjects res = infostore.query("SELECT STATIC, RELATIONSHIPS FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='DatasourceReference' AND SI_NAME='credMapping'");
IDatasourceReference dr = (IDatasourceReference)res.get(0);
UserDatasourceReference mapping =
enterpriseSession.getUserInfo().getUserDatasourceReference(dr.getCUID());
```

CMS クエリの例

すべての DSR の一覧	SELECT STATIC, RELATIONSHIPS FROM CI_SYSTEMOBJECTS WHERE SI_KIND='`DatasourceReference`'
管理者ユーザに関連する DSR の一覧	SELECT SI_ID, SI_KIND, SI_NAME, FROM CI_SYSTEMOBJECTS WHERE CHILDREN("SI_NAME="`User-DatasourceReference`", "SI_ID='12")
デフォルト DSR に関連するユーザの一覧 (レガシー互換用)	SELECT SI_ID, SI_KIND, SI_NAME, SI_DATA FROM CI_SYSTEMOBJECTS WHERE PARENTS("SI_NAME="`User-DatasourceReference`", "SI_CUID='AWRx3R5vojJLgqMKj9X5qhI")

デフォルト DSR のハードコーディングされた CUID は 'AWRx3R5vojJLgqMKj9X5qhI' です。

5.3 セキュリティ

SAP BusinessObjects Business Intelligence platform は、グループ内のユーザーアカウントおよびユーザーメンバーシップを管理するためのセキュリティフレームワークを提供します。アプリケーション、サーバー、およびアクションへの十分な権限をユーザーに付与することで、これらのオブジェクトへのアクセスを制御できます。

このセクションでは、SAP BusinessObjects Business Intelligence platform SDK を使用して、ユーザーおよびオブジェクトのセキュリティ設定をプログラムで管理する方法について説明します。

ユーザーおよびオブジェクトセキュリティを管理するためのクラス

- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
  ユーザーの動作を管理するためのメソッドおよびプロパティを提供します。
- `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup`
  ユーザーグループを管理するためのメソッドおよびプロパティを提供します。
- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
  ユーザーおよびユーザーグループに 1 つのオブジェクトに対するロール、権限、および制限を割り当てることができます。
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr`
  ユーザーおよびユーザーグループに複数のオブジェクトに対するロール、権限、および制限を割り当てることができます。このインターフェイスを使用して、効率とパフォーマンスを向上させるために、Central Management Server（CMS）への複数のリクエストをバッチ処理します。
- `com.businessobjects.sdk.plugin.desktop.customrole.ICustomRole`
  アクセスレベルを表します。アクセスレベルは、ユーザーおよびユーザーグループが頻繁に必要とする一連の権限です。アクセスレベルを使用すると、よく使用されるセキュリティ設定をオブジェクトに効率よく統一的に適用できます。

5.3.1 ユーザとグループ

5.3.1.1 ユーザーグループを作成する

ユーザーグループは、`com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup` インターフェイスで表され、ユーザーおよびサブグループに関する情報を含みます。プログラムでは、いくつかの簡単な API 呼び出しによって新しいユーザーグループを構築できます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. `IInfoStore` インターフェイスの `newInfoObjectCollection` メソッドを呼び出して、`IInfoObjects` コレクションを作成します。

   ```
   IInfoObjects newGroups = infostore.newInfoObjectCollection();
   ```

3. コレクションの `add` メソッドを呼び出して、`IUserGroup` オブジェクトを作成します。

   定数 `IUserGroup.KIND` を使用し、結果を `IUserGroup` オブジェクトにキャストします。

   ```
   IUserGroup newUserGroup = (IUserGroup) newGroups.add(IUserGroup.KIND);
   ```

4. 新しいグループのさまざまなプロパティ（名前、説明など）を設定します。

   ```
   newUserGroup.setTitle("New Group Name");
   newUserGroup.setDescription("New group description.");
   ```

5. 新しいユーザーグループコレクションを Central Management Server (CMS) にコミットし直し、変更を保存します。

   ```
   infostore.commit(newGroups);
   ```

新しいユーザーグループが作成されます。CMS からグループを取得するには、適切な SI_KIND オブジェクトタイプ識別子を使用して、CI_SYSTEMOBJECTS テーブルから照会します。

```
IInfoObjects groups = infostore.query("SELECT SI_ID, SI_NAME "
    + "FROM CI_SYSTEMOBJECTS WHERE SI_KIND='UserGroup' AND SI_NAME='New Group
Name'");
IUserGroup group = (IUserGroup) groups.get(0);
```

New Group Name を、検索するユーザーグループの名前に置き換えます。

例

この例では、BI platform ユーザーグループを作成します。

```
void createUserGroup(IEnterpriseSession enterpriseSession, IUser user) throws
SDKException
{
    IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
    IInfoObjects newGroups = infostore.newInfoObjectCollection();

    IUserGroup newUserGroup = (IUserGroup) newGroups.add(IUserGroup.KIND);
    newUserGroup.setTitle("New Group Name");
    newUserGroup.setDescription("New group description.");

    Set usersOfNewGroup = newUserGroup.getUsers();
    usersOfNewGroup.add(user.getID());
    infostore.commit(newGroups);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.usergroup.IUser
- com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup
- java.util.Set

関連情報

ユーザーを作成する [110 ページ]

5.3.1.2 ユーザーを作成する

ユーザーは com.crystaldecisions.sdk.plugin.desktop.user.IUser インターフェイスで表され、セキュリティアカウント情報、グループメンバーシップ、および識別情報を含みます。プログラムでは、いくつかの簡単な API 呼び出しによって新しいユーザーを構築できます。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. IInfoStore インターフェイスの newInfoObjectCollection メソッドを呼び出して、IInfoObjects コレクションを作成します。

```
IInfoObjects newUsers = infostore.newInfoObjectCollection();
```

3. コレクションの add メソッドを呼び出して、IUser オブジェクトを作成します。

   定数 IUser.KIND を使用し、結果を IUser オブジェクトにキャストします。

```
IUser newUser = (IUser) newUsers.add(IUser.KIND);
```

4. 名前、パスワード、接続の種類など、新しいユーザーのさまざまなプロパティを設定します。

> **注記**
>
> setNewPassword は管理者メソッドです。呼び出し元がユーザーのパスワードを変更する権限を持つ必要があります。changePassword メソッドを使用して、既存のパスワードを確認および変更する機会をユーザーに提供します。ユーザーの既存のパスワードを取得するメソッドはありません。

```
newUser.setTitle("username");
newUser.setNewPassword("password");
```

5. このユーザーのライセンス情報を設定します。

> **注記**
>
> 接続の種類は、ユーザーが指定ユーザーライセンス (IUser.NAMED) と同時接続ユーザーライセンス (IUser.CONCURRENT) のどちらを使用するように設定されているかを示します。

```
newUser.setConnection(IUser.CONCURRENT);
```

6. 新しいユーザーコレクションを Central Management Server (CMS) にコミットし直し、変更を保存します。

```
infostore.commit(newUsers);
```

新規ユーザーが作成されます。CMS からユーザーを取得するには、適切な SI_KIND オブジェクトタイプ識別子を使用して、CI_SYSTEMOBJECTS テーブルから照会します。

```
IInfoObjects users = infostore.query("SELECT SI_ID, SI_NAME "
    + "FROM CI_SYSTEMOBJECTS WHERE SI_KIND='User' AND SI_NAME='username'");
IUser user = (IUser) users.get(0);
```

username を、検索するユーザーアカウントの名前に置き換えます。

例

この例では、同時接続ユーザー用の BI platform ユーザーアカウントを作成します。

```
void createUser(IEnterpriseSession enterpriseSession) throws SDKException
{
```

```
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects newUsers = infostore.newInfoObjectCollection();
  IUser newUser = (IUser) newUsers.add(IUser.KIND);

  newUser.setTitle("username");
  newUser.setNewPassword("password");
  newUser.setConnection(IUser.CONCURRENT);
  infostore.commit(newUsers);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeSecurityCUID.LicenseRestriction`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`

関連情報

ユーザーグループを作成する [109 ページ]

5.3.1.3 ユーザーまたはユーザーグループを削除する

ユーザーまたはユーザーグループをプログラムで削除することができます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリからユーザーを取得します。

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_ID From
   CI_SYSTEMOBJECTS "
       + "Where SI_KIND='USER'AND SI_NAME='username'");
   IInfoObject user = (IInfoObject)infoObjects.get(0);
   ```

   `username` を、削除するユーザーアカウントの名前に置き換えます。

3. `IInfoObjects` コレクションの `delete` メソッドを呼び出して、特定のユーザーを削除します。

   ```
   infoObjects.delete(user);
   ```

4. 変更したユーザーコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

   ```
   infostore.commit(infoObjects);
   ```

例

この例では、リポジトリから username という BI platform ユーザーアカウントを削除します。同様の方法でユーザーグループを削除できます。

```
void deleteUser(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID From CI_SYSTEMOBJECTS
Where SI_KIND='USER' AND SI_NAME='username'");
  IInfoObject user = (IInfoObject)infoObjects.get(0);
  infoObjects.delete(user);
  infostore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

関連情報

ユーザーを作成する [110 ページ]
ユーザーグループを作成する [109 ページ]

5.3.2 アクセス権の設定

5.3.2.1 オブジェクトセキュリティの概要

フォルダ、ドキュメント、ユーザー、サーバーなどのタイプに関係なく、BI platform リポジトリ内のオブジェクトは、オブジェクトにアクセスできる人とその人々がそのオブジェクトに実行できるアクションを示す一連のセキュリティ情報を所有します。必要に応じて、`com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2` インターフェイスと `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr` インターフェイスを使用してプログラムで、オブジェクトに対するロール、権限、制限をユーザーとユーザーグループに割り当てることができます。

ここでは、いくつかの主要なセキュリティ用語を定義します。オブジェクトセキュリティの詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

ロール（アクセスレベル）

ロール（アクセスレベル）は、オブジェクトにセットで適用されるセキュリティ設定をあらかじめ定義したものです。このセットには、最も一般的なケースのセキュリティ設定が含まれています。たとえば、販売グループとマーケティンググループに別のロールを適用できます。販売グループのロールが適用されると、グループメンバーには、マーケティンググループとは異なるオブジェクトへのアクセス権が付与されます。

BI platform には、いくつかの定義済みのロールが含まれています。カスタムロール（カスタムアクセスレベル）を作成したり、ロールごとに権限を付与するか拒否するかを指定することもできます。

ユーザーやグループにロールを割り当てずに、権限を割り当てることもできます。ただし、ベストプラクティスとしては、最初にロールを割り当て、必要に応じて細かな権限を割り当てていくことをお勧めします。ロールに定義された権限の設定は、主体に同じ細かい権限を明示的に付与または拒否して上書きできます。

各ロールに付与される細かい権限の一覧については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

権限

権限は、オブジェクトにアクセスしたり、オブジェクトに対して操作を実行するための許可です。たとえば、レポートに対する「オブジェクトの削除権限」が付与されているユーザーは、システムからレポートを削除することを許可されます。

次の 2 種類の権限があります。

1. 論理値権限
   許可または拒否のいずれかを表す権限。
2. 制限権限
   ある操作を実行する機能を制限する値。たとえば、レポートのインスタンスの最大数です。

権限は、許可、拒否、または指定しないことができます。

BI platform では、Central Management Server（CMS）に認識されている一連のシステム権限を自動的に使用できるようになります。たとえば、`CeSecurityID.Right.VIEW` システム権限を使用して、ユーザーのオブジェクト表示機能を管理できます。さらに、各タイプのオブジェクトには、オブジェクト固有の権限がカスタムセットとして用意されています。たとえば、Crystal レポートには `CeReportRightID.PRINT` 権限が用意されています。この権限は、ユーザーがレポートをプリンタにスケジュールできるようにします。

利用可能な権限

使用可能な権限には、CMS を通して提供されるシステム権限と、オブジェクトのタイプに固有のその他の権限（プラグイン固有権限）があります。

あるオブジェクトタイプに適用されない権限でも、スコープを `CeSecurityOptions.RightScope.DESCENDANTS` に設定して、そのようなオブジェクトに設定できます。この権限は、このオブジェクトの子のうち、適用可能なタイプに対して有効になります。たとえば、`CeSecurityOptions.RightScope.DESCENDANTS` スコープを使用して、フォルダオブジェクトに `CeSecurityID.Right.Schedule` 権限を割り当てることができます。フォルダ自体をスケジュールすること

はできませんが、フォルダにはスケジュール可能なレポートが入っていることがあり、この権限はフォルダ内のすべてのレポートに継承されます。

明示的な権限と継承された権限

権限は、オブジェクトに明示的に割り当てるか、オブジェクトに設定したアクセスレベルから派生させるか、他のオブジェクトから継承することができます。たとえば、フォルダに権限を割り当てると、そのサブフォルダが親フォルダの権限を継承します。また、グループに権限を割り当てると、そのグループに追加されたメンバがグループの権限を継承します。

実効権限

オブジェクトに対する主体の実効権限は、明示的な権限、継承された権限、アクセスレベルから派生した権限のすべての組み合わせです。実効権限を決定するために、BI platform は次のルールを適用します。

- 明示的にまたは継承によって指定されていない権限（許可も拒否もされていない）は拒否される。
- 権限が設定される最も細かいレベルによって、そのアクセスが決定される。つまり、明示的に設定された権限は、継承によって割り当てられた権限より常に優先します。さらに、親オブジェクトから継承された権限は、継承階層のさらに上から継承された権限より優先します。たとえば、ユーザーは親フォルダへのアクセスが拒否されるが、そのフォルダにあるオブジェクトへのアクセスが明示的に許可されることがあります。
- ユーザーやグループに明示的に設定された権限が、同じオブジェクトに明示的に設定された別の権限と矛盾する場合は、権限が拒否される。たとえば、ユーザーがアクセスレベル A と B に属するとします。アクセスレベル A があるオブジェクトに対する権限を許可し、アクセスレベル B がそのオブジェクトに対する権限を拒否する場合、ユーザーはアクセスを拒否されます。

> **注記**
>
> BusinessObjects Enterprise XI 3.x より前は、継承された権限と明示的な権限が競合した場合、権限の継承が無効でない限り、実効権限は常に拒否されました。

次の表は、実効権限の計算方法を示します。

明示的な親オブジェクト/グループの権限	明示的な子の権限	実効権限
許可	指定なし	許可
指定なし	許可	許可
指定なし	指定なし	拒否
拒否	許可	許可
許可	拒否	拒否

オーナー権限

オブジェクトに対して特定のユーザーのオーナー権限を設定できます。たとえば、`CeSecurityID.Right.OWNER_DELETE` は、ユーザーがオーナーである場合にのみ、オブジェクトインスタンスの削除権限を付与します。

制限

制限は、システム内でオブジェクトインスタンスがクリーンアップされる方法を自動化する特定のタイプの権限です。他の権限はオブジェクトに対する権限が主体に付与または拒否されるかどうかを指定する論理値で管理されますが、制限は整数値で管理されます。システム上に残すオブジェクトやユーザー/グループのインスタンスの数をオブジェクトレベルで制限することができます。また、ユーザー/グループのインスタンスをシステム上に残す日数を制限することもできます。他の権限と同様に、制限にも実効と明示があります。

主体

主体は、特定のオブジェクトに対する権限、ロール、制限を割り当てられたユーザーまたはユーザーグループです。主体は、明示または実効のいずれかです。

- 明示主体は、オブジェクトへの特定のタイプのアクセス (権限、ロール、または制限) を具体的に割り当てられたユーザーまたはユーザーグループです。
- 実効主体とは、明示的な権限および親オブジェクトから継承された権限の両方に基づいて計算された、オブジェクトに対するアクセスを持つユーザーまたはユーザーグループです。
  たとえば、オブジェクトに対する明示的な権限を持たないユーザーが明示的な権限を持つユーザーグループに所属することがあります。この場合、そのユーザーはそのオブジェクトの実効主体になります。

関連情報

セキュリティ API の選択 [116 ページ]

5.3.2.2 セキュリティ API の選択

権限のチェックおよび設定は、アプリケーションのパフォーマンスに影響を及ぼす可能性があります。どのようにセキュリティ情報にアクセスするか、アプリケーションのパフォーマンスを維持するためにどのアプローチを使用できるかは、選択する API によって決まります。SDK には、オブジェクトセキュリティへのエントリポイントが 2 つあります。

- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
  このインターフェイスを使用して、1 つのオブジェクトに対してセキュリティのクエリーおよび変更を行います。

- com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr
  このインターフェイスを使用して、複数のオブジェクトの複数のセキュリティクエリーをバッチ処理します。

次のセクションでは、各インターフェイスの使用方法およびパフォーマンスの向上について詳しく説明します。

ISecurityInfo2 インターフェイス

ISecurityInfo2 インターフェイスは、IInfoObject オブジェクトから取得されます。したがって、1 つのオブジェクトの権限のチェックおよび設定に適しています。

オブジェクトに対する主体の権限をチェックするには、ネットワーク経由で Central Management Server（CMS）に複数の呼び出しを行う必要があります。パフォーマンスを改善するには、以下の点を考慮してください。

- checkRight メソッドを複数回呼び出すのではく、checkRights メソッドを使用して、1 回の呼び出しで複数の権限をチェックします。
- 権限をチェックする際は、使用できるパラメータ値がないか、最初に SDK キャッシュをチェックします。すべての checkRight および checkRights メソッドには、キャッシュされたセキュリティ情報があれば、リポジトリを照会する前にそれを使用することを指定するための論理値パラメータがあります。

ISecurityInfo2 インターフェイスを使用してセキュリティ情報を取得するための一般的なワークフローは次のとおりです。

1. ドキュメント、フォルダ、ユーザー、サーバーなどのオブジェクトを取得します。
2. IInfoObject インターフェイスから継承された getSecurityInfo2 メソッドを使用して、ISecurityInfo2 オブジェクトを取得します。
3. ISecurityInfo2 インターフェイスの getEffectivePrincipals または getExplicitPrincipals メソッドを使用して、そのオブジェクトの主体のリストを取得します。
   明示主体とは、特定のオブジェクトに割り当てられた明示的な権限を持つユーザーまたはユーザーグループです。実効主体とは、明示的な権限および親オブジェクトから継承された権限の両方に基づいて計算された、オブジェクトに対する権限を持つユーザーまたはユーザーグループです。これは実効権限と呼ばれます。
4. 個別の主体の権限、ロール、または制限のリストを取得します。
5. 個別の権限、ロール、または制限の情報を取得します。

ISecurityInfoMgr インターフェイス

ISecurityInfoMgr インターフェイスを使用して、複数のセキュリティクエリーを 1 回の呼び出しにバッチ処理します。セキュリティクエリーをバッチ処理すると、CMS へのネットワークトリップ数は減少します。ISecurityInfoMgr インターフェイスのもう 1 つの利点は、ユーザーに表示権限が付与されていないオブジェクトの権限をチェックする方法を提供することです。ISecurityInfoMgr インターフェイスを使用してセキュリティ情報を取得するための一般的なワークフローは次のとおりです。

1. IEnterpriseSession.getSecurityInfoMgr メソッドを使用して、現在のセッションから ISecurityInfoMgr オブジェクトを取得します。
2. 複数のオブジェクトの一連のセキュリティクエリーをバッチ処理します。
   - ISecurityInfoMgr.getSecCache メソッドを使用して、ISecCacheController オブジェクトを返します。ISecCacheController.batch メソッドは、ユーザーセッションが権限を持つオブジェクトに対する一連の非管理者セキュリティクエリーをバッチ処理するために使用されます。

- `ISecurityInfoMgr.getSecCacheAdmin` メソッドを使用して、`ISecCacheControllerAdmin` オブジェクトを返します。`ISecCacheControllerAdmin.batch` メソッドは、複数のオブジェクトに対する一連の管理者セキュリティクエリーをバッチ処理するために使用されます。ユーザーセッションがセキュリティ情報を照会したり設定するには、これらのオブジェクトに対する適切な管理者権限を持つ必要があります。

3. バッチ処理されたクエリーを CMS にコミットします。結果は、SDK キャッシュにロードされます。`ISecCacheController.commit` または `ISecCacheControllerAdmin.commit` メソッドを使用します。

> **注記**
>
> バッチおよびコミット処理中にスローされたエラーおよび例外をキャッチする必要があります。`rollback` メソッドを使用して、バッチ状態を復元します。

4. SDK キャッシュからバッチ処理されたクエリーの結果を取得します。
   - `ISecurityInfoMgr.getRights` メソッドを使用して、非管理者クエリーの結果を含む `ISecRights` オブジェクトを返します。
   - `ISecurityInfoMgr.getRightsAdmin` メソッドを使用して、管理者クエリーの結果を含む `ISecRightsAdmin` オブジェクトを返します (`ISecRightsQueryAdmin` から継承されたメソッド)。

> **注記**
>
> SDK キャッシュは 1 分後に期限が切れます。したがって、CMS への余分なラウンドトリップを回避するため、クエリー結果を含むキャッシュをロードしたら、キャッシュの期限が切れる前にその情報を取得する必要があります。また、キャッシュサイズは使用可能なメモリ量で制限されるため、キャッシュ内のアイテムは必要になるメモリ量に基づいて解放されます。

セキュリティ情報のクエリー以外にも、`ISecRightsAdmin` インターフェイスを使用すると、`batch` メソッド、さまざまな set および remove メソッド、および `commit` メソッドを使用して、複数のオブジェクトのセキュリティの変更をバッチ処理でコミットすることができます。

キャッシュからセキュリティ情報を取得するには、正しい結果を含むキャッシュを事前にロードするための呼び出しを行う必要があります。いくつかの情報は `ISecurityResult` オブジェクトとして取得されるため、その `getResult` メソッドの値を正しいインターフェイスにキャストする必要があります。次の表に、ロードから取得までの手順に関連するメソッド呼び出しと、結果オブジェクトのキャスト先として正しいインターフェイスを一覧します。

ISecCacheController（キャッシュロード）	ISecRights（取得）	Cast ISecurityResult.getResult As
cacheLimit	checkLimit	ILimitResult
cacheRight	checkRight	適用なし。checkRight は、整数を返します。

ISecCacheControllerAdmin（キャッシュロード）	ISecRightsQueryAdmin（取得）	Cast ISecurityResult.getResult As
cacheExplicitLimits	getExplicitLimits	ISecurityLimitAdmin[]
cacheExplicitPrincipals	getExplicitPrincipals	ISecurityPrincipal[]
cacheExplicitRights	getExplicitRights	ISecurityRightAdmin[]
cacheExplicitRoles	getExplicitRoles	ISecurityRoleAdmin[]
cacheKnownRightsByPlugin	getKnownRightsByPlugin	IPluginBasedRightIDs

ISecCacheControllerAdmin（キャッシュロード）	ISecRightsQueryAdmin（取得）	Cast ISecurityResult.getResult As
cacheKnownSecurityInfo	getKnownRoles	IRoleID[]
cacheLimit	checkLimit	ILimitResult
cachePrincipals	getPrincipals	ISecurityPrincipal[]
cacheRight	checkRight	適用なし。checkRight は、整数を返します。
cacheSecurityInfo(int)	getSecurityInfo(int)	ISecurityInfoResult
cacheSecurityInfo(int, int)	getSecurityInfo(int, int)	ISecurityInfoResult

関連情報

1 つのオブジェクトに対する権限の設定 [120 ページ]
複数のオブジェクトに対する権限の設定 [129 ページ]

5.3.2.3 タイプ固有の権限

BI platform オブジェクトには、ユーザーおよびユーザーグループに付与できる一連の権限が含まれます。ISecurityInfo2 または ISecRightsQueryAdmin (ISecurityInfoMgr 経由) インターフェイスの getKnownRightsByPlugin メソッドと checkRight メソッドを組み合わせて使用して、特定のユーザーまたはユーザーグループにオブジェクトの権限が付与されているかどうかを確認できます。getKnownRightsByPlugin メソッドは、オブジェクトタイプ固有のカテゴリ (com.crystaldecisions.sdk.plugin.desktop.pdf.IPDF.PROGID などの ProgID 値) に応じて使用可能な権限をまとめる IPluginBasedRightIDs オブジェクトを返します。

たとえば、フォルダオブジェクトの権限をチェックするとします。IPluginBasedRightIDs インターフェイスでは、たとえば、以下のカテゴリごとに権限がまとめられます。

- Any
  フォルダまたはフォルダ内のすべてのオブジェクトに適用できる一般的な権限。
- CrystalEnterprise.Folder
  フォルダまたはすべてのサブフォルダに適用できる権限。
- CrystalEnterprise.Pdf
  フォルダに含まれるすべての PDF ファイルに適用できる権限。

すべてのオブジェクトに適用されるタイプ固有の権限。たとえば、Crystal レポートファイルの権限を取得した後、取得した IPluginBasedRightIDs オブジェクトには、次のようなカテゴリが含まれます。

- Any - レポートまたはレポートのすべての子インスタンスに適用できる一般的な権限。
- CrystalEnterprise.Report - レポートまたはレポートのすべての子インスタンスに適用できる権限。
- CrystalEnterprise.Pdf - レポートのすべての子 PDF インスタンスに適用できる権限。

IPluginBasedRightIDs オブジェクトを取得したら、すべてのオブジェクトタイプを反復処理して、権限の要素を表す IRightID オブジェクトを取得します。これらの要素を使用して、権限を表す RightDescriptor オブジェクトを構築し、次に、このオブジェクトの checkRight メソッドを呼び出して、権限が付与されているかどうかを確認します。

> ⓘ 注記
>
> タイプ固有の権限の詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

関連情報

オブジェクトの権限をチェックする [124 ページ]

5.3.2.4 1 つのオブジェクトに対する権限の設定

com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2 インターフェイスを使用して、ユーザーおよびユーザーグループに 1 つのオブジェクトに対するロール、権限、制限を割り当てることができます。ここでは、このインターフェイスを使用して一般的なワークフローをプログラムする方法を示します。

関連情報

複数のオブジェクトに対する権限の設定 [129 ページ]

5.3.2.4.1 オブジェクトのセキュリティ情報を取得する

1 つのオブジェクトのセキュリティ情報には、com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2 インターフェイスを使用してアクセスできます。

次の例では、Crystal レポートファイルの実効ロール、権限、および制限を持つ各ユーザーまたはユーザーグループを反復処理して、セキュリティ情報を出力します。

> ⓘ 注記
>
> 複数のオブジェクトの一連のセキュリティクエリーを 1 つの CMS 呼び出しにバッチ処理する場合は、ISecurityInfoMgr インターフェイスを使用し、キャッシュから結果を取得します。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからレポートを取得します。

```
IInfoObjects infoObjects = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_NAME='World Sales Report' And SI_INSTANCE=0");
IInfoObject report = (IInfoObject)infoObjects.get(0);
```

3. getSecurityInfo2 メソッドを呼び出して、レポートの ISecurityInfo2 オブジェクトを取得します。

```
ISecurityInfo2 securityInfo = report.getSecurityInfo2();
```

4. getEffectivePrincipals メソッドを呼び出して、レポートの実効権限を持つすべてのユーザーおよびユーザーグループを取得します。

```
IEffectivePrincipals effectivePrincipals =
securityInfo.getEffectivePrincipals();
```

5. 反復子を使用して、IEffectivePrincipals コレクションに含まれる個別の主体にアクセスします。

```
Iterator it = effectivePrincipals.iterator();
while (it.hasNext())
{
    IEffectivePrincipal effectivePrincipal = (IEffectivePrincipal)it.next();
    ...
}
```

6. レポートに割り当てられている各主体のすべての実効ロール（アクセスレベル）を取得します。
   a. getRoles メソッドを呼び出して、実効ロールのコレクションを取得します。

   ```
   IEffectiveRoles effectiveRoles = effectivePrincipal.getRoles();
   ```

   b. 反復子を使用して、個別のロールを取得します。

   1 つのロールを取得したら、IEffectiveRole インターフェイスの個別のアクセッサメソッドを使用して、ロールに関する目的の情報にアクセスします。

   ```
   Iterator roleIT = effectiveRoles.iterator();
   while (roleIT.hasNext())
   {
       IEffectiveRole effectiveRole = (IEffectiveRole)roleIT.next();
       ...
   }
   ```

7. レポートに割り当てられている各主体のすべての実効権限を取得します。
   a. getRights メソッドを呼び出して、実効権限のコレクションを取得します。

   ```
   IEffectiveRights effectiveRights = effectivePrincipal.getRights();
   ```

   b. 反復子を使用して、個別の権限を取得します。

   1 つの権限を取得したら、IEffectiveRight インターフェイスの個別のアクセッサメソッドを使用して、権限に関する情報にアクセスします。

   ```
   Iterator rightIT = effectiveRights.iterator();
   while (rightIT.hasNext())
   {
       IEffectiveRight effectiveRight = (IEffectiveRight)rightIT.next();
       ...
   }
   ```

8. レポートに割り当てられている各主体のすべての有効な制限を取得します。

a. getlimits メソッドを呼び出して、有効な制限のコレクションを取得します。

```
IEffectiveLimits effectiveLimits = effectivePrincipal.getLimits();
```

b. 反復子を使用して、個別の制限を取得します。

1 つの制限を取得したら、IEffectiveLimit インターフェイスの個別のアクセッサメソッドを使用して、制限に関する情報にアクセスします。

```
Iterator limitIT = effectiveLimits.iterator();
while (limitIT.hasNext())
{
    IEffectiveLimit effectiveLimit = (IEffectiveLimit)limitIT.next();
    ...
}
```

例

この例では、レポートの実効セキュリティ情報を取得し、結果を HTML テーブルに表示する String オブジェクトを返します。

```
String viewEffectiveSecurityInfo(IEnterpriseSession enterpriseSession) throws
SDKException, IOException
{
  String resultString = "";

  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID From CI_INFOOBJECTS
Where SI_NAME='World Sales Report' And SI_INSTANCE=0");


  IInfoObject report = (IInfoObject)infoObjects.get(0);
  ISecurityInfo2 securityInfo = report.getSecurityInfo2();
  IEffectivePrincipals effectivePrincipals =
securityInfo.getEffectivePrincipals();
  Iterator it = effectivePrincipals.iterator();
  while (it.hasNext())
  {
    IEffectivePrincipal effectivePrincipal = (IEffectivePrincipal)it.next();
    IEffectiveRoles effectiveRoles = effectivePrincipal.getRoles();
    Iterator roleIT = effectiveRoles.iterator();

    resultString += "<b>Effective Principal: </b>" +
effectivePrincipal.getName() + "<br>";
    resultString += "<b>Effective Roles:</b>";

    resultString += "<table  border='1'><tr><th>getTitle</th><th>isInherited</
th></tr>";
    while (roleIT.hasNext())
    {
      IEffectiveRole effectiveRole = (IEffectiveRole)roleIT.next();
      resultString += "<tr><td>" + effectiveRole.getTitle() + "</td><td>" +
effectiveRole.isInherited() + "</td></tr>";
    }
    resultString += "</table>";

    IEffectiveRights effectiveRights = effectivePrincipal.getRights();
    Iterator rightIT = effectiveRights.iterator();

    resultString += "<b>Effective Rights:</b>";
    resultString += "<table border='1'><tr><th>getDescription</
th><th>isInherited</th></tr>";
```

```
    while (rightIT.hasNext())
    {
      IEffectiveRight effectiveRight = (IEffectiveRight)rightIT.next();
      resultString += "<tr><td>" +
effectiveRight.getDescription(java.util.Locale.ENGLISH) + "</td><td>" +
effectiveRight.isInherited() + "</td></tr>";
    }
    resultString += "</table>";

    IEffectiveLimits effectiveLimits = effectivePrincipal.getLimits();
    Iterator limitIT = effectiveLimits.iterator();

    resultString += "<b>Effective Limits:</b>";
    resultString += "<table border='1'><tr><th>getDescription</th><th>getValue</
th></tr>";
    while (limitIT.hasNext())
    {
      IEffectiveLimit effectiveLimit = (IEffectiveLimit)limitIT.next();
      resultString += "<tr><td>" +
effectiveLimit.getDescription(java.util.Locale.ENGLISH) + "</td><td>" +
effectiveLimit.getValue() + "</td></tr>";
    }
    resultString += "</table> <hr>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IEffectiveLimit
- com.crystaldecisions.sdk.occa.infostore.IEffectiveLimits
- com.crystaldecisions.sdk.occa.infostore.IEffectivePrincipal
- com.crystaldecisions.sdk.occa.infostore.IEffectivePrincipals
- com.crystaldecisions.sdk.occa.infostore.IEffectiveRight
- com.crystaldecisions.sdk.occa.infostore.IEffectiveRights
- com.crystaldecisions.sdk.occa.infostore.IEffectiveRole
- com.crystaldecisions.sdk.occa.infostore.IEffectiveRoles
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2
- java.io.IOException
- java.util.Iterator

関連情報

オブジェクトセキュリティの概要 [113 ページ]
複数のオブジェクトのセキュリティ情報を取得する [129 ページ]

5.3.2.4.2 オブジェクトの権限をチェックする

この例では、レポートに対するすべての権限を反復処理し、各権限がユーザーに付与されているかどうかをチェックし、その結果を出力します。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからレポートを取得します。

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_NAME='World Sales Report' And SI_INSTANCE=0");
IInfoObject report = (IInfoObject)reports.get(0);
```

3. getSecurityInfo2 メソッドを呼び出して、レポートの ISecurityInfo2 オブジェクトを取得します。

```
ISecurityInfo2 securityInfo = report.getSecurityInfo2();
```

4. getKnownRightsByPlugin メソッドを呼び出して、IPluginBasedRightIDs オブジェクトを取得します。

```
IPluginBasedRightIDs pluginBasedRightIDs =
securityInfo.getKnownRightsByPlugin();
```

5. IPluginBasedRightIDs オブジェクトの getPluginRights メソッドを呼び出して、タイプ固有の権限セットをそれぞれ反復処理します。

   getPluginRights メソッドは java.util.Map オブジェクトを返します。Map オブジェクトのキーは、さまざまなオブジェクトタイプを表す ProgID 文字列 (com.crystaldecisions.sdk.plugin.desktop.pdf.IPDF.PROGID) です。これらの値は、個別のタイプ固有の権限に対する IRightID オブジェクトを含む java.util.Set オブジェクトです。

```
java.util.Map pluginRights = pluginBasedRightIDs.getPluginRights();
java.util.Iterator it = pluginRights.keySet().iterator();
while (it.hasNext())
{
    Object progID = it.next();
    ...
}
```

6. タイプごとに IRightID オブジェクトのセットを取得し、そのセットを反復処理します。

```
java.util.Set rightIDs = (java.util.Set)pluginRights.get(progID);
java.util.Iterator rIt = rightIDs.iterator();

while (rIt.hasNext())
{
    IRightID key = (IRightID)rIt.next();
    ...
}
```

7. 各 IRightID オブジェクトに対して、チェックする権限を表す新しい RightDescriptor オブジェクトを構築します。

各 IRightID オブジェクトには、権限の基本 ID、権限が属するオブジェクトタイプ、および権限がオブジェクトの所有者に適用されるかどうかを示すフラグが含まれます。この情報は、RightDescriptor コンストラクタへの入力として使用されます。

```
RightDescriptor rightToCheck = new RightDescriptor(key.getBaseID(),
key.getRightPluginKind(), key.isOwner(),
CeSecurityOptions.RightScope.CURRENT_OBJECT, progID);
```

コンストラクタの 4 番目のパラメータは、スコープを指定します。スコープは、権限が割り当てられるレベルです。CeSecurityOptions.RightScope.CURRENT_OBJECT は、権限が現在のオブジェクト（権限が設定されているオブジェクト）に適用されることを示し、CeSecurityOptions.RightScope.DESCENDANTS は、権限が現在のオブジェクトの子孫に適用されることを示します。最後のパラメータは、権限が適用されるオブジェクトのタイプを指定します。たとえば、フォルダに割り当てられ、new RightDescriptor(CeSecurityID.Right.View, null, false, CeSecurityOptions.RightScope.DESCENDANTS, IReport.KIND) として構築される権限は、そのフォルダに含まれるすべての Crystal レポートオブジェクトだけに適用されます。

8. 新しく構築された RightDescriptor オブジェクトとユーザーの ID を入力として使用し、checkRight メソッドを呼び出します。

```
securityInfo.checkRight(rightToCheck, user.getID(), true)
```

例

この例では、レポートに設定できるすべての権限を反復処理し、特定のユーザーにこれらの権限が付与されているかどうかをチェックします。結果は、HTML テーブル形式で作成され、String オブジェクトとして返されます。

```
String checkRights(IEnterpriseSession enterpriseSession, IUser user) throws
SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String resultString = "Rights for user '" + user.getTitle() + "' for 'World
Sales Report'.<p>";

  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS Where
SI_NAME='World Sales Report' And SI_INSTANCE=0");
  IInfoObject report = (IInfoObject)reports.get(0);
  ISecurityInfo2 securityInfo = report.getSecurityInfo2();

  IPluginBasedRightIDs pluginBasedRightIDs =
securityInfo.getKnownRightsByPlugin();
  Map pluginRights = pluginBasedRightIDs.getPluginRights();
  Iterator it = pluginRights.keySet().iterator();

  while (it.hasNext())
  {
    Object progID = it.next();
    Set rightIDs = (java.util.Set)pluginRights.get(progID);
    Iterator rIt = rightIDs.iterator();

    resultString += "<p><p><b>Plugin Type: </b>" + progID.toString() + "<br>";
    resultString += "<table><tr><td><b>Right</b></td><td><b>Is granted</b></td></
tr>";
    while (rIt.hasNext())
    {
      IRightID key = (IRightID)rIt.next();
```

```
      RightDescriptor rightToCheck = new RightDescriptor(key.getBaseID(),
key.getRightPluginKind(), key.isOwner(),
CeSecurityOptions.RightScope.CURRENT_OBJECT, progID);
      boolean hasRight = securityInfo.checkRight(rightToCheck, user.getID(),
true);
      resultString += "<tr><td>" + key.toString()+ "</td><td>" + hasRight + "</
td></tr>";
    }
    resultString += "</table>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IPluginBasedRightIDs
- com.crystaldecisions.sdk.occa.infostore.IRightID
- com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2
- com.crystaldecisions.sdk.occa.infostore.RightDescriptor
- com.crystaldecisions.sdk.occa.security.CeSecurityOptions
- com.crystaldecisions.sdk.plugin.desktop.user.IUser
- java.io.IOException
- java.util.Iterator
- java.util.Map
- java.util.Set

関連情報

タイプ固有の権限 [119 ページ]
オブジェクトセキュリティの概要 [113 ページ]
複数のオブジェクトに対する権限をチェックする [134 ページ]
1 つのオブジェクトの明示的な権限を設定する [126 ページ]

5.3.2.4.3 1 つのオブジェクトの明示的な権限を設定する

アクセスレベル (ロール) は、多くのユーザーおよびグループにオブジェクトに対する共通の権限セットを適用するために使用されますが、ユーザーおよびグループにオブジェクトに対する個別の権限を明示的に割り当てることもできます。

この例では、Crystal Reports のデータを最新表示するための権限を明示的にユーザーに付与します。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからレポートを取得します。

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_NAME='World Sales Report' And SI_INSTANCE=0");
IInfoObject report = (IInfoObject)reports.get(0);
```

3. getSecurityInfo2 メソッドを呼び出して、レポートの ISecurityInfo2 オブジェクトを取得します。

```
ISecurityInfo2 securityInfo = report.getSecurityInfo2();
```

4. レポートの明示主体のコレクションにユーザーを追加します。

> ⓘ 注記
>
> この例では、既にユーザーアカウントを取得しているとします。

```
IExplicitPrincipal explicitPrincipal = explicitPrincipals.add(user.getID());
```

5. オンデマンドでレポートデータを最新表示する権限を表す RightDescriptor オブジェクトを作成します。

   CeReportRightID インターフェイスには、Crystal Reports セキュリティ権限を指定する定数フィールド値が含まれます。CeSecurityID.Right インターフェイスは、一般的なオブジェクト権限を定義します。他のオブジェクトタイプに固有の権限は、CeWebIRightID、CeFullClientRightID、CeUniverseRightID などのインターフェイスで指定されます。

```
RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
```

> ⓘ 注記
>
> このコンストラクタを使用して作成された権限は、1 つのオブジェクトに CeSecurityOptions.RightScope.CURRENT_OBJECT スコープと CeSecurityOptions.RightScope.DESCENDANTS スコープの 2 つの権限を追加します。このコンストラクタは、権限の設定にのみ使用します。権限情報を取得するには、パラメータとしてスコープおよび適用可能なオブジェクトタイプも要求する RightDescriptor コンストラクタを使用する必要があります。

6. 新しく作成した権限を主体の明示的な権限のコレクションに追加します。

```
IExplicitRights reportRights = explicitPrincipal.getRights();
IExplicitRight reportRight = reportRights.add(refreshReportRight);
```

7. setGranted メソッドを使用して、ユーザーに権限を付与します。

```
reportRight.setGranted(true);
```

8. 変更したレポートコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(reports);
```

> ⓘ 注記
>
> 変更されて CMS にコミットし直されるのは、ユーザーアカウントではなくレポートです。権限は、レポートやフォルダなどのオブジェクトにアクセスするユーザーおよびグループではなく、それらのオブジェクトに設定されます。

例

この例では、ユーザーに `World Sales Report.rpt` という名前のレポートのデータを最新表示するための明示的な権限を付与します。

```
void setExplicitRight(IEnterpriseSession enterpriseSession, IUser user) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS Where
SI_NAME='World Sales Report' And SI_INSTANCE=0");
  IInfoObject report = (IInfoObject)reports.get(0);

  ISecurityInfo2 securityInfo = report.getSecurityInfo2();
  IExplicitPrincipals explicitPrincipals = securityInfo.getExplicitPrincipals();
  IExplicitPrincipal explicitPrincipal = explicitPrincipals.add(user.getID());

  RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
  IExplicitRights reportRights = explicitPrincipal.getRights();
  IExplicitRight reportRight = reportRights.add(refreshReportRight);
  reportRight.setGranted(true);
  infostore.commit(reports);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitPrincipal`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitPrincipals`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitRight`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitRights`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`
- `com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`

関連情報

オブジェクトセキュリティの概要 [113 ページ]
複数のオブジェクトの明示的な権限を設定する [137 ページ]
オブジェクトの権限をチェックする [124 ページ]

5.3.2.5 複数のオブジェクトに対する権限の設定

`com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr` インターフェイスを使用して、ユーザーおよびユーザーグループに複数のオブジェクトに対する役割、権限、制限を割り当てることができます。このセクションでは、このインターフェイスを使用して、効率とパフォーマンスを向上させるために、Central Management Server（CMS）への複数のリクエストをバッチ処理する方法を示します。

関連情報

1 つのオブジェクトに対する権限の設定 [120 ページ]

5.3.2.5.1 複数のオブジェクトのセキュリティ情報を取得する

複数のオブジェクトの一連のセキュリティクエリーを 1 つの CMS 呼び出しにバッチ処理する場合は、`com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr` インターフェイスを使用し、キャッシュから結果を取得します。

次の例では、特定のユーザーのフォルダ内にあるすべての Crystal レポートに関する実効セキュリティ情報 (ロール、権限、制限) を取得します。結果は HTML テーブルで表示されます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. `getSecurityInfoMgr` メソッドを呼び出して、セッションの `ISecurityInfoMgr` オブジェクトを取得します。

```
ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
```

3. リポジトリからフォルダ内のすべてのレポートを取得します。

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID() + "
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");
```

4. 権限をチェックするための一連のクエリーをバッチ処理し、それらを CMS にコミットします。

a. ISecurityInfoMgr インターフェイスの getSecCacheAdmin メソッドを呼び出して、ISecCacheControllerAdmin オブジェクトを取得します。

```
ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
```

ISecCacheControllerAdmin インターフェイスは、複数のセキュリティ要求をバッチ処理し、その結果を SDK キャッシュにロードするために使用されます。

b. batch メソッドを呼び出してバッチ処理を開始します。

```
secCacheControllerAdmin.batch();
```

c. エラーが発生した場合にすべてのバッチクエリーをロールバックするための try ブロックを作成します。

```
try {
    ...
}
catch (Throwable e) {
    secCacheControllerAdmin.rollback();
    ...
}
```

d. フォルダ内の各レポートを反復処理して、cacheSecurityInfo メソッドを呼び出します。

```
for (Object object : reports)
{
    IInfoObject report = (IInfoObject)object;
    secCacheControllerAdmin.cacheSecurityInfo(report.getID(),
user.getID());
}
```

cacheSecurityInfo メソッドは、特定の主体であるユーザーのオブジェクトに対するすべての実効セキュリティ情報 (ロール、権限、制限) を要求するために使用されます。メソッドを呼び出すたびに追加の要求がバッチ処理されますが、CMS への要求は送信されません。

e. commit メソッドを呼び出して、クエリーを送信し、その結果を SDK キャッシュにロードします。

```
secCacheControllerAdmin.commit();
```

バッチ処理されたクエリーの結果は、SDK キャッシュにロードされるようになりました。この結果は、ISecurityInfoMgr.getRightsAdmin メソッドを使用して取得できます。

> **注記**
>
> SDK キャッシュは 1 分後に期限が切れます。したがって、CMS への余分なラウンドトリップを回避するため、クエリー結果を含むキャッシュをロードしたら、キャッシュの期限が切れる前にその情報を取得する必要があります。また、キャッシュサイズは使用可能なメモリ量で制限されるため、キャッシュ内のアイテムは必要になるメモリ量に基づいて解放されます。

5. ISecurityInfoMgr インターフェイスの getRightsAdmin メソッドを呼び出して、ISecRightsAdmin オブジェクトを取得します。

```
ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
```

6. レポートごとにステップを繰り返すループを作成します。

```
for (Object object : reports)
{
    IInfoObject report = (IInfoObject)object;
```

```
    ...
}
```

7. レポートごとに、ISecRightsAdmin オブジェクトの getSecurityInfo メソッドを呼び出して、セキュリティクエリーの結果を取得します。

```
ISecurityResult securityResult =
secRightsAdmin.getSecurityInfo(report.getID(), user.getID());
```

結果は、ISecurityResult オブジェクトとして取得されます。このオブジェクトは、クエリー結果を含む汎用 java.lang.Object オブジェクトと要求のステータスを示すフラグの 2 つの部分で構成されます。

8. ISecurityResult オブジェクトの getResult メソッドを呼び出し、結果を ISecurityInfoResult オブジェクトにキャストします。

```
ISecurityInfoResult securityInfoResult =
(ISecurityInfoResult)securityResult.getResult();
```

この例では、ISecurityInfoResult は結果のキャスト先として正しいインターフェイスです。これにより、レポートに対するすべてのロール、権限、制限にアクセスできます。

9. 特定の主体であるユーザーの各レポートに対するすべての実効ロール (アクセスレベル) を取得します。
   a. getRoles メソッドを呼び出して、実効ロールのコレクションを取得します。

```
ISecurityRoleAdmin[] securityRoleAdminArray =
securityInfoResult.getRoles();
```

   b. レポートに対するロールごとにステップを繰り返すループを作成します。

```
for (ISecurityRoleAdmin securityRoleAdmin : securityRoleAdminArray)
{
    ...
}
```

   1 つのロールを取得したら、ISecurityRoleAdmin インターフェイスの個別のアクセッサメソッドを使用して、ロールに関する目的の情報にアクセスします。

10. 特定の主体であるユーザーの各レポートに対するすべての実効権限を取得します。
    a. getRights メソッドを呼び出して、実効権限のコレクションを取得します。

```
ISecurityRightAdmin[] securityRightAdminArray =
securityInfoResult.getRights();
```

    b. レポートに対する権限ごとにステップを繰り返すループを作成します。

```
for (ISecurityRightAdmin securityRightAdmin : securityRightAdminArray)
{
    ...
}
```

    1 つの権限を取得したら、ISecurityRightAdmin インターフェイスの個別のアクセッサメソッドを使用して、権限に関する目的の情報にアクセスします。

11. 特定の主体であるユーザーの各レポートに対するすべての実効制限を取得します。
    a. getLimits メソッドを呼び出して、有効な制限のコレクションを取得します。

```
ISecurityLimitAdmin[] securityLimitAdminArray =
securityInfoResult.getLimits();
```

b. レポートに対する制限ごとにステップを繰り返すループを作成します。

```
for (ISecurityLimitAdmin securityLimitAdmin : securityLimitAdminArray)
{
    ...
}
```

1 つの制限を取得したら、ISecurityRightAdmin インターフェイスの個別のアクセッサメソッドを使用して、制限に関する目的の情報にアクセスします。

例

この例では、特定のユーザーのフォルダ内にあるすべての Crystal レポートに関する実効セキュリティ情報 (ロール、権限、制限) を取得します。結果は、HTML テーブル形式で作成され、String オブジェクトとして返されます。

```
String  viewSecurityInfo(IEnterpriseSession enterpriseSession, IUser user,
IFolder folder) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String resultString = "";
  ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS Where
SI_PARENT_FOLDER=" + folder.getID() + "And SI_KIND='CrystalReport' And
SI_INSTANCE=0");
  ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
  secCacheControllerAdmin.batch();
  try
  {
    for (Object object : reports)
    {
      IInfoObject report = (IInfoObject)object;
      secCacheControllerAdmin.cacheSecurityInfo(report.getID(), user.getID());
    }
    secCacheControllerAdmin.commit();
  }
  catch (Throwable e)
  {
    secCacheControllerAdmin.rollback();
  }

  ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
  for (Object object : reports)
  {
    IInfoObject report = (IInfoObject)object;
    resultString += "<b>Report: <i>" + report.getTitle() + "</i></b><br>";

    ISecurityResult securityResult =
secRightsAdmin.getSecurityInfo(report.getID(), user.getID());
    ISecurityInfoResult securityInfoResult =
(ISecurityInfoResult)securityResult.getResult();
    resultString += "<b>Roles:</b>";
    resultString += "<table border='1'><tr><th>getTitle</
th><th>getPrincipalName</th><th>isInherited</th></tr>";
    ISecurityRoleAdmin[] securityRoleAdminArray = securityInfoResult.getRoles();
    for (ISecurityRoleAdmin securityRoleAdmin : securityRoleAdminArray)
    {
      resultString += "<tr><td>" + securityRoleAdmin.getTitle() + "</td><td>" +
securityRoleAdmin.getPrincipalName() + "</td><td>" +
securityRoleAdmin.isInherited() + "</td></tr>";
```

```
    }
    resultString += "</table>";
    resultString += "<b>Rights:</b><table border='1'><tr><th>getDescription</
th><th>isInherited</th></tr>";
    ISecurityRightAdmin[] securityRightAdminArray =
securityInfoResult.getRights();
    for (ISecurityRightAdmin securityRightAdmin : securityRightAdminArray)
    {
      String rightDescription =
securityRightAdmin.getDescription(java.util.Locale.ENGLISH);
      resultString += "<tr><td>" + rightDescription + "</td><td>" +
securityRightAdmin.isInherited() + "</td></tr>";
    }
    resultString += "</table>";

    resultString += "<b>Limits:</b><table border='1'><tr><th>getDescription</
th><th>getValue</th></tr>";
    ISecurityLimitAdmin[] securityLimitAdminArray =
securityInfoResult.getLimits();
    for (ISecurityLimitAdmin securityLimitAdmin : securityLimitAdminArray)
    {
      resultString += "<tr><td>" +
securityLimitAdmin.getDescription(java.util.Locale.ENGLISH) + "</td><td>" +
securityLimitAdmin.getValue() + "</td></tr>";
    }
    resultString += "</table><hr>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISecurityLimitAdmin
- com.crystaldecisions.sdk.occa.infostore.ISecurityRightAdmin
- com.crystaldecisions.sdk.occa.infostore.ISecurityRoleAdmin
- com.crystaldecisions.sdk.occa.security.ISecCacheControllerAdmin
- com.crystaldecisions.sdk.occa.security.ISecRightsAdmin
- com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr
- com.crystaldecisions.sdk.occa.security.ISecurityInfoResult
- com.crystaldecisions.sdk.occa.security.ISecurityResult
- com.crystaldecisions.sdk.plugin.desktop.folder.IFolder
- com.crystaldecisions.sdk.plugin.desktop.user.IUser
- java.io.IOException

関連情報

オブジェクトセキュリティの概要 [113 ページ]
オブジェクトのセキュリティ情報を取得する [120 ページ]

5.3.2.5.2 複数のオブジェクトに対する権限をチェックする

この例では、あるフォルダ内のすべての Crystal レポートを反復処理して、特定のユーザーが各レポートに対してデータを最新表示する権限があるかどうかをチェック (`CeReportRightID.REFRESH_ON_DEMAND`) し、結果を出力します。この例では、セッションにレポートに対する管理者権限があると仮定し、`ISecurityInfoMgr` インターフェイスを使用して一連のクエリーをバッチ処理し、SDK キャッシュに情報をロードし、その結果を取得します。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. `getSecurityInfoMgr` メソッドを呼び出して、セッションの `ISecurityInfoMgr` オブジェクトを取得します。

```
ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
```

3. リポジトリからフォルダ内のすべてのレポートを取得します。

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID() "
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");
```

4. レポートを最新表示するための権限を表す新しい `RightDescriptor` オブジェクトを構築します。

```
RightDescriptor rightToCheck = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, "CrystalEnterprise.Report");
```

   コンストラクタの 4 番目のパラメータは、スコープを指定します。スコープは、権限が割り当てられるレベルです。`CeSecurityOptions.RightScope.CURRENT_OBJECT` は、権限が現在のオブジェクト（権限が設定されているオブジェクト）に適用されることを示し、`CeSecurityOptions.RightScope.DESCENDANTS` は、権限が現在のオブジェクトの子孫に適用されることを示します。最後のパラメータは、権限が適用されるオブジェクトのタイプを指定します。たとえば、フォルダに割り当てられ、`new RightDescriptor(CeSecurityID.Right.View, null, false, CeSecurityOptions.RightScope.DESCENDANTS, IReport.KIND)` として構築される権限は、そのフォルダに含まれるすべての Crystal レポートオブジェクトだけに適用されます。

5. 権限をチェックするための一連のクエリーをバッチ処理し、それらを CMS にコミットします。
   a. `ISecurityInfoMgr` インターフェイスの `getSecCacheAdmin` メソッドを呼び出して、`ISecCacheControllerAdmin` オブジェクトを取得します。

```
ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
```

   `ISecCacheControllerAdmin` インターフェイスは、複数のセキュリティ要求をバッチ処理し、その結果を SDK キャッシュにロードするために使用されます。

   b. `batch` メソッドを呼び出してバッチ処理を開始します。

```
secCacheControllerAdmin.batch();
```

   c. エラーが発生した場合にすべてのバッチクエリーをロールバックするための `try` ブロックを作成します。

```
try {
    ...
}
```

```
catch (Throwable e) {
    secCacheControllerAdmin.rollback();
    ...
}
```

d. フォルダ内の各レポートを反復処理して、cacheRight メソッドを呼び出します。

```
for (Object object : reports)
{
    IInfoObject report = (IInfoObject)object;
    secCacheControllerAdmin.cacheRight(rightToCheck, user.getID(),
report.getID());
}
```

cacheRight メソッドは、主体である各ユーザーがレポートに対して指定された権限があるかどうかを要求するために使用されます。メソッドを呼び出すたびに追加の要求がバッチ処理されますが、CMS への要求は送信されません。

e. commit メソッドを呼び出して、クエリーを送信し、その結果を SDK キャッシュにロードします。

```
secCacheControllerAdmin.commit();
```

バッチ処理されたクエリーの結果は、SDK キャッシュにロードされるようになりました。この結果は、ISecurityInfoMgr.getRightsAdmin メソッドを使用して取得できます。

> **注記**
>
> SDK キャッシュは 1 分後に期限が切れます。したがって、CMS への余分なラウンドトリップを回避するため、クエリー結果を含むキャッシュをロードしたら、キャッシュの期限が切れる前にその情報を取得する必要があります。また、キャッシュサイズは使用可能なメモリ量で制限されるため、キャッシュ内のアイテムは必要になるメモリ量に基づいて解放されます。

6. SDK キャッシュからバッチ処理されたクエリーの結果を取得します。

a. ISecurityInfoMgr インターフェイスの getRightsAdmin メソッドを呼び出して、ISecRightsAdmin オブジェクトを取得します。

```
ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
```

この例の ISecRightsAdmin インターフェイスは、ロードされた結果を SDK キャッシュから取得するために使用されます。このインターフェイスを使用して、複数の管理セキュリティ変更をバッチ処理し、それらの変更を CMS にコミットすることもできます。

b. フォルダ内の各レポートを反復処理して、checkRight メソッドを呼び出します。

checkRight メソッドは、主体であるユーザーがレポートに対して指定された権限を持つかどうかを示す整数（2=許可、3=拒否）を返します。

```
String resultString = "";
for (Object object : reports)
{
  IInfoObject report = (IInfoObject)object;
  resultString += "<b>Report: </b>" + report.getTitle() + " - <i>User has
right granted? </i>" + secRightsAdmin.checkRight(rightToCheck,
user.getID(), report.getID()) + "<br>";
}
```

例

この例では、フォルダ内のすべての Crystal レポートを反復処理して、特定のユーザーに各レポートのデータ (`CeReportRightID.REFRESH_ON_DEMAND`) を最新表示するための権限が付与されているかどうかをチェックします。結果は、HTML テーブル形式で作成され、String オブジェクトとして返されます。

```
String checkRights(IEnterpriseSession enterpriseSession, IUser user, IFolder
folder) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String resultString = "";

  ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS Where
SI_PARENT_FOLDER=" + folder.getID() + "And SI_KIND='CrystalReport' And
SI_INSTANCE=0");
  RightDescriptor rightToCheck = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, "CrystalEnterprise.Report");
  ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
  secCacheControllerAdmin.batch();
  try
  {
    for (Object object : reports)
    {
      IInfoObject report = (IInfoObject)object;
      secCacheControllerAdmin.cacheRight(rightToCheck, user.getID(),
report.getID());
    }
    secCacheControllerAdmin.commit();
  }
  catch (Throwable e)
  {
    secCacheControllerAdmin.rollback();
  }
  ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
  resultString += "<b>Right: <i>CeReportRightID.REFRESH_ON_DEMAND (granted=2,
denied=3)</i></b><br>";
  for (Object object : reports)
  {
    IInfoObject report = (IInfoObject)object;
    resultString += "<b>Report: </b>" + report.getTitle() + " - <i>User has
right granted? </i>" + secRightsAdmin.checkRight(rightToCheck, user.getID(),
report.getID()) + "<br>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`
- `com.crystaldecisions.sdk.occa.security.CeSecurityOptions`
- `com.crystaldecisions.sdk.occa.security.ISecCacheControllerAdmin`

- com.crystaldecisions.sdk.occa.security.ISecRightsAdmin
- com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr
- com.crystaldecisions.sdk.plugin.desktop.folder.IFolder
- com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID
- com.crystaldecisions.sdk.plugin.desktop.report.IReport
- com.crystaldecisions.sdk.plugin.desktop.user.IUser
- java.io.IOException

関連情報

オブジェクトセキュリティの概要 [113 ページ]
オブジェクトの権限をチェックする [124 ページ]
複数のオブジェクトの明示的な権限を設定する [137 ページ]

5.3.2.5.3 複数のオブジェクトの明示的な権限を設定する

アクセスレベル (ロール) は、多くのユーザーおよびグループにオブジェクトに対する共通の権限セットを適用するために使用されますが、ユーザーおよびグループにオブジェクトに対する個別の権限を明示的に割り当てることもできます。

この例では、フォルダ内のすべての Crystal レポートのデータ (CeReportRightID.REFRESH_ON_DEMAND) を最新表示するための明示的な権限を主体であるユーザーに付与します。この例では、セッションにレポートの管理者権限があるとし、ISecurityInfoMgr インターフェイスを使用して、権限を設定するための一連の操作をバッチ処理し、それらを Central Management Server (CMS) にコミットします。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. getSecurityInfoMgr メソッドを呼び出して、セッションの ISecurityInfoMgr オブジェクトを取得します。

```
ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
```

3. リポジトリからフォルダ内のすべてのレポートを取得します。

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID() "
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");
```

4. レポートを最新表示するための権限を表す新しい RightDescriptor オブジェクトを構築します。

   CeReportRightID インターフェイスには、Crystal Reports セキュリティ権限を指定する定数フィールド値が含まれます。CeSecurityID.Right インターフェイスは、一般的なオブジェクト権限を定義します。他のオブジェクトタイプに固有の権限は、CeWebIRightID、CeFullClientRightID、CeUniverseRightID などのインターフェイスで指定されます。

```
RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
```

> ⓘ 注記
>
> このコンストラクタを使用して作成された権限は、1 つのオブジェクトに `CeSecurityOptions.RightScope.CURRENT_OBJECT` スコープと `CeSecurityOptions.RightScope.DESCENDANTS` スコープの 2 つの権限を追加します。このコンストラクタは、権限の設定にのみ使用します。権限情報を取得するには、パラメータとしてスコープおよび適用可能なオブジェクトタイプも要求する `RightDescriptor` コンストラクタを使用します。

5. 権限を設定するための一連の要求をバッチ処理し、それらを CMS にコミットします。
   a. `ISecurityInfoMgr` インターフェイスの `getRightsAdmin` メソッドを呼び出して、`ISecRightsAdmin` オブジェクトを取得します。

      ```
      ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
      ```

      この例では、`ISecRightsAdmin` インターフェイスを使用して、複数の管理セキュリティ変更をバッチ処理し、それらの変更を CMS にコミットします。このインターフェイスを使用して、SDK キャッシュから読み込まれた結果を取得することもできます。
   b. `batch` メソッドを呼び出してバッチ処理を開始します。

      ```
      secRightsAdmin.batch();
      ```

   c. エラーが発生した場合にすべてのバッチ変更をロールバックするための `try` ブロックを作成します。

      ```
      try {
          ...
      }
      catch (Throwable e) {
          secRightsAdmin.rollback();
      }
      ```

   d. フォルダ内の各レポートを反復処理して、`setRight` メソッドを呼び出します。

      4 番目のパラメータは、`RightDescriptor` オブジェクトで記述された権限を主体に付与または拒否するかどうかを示す論理値です。

      ```
      for (Object object : reports)
      {
          IInfoObject report = (IInfoObject)object;
          secRightsAdmin.setRight(refreshReportRight, user.getID(),
      report.getID(), true);
      }
      ```

   e. `commit` メソッドを呼び出して、権限変更のバッチを送信します。

      ```
      secRightsAdmin.commit();
      ```

例

この例では、フォルダ内のすべての Crystal レポートを反復処理して、特定のユーザーに各レポートのデータ (`CeReportRightID.REFRESH_ON_DEMAND`) を最新表示するための明示的な権限を付与します。

```
void setExplicitRight(IEnterpriseSession enterpriseSession, IUser user, IFolder
folder) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

```
  ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID()
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");

  RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
  ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
  secRightsAdmin.batch();
  try
  {
          for (Object object : reports)
    {
      IInfoObject report = (IInfoObject)object;
      secRightsAdmin.setRight(refreshReportRight, user.getID(), report.getID(),
true);
    }
    secRightsAdmin.commit();
  }
  catch (Throwable e)
  {
    secRightsAdmin.rollback();
  }
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.RightDescriptor
- com.crystaldecisions.sdk.occa.security.ISecRightsAdmin
- com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr
- com.crystaldecisions.sdk.plugin.desktop.folder.IFolder
- com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID
- com.crystaldecisions.sdk.plugin.desktop.report.IReport
- com.crystaldecisions.sdk.plugin.desktop.user.IUser
- java.io.IOException

関連情報

オブジェクトセキュリティの概要 [113 ページ]
1 つのオブジェクトの明示的な権限を設定する [126 ページ]
複数のオブジェクトに対する権限をチェックする [134 ページ]

5.3.2.6 カスタムアクセスレベルを作成する

アクセスレベル（ロール）は、オブジェクトに対してよく使用される権限セットを多数のユーザーとグループに適用するために使用されます。BI platform は、いくつかの事前定義されたアクセスレベルでインストールされま

す。ただし、独自のアクセスレベルを作成してカスタマイズすることもできます。この機能により、セキュリティに関連する管理コストやメンテナンスコストを大幅に削減できます。SDK を使用してプログラムで新しいカスタムアクセスレベルを構築できます。

この例では、リポジトリから任意のオブジェクトを削除する権限を与えるカスタムアクセスレベルを作成します。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. `IInfoStore` インターフェイスの `newInfoObjectCollection` メソッドを呼び出して、`IInfoObjects` コレクションを作成します。

```
IInfoObjects newCustomRoles = infostore.newInfoObjectCollection();
```

3. コレクションの `add` メソッドを呼び出して、`ICustomRole` オブジェクトを作成します。

   定数 `ICustomRole.KIND` を使用し、結果を `ICustomRole` オブジェクトにキャストします。

```
ICustomRole newCustomRole = (ICustomRole)
newCustomRoles.add(ICustomRole.KIND);
```

4. 新しいアクセスレベルのさまざまなプロパティ（名前、説明など）を設定します。

```
newCustomRole.setTitle("Delete Objects");
newCustomRole.setDescription("This access level grants members to delete
objects from the repository.");
```

5. リポジトリからオブジェクトを削除する権限を表す新しい `RightDescriptor` オブジェクトを作成します。

   カスタムアクセスレベルを作成するときは、`CeSecurityOptions.RightScope` オブジェクトと共に、権限のスコープを指定する `RightDescriptor` コンストラクタを使用します。

   `CeSecurityID.Right` インターフェイスは、一般的なオブジェクト権限を定義します。他のオブジェクトタイプに固有の権限は、`CeReportRightID`、`CeWebIRightID`、`CeFullClientRightID`、`CeUniverseRightID` などのインターフェイスで指定されます。

```
RightDescriptor deleteObjectsRight = new
RightDescriptor(CeSecurityID.Right.DELETE, "", false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, CeSecurityOptions.ANY_OBJTYPE);
```

6. 新しく作成した権限をアクセスレベルの明示的な権限のコレクションに追加します。

   > ⓘ 注記
   >
   > `addRoleRight` メソッド呼び出しの論理値型引数 `true` は、アクセスレベルに権限を付与します。このケースでは、`IRoleRight.setGranted` メソッドを使用する必要がありません。

```
IRoleRights roleRights = newCustomRole.getRoleRights();
IRoleRight roleRight = roleRights.addRoleRight(deleteObjectsRight, true);
```

7. 新しいカスタムアクセスレベルを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(newCustomRoles);
```

新しいカスタムアクセスレベルが作成されます。レポートなどのオブジェクトの既存の主体にこのアクセスレベルを割り当てるには、次のように、主体の既存のアクセスレベルのリストにこのアクセスレベルを追加します。

```
IExplicitRoles customRoles = principal.getRoles();
```

```
IExplicitRole customRole = customRoles.add(newCustomRole.getID());
```

このコードでは、principal という名前の IExplicitPrincipal インスタンスがあるとします。アクセスレベルを追加したら、IInfoStore.commit メソッドを使用して、セキュリティ設定を変更したレポートまたはオブジェクトを CMS にコミットする必要があります。

例

この例では、リポジトリから任意のオブジェクトを削除する権限を与えるカスタムアクセスレベルを作成します。

```
void createCustomAccessLevel(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects newCustomRoles = infostore.newInfoObjectCollection();
  ICustomRole newCustomRole = (ICustomRole) newCustomRoles.add(ICustomRole.KIND);

  newCustomRole.setTitle("Delete Objects");
  newCustomRole.setDescription("This access level grants members to delete
objects from the repository.");

  RightDescriptor deleteObjectsRight = new
RightDescriptor(CeSecurityID.Right.DELETE, "", false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, CeSecurityOptions.ANY_OBJTYPE);
  IRoleRights roleRights = newCustomRole.getRoleRights();
  IRoleRight roleRight = roleRights.addRoleRight(deleteObjectsRight, true);
  infostore.commit(newCustomRoles);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.customrole.ICustomRole
- com.businessobjects.sdk.plugin.desktop.customrole.IRoleRight
- com.businessobjects.sdk.plugin.desktop.customrole.IRoleRights
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.CeSecurityID
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.RightDescriptor
- com.crystaldecisions.sdk.occa.security.CeSecurityOptions

関連情報

オブジェクトセキュリティの概要 [113 ページ]
1 つのオブジェクトの明示的な権限を設定する [126 ページ]
複数のオブジェクトの明示的な権限を設定する [137 ページ]

5.4 スケジューリング

スケジュールは、Central Management Server（CMS）内に保存されている `InfoObject` を選択し、指定された時刻にそれを自動的に実行するための処理です。最もよくスケジュールされる `InfoObject` はレポートとドキュメントですが、プログラムやオブジェクトパッケージなどの他の `InfoObject` もスケジュールできます。

`InfoObject` は、さまざまな方法でスケジュールできます。カスタム値、ファイル形式、出力先、パラメータを指定できます。カレンダーを使用してレポートをスケジュールすることもできます。

スケジュールには、次のさまざまなオプションがあります。

- `InfoObject` は、スケジュールの設定後即座に実行するようにスケジュールすることも、将来のある時点で実行するようにスケジュールすることもできます。
- `InfoObject` は、一度だけ実行するようにスケジュールすることも、定期的に繰り返し実行するようにスケジュールすることもできます。
- `InfoObject` は、Crystal レポートファイル、Excel ファイル、Word ファイル、PDF ファイルなどのさまざまな形式でエクスポートするようにスケジュールできます。
- `InfoObject` は、ファイルリポジトリサーバーに内部的に保存するようにスケジュールできます。さまざまな出力先にスケジュールすることもできます。

特定のスケジュール情報を管理するには、`IInfoObject` オブジェクトの各インスタンスに関連付けられている `SchedulingInfo` プロパティにアクセスします。スケジュール情報を決定し、CMS リポジトリにコミットしたら、`IInfoStore.schedule` メソッドを使用してスケジュールオブジェクトを選択できます。このメソッドは、コレクション内のオブジェクトの実行可能インスタンスを作成します。実行可能インスタンスは、スケジュールされた親 `InfoObject` の子です。スケジュールの詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

スケジュールに使用されるクラス

- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
  繰り返しオプション、出力先オプションなどの、オブジェクトのスケジュールプロパティが含まれます。
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
  オブジェクトが実行される間隔を指定します。
- `com.crystaldecisions.sdk.plugin.desktop.calendar.ICalendar`
  ビジネスカレンダーのコレクションとカレンダーの実行日に関連する情報にアクセスするためのメソッドとプロパティを提供します。

5.4.1 スケジュールのワークフロー

SDK を使用してレポートをスケジュールする際の基本ワークフローは、次のとおりです。

レポートをスケジュールする

1. `InfoStore` オブジェクトクエリーによってレポートオブジェクトを取得します。
2. `InfoObject.SchedulingInfo` プロパティによって `SchedulingInfo` オブジェクトを取得します。
3. レポートのスケジュールのオプションを設定します。
4. `ReportFormatOptions.format` プロパティを使用して、レポートの出力書式を設定します。
5. レポートの出力先と出力先オプションを設定します。
6. レポートをスケジュールします。

> **注記**
>
> Job Server で設定された出力先オプションは、レポート内で設定されている出力先オプションによって上書きされます。

5.4.2 出力先

オブジェクトをスケジュールすると、そのオブジェクトのインスタンスが特定の出力先に送信されます。サポートされている出力先は、次のとおりです。

出力先	説明
FTP	FTP を使用してインスタンスをサーバーへコピーできます。
SMTP	電子メールでインスタンスを送信できます。
アンマネージドディスク	インスタンスをローカル ファイル システムまたはネットワーク ファイル システムにコピーできます。
受信ボックス（マネージド）	インスタンスをユーザーの Enterprise 受信ボックスにスケジュールできます。
プリンタ	インスタンスをローカルプリンタまたはネットワークプリンタへ出力できます。

> **注記**
>
> - 受信ボックスはマネージド出力先の一例です。マネージド出力先は、BI platform 内に存在する場所です。
> - プリンタは、RPT 形式から印刷する Crystal レポート固有の出力先です。他の出力先は、スケジュールされたオブジェクトとサポートされている形式タイプのすべてに適用できます。Crystal レポートのプリンタオプションは、`com.crystaldecisions.sdk.plugin.desktop.report.IReport` インターフェイスから直接設定します。

出力先は、次の 2 とおりの方法で設定します。

- 適切な Job Server のデフォルトの出力先を使用する。
  Job Server のデフォルトの出力先は、`IServerDestination` インターフェイスから設定できます。
- スケジュールされた時刻にオブジェクトの出力先を設定する出力先プラグインを使用する。
  スケジュールされたオブジェクトの出力先は、`IDestinationPlugin` インターフェイスの適切なサブインターフェイスから設定します。
  - `com.crystaldecisions.sdk.plugin.destination.ftp.IFTP`
  - `com.crystaldecisions.sdk.plugin.destination.smtp.ISMTP`

- `com.crystaldecisions.sdk.plugin.destination.managed.IManaged`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanaged`

> **注記**
>
> その出力先にジョブをスケジュールする前に、適切な Job Server 上の出力先を有効にする必要があります。詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

スケジュールまたは出力先に送信できるオブジェクト

オブジェクトタイプ	スケジュール	送信
Crystal レポート	はい	はい
オブジェクトパッケージ	はい	はい
プログラム	はい	はい
Excel ファイル	いいえ	はい
Word ファイル	いいえ	はい
PDF ファイル	いいえ	はい
テキストファイル	いいえ	はい
RTF ファイル	いいえ	はい
PowerPoint ファイル	いいえ	はい
ハイパーリンク	いいえ	はい
Web Intelligence ドキュメント	はい	はい
パブリケーション	はい	はい

オブジェクトタイプの出力先

オブジェクトタイプ	アンマネージドディスク	FTP	SMTP（ファイル）|（リンク）	受信ボックス（ファイル）|（リンク）	お気に入りフォルダ（ファイル）|（リンク）
Crystal レポート	はい	はい	はい | はい	はい | はい	はい | はい

オブジェクトタイプ	アンマネージドディスク	FTP	SMTP（ファイル）| （リンク）	受信ボックス（ファイル）| （リンク）	お気に入りフォルダ（ファイル）| （リンク）
オブジェクトパッケージ	いいえ	いいえ	いいえ | いいえ	はい | はい	はい | はい
プログラム	はい	はい	はい | はい	はい | はい	はい | はい
Excel ファイル	はい	はい	はい | はい	はい | はい	はい | はい
Word ファイル	はい	はい	はい | はい	はい | はい	はい | はい
PDF ファイル	はい	はい	はい | はい	はい | はい	はい | はい
テキストファイル	はい	はい	はい | はい	はい | はい	はい | はい
RTF ファイル	はい	はい	はい | はい	はい | はい	はい | はい
PowerPoint ファイル	はい	はい	はい | はい	はい | はい	はい | はい
ハイパーリンク	いいえ	いいえ	いいえ | はい	はい | はい	はい | はい
Web Intelligence ドキュメント	いいえ	いいえ	いいえ | はい	はい | はい	はい | はい

5.4.2.1 出力先オプションの設定

対応するオプションインターフェイスから各出力先のオプションを設定できます。

- `com.crystaldecisions.sdk.plugin.destination.ftp.IFTPOptions`
- `com.crystaldecisions.sdk.plugin.destination.smtp.SMTPOptions`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions`
- `com.crystaldecisions.sdk.plugin.destination.managed.IManagedOptions`

オプションは、Job Server のデフォルトの出力先を使用してグローバルに適用したり、出力先プラグインを使用して個別に適用することができます。

Job Server のグローバルな出力先オプション

グローバルな出力先オプションは、Job Server 上で設定されるデフォルトの出力先オプションです。出力先プラグインから出力先オプションを設定しない場合は、Job Server のデフォルトの出力先オプションが使用されます。主に次の 2 つの状況を考慮する必要があります。

- `IInfoStore.schedule` メソッドを使用してスケジュールされるオブジェクト。
  スケジュールされたオブジェクトを処理する Job Server のデフォルトの出力先オプションが使用されます。たとえば、Crystal Reports Job Server です。
- `IInfoStore.sendTo` メソッドを使用して送信されるオブジェクト。
  Destination Job Server のデフォルトの出力先オプションが使用されます。Destination Job Server は、スケジュールや処理を行わない点で、他の Job Server とは異なります。

`IServerDestination.getConfigGlobalOptions` メソッドを使用して、プログラムで Job Server 上のデフォルトの出力先オプションを変更できます。

次の例は、すべての Crystal Reports Job Server のデフォルトの出力先を有効にして、それをローカル ファイル システムの `C:¥` フォルダに設定します。これで、この Job Server の出力先が使用されるように、オブジェクトのスケジュール情報が `CrystalEnterprise.DiskUnmanaged` に設定されます。

```
void setJobServerDefaultDestination(IEnterpriseSession enterpriseSession,
IInfoObject report) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects servers = infostore.query( "Select SI_ID, SI_HOSTED_SERVICES From
CI_SYSTEMOBJECTS "
    + "Where SI_KIND='Server' And SI_SERVER_KIND='jobserver' "
    + "And SI_NAME LIKE '%CrystalReportsJobServer%'");

  for (int i = 0; i < servers.size(); i++)
  {
    IServer crJobServer = (IServer)servers.get(i);
    IServerDestinations serverDestinations = crJobServer.getServerDestinations();
    IServerDestination diskUnmanagedDestination =
serverDestinations.getServerDestination("CrystalEnterprise.DiskUnmanaged");

    diskUnmanagedDestination.setRequestedStatus(true);

    IDiskUnmanagedOptions diskUnmanagedOptions =
(IDiskUnmanagedOptions)diskUnmanagedDestination.getConfigGlobalOptions();
    List outputFilePaths = diskUnmanagedOptions.getDestinationFiles();
    outputFilePaths.clear();
    outputFilePaths.add("C:¥¥");
  }
  infostore.commit(servers);

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.getDestinations().add("CrystalEnterprise.DiskUnmanaged");
}
```

出力先プラグインオプション

出力先プラグインは、Job Server 上のデフォルトのオプションを上書きする出力先オプションを設定する方法を提供します。Central Management Server（CMS）から適切なプラグインを照会し、`IDestinationPlugin.getScheduleOptions` メソッドを使用して適切な出力先タイプのオプションにアクセスして変更します。

プラグインの出力先オプションを変更した後で、`IDestination.setFromPlugin` メソッドを使用して出力先プラグインをオブジェクトに適用してから、そのオブジェクトをスケジュールまたは送信できます。

次の例は、アンマネージドディスク出力先プラグインを照会し、その送信オプションをローカル ファイル システムの C:¥ フォルダに設定します。これで、出力先プラグインがオブジェクトのスケジュール情報に適用されます。

```
void setDestinationToUnmanagedDisk(IEnterpriseSession enterpriseSession,
IInfoObject report) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects diskUnmanagedInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID "
    + "From CI_SYSTEMOBJECTS Where SI_NAME = 'CrystalEnterprise.DiskUnmanaged'");
  IInfoObject infoObject = (IInfoObject)diskUnmanagedInfoObjects.get(0);
  IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
  IDiskUnmanaged diskUnmanaged = (IDiskUnmanaged)destinationPlugin;
  IDiskUnmanagedOptions diskUnmanagedOptions =
(IDiskUnmanagedOptions)diskUnmanaged.getScheduleOptions();
  List outputFilePaths = diskUnmanagedOptions.getDestinationFiles();
  outputFilePaths.clear();
  outputFilePaths.add("C:¥¥");

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.DiskUnmanaged");
  destination.setFromPlugin(destinationPlugin);
}
```

注記

その出力先にジョブをスケジュールする前に、適切な Job Server 上の出力先を有効にする必要があります。詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPlugin`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerDestination`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerDestinations`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanaged`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions`
- `java.io.IOException`
- `java.util.List`

5.4.2.2 スケジュールされたオブジェクトの出力先を設定する

Central Management Server (CMS) を照会して取得した出力先プラグイン (`IDestinationPlugin` とそのサブインターフェイス) を使用して、スケジュールされたオブジェクトの出力先を設定できます。このように設定した

出力先は、Job Server 上のデフォルトの設定を上書きします。これらの例では、getSchedulingInfo メソッドを使用して ISchedulingInfo オブジェクトを取得しているとします。

例: 出力先を FTP に設定

この例では、スケジュールされたオブジェクトを scheduledReport.pdf という名前のファイルとして、ポート 21 で受信待機している servername.domain という名前の FTP サーバー上の ftpdir フォルダに送信するように指定します。

```
void setDestinationToFTP(IEnterpriseSession enterpriseSession, IInfoObject
report) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

  IInfoObjects ftpInfoObjects = infostore.query("Select SI_DEST_SCHEDULEOPTIONS,
SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME = 'CrystalEnterprise.Ftp'");
  IInfoObject ftpInfoObject = (IInfoObject)ftpInfoObjects.get(0);

  IDestinationPlugin destinationPlugin = (IDestinationPlugin)ftpInfoObject;
  IFTP ftp = (IFTP)destinationPlugin;

  IFTPOptions ftpOptions = (IFTPOptions)ftp.getScheduleOptions();
  ftpOptions.setServerName("servername.domain");
  ftpOptions.setPort(Integer.parseInt("21"));
  ftpOptions.setUserName("username");
  ftpOptions.setPassword("password");
  List files = ftpOptions.getDestinationFiles();
  files.add("ftpdir¥¥scheduledReport.pdf");
  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.Ftp");
  destination.setFromPlugin(destinationPlugin);
}
```

> **注記**
>
> FTP サーバーにスケジュールするときに設定できるオプションの詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスで、com.crystaldecisions.sdk.plugin.destination.ftp.IFTPOptions インターフェイスを参照してください。

例: 出力先を SMTP に設定

この例では、スケジュールされたオブジェクトを、ポート 25 で受信待機している servername.domain という名前の SMTP サーバーを通してリレーされる 2 つのアドレスに電子メールで送信するように指定します。

```
void setDestinationToSMTP(IEnterpriseSession enterpriseSession, IInfoObject
report) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

```
  IInfoObjects smtpInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME =
'CrystalEnterprise.Smtp'");
  IInfoObject infoObject = (IInfoObject)smtpInfoObjects.get(0);

  IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
  ISMTP smtp = (ISMTP)destinationPlugin;

  ISMTPOptions smtpOptions = (ISMTPOptions)smtp.getScheduleOptions();
  smtpOptions.setDomainName("domain");
  smtpOptions.setServerName("servername");
  smtpOptions.setPort(Integer.parseInt("25"));
  smtpOptions.setSMTPAuthenticationType(ISMTPOptions.CeSMTPAuthentication.LOGIN);
  smtpOptions.setSMTPUserName("username");
  smtpOptions.setSMTPPassword("password");
  smtpOptions.setSubject("Email subject");
  smtpOptions.setMessage("This is the email message.");

  List toAddresses = smtpOptions.getToAddresses();
  toAddresses.add("user1@emaildomain.com");
  toAddresses.add("user2@emaildomain.com");*/

  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.Smtp");
  destination.setFromPlugin(destinationPlugin);
}
```

> **注記**
>
> 電子メールの出力先にスケジュールするときに設定できるオプションの詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム *Java API* リファレンスで `com.crystaldecisions.sdk.plugin.destination.smtp.ISMTPOptions` インターフェイスを参照してください。

例: 出力先をアンマネージドディスクに設定

この例では、スケジュールされたオブジェクトをローカル ファイル システム上の `C:¥` フォルダに送信するように指定します。

```
void setDestinationToUnmanagedDisk(IEnterpriseSession enterpriseSession,
IInfoObject report) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects diskUnmanagedInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME =
'CrystalEnterprise.DiskUnmanaged'");
  IInfoObject infoObject = (IInfoObject)diskUnmanagedInfoObjects.get(0);

  IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
  IDiskUnmanaged diskUnmanaged = (IDiskUnmanaged)destinationPlugin;

  IDiskUnmanagedOptions diskUnmanagedOptions =
(IDiskUnmanagedOptions)diskUnmanaged.getScheduleOptions();
  List outputFilePaths = diskUnmanagedOptions.getDestinationFiles();
  outputFilePaths.clear();
  outputFilePaths.add("C:¥¥");

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.DiskUnmanaged");
```

```
  destination.setFromPlugin(destinationPlugin);
}
```

> ⓘ 注記
>
> ファイル システムの場所にスケジュールするときに設定できるオプションの詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスで `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions` インターフェイスを参照してください。

例: 出力先を受信ボックスに設定

この例では、スケジュールされたオブジェクトをユーザーの受信ボックスにコピーするように指定します。

```
void setDestinationToInbox(IEnterpriseSession enterpriseSession, IUser user,
IInfoObject report) throws SDKException, IOException
{
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

   IInfoObjects managedInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME =
'CrystalEnterprise.Managed'");
   IInfoObject infoObject = (IInfoObject)managedInfoObjects.get(0);

   IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
   IManaged managed = (IManaged)destinationPlugin;

   IManagedOptions managedOptions =
(IManagedOptions)managed.getScheduleOptions();

managedOptions.setDestinationOption(IManagedOptions.CeDestinationOption.ceInbox);
   managedOptions.setSendOption(IManagedOptions.CeManagedSendOption.ceCopy);
   managedOptions.setIncludeInstance(true);
   managedOptions.getDestinations().add(user.getID());

   IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.Managed");
   destination.setFromPlugin(destinationPlugin);
}
```

> ⓘ 注記
>
> 受信ボックスにスケジュールするときに設定できるオプションの詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム *Java API* リファレンスで `com.crystaldecisions.sdk.plugin.destination.managed.IManagedOptions` インターフェイスを参照してください。

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPlugin`

- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo
- com.crystaldecisions.sdk.plugin.desktop.user.IUser
- com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanaged
- com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions
- com.crystaldecisions.sdk.plugin.destination.ftp.IFTP
- com.crystaldecisions.sdk.plugin.destination.ftp.IFTPOptions
- com.crystaldecisions.sdk.plugin.destination.managed.IManaged
- com.crystaldecisions.sdk.plugin.destination.managed.IManagedOptions
- com.crystaldecisions.sdk.plugin.destination.smtp.ISMTP
- com.crystaldecisions.sdk.plugin.destination.smtp.ISMTPOptions
- java.io.IOException
- java.util.List

5.4.3 レポートをスケジュールする

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. 指定されたレポートオブジェクトを照会して取得します。

```
IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
IReport report = (IReport) infoObjects.get(0);
```

3. getSchedulingInfo メソッドを呼び出して、ISchedulingInfo オブジェクトのインスタンスを取得します。

```
ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

4. スケジュールオプションを設定します。

   このレポートは、即座に一度だけ実行するように設定しましょう。

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

5. getReportFormatOptions メソッドを呼び出して、レポートインスタンスの形式を取得します。

```
IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();
```

6. setFormat メソッドを呼び出して、形式の種類を設定します。形式の種類は、CeReportFormat クラスのいずれかの値を取ることができます。

> ⓘ 注記
>
> 選択した出力形式のプロパティも設定できます。

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. Crystal レポートのコピーをプリンタに出力するには、レポートのプリンタオプションを変更します。

   プリンタへの出力をスケジュールできるのは、Crystal レポートのみです。`"¥¥¥¥servername¥¥printername"` を、使用するプリンタ名に置き換えます。

   > ⓘ 注記
   >
   > プリンタにアクセスするには、プリンタへのアクセスに必要なネットワーク権限を持つユーザーアカウントを使用して、Server Intelligence Agent を実行している必要があります。サーバーの管理と設定の詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

```
IReportPrinterOptions printerOptions = report.getReportPrinterOptions();
printerOptions.setCopies(1);
printerOptions.setEnabled(true);
printerOptions.setFromPage(1);
printerOptions.setToPage(1);
printerOptions.setPrinterName("¥¥¥¥servername¥¥printername");
```

8. `IInfoStore` オブジェクトを使用してレポートをスケジュールします。

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

例

この例では、Crystal レポートを取得し、デフォルト出力先にレポートをスケジュールして、レポートの最初のページをプリンタに出力します。

```
void scheduleReport(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
  IReport report = (IReport) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;
  IReportPrinterOptions printerOptions = report.getReportPrinterOptions();
  printerOptions.setCopies(1);
  printerOptions.setEnabled(true);
  printerOptions.setFromPage(1);
  printerOptions.setToPage(1);
  printerOptions.setPrinterName("¥¥¥¥servername¥¥printername");
```

```
  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.CeScheduleType
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo
- com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions
- com.crystaldecisions.sdk.plugin.desktop.common.IReportPrinterOptions
- com.crystaldecisions.sdk.plugin.desktop.report.IReport

関連情報

スケジュールされたオブジェクトの出力先を設定する [147 ページ]

5.4.4 カスタム値を使用してレポートをスケジュールする

レポートをスケジュールする際、作成されるインスタンスにカスタム値を追加することができます。たとえば、管理者はインスタンスにユーザーの電子メールアドレスやユーザー名を組み込み、特定のユーザーが作成したインスタンスを追跡することができます。これは InfoObject を使用して行います。この例では、InfoObject を使用して電子メールアドレスを保存します。その後、CMS 上でその電子メールアドレスを含む全インスタンスを照会します。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. 指定されたレポートオブジェクトを照会して取得します。

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
   From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
   Report' and SI_INSTANCE = 0");
   IInfoObject report = (IInfoObject) infoObjects.get(0);
   ```

3. InfoObject.SchedulingInfo プロパティを使用して、SchedulingInfo オブジェクトを取得します。

   ```
   ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
   ```

4. スケジュールオプションを設定します。

このレポートは、即座に一度だけ実行するように設定しましょう。

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

5. ReportFormatOptions プロパティを使用して、レポートインスタンスの形式を取得します。

```
IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

6. Format プロパティを使用して、形式の種類を設定します。形式の種類は CeReportFormat のいずれかの値を取ることができます。

> **注記**
>
> 選択した出力形式のプロパティも設定できます。

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. InfoObject のプロパティ バッグを使用して、レポートにカスタム値を追加します。

> **注記**
>
> カスタム値を追加するには、キーも与える必要があります。キーにより、InfoStore Query メソッドを使用してカスタム値を取得できます。カスタム値を含むレポートを取得するには、クエリーの WHERE 句内でキーを使用します。プロパティ バッグにそのプロパティがないと、結果は返されません。

この例のキー（値が "Email" の文字列）は、後で値を取得するために使用します。

```
report.properties().setProperty("Email", email);
```

8. InfoStore オブジェクトを使用してレポートをスケジュールします。

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

例

この例では、Crystal レポートを取得し、このレポートに電子メールアドレスを追加して、レポートをデフォルト送信先にスケジュールします。

```
void scheduleWithCustomValues(IEnterpriseSession enterpriseSession, String
email) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE = 0");
  IInfoObject report = (IInfoObject) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;
  report.properties().setProperty("Email", email);

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

関連情報

スケジュールされたオブジェクトの出力先を設定する [147 ページ]

5.4.5 特定のパラメータ値を含むレポートをスケジュールする

この例では、特定のパラメータ値を含むレポートをスケジュールする方法を示します。変更したパラメータを持つレポートをスケジュールするために、そのパラメータを CMS にコミットする必要はありません。レポートをスケジュールするとき、CMS に格納されたパラメータではなく、現在の `ReportInterface` のパラメータが使用されます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. 指定されたレポートオブジェクトを照会して取得します。

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
   From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
   Report' and SI_INSTANCE = 0");
   IInfoObject report = (IInfoObject) infoObjects.get(0);
   ```

3. `InfoObject.SchedulingInfo` プロパティを使用して、`SchedulingInfo` オブジェクトのインスタンスを取得します。

   ```
   ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
   ```

4. スケジュールオプションを設定します。

このレポートは、即座に一度だけ実行するように設定しましょう。

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

5. ReportFormatOptions プロパティを使用して、レポートインスタンスの形式を取得します。

```
IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

6. Format プロパティを使用して、形式の種類を設定します。形式の種類は CeReportFormat のいずれかの値を取ることができます。

> ⓘ 注記
>
> 選択した出力形式のプロパティも設定できます。

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. レポートパラメータを取得して操作します。

次のコードは、レポート内の各パラメータを反復処理し、離散値の String パラメータを変更します。

```
for (int i = 0; i < ((IReport)report).getReportParameters().size(); i++)
{
  IReportParameter reportParameter = (IReportParameter)
((IReport)report).getReportParameters().get(i);
  if (reportParameter.isDiscreteValueSupported() &&
reportParameter.getValueType() ==
IReportParameter.ReportVariableValueType.STRING)
  {
    IReportParameterValues parameterValue =
reportParameter.getCurrentValues();
    IParameterFieldDiscreteValue discreteValue =
(IParameterFieldDiscreteValue)parameterValue.getValues(1).getValue(0);
    discreteValue.setValue("A String");
  }
}
```

> ⓘ 注記
>
> このサンプルでは、getParameterName メソッドのパラメータフィールドとして end_date を指定します。新しいパラメータ値 end_date は、addSingleValue メソッドによって加算されます。パラメータ値が存在する場合は、addSingleValue メソッドではなく、get メソッドを使用して既存のパラメータ値を取得します。

8. InfoStore オブジェクトを使用してレポートをスケジュールします。

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

例

この例では、Crystal レポートを取得し、レポートパラメータを変更して、レポートをデフォルト送信先にスケジュールします。

```
void scheduleWithParameters(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE = 0");
  IInfoObject report = (IInfoObject) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;

  for (int i = 0; i < ((IReport)report).getReportParameters().size(); i++)
  {
    IReportParameter reportParameter = (IReportParameter)
((IReport)report).getReportParameters().get(i);
    if (reportParameter.isDiscreteValueSupported() &&
reportParameter.getValueType() ==
IReportParameter.ReportVariableValueType.STRING)
    {
      IReportParameterValues parameterValue = reportParameter.getCurrentValues();
      IParameterFieldDiscreteValue discreteValue =
(IParameterFieldDiscreteValue)parameterValue.getValues(1).getValue(0);
      discreteValue.setValue("A String");
    }
  }

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

> **注記**
>
> このサンプルでは、rascore.jar ファイルと serialization.jar ファイルに依存する getReportParameters メソッドを使用します。したがって、このサンプルを実行するには、両方のファイルをプロジェクトにインポートする必要があります。

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.CeScheduleType
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo

- com.crystaldecisions.sdk.occa.report.data.IParameterFieldDiscreteValue
- com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions
- com.crystaldecisions.sdk.plugin.desktop.common.IReportParameter
- com.crystaldecisions.sdk.plugin.desktop.common.IReportParameterValues
- com.crystaldecisions.sdk.plugin.desktop.report.IReport

関連情報

スケジュールされたオブジェクトの出力先を設定する [147 ページ]

5.4.6 カレンダーを使用してレポートをスケジュールする

このチュートリアルでは、カレンダーを使用した操作の実行方法を示します。ビジネスカレンダーを使用すると、ビジネス要件に基づいて実行日を組み込むためのカスタムスケジュールを生成できます。たとえば、すべての平日（月曜から金曜まで）を実行日に設定するビジネスカレンダーを作成できます。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. PluginManager オブジェクトのインスタンスを取得して、新しい Calendar オブジェクトを作成します。

```
IPluginMgr pluginMgr = infostore.getPluginMgr();
IPluginInfo calendarPlugin =
pluginMgr.getPluginInfo("CrystalEnterprise.Calendar");
IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
newInfoObjects.add (calendarPlugin);
IInfoObject iObject = (IInfoObject)newInfoObjects.get(0);
```

3. カレンダーのタイトルと説明を設定します。

```
iObject.setTitle("My Calendar");
iObject.setDescription("A description of my calendar");
```

4. カレンダーに日付と時刻を追加します。

   このコードは、2010 年 1 月 1 日から 2010 年 12 月 31 日までの毎週月曜日に実行されるようにカレンダーを設定します。

```
int startDay = 01;
int startMonth = 0;
int startYear = 2010;
int endDay = 31;
int endMonth = 11;
int endYear = 2010;

int dayOfWeek = java.util.Calendar.MONDAY;
int weekOfMonth = IBusinessCalendarDay.ALL;
ICalendar calendar = (ICalendar)iObject;
IBusinessCalendarDays days = calendar.getDays();
int calendarID = days.getNextGroupID();
days.add(startDay, startMonth, startYear, endDay, endMonth, endYear,
dayOfWeek, weekOfMonth, calendarID);
```

5. `infoObject` を CMS にコミットします。

```
infostore.commit(newInfoObjects);
```

6. 指定されたレポートオブジェクトを照会して取得します。

```
InfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS "
  + "Where SI_KIND='CrystalReport' and SI_NAME='"
  + reportName + "' and SI_INSTANCE = 0");  IInfoObject report =
(IInfoObject) infoObjects.get(0);
IInfoObject report = (IInfoObject) infoObjects.get(0);
```

7. `InfoObject.SchedulingInfo` プロパティを使用して、`SchedulingInfo` オブジェクトを取得します。

```
ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

8. カレンダーを使用するためのスケジュールオプションを設定します。

```
scheduleInfo.setType(CeScheduleType.CALENDAR);
scheduleInfo.setRightNow(false);
scheduleInfo.setCalendarTemplate(calendarID);
```

9. `ReportFormatOptions` プロパティを使用して、レポートインスタンスの形式を取得します。

```
IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

10. `Format` プロパティを使用して、形式の種類を設定します。形式の種類は `CeReportFormat` のいずれかの値を取ることができます。

> **注記**
>
> 選択した出力形式のプロパティも設定できます。

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

11. `InfoStore` オブジェクトを使用してレポートをスケジュールします。

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

例

この例は、新しいカレンダーを作成し、Crystal レポートを取得し、カレンダーの日付を使用してレポートをスケジュールします。このカレンダーは、2010 年 1 月 1 日から 2010 年 12 月 31 日までの毎週月曜日に実行されるようにレポートをスケジュールします。

```
void scheduleWithCalendar(IEnterpriseSession enterpriseSession , String
reportName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IPluginMgr pluginMgr = infostore.getPluginMgr();
```

```
  IPluginInfo calendarPlugin =
pluginMgr.getPluginInfo("CrystalEnterprise.Calendar");

  IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
  newInfoObjects.add (calendarPlugin);

  IInfoObject iObject = (IInfoObject)newInfoObjects.get(0);
  iObject.setTitle("My Calendar");
  iObject.setDescription("A description of my calendar");
  int startDay = 01;
  int startMonth = 0;
  int startYear = 2010;
  int endDay = 31;
  int endMonth = 11;
  int endYear = 2010;

  int dayOfWeek = java.util.Calendar.MONDAY;
  int weekOfMonth = IBusinessCalendarDay.ALL;

  ICalendar calendar = (ICalendar)iObject;
  IBusinessCalendarDays days = calendar.getDays();
  int calendarID = days.getNextGroupID();
  days.add(startDay, startMonth, startYear, endDay, endMonth, endYear,
dayOfWeek, weekOfMonth, calendarID);

  infostore.commit(newInfoObjects);
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS "
          + "Where SI_KIND='CrystalReport' and SI_NAME='"
          + reportName + "' and SI_INSTANCE = 0");
  IInfoObject report = (IInfoObject) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

  scheduleInfo.setType(CeScheduleType.CALENDAR);
  scheduleInfo.setRightNow(false);
  scheduleInfo.setCalendarTemplate(calendarID);

  IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.CeScheduleType
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo
- com.crystaldecisions.sdk.occa.pluginmgr.IPluginInfo
- com.crystaldecisions.sdk.occa.pluginmgr.IPluginMgr
- com.crystaldecisions.sdk.plugin.desktop.calendar.IBusinessCalendarDay
- com.crystaldecisions.sdk.plugin.desktop.calendar.IBusinessCalendarDays

- com.crystaldecisions.sdk.plugin.desktop.calendar.ICalendar
- com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions
- com.crystaldecisions.sdk.plugin.desktop.report.IReport
- java.util.Calendar

5.4.7 イベントを使用してレポートをスケジュールする

イベントがトリガーされたときに処理されるレポートをスケジュールできます。たとえば、別のレポートの処理が正常に完了した後にレポートを実行したり、カスタムイベントが手動でトリガーされたときにレポートを実行するようにスケジュールできます。レポートが処理されるには、スケジュールの条件がすべて満たされている必要があります。イベントがトリガーされても、他のスケジュール条件が満たされていない場合、レポートは処理されません。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. スケジュールするレポートを照会して取得します。

```
IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
IReport report = (IReport) infoObjects.get(0);
```

3. レポートを処理する前に発生する必要があるイベントを照会して取得します。

   この例では、My Event という名前のイベントを照会します。

```
IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_NAME = 'My Event'");
IEvent event = (IEvent)events.get(0);
```

4. IReport オブジェクトの getSchedulingInfo メソッドを呼び出して、ISchedulingInfo オブジェクトのインスタンスを取得します。

```
ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

5. レポートのスケジュールのオプションを設定します。

   この例では、レポートを 1 回、即座に実行するように設定します。

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

6. スケジュールされたレポートの出力形式を設定します。

   この例では、出力形式を Crystal レポートに設定します。

```
IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. レポートを処理する前にトリガーされる必要がある依存イベントのリストを取得します。

```
IEvents scheduleEvents = scheduleInfo.getDependencies();
```

8. レポートを処理する前に発生する必要があるイベントを依存イベントのリストに追加します。

```
int eventID = event.getID();
scheduleEvents.add(eventID);
```

9. レポートをスケジュールします。

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

例

この例では、イベント My Event がトリガーされたときに、レポートを 1 回、即座に実行するようにスケジュールします。

```
void scheduleUsingEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
  IReport report = (IReport) infoObjects.get(0);

  IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_NAME = 'My Event'");
  IEvent event = (IEvent)events.get(0);

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;
  IEvents scheduleEvents = scheduleInfo.getDependencies();
  int eventID = event.getID();
  scheduleEvents.add(eventID);
  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.CeScheduleType
- com.crystaldecisions.sdk.occa.infostore.IEvents
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo
- com.crystaldecisions.sdk.occa.pluginmgr.IPluginInfo

- `com.crystaldecisions.sdk.occa.pluginmgr.IPluginMgr`
- `com.crystaldecisions.sdk.plugin.desktop.event.IEvent`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`
- `java.util.Calendar`

5.5 イベントとアラート

イベントは、SAP BusinessObjects Business Intelligence platform システム内で発生したアクションをキャプチャします。イベントの例には、ジョブの処理が終了した場合、システムファイルが作成された場合、ドキュメントのアラートがトリガーされた場合、パフォーマンスのしきい値を超えた場合などがあります。イベントは、`com.crystaldecisions.sdk.plugin.desktop.event.IEvent` インターフェイスを実装する、Central Management Server (CMS) 内のオブジェクトによって定義されます。

ジョブのスケジュールでイベントを使用することで、イベントがトリガーされたときにジョブを実行することができます。また、イベントがトリガーされたときにユーザーに通知することもできます。それには、イベントをアラート可能に設定し、そのイベントにユーザーを登録します。たとえば、レポートの処理が失敗したときにトリガーされるイベントを作成することができます。イベントがトリガーされた際に、アラート通知を購読ユーザーに送信したり、エラーレポートを実行する別のジョブを開始することができます。

この SDK を使用して、スケジュール、ファイル、およびカスタムイベントを作成できます。また、イベントのアラートを有効に設定したり、ユーザーをアラート可能なイベントに登録したり、アラート通知を表示することもできます。

この SDK を使用して、次のようなさまざまなイベントを作成できます。

- スケジュールイベント
  スケジュールイベントは、スケジュールジョブの処理終了時にトリガーされます。イベントは、ジョブが正常に終了したとき、ジョブは終了したが失敗だったとき、または少なくともジョブが終了したときにトリガーするように設定します。スケジュールイベントは、`com.crystaldecisions.sdk.plugin.desktop.event.IScheduleEvent` インターフェイスで表されます。
- ファイルイベント
  ファイルイベントはファイルの作成時にトリガーされます。たとえば、システム内にログファイルが作成されたときにトリガーされるように、ファイルイベントを定義できます。ファイルイベントは、`com.crystaldecisions.sdk.plugin.desktop.event.IFileEvent` インターフェイスで表されます。
- カスタムイベント
  カスタムイベントは、UI または SDK を使用して、手動でトリガーされます。カスタムイベントは、`com.crystaldecisions.sdk.plugin.desktop.event.IUserEvent` インターフェイスで表されます。

関連情報

イベントを使用してレポートをスケジュールする [161 ページ]

5.5.1 スケジュールイベントを作成する

スケジュールジョブの処理終了時にトリガーされるスケジュールイベントを作成できます。イベントは、ジョブが正常に終了したとき、ジョブは終了したが失敗だったとき、または少なくともジョブが終了したときにトリガーするように設定できます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. 新しい `IInfoObject` コレクションを作成し、これに新しい `IScheduleEvent` オブジェクトを追加します。

```
IInfoObjects infoObjects = infostore.newInfoObjectCollection();
IScheduleEvent event =
(IScheduleEvent)infoObjects.add(IScheduleEvent.SPECIFIC_KIND);
```

3. イベントのタイトルと説明を設定します。

```
event.setTitle("Schedule Event");
event.setDescription("Schedule event description");
```

4. イベントに対するアラートを有効にするには、`setAlertEnabled` メソッドを呼び出します。

```
event.setAlertEnabled(true);
```

5. イベントをトリガーするスケジュール上の結果を設定します。

   ジョブの正常完了時にイベントをトリガーするには、`IScheduleEvent.SUCCESS` を使用します。ジョブの失敗時にイベントをトリガーするには、`IScheduleEvent.FAILURE` を使用します。正常に終了したかどうかにかかわらず、ジョブの終了時にイベントをトリガーするには、`IScheduleEvent.BOTH` を使用します。

```
event.setDependencyType(IScheduleEvent.SUCCESS);
```

6. 新しいイベントコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(infoObjects);
```

例

次の例は、ジョブの正常完了時に発生するスケジュールイベントを作成します。

```
void createScheduleEvent(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  IScheduleEvent event =
(IScheduleEvent)infoObjects.add(IScheduleEvent.SPECIFIC_KIND);
  event.setTitle("Schedule Event");
  event.setDescription("Schedule event description");
  event.setAlertEnabled(true);
  event.setDependencyType(IScheduleEvent.SUCCESS);
  infostore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.plugin.desktop.event.IScheduleEvent
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

5.5.2 ファイルイベントを作成する

ファイルが作成されるとトリガーされるファイルイベントを作成できます。たとえば、ファイルイベントを使用して、外部プロセスによって生成されたログファイルがファイル システム上に置かれたときにレポートを実行するようなスケジュールを作成できます。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. 新しい IInfoObject コレクションを作成し、これに新しい IFileEvent オブジェクトを追加します。

```
IInfoObjects infoObjects = infostore.newInfoObjectCollection();
IFileEvent event = (IFileEvent)infoObjects.add(IFileEvent.SPECIFIC_KIND);
```

3. イベントのタイトルと説明を設定します。

```
event.setTitle("File Event");
event.setDescription("File event description.");
```

4. 監視するファイルのパスを設定します。

   次の例は、c:¥log.txt へのファイルパスを設定します。このファイルがファイル システム上に表示されると、イベントがトリガーされます。

```
event.setFileName("C:¥¥log.txt");
```

5. イベントを監視するイベントサーバーの名前を設定します。

   MyNodeName.EventServer を、イベントを監視するイベントサーバーの名前に置き換えます。

```
event.setServerFriendlyName("MyNodeName.EventServer");
```

6. イベントに対するアラートを有効にするには、setAlertEnabled メソッドを呼び出します。

```
event.setAlertEnabled(true);
```

7. アラートメッセージを追加するには、getDefaultAlertMessages メソッドから返されるコレクションに文字列を追加します。

```
event.getDefaultAlertMessages().add("The event was triggered.");
```

8. 新しいイベントコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(infoObjects);
```

例

次の例は、ファイル c:¥log.txt がファイル システム上に置かれたときにトリガーされるファイルイベントを作成します。

```
void createFileEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  IFileEvent event = (IFileEvent)infoObjects.add(IFileEvent.SPECIFIC_KIND);
  event.setTitle("File Event");
  event.setDescription("File event description.");
  event.setFileName("C:¥¥log.txt");
  event.setServerFriendlyName("MyNodeName.EventServer");
  event.setAlertEnabled(true);
  event.getDefaultAlertMessages().add("The event was triggered.");
  infostore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.plugin.desktop.event.IFileEvent
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

5.5.3 カスタムイベントを作成する

カスタムイベントを作成し、それらを必要に応じて手動でトリガーすることができます。たとえば、カスタムイベントが手動でトリガーされるまで待機するスケジューリングジョブを設定できます。アラート可能なカスタムイベントをトリガーすることで、イベント購読者にアラート通知を送信できます。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ```

2. 新しい IInfoObject コレクションを作成し、これに新しい IUserEvent オブジェクトを追加します。

   ```
   InfoObjects infoObjects = infostore.newInfoObjectCollection();
   IUserEvent event = (IUserEvent)infoObjects.add(IUserEvent.SPECIFIC_KIND);
   ```

3. イベントのタイトルと説明を設定します。

   ```
   event.setTitle("My Event");
   event.setDescription("Event description.");
   ```

4. イベントに対するアラートを有効にするには、setAlertEnabled メソッドを呼び出します。

   ```
   event.setAlertEnabled(true);
   ```

5. アラートメッセージを追加するには、getDefaultAlertMessages メソッドから返されるコレクションに文字列を追加します。

   ```
   event.getDefaultAlertMessages().add("The event was triggered.");
   ```

6. 新しいイベントコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(infoObjects);
```

例

次の例は、手動でトリガーできる `My Event` という名前のカスタムイベントを作成しています。

```
void createCustomEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  IUserEvent event = (IUserEvent)infoObjects.add(IUserEvent.SPECIFIC_KIND);
  event.setTitle("My Event");
  event.setDescription("Event description.");
  event.setAlertEnabled(true);
  event.getDefaultAlertMessages().add("The event was triggered.");
  infostore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.event.IUserEvent`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.5.4 イベントのアラートを有効にする

次の例は、イベントのアラートを有効にする方法を示しています。アラート可能なイベントがトリガーされると、そのイベントの購読者にアラート通知が送信されます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Central Management Server（CMS）にイベントを照会して取得します。

   次の例は、`My Event` という名前のイベントを取得しています。

```
IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_KIND = 'Event' and SI_NAME = 'My Event'");
IEvent event = (IEvent)events.get(0);
```

3. `setAlertEnabled` メソッドを呼び出して、イベントのアラートを有効にします。

```
event.setAlertEnabled(true);
```

4. 変更したイベントコレクションを CMS にコミットし直し、変更を保存します。

```
infostore.commit(events);
```

例

次の例は、イベントのアラートを有効にし、アラートメッセージを設定する方法を示しています。

```
void enableAlerting(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_KIND = 'Event' and SI_NAME = 'My Event'");
  IEvent event = (IEvent)events.get(0);
  event.setAlertEnabled(true);
  infostore.commit(events);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.event.IEvent`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.5.5 イベントを購読するには

ユーザーまたはグループをイベントに登録するには、そのイベントのアラートを事前に有効にする必要があります。

ユーザーまたはグループをイベントに登録できます。イベントがトリガーされると、そのイベントの購読者にアラート通知が送信されます。

> **注記**
>
> グループをイベントに登録している場合は、そのグループの個別のメンバーを購読者から削除できます。それには、`IEventBase` クラスの `getExcludedSubscribers` メソッドから返される一覧にメンバーを追加します。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ```

2. InfoStore にイベントを照会して取得します。

   次の例は、`My Event` という名前のイベントを取得しています。

   ```
   IInfoObjects events = infostore.query("Select SI_ID, SI_EVENT_SUBSCRIBERS
   from CI_SYSTEMOBJECTS where SI_KIND = 'Event' and SI_NAME = 'My Event'");
   IEvent event = (IEvent)events.get(0);
   ```

3. InfoStore に `IUser` オブジェクトを照会して取得します。

   次の例は、`Administrator` ユーザーを照会しています。

   ```
   IInfoObjects subscribers = infostore.query("Select SI_ID from
   CI_SYSTEMOBJECTS where SI_NAME = 'Administrator' and SI_KIND = 'User'");
   IUser subscriber = (IUser)subscribers.get(0);
   ```

4. 購読者オブジェクトからユーザー ID を取得します。

```
int subscriberID = subscriber.getID();
```

5. このイベントのアラート購読者の一覧にユーザー ID を追加します。

```
IAlertSubscriptions subscriptions = event.getAlertSubscriptions();
IAlertSubscription subscription = subscriptions.add(subscriberID);
```

6. アラート通知送信オプションを設定します。

   次の例は、受信ボックスへの配信方法を設定しています。

```
Set<String> deliveryMethods = subscription.getDeliveryMethods();
deliveryMethods.add(IAlertSubscription.CeAlertDeliveryMethod.INBOX);
```

7. 変更されたイベントコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(events);
```

例

次の例は、`My Event` という名前のアラート可能なイベントにユーザーを登録する方法を示しています。

```
void subscribeAlert(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects events = infostore.query("Select SI_ID, SI_EVENT_SUBSCRIBERS from
CI_SYSTEMOBJECTS where SI_KIND = 'Event' and SI_NAME = 'My Event'");
  IEvent event = (IEvent)events.get(0);

  IInfoObjects subscribers = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_NAME = 'Administrator' and SI_KIND = 'User'");
  IUser subscriber = (IUser)subscribers.get(0);
  int subscriberID = subscriber.getID();

  IAlertSubscriptions subscriptions = event.getAlertSubscriptions();
  IAlertSubscription subscription = subscriptions.add(subscriberID);
  Set<String> deliveryMethods = subscription.getDeliveryMethods();
  deliveryMethods.add(IAlertSubscription.CeAlertDeliveryMethod.INBOX);

  infostore.commit(events);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.common.IAlertSubscription`
- `com.crystaldecisions.sdk.plugin.desktop.common.IAlertSubscriptions`
- `com.crystaldecisions.sdk.plugin.desktop.event.IEvent`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`

- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- java.util.Set

5.5.6 カスタムイベントを発生させるには

カスタムイベントをトリガーするには、Central Management Server（CMS）に事前にコミットする必要があります。カスタムイベントの作成とトリガーを同じコミットアクションで実行することはできません。

カスタムイベントをプログラムでトリガーして、そのイベントを待機するオブジェクトをスケジュールしたり、そのイベントに登録されているユーザーやグループにアラート通知を送信することができます。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. InfoStore にユーザーイベントを照会して取得します。

```
IInfoObjects events = infostore.query("Select SI_ID,
SI_DEFAULT_ALERT_MESSAGES * from CI_SYSTEMOBJECTS where SI_SPECIFIC_KIND =
'UserEvent' and SI_NAME = 'My Event'");
IUserEvent event = (IUserEvent) events.get(0);
```

3. イベントをトリガーします。

```
event.trigger();
```

4. 変更したイベントコレクションを CMS にコミットし直し、変更を保存します。

```
infostore.commit(events);
```

例

次の例は、My Event という名前のカスタムイベントをトリガーする方法を示しています。

```
void triggerUserEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects events = infostore.query("Select SI_ID, SI_DEFAULT_ALERT_MESSAGES
from CI_SYSTEMOBJECTS where SI_SPECIFIC_KIND = 'UserEvent' and SI_NAME = 'My
Event'");
  IUserEvent event = (IUserEvent) events.get(0);
  event.trigger();
  infostore.commit(events);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

- com.crystaldecisions.sdk.plugin.desktop.event.IUserEvent

5.5.7 アラート通知を表示する

アラート可能なイベントがトリガーされると、そのイベントに登録しているユーザーとグループにアラート通知が送信されます。アラート通知には、アラートメッセージ、アラートをトリガーしたイベントの ID、およびそのイベントに登録しているユーザーとグループの一覧が入っています。次の例は、ユーザーに送信されたアラート通知を取得する方法を示します。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. ユーザーを照会して取得し、クエリーに SI_RECEIVED_ALERTNOTIFICATIONS プロパティを含めます。

   次の例は、管理者ユーザーを照会します。

```
IInfoObjects users = infostore.query( "Select SI_ID,
SI_RECEIVED_ALERTNOTIFICATIONS from CI_SYSTEMOBJECTS where SI_KIND='user' and
SI_NAME='Administrator'");
IUser user = (IUser)users.get(0);
```

3. このユーザーの受信アラート通知を取得します。

```
IReceivedAlertNotifications receivedAlertNotifications =
user.getReceivedAlertNotifications();
IReceivedAlertNotification notification =
receivedAlertNotifications.iterator().next();
```

4. アラート通知の ID を取得します。

```
int alertID = notification.getID();
```

5. アラート通知の ID を使用してアラート通知を検索することでアラート通知を照会および取得し、クエリーに SI_ALERT_MESSAGES プロパティを含めます。

```
IInfoObjects alertNotifications = infostore.query("Select SI_ID,
SI_ALERT_MESSAGES from CI_SYSTEMOBJECTS where SI_KIND = 'AlertNotification'
AND SI_ID = " + alertID);
IAlertNotification alertNotification = (IAlertNotification)
alertNotifications.iterator().next();
```

6. IAlertNotification クラスの getMessages メソッドを使用して、アラートメッセージを取得します。

```
List<String>alertMessages = alertNotification.getMessages();
String alertMessage = (String) alertMessages.get(0);
...
```

例

次の例は、ユーザーに送信されたアラート通知を表示する方法を示します。

```
void getAlertNotifications(IEnterpriseSession enterpriseSession) throws
SDKException
```

```
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects users = infostore.query( "Select SI_ID,
SI_RECEIVED_ALERTNOTIFICATIONS from CI_SYSTEMOBJECTS where SI_KIND='user' and
SI_NAME='Administrator'");
  IUser user = (IUser)users.get(0);
  IReceivedAlertNotifications receivedAlertNotifications =
user.getReceivedAlertNotifications();
  IReceivedAlertNotification notification =
receivedAlertNotifications.iterator().next();
  int alertID = notification.getID();
  IInfoObjects alertNotifications = infostore.query("Select SI_ID,
SI_ALERT_MESSAGES from CI_SYSTEMOBJECTS where SI_KIND = 'AlertNotification' AND
SI_ID = " + alertID);
  IAlertNotification alertNotification = (IAlertNotification)
alertNotifications.iterator().next();
  List<String>alertMessages = alertNotification.getMessages();
  String alertMessage = (String) alertMessages.get(0);
    ...
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.alertnotification.IAlertNotification`
- `com.businessobjects.sdk.plugin.desktop.common.IRecievedAlertNotification`
- `com.businessobjects.sdk.plugin.desktop.common.IRecievedAlertNotifications`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `java.util.List`

5.6 パブリケーション

パブリケーションは、定期的に複数の受信者に配布するようにスケジュールできるオブジェクトの集合です。受信者は、パブリケーションを電子メールで受け取ることができる BI platform ユーザーまたは外部ユーザーです。オブジェクトの集合には 1 つ以上のレポートドキュメント（Crystal Reports ドキュメントまたは Web Intelligence ドキュメント）が含まれ、Excel スプレッドシートや画像などの静的オブジェクトを含めることもできます。パブリケーション内のレポートドキュメントの内容は、受信者に関係のある情報だけが表示されるように個人用にカスタマイズできます。

パブリケーションには、スケジュールされたパブリケーションを受け取る受信者（ユーザーおよびユーザーグループ）のリストが含まれます。電子メール受信者も、動的データソースから Crystal Reports または Desktop Intelligence ドキュメントなどを取得できます。

パブリケーションには、プロファイルおよびプロファイルターゲットのリストも含まれます。これにより、パブリケーションの作成者は、各受信者に割り当てられたプロファイル値に基づいて、パブリケーション内のレポートの内容を個人用にカスタマイズできます。

ユーザーは、セントラル管理コンソール（CMC）または BI 起動パッドのいずれかを使用してパブリケーションを作成および管理できます。パブリケーションをプログラムで作成および管理することもできます。このセクションでは、SDK を使用してパブリケーションを作成および管理する方法について説明します。

> ⓘ 注記
>
> 公開機能の詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム ユーザーガイドを参照してください。

パブリケーションの管理に使用されるクラス

- `com.businessobjects.sdk.plugin.desktop.publication`
  `IPublication` インターフェイスといくつかのサポートクラスが含まれます。
- `com.crystaldecisions.sdk.occa.infostore`
  出力先を設定するためのクラスが含まれます。
- `com.crystaldecisions.sdk.plugin.desktop.common`
  出力形式と動的受信者を設定するためのクラスが含まれます。

5.6.1 公開のワークフロー

SDK を使用してドキュメントを公開する際の基本ワークフローは、次のとおりです。

1. リポジトリにパブリケーションを作成します。
2. パブリケーションにレポートドキュメントを追加します。1 つ以上の Crystal レポートまたは Web Intelligence ドキュメントを含めることができますが、すべてのドキュメントの種類を同じにする必要があります。
3. パブリケーションに受信者を追加します。パブリケーションの受信者として BI platform ユーザーおよびユーザーグループを登録できます。各動的受信者の名前、電子メールアドレス、および（オプションで）プロファイル値が記述された Crystal Reports または Desktop Intelligence ドキュメントなどのデータソースを指定して、動的受信者を登録することもできます。
4. BI platform ユーザーごとにレポートのパーソナライズに使用されるプロファイル値を指定します。
5. パブリケーションが配信される出力先を設定し、出力先ごとに各パブリケーションドキュメントの出力形式を指定します。
6. パブリケーションに静的ドキュメントを追加します。
7. パブリケーションをスケジュールします。

このセクションのトピックでは、各手順の実行方法を詳細に説明します。

5.6.2 パブリケーションを作成するには

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. CMS リポジトリに対してクエリーを実行して、パブリケーションを作成するフォルダオブジェクトを取得します。

   ```
   String folderQuery = "SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_KIND= '"
   ```

```
  + IFolder.FOLDER_KIND + "' AND SI_NAME = 'Feature Samples'";
IInfoObjects folders = infostore.query(folderQuery);
IInfoObject folder = (IInfoObject) folders.get(0);
```

3. パブリケーション情報オブジェクトを作成し、親 ID をフォルダオブジェクトの ID に設定します。

```
IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
IPublication publication = (IPublication)
newInfoObjects.add(IPublication.KIND);
publication.setTitle("Test Publication");
publication.setDescription("This is a sample publication");
publication.setParentID(folder.getID());
```

4. パブリケーションを CMS リポジトリに保存します。

```
publication.save();
```

例

次のコードは、CMS の `Feature Samples` フォルダにパブリケーションを作成します。

```
public void createPublication(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");

  String folderQuery = "SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_KIND= '"
     + IFolder.FOLDER_KIND + "' AND SI_NAME = 'Feature Samples'";
  IInfoObjects folders = infostore.query(folderQuery);
  IInfoObject folder = (IInfoObject) folders.get(0);

  IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
  IPublication publication = (IPublication)
newInfoObjects.add(IPublication.KIND);

  publication.setTitle("Test Publication");
  publication.setDescription("This is a sample publication");
  publication.setParentID(folder.getID());
  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

関連情報

パブリケーションにレポートドキュメントを追加するには [176 ページ]
パブリケーションに静的ドキュメントを追加するには [177 ページ]
パブリケーションにユーザーを追加するには [180 ページ]
パブリケーションにグループを追加するには [182 ページ]
動的受信者向けにドキュメントをパーソナライズする [189 ページ]
出力先および出力形式を設定するには [194 ページ]

5.6.3 パブリケーションへのドキュメントの追加

5.6.3.1 レポートドキュメントと静的ドキュメント

パブリケーションには、次の 2 種類のドキュメントを含めることができます。

- レポート（動的）ドキュメント。
- 静的ドキュメント。

レポートドキュメントは、パブリケーションのメインコンポーネントです。パブリケーションには、少なくとも 1 つのレポートドキュメントが含まれる必要があります。

パブリケーションは、Crystal レポートおよび Web Intelligence ドキュメントをサポートすることができます。レポートオブジェクトの `SI_KIND` プロパティに指定されるレポートオブジェクトの種類は、次のいずれかの値である必要があります。

- `CrystalReport`
- `WebI`
- `FullClient`
- `FullClientAddin`
- `FullClientTemplate`

> **注記**
>
> パブリケーション内に複数のレポートドキュメントが存在する場合は、すべてのレポートドキュメントの種類が同じである必要があります。

静的ドキュメントは、レポートドキュメント以外のドキュメントです。Excel スプレッドシート、画像、PDF などのドキュメントがサポートされています。パブリケーションには、静的ドキュメントをいくつでも含めることができます。

関連情報

パブリケーションにレポートドキュメントを追加するには [176 ページ]
パブリケーションに静的ドキュメントを追加するには [177 ページ]

5.6.3.2 パブリケーションにレポートドキュメントを追加するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを com.businessobjects.sdk.desktop.plugin.publication.IPublication にキャストします。クエリーには、SI_PUBLICATION_DOCUMENTS プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND + "' AND SI_NAME =
'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. クエリーを実行して、パブリケーションに追加するレポートオブジェクトを取得します。クエリーには、SI_PROCESSINFO プロパティを入れる必要があります。

   パブリケーションは、Crystal レポートおよび Web Intelligence ドキュメントをサポートすることができます。レポートオブジェクトの種類は、次のいずれかの値にする必要があります。

   - CrystalReport
   - WebI
   - FullClient
   - FullClientAddin
   - FullClientTemplate

```
String documentQuery = "SELECT SI_ID, SI_PROCESSINFO FROM CI_INFOOBJECTS "
  + "WHERE SI_NAME='World Sales Report' AND SI_INSTANCE = 0";
IInfoObjects documents = infostore.query(documentQuery);
IInfoObject document = (IInfoObject) documents.get(0);
```

4. パブリケーションオブジェクトの getDocuments メソッドを呼び出して、コレクションにレポートオブジェクト ID を追加します。

```
Collection publicationDocuments = publication.getDocuments();
Integer documentID = new Integer(document.getID());
publicationDocuments.add(documentID);
```

5. パブリケーションオブジェクトの setDocumentProcessingInfo メソッドを使用して、レポートオブジェクトのドキュメント処理情報をパブリケーションにコピーします。

   setDocumentProcessingInfo の最初のパラメータは、レポートドキュメントの ID です。2 番目のパラメータは、ドキュメントの種類です。3 番目のパラメータは、処理情報を含むプロパティ バッグです。

```
publication.setDocumentProcessingInfo(documentID, document.getKind(),
document.getProcessingInfo().properties());
```

6. パブリケーションを CMS リポジトリに保存します。

```
publication.save();
```

例

```
public void addReportToPublication(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
    + "WHERE SI_KIND = '" + IPublication.KIND + "' AND SI_NAME = 'MyPublication'
AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String documentQuery = "SELECT SI_ID, SI_PROCESSINFO FROM CI_INFOOBJECTS "
    + "WHERE SI_NAME='World Sales Report' AND SI_INSTANCE = 0";
  IInfoObjects documents = infostore.query(documentQuery);
  IInfoObject document = (IInfoObject) documents.get(0);

  Collection publicationDocuments = publication.getDocuments();
  Integer documentID = new Integer(document.getID());
  publicationDocuments.add(documentID);

  publication.setDocumentProcessingInfo(documentID, document.getKind(),
document.getProcessingInfo().properties());

  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `java.util.Collection`

関連情報

レポートドキュメントと静的ドキュメント [175 ページ]
パブリケーションに静的ドキュメントを追加するには [177 ページ]

5.6.3.3 パブリケーションに静的ドキュメントを追加するには

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. CMS リポジトリに対してクエリーを実行して、パブリケーションオブジェクトを取得します。クエリーには、SI_PUBLICATION_DOCUMENTS および SI_SCHEDULEINFO プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_SCHEDULEINFO "
  + "FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND
  + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. クエリーを実行して、パブリケーションに追加する静的ドキュメントオブジェクトを取得します。

```
String staticDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='a_jpg_image'";
IInfoObjects staticDocuments = infostore.query(staticDocumentQuery);
IInfoObject staticDocument = (IInfoObject) staticDocuments.get(0);
```

4. パブリケーションオブジェクトの getDocuments メソッドを呼び出して、コレクションに静的ドキュメントの ID を追加します。

```
Collection publicationDocuments = publication.getDocuments();
Integer staticDocumentID = new Integer(staticDocument.getID());
publicationDocuments.add(staticDocumentID);
```

5. パブリケーションのスケジュール情報から宛先を取得します。少なくとも 1 つの宛先がパブリケーションに合わせて設定されている必要があります。

```
ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
IDestinations destinations = schedulingInfo.getDestinations();
```

6. 適切な宛先の静的ドキュメントコレクションに静的ドキュメント ID を追加します。

```
Iterator destinationsIter = destinations.iterator();
while(destinationsIter.hasNext())
{
  IDestination destination = (IDestination) destinationsIter.next();
  IDestinationStaticDocuments destinationDocs =
destination.getDestinationStaticDocuments();
  IDestinationStaticDocument newStaticDoc = destinationDocs.add();
  newStaticDoc.setStaticDocumentID(staticDocumentID);
}
```

7. パブリケーションを保存します。

```
publication.save();
```

例

```
public void addStaticDocumentToPublication(IEnterpriseSession enterpriseSession)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_SCHEDULEINFO "
    + "FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND
    + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);
```

```
  String staticDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='a_jpg_image'";
  IInfoObjects staticDocuments = infostore.query(staticDocumentQuery);
  IInfoObject staticDocument = (IInfoObject) staticDocuments.get(0);

  Collection publicationDocuments = publication.getDocuments();
  Integer staticDocumentID = new Integer(staticDocument.getID());
  publicationDocuments.add(staticDocumentID);

  ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
  IDestinations destinations = schedulingInfo.getDestinations();

  Iterator destinationsIter = destinations.iterator();
  while(destinationsIter.hasNext())
  {
    IDestination destination = (IDestination) destinationsIter.next();
    IDestinationStaticDocuments destinationDocs =
destination.getDestinationStaticDocuments();
    IDestinationStaticDocument newStaticDoc = destinationDocs.add();
    newStaticDoc.setStaticDocumentID(staticDocumentID);
  }
  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.publication.IPublication
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IDestination
- com.crystaldecisions.sdk.occa.infostore.IDestinationStaticDocument
- com.crystaldecisions.sdk.occa.infostore.IDestinationStaticDocuments
- com.crystaldecisions.sdk.occa.infostore.IDestinations
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo
- java.util.Collection
- java.util.Iterator

関連情報

出力先および出力形式を設定するには [194 ページ]
レポートドキュメントと静的ドキュメント [175 ページ]
パブリケーションにレポートドキュメントを追加するには [176 ページ]

5.6.4 パブリケーションへの受信者の追加

5.6.4.1 Enterprise 受信者と動的受信者

パブリケーションは、2 種類の受信者に配信できます。

- Enterprise 受信者。
- 動的受信者。

Enterprise 受信者は、SAP BusinessObjects Business Intelligence platform にアクセスでき、パブリケーションに対して少なくとも読み取り許可が与えられている個人です。これらの受信者は、通常、マネージド出力先を通して配信されたパブリケーションを個人用受信ボックスに受信します。

動的受信者は、SAP BusinessObjects Business Intelligence platform 内にユーザーアカウントを持っていない個人です。これらの受信者は、各受信者の一意の識別子、電子メールアドレス、およびプロファイル情報（ある場合）を提供する外部データソースを通して識別されます。動的受信者は、通常、SMTP 出力先を通して配信されたパブリケーションを電子メールで受信します。

このセクションでは、これらの受信者をパブリケーションに含めるさまざまな方法について説明します。

関連情報

パブリケーションにユーザーを追加するには [180 ページ]
パブリケーションにグループを追加するには [182 ページ]
パブリケーションに動的受信者を追加するには [183 ページ]

5.6.4.2 パブリケーションにユーザーを追加するには

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを `com.businessobjects.sdk.desktop.plugin.publication.IPublication` にキャストします。クエリーには、`SI_PRINCIPALS` プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
  + IPublication.KIND
  + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. クエリーを実行して、パブリケーションに追加するユーザーを取得します。

```
String userQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
  + IUser.KIND + "' AND SI_NAME = 'Administrator'";
IInfoObjects users = infostore.query(userQuery);
IInfoObject user = (IInfoObject) users.get(0);
```

4. パブリケーションオブジェクトの subscribe メソッドを呼び出します。パラメータをユーザーオブジェクトの ID に設定します。

> ⓘ 注記
>
> ユーザーがパブリケーションを購読するには、そのパブリケーションに対する「オブジェクトの表示」システム権限が必要です。

```
Integer userID = new Integer(user.getID());
publication.subscribe(userID);
```

5. パブリケーションを保存します。

```
publication.save();
```

例

```
public void addUserToPublication(IEnterpriseSession enterpriseSession, String
username) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
    + IPublication.KIND
    + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String userQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
    + IUser.KIND + "' AND SI_NAME = '" + username + "'";
  IInfoObjects users = infostore.query(userQuery);
  IInfoObject user = (IInfoObject) users.get(0);

  Integer userID = new Integer(user.getID());
  publication.subscribe(userID);

  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.publication.IPublication
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.user.IUser

関連情報

パブリケーションにグループを追加するには [182 ページ]

5.6.4.3 パブリケーションにグループを追加するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを com.businessobjects.sdk.desktop.plugin.publication.IPublication にキャストします。クエリーには、SI_PRINCIPALS プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
  + IPublication.KIND + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. クエリーを実行して、パブリケーションに追加するユーザーグループを取得します。

> **注記**
>
> ユーザーがパブリケーションを購読するには、そのパブリケーションに対する「オブジェクトの表示」システム権限が必要です。

```
String groupQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
  + IUserGroup.KIND + "' AND SI_NAME = 'Administrators'";
IInfoObjects groups = infostore.query(groupQuery);
IInfoObject group = (IInfoObject) groups.get(0);
```

4. パブリケーションオブジェクトの主体コレクションを取得し、グループの ID をそのコレクションに追加します。

```
Integer groupID = new Integer(group.getID());
Collection principals = publication.getPrincipals();
principals.add(groupID);
```

5. パブリケーションを保存します。

```
publication.save();
```

例

```
public void addGroupToPublication(IEnterpriseSession enterpriseSession, String
groupName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

```
  String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
    + IPublication.KIND + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String groupQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
    + IUserGroup.KIND + "' AND SI_NAME = '" + groupName + "'";
  IInfoObjects groups = infostore.query(groupQuery);
  IInfoObject group = (IInfoObject) groups.get(0);

  Integer groupID = new Integer(group.getID());
  Collection principals = publication.getPrincipals();
  principals.add(groupID);

  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup`
- `java.util.Collection`

関連情報

パブリケーションにユーザーを追加するには [180 ページ]

5.6.4.4 パブリケーションに動的受信者を追加するには

- 動的受信者データを含む Crystal Reports または Web Intelligence ドキュメントが必要です。ドキュメントには以下が含まれます。
  - 一意の識別子を含む列。これらの識別子は公開エンジンによって内部的に使用され、パブリケーションのコンテンツとしては表示されません。
  - 受信者の電子メールアドレスを含む列。(パブリケーションが SMTP 出力先に配信される場合、この列は必須です。そうでない場合、この列はオプションです。)
  - 各受信者のフルネームを含む列。(オプション)
  - パーソナライゼーション情報を含む列。(オプション)

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを com.businessobjects.sdk.desktop.plugin.publication.IPublication にキャストします。クエリーには、SI_PUBLICATION_EVENT_HANDLERS プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND + "' "
  + "AND SI_NAME = 'MyPublication' "
  + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. CMS リポジトリに対して、動的受信者情報を含むドキュメントを取得するためのクエリーを実行します。

```
String providerDocumentName = "HowTo_CreatePublication_DynamicRecipients";
String providerDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='" + providerDocumentName + "'";
IInfoObjects providerDocuments = infostore.query(providerDocumentQuery);
IInfoObject providerDocument = (IInfoObject) providerDocuments.get(0);
int providerDocumentID = providerDocument.getID();
```

4. パブリケーションに関連付けられているパブリケーションイベントハンドラのコレクションを取得し、動的受信者を取得するためのハンドラを追加します。

```
IPublicationEventHandlers handlers =
publication.getPublicationEventHandlers();
IPublicationEventHandler handler =
handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);
```

パブリケーションイベントハンドラは、公開処理をカスタマイズするためのパブリケーション拡張（プラグイン）を呼び出すために使用されます。IPublicationEventHandlers コレクションの addPublicationEventHandler メソッドは、追加するイベントハンドラのタイプを指定する 1 つのパラメータを受け取ります。CePropertyID.SI_ON_READ_RECIPIENTS 定数は、パブリケーションが受信者のリストを作成するとイベントハンドラが呼び出されるように指定します。

5. 動的受信者データプロバイダーを使用するようにパブリケーションイベントハンドラを設定し、動的受信者情報を含むドキュメントの ID を指定します。

```
IPublicationEventHandlerInfo handlerInfo = handler.add();
handlerInfo.setPluginClassName("com.businessobjects.publisher.dynamicrecipient
s.crystalreports.CRDataProvider");
handlerInfo.setDynamicDataDocumentID(providerDocumentID);
```

IPublicationEventHandlerInfo の setPluginClassName メソッドは、イベントを処理するために呼び出されるデータプロバイダーの完全修飾クラス名を設定します。動的受信者を読み取るためのデータプロバイダーとして、次の 2 つの組み込みデータプロバイダーが用意されています。

- Crystal Reports ドキュメントから動的受信者情報を読み取る com.businessobjects.publisher.dynamicrecipients.crystalreports.CRDataProvider クラス。
- Web Intelligence ドキュメントから動的受信者情報を読み取る com.businessobjects.publisher.dynamicrecipients.rebean.REDataProvider クラス。

また、カスタム動的受信者データプロバイダーを使用することもできます。

6. プロファイル値マッピングコレクションを取得し、動的受信者ドキュメント内で各動的受信者の名前、氏名、および電子メールアドレスを示す列を指定します。

```
IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
```

```
valueMappings.setName("person.ID");
valueMappings.setFullName("person.full_name");
valueMappings.setEmailAddress("person.email");
```

これらの設定は公開時に使用されて、%SI_OWNER%、%SI_USERFULLNAME%、および %SI_EMAIL_ADDRESS% プレースホルダの値を提供します。

> **注記**
>
> setName に渡す列名は、動的受信者の一意の識別子を含む列を指定する必要があります。これらの値は内部的に使用され、パブリケーションの出力には含まれません。
>
> setEmailAddress に渡す列名は、各動的受信者の有効な電子メールアドレスを含む列を指定する必要があります。

7. パブリケーションを保存します。

```
publication.save();
```

例

次の例は、Crystal Reports ドキュメントから動的受信者を読み取り、それらをパブリケーションに追加します。

```
public void addDynamicRecipientsToPublication(IEnterpriseSession
enterpriseSession, String publicationName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
    + "WHERE SI_KIND = '" + IPublication.KIND + "' "
    + "AND SI_NAME = '" + publicationName + "' "
    + "AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String providerDocumentName = "HowTo_CreatePublication_DynamicRecipients";
  String providerDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='" + providerDocumentName + "'";
  IInfoObjects providerDocuments = infostore.query(providerDocumentQuery);
  IInfoObject providerDocument = (IInfoObject) providerDocuments.get(0);
  int providerDocumentID = providerDocument.getID();

  IPublicationEventHandlers handlers = publication.getPublicationEventHandlers();
  IPublicationEventHandler handler =
handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);

  IPublicationEventHandlerInfo handlerInfo = handler.add();

handlerInfo.setPluginClassName("com.businessobjects.publisher.dynamicrecipients.c
rystalreports.CRDataProvider");
  handlerInfo.setDynamicDataDocumentID(providerDocumentID);

  IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
  valueMappings.setName("person.ID");
  valueMappings.setFullName("person.full_name");
  valueMappings.setEmailAddress("person.email");
  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDynaRecipientProfileValueMappings`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandler`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlerInfo`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlers`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CePropertyID`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

関連情報

動的受信者データプロバイダー [203 ページ]

5.6.5 パブリケーションドキュメントの個人用カスタマイズ

5.6.5.1 Enterprise 受信者向けにドキュメントをパーソナライズする

Enterprise 受信者のドキュメントをパーソナライズするには、プロファイルが存在し、ユーザーにプロファイル値が割り当てられている必要があります。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを `com.businessobjects.sdk.desktop.plugin.publication.IPublication` にキャストします。クエリーには、`SI_PUBLICATION_DOCUMENTS` および `SI_PUBLICATION_DOCUMENTPROFILE_TARGETS` プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND + "' "
  + "AND SI_NAME = 'MyPublication"' "
  + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. パブリケーションオブジェクトの getSchedulableDocuments メソッドを呼び出し、コレクションからレポートドキュメントの ID を取得します。

```
Collection documents = publication.getSchedulableDocuments();
IInfoObject document = (IInfoObject) documents.toArray()[0];
Integer documentID = new Integer(document.getID());
```

4. クエリーを実行して、レポートに追加するプロファイルを取得します。

```
String profileQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
  + IProfile.KIND + "' AND SI_NAME = 'country'";
IInfoObjects profiles = infostore.query(profileQuery);
IInfoObject profile = (IInfoObject) profiles.get(0);
```

5. パブリケーションオブジェクトの getPublicationDocumentProfileTargets メソッドを呼び出します。

   これは、パブリケーションに関連付けられたプロファイルターゲットが入った IPublicationDocumentProfileTargets コレクションを返します。各プロファイルターゲットには、パブリケーション内のレポートドキュメント用の変数マッピングのコレクションが入っています。

```
IPublicationDocumentProfileTargets targets =
publication.getPublicationDocumentProfileTargets();
```

6. プロファイルターゲットコレクションの add メソッドを呼び出します。引数をレポートドキュメントの ID に設定します。

   これは、指定されたレポートドキュメントの新しいプロファイルターゲットを作成し、コレクションに追加します。

```
IPublicationDocumentProfileTarget target = targets.add(documentID.intValue());
```

7. ターゲットオブジェクトの getVariableMappings メソッドを呼び出します。

   これは、レポートドキュメントに関連付けられた変数マッピングが入った IPublicationDocumentVariableMappings コレクションを返します。

```
IPublicationDocumentVariableMappings variableMappings =
target.getVariableMappings();
```

8. 変数マッピングコレクションの add メソッドを呼び出します。

   これは、新しい変数マッピングを作成し、コレクションに追加します。

```
IPublicationDocumentVariableMapping variableMapping = variableMappings.add();
```

9. 変数マップオブジェクトの setVariableName メソッドを呼び出します。引数を、プロファイル値と比較する値が入ったレポートドキュメント内の列の名前に設定します。

   > ⓘ 注記
   >
   > Crystal Reports の列名は、中かっこで囲む必要があります。たとえば、列名が **Customer.Country** の場合は、変数名を **{Customer.Country}** に設定します。

```
variableMapping.setVariableName("{Customer.Country}");
```

10. 変数マップオブジェクトの setProfileID メソッドを呼び出します。引数をプロファイルの ID に設定します。

```
variableMapping.setProfileID(profile.getID());
```

11. パブリケーションを保存します。

```
publication.save();
```

例

次のコードは、プロファイルをパブリケーション内のレポートドキュメントに追加し、そのプロファイルをレポート内の **{Customer.Country}** 列にマップします。

```
public void personalizeDocumentForEnterpriseRecipients(IEnterpriseSession
enterpriseSession, String publicationName, String profileName) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
      + "WHERE SI_KIND = '" + IPublication.KIND + "' "
      + "AND SI_NAME = '" + publicationName + "' "
      + "AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  Collection documents = publication.getSchedulableDocuments();
  IInfoObject document = (IInfoObject) documents.toArray()[0];
  Integer documentID = new Integer(document.getID());

  String profileQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
    + IProfile.KIND + "' AND SI_NAME = '" + profileName + "'";
  IInfoObjects profiles = infostore.query(profileQuery);
  IInfoObject profile = (IInfoObject) profiles.get(0);

  IPublicationDocumentProfileTargets targets =
publication.getPublicationDocumentProfileTargets();
  IPublicationDocumentProfileTarget target = targets.add(documentID.intValue());

  IPublicationDocumentVariableMappings variableMappings =
target.getVariableMappings();
  IPublicationDocumentVariableMapping variableMapping = variableMappings.add();
  variableMapping.setVariableName("{Customer.Country}");
  variableMapping.setProfileID(profile.getID());

  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.profile.IProfile
- com.businessobjects.sdk.plugin.desktop.publication.IPublication
- com.businessobjects.sdk.plugin.desktop.publication.IPublicationDocumentProfileTarget
- com.businessobjects.sdk.plugin.desktop.publication.IPublicationDocumentProfileTargets
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject

- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.common.IPublicationDocumentVariableMapping
- com.crystaldecisions.sdk.plugin.desktop.common.IPublicationDocumentVariableMappings
- java.util.Collection

関連情報

パブリケーションに動的受信者を追加するには [183 ページ]
動的受信者向けにドキュメントをパーソナライズする [189 ページ]

5.6.5.2 動的受信者向けにドキュメントをパーソナライズする

動的受信者向けにドキュメントをパーソナライズするには、動的受信者を含むパブリケーションが必要です。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを com.businessobjects.sdk.desktop.plugin.publication.IPublication にキャストします。クエリーには、SI_PUBLICATION_DOCUMENTS および SI_DYNAMIC_RECIPIENTS_PROFILE_VALUE_MAPPINGS プロパティを入れる必要があります。

```
 String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_DYNAMIC_RECIPIENTS_PROFILE_VALUE_MAPPINGS "
  + " FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND + "' "
  + " AND SI_NAME = '" + publicationName + "' " + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. パブリケーションオブジェクトの getSchedulableDocuments メソッドを呼び出し、コレクションからレポートドキュメントの ID を取得します。

```
Collection reportDocuments = publication.getSchedulableDocuments();
IInfoObject reportDocument = (IInfoObject) reportDocuments.toArray()[0];
Integer reportDocumentID = reportDocument.getID();
```

4. パブリケーションの動的受信者プロファイル値マッピングのコレクションを取得し、動的受信者ドキュメント内で各動的受信者の名前、氏名、および電子メールアドレスを示す列を指定します。

```
IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
valueMappings.setName("person.given_name");
valueMappings.setFullName("person.family_name");
valueMappings.setEmailAddress("person.email");
```

5. プロバイダー列コレクションを取得し、レポートドキュメントのプロファイルと比較するプロファイル値が含まれている動的受信者の列を追加します。

```
IDynamicRecipientProviderColumns providerColumns =
valueMappings.getProviderColumns();
providerColumns.addProviderColumn(1, "person.given_name");
providerColumns.addProviderColumn(2, "person.family_name");
providerColumns.addProviderColumn(3, "person.email");
providerColumns.addProviderColumn(4, "person.country");
```

   IDynamicRecipientProviderColumns コレクションの addProviderColumn メソッドは、2 つ目パラメータを受け取ります。最初のパラメータは、コレクション内の列を識別するための 1 から始まるインデックスです。2 番目のパラメータは、動的受信者プロバイダードキュメント内の列名を指定します。

6. ドキュメントごとのプロファイルマッピングを取得します。

   このコレクションは、動的受信者プロファイル値マッピングのコレクションを保存するために使用されます。各コレクションには、パブリケーションの 1 つのスケジュール可能ドキュメントのプロファイル値マッピングが含まれています。

```
IDynamicRecipientPerDocProfileMappings perDocProfileMappings =
valueMappings.getPerDocProfileValueMappings();
```

7. ドキュメントプロファイルマッピングコレクションを追加します。マッピングのソースドキュメント ID をスケジュール可能ドキュメントの ID に設定します。

```
IDynamicRecipientPerDocProfileMapping mapping = perDocProfileMappings.add();
mapping.setSourceDocumentID(reportDocumentID);
```

8. ドキュメントプロファイルマッピングに関連付けられた変数マッピングコレクションを取得し、それに変数マッピングを追加します。

   IDynamicRecipientVariableMappings の addVariableMapping メソッドは、3 つのパラメータを受け取ります。最初のパラメータは、IDynamicRecipientProviderColumns コレクション内のプロバイダー列のインデックスを指定し、2 つ目は動的受信者プロバイダードキュメント内の変数の名前です。3 番目のパラメータは、最初のパラメータと同じ値に設定する必要があります。

```
IDynamicRecipientVariableMappings varMappings =
mapping.getDynamicRecipientVariableMappings();
varMappings.addVariableMapping(4, "{Customer.Country}", 4);
```

> ⓘ 注記
>
> 変数名は、ソースドキュメントと同じ構文を使用する必要があります。たとえば、Crystal Reports の列名は、中かっこで囲む必要があります。

9. パブリケーションを保存します。

```
publication.save();
```

例

```
public void personalizeDocumentForDynamicRecipients(IEnterpriseSession
enterpriseSession, String publicationName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

```
  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_DYNAMIC_RECIPIENTS_PROFILE_VALUE_MAPPINGS "
    + " FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND + "' "
    + " AND SI_NAME = '" + publicationName + "' " + "AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  Collection reportDocuments = publication.getSchedulableDocuments();
  IInfoObject reportDocument = (IInfoObject) reportDocuments.toArray()[0];
  Integer reportDocumentID = reportDocument.getID();

  IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
  valueMappings.setName("person.given_name");
  valueMappings.setFullName("person.family_name");
  valueMappings.setEmailAddress("person.email");

  IDynamicRecipientProviderColumns providerColumns =
valueMappings.getProviderColumns();
  providerColumns.addProviderColumn(1, "person.given_name");
  providerColumns.addProviderColumn(2, "person.family_name");
  providerColumns.addProviderColumn(3, "person.email");
  providerColumns.addProviderColumn(4, "person.country");

  IDynamicRecipientPerDocProfileMappings perDocProfileMappings =
valueMappings.getPerDocProfileValueMappings();

  IDynamicRecipientPerDocProfileMapping mapping = perDocProfileMappings.add();
  mapping.setSourceDocumentID(reportDocumentID);

  IDynamicRecipientVariableMappings varMappings =
mapping.getDynamicRecipientVariableMappings();
  varMappings.addVariableMapping(4, "{Customer.Country}", 4);
  publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.publication.IDynamicRecipientPerDocProfileMapping
- com.businessobjects.sdk.plugin.desktop.publication.IDynamicRecipientPerDocProfileMappings
- com.businessobjects.sdk.plugin.desktop.publication.IDynamicRecipientProviderColumns
- com.businessobjects.sdk.plugin.desktop.publication.IPublication
- com.businessobjects.sdk.plugin.desktop.publication.IPublicationDynaRecipientProfileValueMappings
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.common.IDynamicRecipientVariableMappings
- java.util.Collection

関連情報

パブリケーションに動的受信者を追加するには [183 ページ]
動的受信者データプロバイダー [203 ページ]

5.6.5.3 プロファイルの作成と管理

この JSP の例は、プロファイルを作成および管理する方法を示します。

例

```
<!-- begin JSP segment -->
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.businessobjects.sdk.plugin.desktop.profile.*" %>
<%@ page import="com.businessobjects.sdk.plugin.desktop.common.*" %>
<%@ page import="com.crystaldecisions.sdk.exception.SDKException" %>
<%@ page import="com.crystaldecisions.sdk.occa.infostore.*" %>
<%@ page import="com.crystaldecisions.sdk.framework.*" %>
<%@ page import="java.util.*"%>
<%!
/* Creates a new profile.
 *
 * Parameters:
 *     iStore - the IInfoStore object from the current session.
 *  name - The name of the profile.
 *  description - The description of the profile.
 *
 * Returns:
 *  String - the output String that contains the relevant information.
 */
String addProfile(IInfoStore iStore, String name, String description)
{
    String outString = "";
    try
    {
        /* Create a new profile object. */
        IInfoObjects newcollection = iStore.newInfoObjectCollection();
        IProfile newProfile = (IProfile)newcollection.add("Profile");

        // Set the profile object's InfoObject properties.
        newProfile.setTitle(name);
        newProfile.setDescription(description);

        // Save the new profile to the CMS.
        iStore.commit(newcollection);

        outString = "<B>Profile created.</B><BR>" +
                    "<B>Name:</B> " + name + "<BR>" +
                    "<B>Description:</B> " + description + "<BR><BR>";
    }
    catch(SDKException e)
    {
        throw new Error("An error has occurred: "
                        + e.getMessage());
    }
```

```
    return outString;
}
/* Assigns a profile value to a group.
 *
 * Parameters:
 *     iStore - the IInfoStore object from the current session.
 *  profileName - The name of the profile.
 *  groupName - The name of the group.
 *  profileValue - The profile value.
 *
 * Returns:
 *  String - the output String that contains the relevant information.
 */
String assignProfileValueToGroup(IInfoStore iStore, String profileName, String
groupName, String profileValue)
{
    String outString = "";
    try
    {
        // Retrieve the profile ID from the profile name.
        IInfoObjects profiles = iStore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = 'Profile' AND SI_NAME = '" + profileName + "'");
        IProfile profile = (IProfile) profiles.get(0);
        int profileID = profile.getID();

        // Retrieve the group as an ISystemPrincipal object.
        IInfoObjects groups = iStore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = 'UserGroup' AND SI_NAME = '" + groupName + "'");
        ISystemPrincipal groupPrincipal = (ISystemPrincipal) groups.get(0);

        // Create the association between principal and profile.
        IProfileValues profileValues = groupPrincipal.getProfileValues();
        IProfileValue profileValueObject = profileValues.add(profileID);

        // Set the profile value.
        profileValueObject.setFormula(profileValue);

        // Save the changes to the CMS.
        iStore.commit(groups);

        outString = "<B>Profile value assigned to group.</B><BR>" +
                    "<B>Group:</B> " + groupName + "<BR>" +
                    "<B>Value:</B> " + profileValue + "<BR><BR>";
    }
    catch(SDKException e)
    {
        throw new Error("An error has occurred: "
                        + e.getMessage());
    }

    return outString;
}
%>
<!-- end JSP segment -->
<!-- begin HTML segment -->
<html>
<head>
</head>
<body>
<%
/* Retrieve the IInfoStore object from the current session.
 */
IInfoStore iStore = (IInfoStore) session.getAttribute("InfoStore");
/* Enter your profile name here. */
String PROFILE_NAME = "new_profile";
/* Enter your profile description here. */
String PROFILE_DESCRIPTION = "new_description";
/* Enter your group name here. */
```

```
String GROUP_NAME = "group_name";
/* Enter your profile value here. */
String PROFILE_VALUE = "=¥"Vancouver¥"";
out.println(addProfile(iStore, PROFILE_NAME, PROFILE_DESCRIPTION));
out.println(assignProfileValueToGroup(iStore, PROFILE_NAME, GROUP_NAME,
PROFILE_VALUE));
%>
</body>
</html>
<!-- end HTML segment -->
```

5.6.6 出力先および出力形式の設定

5.6.6.1 出力先および出力形式を設定するには

パブリケーション内のドキュメントに出力先と出力形式を設定するには、少なくとも 1 つのレポートドキュメントを含むパブリケーションが必要です。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを `com.businessobjects.sdk.desktop.plugin.publication.IPublication` にキャストします。クエリーには、`SI_SCHEDULEINFO` プロパティを入れる必要があります。

   ```
   String publicationQuery = "SELECT SI_ID, SI_PROCESSINFO, SI_SCHEDULEINFO, "
     + "SI_PUBLICATION_DOCUMENTS FROM CI_INFOOBJECTS "
     + "WHERE SI_KIND = '" + IPublication.KIND
     + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE=0";
   IInfoObjects publications = infostore.query(publicationQuery);
   IPublication publication = (IPublication) publications.get(0);
   ```

3. 出力先プラグインオブジェクトを取得するためのクエリーを実行します。

   ```
   String pluginQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_PARENTID=29 AND
   "
     + "SI_NAME = '" + IManaged.PROGID + "'";
   IInfoObjects plugins = infostore.query(pluginQuery);
   IDestinationPlugin inboxPlugin = (IDestinationPlugin) plugins.get(0);
   ```

4. パブリケーションのスケジュール情報に出力先を追加します。

   ```
   ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
   IDestinations destinations = schedulingInfo.getDestinations();
   IDestination inboxDestination = destinations.add(IManaged.PROGID);
   inboxDestination.setFromPlugin(inboxPlugin);
   inboxDestination.setDeliverPerUser(true);
   ```

5. スケジュール可能なドキュメントのコレクションからレポートドキュメントを取得します。

   ```
   Collection documents = publication.getSchedulableDocuments();
   IInfoObject document = (IInfoObject) documentIter.next();
   Integer documentID = new Integer(document.getID());
   ```

6. 出力先に出力形式情報を追加します。

   サポートされる形式は、

   `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions.CeReportForm`

at インターフェイスで定義されています。ドキュメントの種類は、それぞれ異なる出力形式に対応します。詳細については、サポートされる出力形式 [197 ページ]を参照してください。

```
IDestinationFormat formats = inboxDestination.getDestinationFormats();
IDestinationFormat format = formats.add();
format.setSourceDocumentID(documentID);
IFormatInfo formatInfo = format.getFormatInfos().add();
formatInfo.setSourceDocumentKind(document.getKind());
formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);
```

7. ドキュメント処理情報をレポートからパブリケーションにコピーします。

```
IProcessingPublicationInfo processingPubInfo =
        (IProcessingPublicationInfo)
publication.getDocumentProcessingInfoObject(documentID);
IInfoObject reportDoc = (IInfoObject) processingPubInfo;
IFormatInfo formatInfo = processingPubInfo.getFormatInfos().add();
formatInfo.setSourceDocumentKind(document.getKind());
formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);
publication.setDocumentProcessingInfo(documentID, document.getKind(),
        reportDoc.getProcessingInfo().properties());
```

パブリケーションオブジェクトの `setDocumentProcessingInfo` メソッドを使用して、レポートのドキュメント処理情報をパブリケーションにコピーします。`setDocumentProcessingInfo` の最初のパラメータは、レポートドキュメントの ID です。2 番目のパラメータは、ドキュメントの種類です。3 番目のパラメータは、処理情報を含むプロパティ バッグです。

8. パブリケーションを保存します。

```
publication.save();
```

例

次の例は、各ドキュメントを受信ボックス出力先に配信し、出力形式を PDF に設定するようにパブリケーションを設定します。

```
public void configurePublicationDestination(IEnterpriseSession
enterpriseSession, String publicationName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String publicationQuery = "SELECT SI_ID, SI_PROCESSINFO, SI_SCHEDULEINFO, "
    + "SI_PUBLICATION_DOCUMENTS FROM CI_INFOOBJECTS "
    + "WHERE SI_KIND = '" + IPublication.KIND
    + "' AND SI_NAME = '" + publicationName + "' AND SI_INSTANCE=0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String pluginQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_PARENTID=29 AND "
    + "SI_NAME = '" + IManaged.PROGID + "'";
  IInfoObjects plugins = infostore.query(pluginQuery);
  IDestinationPlugin inboxPlugin = (IDestinationPlugin) plugins.get(0);

  ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
  IDestinations destinations = schedulingInfo.getDestinations();
  IDestination inboxDestination = destinations.add(IManaged.PROGID);
  inboxDestination.setFromPlugin(inboxPlugin);
  inboxDestination.setDeliverPerUser(true);
```

```
  Collection documents = publication.getSchedulableDocuments();
  Iterator documentIter = documents.iterator();

  while(documentIter.hasNext())
  {
    IInfoObject document = (IInfoObject) documentIter.next();
    Integer documentID = new Integer(document.getID());

    IDestinationFormat format = inboxDestination.getDestinationFormats().add();
    format.setSourceDocumentID(documentID);
    IFormatInfo formatInfo = format.getFormatInfos().add();
    formatInfo.setSourceDocumentKind(document.getKind());
    formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);

    IProcessingPublicationInfo processingPubInfo =(IProcessingPublicationInfo)
publication.getDocumentProcessingInfoObject(documentID);
    IInfoObject reportDoc = (IInfoObject) processingPubInfo;
    formatInfo = processingPubInfo.getFormatInfos().add();
    formatInfo.setSourceDocumentKind(document.getKind());
    formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);
    publication.setDocumentProcessingInfo(documentID, document.getKind(),
    reportDoc.getProcessingInfo().properties());
   }
   publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.publication.IPublication
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IDestination
- com.crystaldecisions.sdk.occa.infostore.IDestinationFormat
- com.crystaldecisions.sdk.occa.infostore.IDestinationPlugin
- com.crystaldecisions.sdk.occa.infostore.IDestinations
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo
- com.crystaldecisions.sdk.plugin.desktop.common.IFormatInfo
- com.crystaldecisions.sdk.plugin.desktop.common.IProcessingPublicationInfo
- com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOption
- com.crystaldecisions.sdk.plugin.destination.managed.IManaged
- java.util.Collection
- java.util.Iterator

出力先を設定したら、その出力先に配信する必要がある静的ドキュメントを追加できます。

関連情報

サポートされる出力形式 [197 ページ]

パブリケーションに静的ドキュメントを追加するには [177 ページ]

5.6.6.2 サポートされる出力形式

サポートされる出力形式は、`com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions.CeReportFormat` インターフェイスで定義されています。

次の表は、サポートされるすべての出力形式と各出力形式をサポートするレポートドキュメントの種類を示します。

出力形式の定数	Crystal Reports	Web Intelligence
CRYSTAL_REPORT	はい	
EXCEL	はい	はい
EXCEL_DATA_ONLY	はい	
MHTML	はい	はい
PDF	はい	はい
RTF	はい	
RTF_EDITABLE	はい	
TEXT_CHARACTER_SEPARATED	はい	
TEXT_PAGINATED	はい	
TEXT_PLAIN	はい	
TEXT_TAB_SEPARATED	はい	
TEXT_TAB_SEPARATED_TEXT	はい	
USER_DEFINED	該当なし	該当なし
WORD	はい	
XML	はい	

特定の出力形式の詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム ユーザーガイドを参照してください。

関連情報

出力先および出力形式を設定するには [194 ページ]

5.6.7 パブリケーション例外のトラブルシューティング

パブリケーションを開始する前に、`IPublication` の `checkPublicationIntegrity` メソッドを呼び出して整合性チェックを実行できます。整合性チェックでは、パブリケーションオブジェクトを分析し、公開を妨げるよ

うな設定エラーがないかどうかを確認します。これは、特に、サイズが大きく長時間実行されるパブリケーションで設定エラーによる例外の発生を避けたい場合に役立ちます。

次の表で、整合性チェックや公開処理の際にスローされる一般的な例外を示し、それらの解決方法を説明します。

例外	説明および解決方法
`MultipleDocumentKinds`	この例外は、パブリケーションに複数の種類のスケジュール可能なドキュメントが含まれる場合にスローされます。パブリケーションには複数のスケジュール可能なドキュメントを含めることができますが、これらのドキュメントはすべて同じ種類にする必要があります。たとえば、1 つのパブリケーションに 2 つの Web Intelligence ドキュメントを含めることができますが、Web Intelligence ドキュメントと Crystal Reports ドキュメントを 1 つずつ含めることはできません。 この問題を解決するには、`IPublication` の `getDocuments` メソッドから返されるドキュメントのコレクションを変更して、スケジュール可能なドキュメントが 1 種類だけ（Crystal Reports ドキュメントまたは Web Intelligence ドキュメントのみ）含まれるようにします。他の種類のスケジュール可能なドキュメントを公開するには、追加のパブリケーションを作成します。 この制限は、静的ドキュメントには適用されないことに注意してください。静的ドキュメントのパブリケーションには、複数の種類のドキュメントを含めることができます。
`MissingProcessingInfo`	パブリケーションに、処理情報のないスケジュール可能なドキュメントが含まれます。 次の 2 つの解決方法が考えられます。 • ドキュメントをパブリケーションから削除します。 • `IPublication` の `setDocumentProcessingInfo` メソッドを使用して、ドキュメントに処理情報を追加します。
`InvalidFormatKind`	パブリケーションに、無効なソースドキュメントの種類を指定する形式情報が含まれます。 次の 2 つの解決方法が考えられます。 • `IFormatInfos` コレクションから無効な `IFormatInfo` オブジェクトを削除します。このエントリがドキュメント処理情報に存在する場合は、`IProcessingPublicationInfo.getFormatInfos()` を使用してコレクションを取得します。このエントリが出力先に存在する場合は、`IDestinationFormat.getFormatInfos()` を使用します。 • 無効な `IFormatInfo` オブジェクトの `setSourceDocumentKind` メソッドを使用して、種

例外	説明および解決方法
	類を CrystalReports、WebIntelligence、または DesktopIntelligence のいずれかに変更します。
`DuplicateFormatInfo`	パブリケーションに、重複するドキュメント形式情報が含まれます。 重複する種類または形式情報が含まれる `IFormatInfos` コレクションのすべてのエントリを削除します。
`InvalidDestinationDocument`	パブリケーションに、無効な出力先ドキュメントが含まれます。 次の 2 つの解決方法が考えられます。 • 無効なドキュメントを `IDestinationFormats` コレクションから削除します。 • `IPublication` の `getDocuments` メソッドから返されるコレクションにドキュメントを追加します。
`InvalidDestinationFormat`	パブリケーションに、無効な形式情報を持つ出力先ドキュメントが含まれます。 次の 2 つの解決方法が考えられます。 • 出力先に記述されている形式と一致する `IFormatInfo` エントリをドキュメントに追加します。 • `IDestinationFormats` コレクションから無効な `IFormatInfo` オブジェクトを削除します。
`DuplicateDestinationMergeFormat`	パブリケーションに、重複するマージ形式を持つ出力先が含まれます。 `IDestinationPluginArtifactFormats` コレクションから重複するマージ形式を削除します。
`InvalidDestinationStaticDocument`	パブリケーションの出力先に無効な静的ドキュメントが含まれます。 次の 2 つの解決方法が考えられます。 • 静的ドキュメントを `IDestinationStaticDocuments` コレクションから削除します。 • `IPublication` の `getDocuments` メソッドから返されるコレクションに静的ドキュメントを追加します。
`DuplicateDestinationStaticDocument`	パブリケーションの出力先に重複する静的ドキュメントが含まれます。 この問題を解決するには、`IDestinationStaticDocuments` コレクションから重複するエントリを削除します。

5.6.8 パブリケーション拡張

5.6.8.1 パブリケーション拡張とは

パブリケーション拡張によってパブリケーション処理をカスタマイズできます。たとえば、配信後パブリケーション拡張を使用して、高ボリュームパブリケーションのスコープバッチをクリーンアップできます。また、配信前パブリケーション拡張を使用して、パーソナライズされたパブリケーションの内容にアーティファクトを追加できます。

パブリケーション拡張には次の 3 種類があります。

- 動的受信者データプロバイダー。これを使用して、外部データソースからパブリケーション受信者を追加できます。
- ポスト処理（配信前）プラグイン。これを使用して、パブリケーションの内容がパーソナライズされた後の処理ロジックを追加できます。
- 配布完了（配信後）プラグイン。これを使用して、パブリケーションの内容が配信された後の処理ロジックを追加できます。

このセクションでは、さまざまなパブリケーション拡張機能について説明し、それらをプログラムで作成、デプロイ、および起動する方法について説明します。

5.6.8.2 カスタムパブリケーション拡張のビルドおよびデプロイメント

パブリケーション拡張をビルドするには、標準の SAP BusinessObjects Business Intelligence platform SDK ライブラリのほかに、クラスパスに `pub_common.jar` を含める必要があります。Windows では、このファイルのデフォルトの場所は `C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib` です。

> **注記**
>
> `PostProcessingPluginHelper` クラスを使用してポスト処理プラグインを作成する場合は、`pub_common.jar` と標準のライブラリの他に、クラスパスに `pub_processing.jar` を含める必要があります。このライブラリは、`pub_common.jar` と同じディレクトリに置かれています。

パブリケーション拡張をビルドしたら、それをパブリケーション拡張ディレクトリにデプロイします。Windows では、このディレクトリのデフォルトの場所は `C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥publishingPlugins` です。その後、Adaptive Processing Server を再起動します。

5.6.8.3 パブリケーションでパブリケーション拡張を呼び出す

パブリケーションでパブリケーション拡張を呼び出すには、拡張 jar ファイルを正しくデプロイしておく必要があります。

1. CMS リポジトリに対してクエリーを実行してパブリケーションオブジェクトを取得し、そのオブジェクトを com.businessobjects.sdk.desktop.plugin.publication.IPublication にキャストします。クエリーには、SI_PUBLICATION_DOCUMENTS、SI_PUBLICATION_EVENT_HANDLERS、および SI_SCHEDULEINFO プロパティを入れる必要があります。

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS, "
        + "SI_PUBLICATION_EVENT_HANDLERS, SI_SCHEDULEINFO FROM CI_INFOOBJECTS
"
        + "WHERE SI_KIND = '" + IPublication.KIND + "' AND SI_NAME =
'MyPublication' "
        + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

2. パブリケーションに関連付けられているパブリケーションイベントハンドラのコレクションを取得し、呼び出すパブリケーション拡張のタイプに対応するイベントハンドラを追加します。

   IPublicationEventHandlers コレクションの addPublicationEventHandler メソッドは、追加するイベントハンドラのタイプを指定する 1 つのパラメータを受け取ります。com.crystaldecisions.sdk.occa.infostore.CePropertyID に定義されている次の定数がサポートされています。

定数	説明
SI_ON_READ_RECIPIENTS	公開エンジンがパブリケーションの受信者のリストを作成するときに、動的受信者データプロバイダーを呼び出します。
SI_ON_POST_PROCESS_PERSONALIZED_DOCS	公開エンジンがパブリケーション内容のパーソナライズを完了し、それを受信者に配信する準備ができた後で、ポスト処理プラグインを呼び出します。
SI_ON_AFTER_DELIVER_SCOPE_BATCH	公開エンジンが受信者にパブリケーション内容を配信した後で、配布完了プラグインを呼び出します。

```
IPublicationEventHandlers handlers =
publication.getPublicationEventHandlers();
IPublicationEventHandler handler =

handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);
```

> ⓘ 注記
>
> パブリケーションには、タイプごとに 1 つのイベントハンドラだけを含めることができます。イベントハンドラを追加するときに、そのタイプのイベントハンドラがコレクションに既にある場合は、既存のイベントハンドラが上書きされます。

3. イベントハンドラでパブリケーション拡張名とクラス名を設定します。

   IPublicationEventHandlerInfo の setName メソッドは、このパブリケーションのコンテキストでパブリケーション拡張に関連付ける名前を設定します。(SAP BusinessObjects Business Intelligence platform の UI を使用してパブリケーションに関連付けることができる拡張の場合、これは表示される名前になります)

   IPublicationEventHandlerInfo の setPluginClassName メソッドは、イベントを処理するために呼び出されるパブリケーション拡張の完全修飾クラス名を設定します。クラス名は、ビルトイン拡張またはカスタム拡張の 1 つを指定できます。

```
String pluginClass =
```

```
"com.businessobjects.publisher.dynamicrecipients.crystalreports.CRDataProvider
";
IPublicationEventHandlerInfo handlerInfo = handler.add();
handlerInfo.setName("My Plugin");
handlerInfo.setPluginClassName(pluginClass);
```

4. パブリケーション拡張に追加情報を渡す必要がある場合は、汎用パラメータを使用します。

```
handlerInfo.setGenericParameter("parameterValue");
```

5. パブリケーション拡張がポスト処理プラグインの場合は、プラグインがサポートする出力先ごとにアーティファクトの出力形式を設定します。

```
IDestinations destinations =
publication.getSchedulingInfo().getDestinations();
Iterator destinationsIter = destinations.iterator();
while(destinationsIter.hasNext())
{
    IDestination destination = (IDestination) destinationsIter.next();
    IDestinationPluginArtifactFormats formats =
destination.getDestinationPluginArtifactFormats();
    IDestinationPluginArtifactFormat format = formats.add();
    format.setFormat(CeFormatNames.PDF);

format.setDistributionMode(IDestinationPluginArtifactFormat.CeDistributionMode
.FILTER_EXCLUDE_SOURCE_DOCUMENTS);
}
```

パブリケーション拡張は、パブリケーション処理の適切な段階で呼び出されます。これは、イベントハンドラのタイプによって決まります。

例

次の例は、パブリケーションを取得し、カスタムデータプロバイダーのパブリケーション拡張を呼び出すイベントハンドラを追加します。

```
void setCustomDynamicRecipientDataProviderForPublication(IInfoStore infostore,
String publicationName) throws SDKException
{
  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
                + "WHERE SI_KIND = '"+ IPublication.KIND + "' "+ "AND SI_NAME =
'"+ publicationName
                + "' "    + "AND SI_INSTANCE = 0";

        IInfoObjects publications = infostore.query(publicationQuery);
        IPublication publication = (IPublication) publications.get(0);
        IPublicationEventHandlers handlers =
publication.getPublicationEventHandlers();
        IPublicationEventHandler handler =
handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);

        IPublicationEventHandlerInfo handlerInfo = handler.add();
        handlerInfo.setName("MySQL Recipient Provider");

handlerInfo.setPluginClassName("com.businessobjects.sample.MySqlRecipientProvider
");

handlerInfo.setGenericParameter("localhost;3306;dbname;dbuser;dbpassword;select
* from person");
```

```
        IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
        valueMappings.setName("given_name");
        valueMappings.setFullName("family_name");
        valueMappings.setEmailAddress("email");
        publication.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDynaRecipientProfileValueMappings`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandler`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlerInfo`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlers`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.CePropertyID`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPluginArtifactFormat`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPluginArtifactFormats`
- `com.crystaldecisions.sdk.occa.infostore.IDestinations`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.common.CeFormatNames`
- `java.util.Iterator`

関連情報

カスタムパブリケーション拡張のビルドおよびデプロイメント [200 ページ]
ポスト処理プラグイン [206 ページ]
配布完了プラグイン [209 ページ]

5.6.8.4 動的受信者データプロバイダー

動的受信者データプロバイダーは、外部のデータソースから受信者を取得してパブリケーションの購読者にするパブリケーション拡張です。BI platform には、次の動的受信者データプロバイダーが組み込まれています。

- Crystal Reports ドキュメントから動的受信者情報を読み取る `com.businessobjects.publisher.dynamicrecipients.crystalreports.CRDataProvider`。
- Web Intelligence ドキュメントから動的受信者情報を読み取る `com.businessobjects.publisher.dynamicrecipients.rebean.REDataProvider`。

カスタムデータプロバイダーを作成して、他のデータソースから動的受信者情報を読み取ることができます。

> **注記**
>
> カスタム動的受信者データプロバイダーは、セントラル管理コンソールまたは BI 起動パッドからは使用できません。プログラムでパブリケーションに追加することだけができます。プログラムで動的受信者をパブリケーションに追加する方法については、パブリケーションに動的受信者を追加するには [183 ページ]を参照してください。

カスタムデータプロバイダーは、`com.businessobjects.publisher.dynamicrecipients.IDataProvider` インターフェイスを実装する Java クラスで構成されます。このインターフェイスには 4 つのメソッドがあり、これをデータプロバイダークラスで実装する必要があります。

メソッド	説明
`getRecipients`	このメソッドは、動的受信者を含む `java.sql.ResultSet` を返す必要があります。このパラメータは、拡張を呼び出すために使用されたパラメータなど、コンテキストに関する情報が含まれる `com.businessobjects.publisher.dynamicrecipients.IDataProviderContext` オブジェクトです。
`getSources`	このメソッドは、内部のデータプロバイダーによって実装されます。カスタムデータプロバイダーは、null を返すためにこのメソッドを実装する必要があります。その他のロジックを含めることはできません。
`keepAlive`	このメソッドは、一定の間隔で呼び出され、データソースに ping を実行してデータソースがアクティブであるかどうかを確認します。データソース接続を維持するために必要なすべてのコードは、このメソッド内に実装する必要があります。データソース接続が有効な場合、このメソッドは true を返す必要があります。
`close`	このメソッドを使用して、クリーンアップを実行できます。たとえば、データベース接続を閉じるために必要なすべてのコードは、このメソッド内に実装する必要があります。

`IDataProvider` インターフェイスと `IDataProviderContext` インターフェイスの詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスを参照してください。

例

次のデータプロバイダークラスは、MySQL データベースから動的受信者を取得します。汎用パラメータからセミコロンで区切られた文字列を受け取ります。文字列の形式は次のようになります。

```
<dbhost>;<port>;<dbname>;<user>;<password>;<query>
```

説明

- `dbhost` は、MySQL データベースサーバーのホスト名です。
- `port` は、MySQL データベースサーバーのポート番号です。
- `dbname` は、動的受信者が含まれるデータベースの名前です。

- user は、少なくとも指定されたデータベースへの読み取り権限を持つ MySQL ユーザーのユーザー名です。
- password は、指定されたユーザーのパスワードです。
- query は、動的受信者データを取得する SQL クエリーです。

> **注記**
>
> この例では、Connector/J 5.0 MySQL JDBC ドライバが必要です。このドライバの GPL（General Public License）バージョンは、http://www.mysql.com/products/connector/j からダウンロードできます。クラスパスに mysql-connector-java-5.0.4-bin.jar ライブラリが追加されていることを確認してください。

```
package com.businessobjects.sample;
import com.businessobjects.publisher.dynamicrecipients.*;
import com.businessobjects.foundation.logging.*;
import java.util.List;
import java.sql.*;
public class MySqlRecipientProvider implements IDataProvider {
    private ILogger logger =
LoggerManager.getLogger(MySqlRecipientProvider.class);
    private static final long serialVersionUID = 1L;
 private Connection connection;
 private Statement statement;
 private ResultSet resultSet;

 public ResultSet getRecipients(IDataProviderContext context) throws
SQLException, Exception
 {
   String param = context.getGenericParameter();
   String[] parts = param.split(";");

   if(parts.length < 6)
   {
     logger.error("Missing parameter values or incorrect parameter format. "
               + "Expected format:
<dbhost>;<port>;<dbname>;<user>;<password>;<query>");
     return null;
   }

   String host = parts[0];
   String port = parts[1];
   String dbname = parts[2];
   String user = parts[3];
   String password = parts[4];
   String query = parts[5];

   new com.mysql.jdbc.Driver();

   String dbUrl = "jdbc:mysql://" + host + ":" + port + "/" + dbname;
   connection = (Connection) DriverManager.getConnection(dbUrl, user, password);

   statement = connection.createStatement();
   resultSet = statement.executeQuery(query);
   return resultSet;
 }
 public List getSources(IDataProviderContext context) throws SQLException,
Exception
 {
   return null;
 }
 public boolean keepAlive() throws SQLException, Exception
 {
   return true;
 }
 public void close() throws SQLException, Exception
```

```
  {
    resultSet.close();
    statement.close();
    connection.close();
   }
}
```

関連情報

Enterprise 受信者と動的受信者 [180 ページ]
パブリケーションに動的受信者を追加するには [183 ページ]
カスタムパブリケーション拡張のビルドおよびデプロイメント [200 ページ]

5.6.8.5 ポスト処理プラグイン

ポスト処理プラグインは、一般にパブリケーションの内容を操作するために、公開処理のパーソナライゼーション段階中に呼び出されるパブリケーション拡張です。

ポスト処理プラグインは、プログラムで、またはセントラル管理コンソール（CMC）からパブリケーションに追加できます。

> **注記**
> BI 起動パッドアプリケーションでは、UI を使用してパブリケーションにポスト処理プラグインを指定できません。

BI platform には、次のポスト処理プラグインが組み込まれています。

- `com.businessobjects.publisher.postprocessing.plugin.ZipMergePlugin`。これは、内容を 1 つの zip アーカイブにマージします。
- `com.businessobjects.publisher.postprocessing.plugin.PdfMergePlugin`。これは、内容を 1 つの Adobe Acrobat Reader ドキュメントにマージします。

カスタムポスト処理プラグインを作成して、公開処理にカスタム動作を追加できます。

カスタムポスト処理プラグインは、`com.businessobjects.publisher.postprocessing.IPublicationPostProcessingPlugin` インターフェイスを実装する Java クラスで構成されます。このインターフェイスには 2 つのメソッドがあり、これをプラグインクラスで実装する必要があります。

メソッド	説明
`handle`	このメソッドは、各出力先とスコープに対して 1 回呼び出されます。これは、プラグインによって作成されて現在の出力先とスコープに関連付けられた受信者に配信されるアーティファクトを含む `IInfoObjects` コレクションを返す必要があります。

メソッド	説明
	このパラメータは、現在のスコープ、出力先、およびプラグインを呼び出すために使用されたパラメータなど、コンテキストに関する情報が含まれる `com.businessobjects.publisher.postprocessing.IPublicationPostProcessingContext` オブジェクトです。
`getTargetDestinations`	このメソッドは、プラグインが呼び出される出力先を示す `com.businessobjects.publisher.postprocessing.PluginTargetDestination` のコレクションを返す必要があります。

`com.businessobjects.publisher.postprocessing.PostProcessingPluginHelper` は、ポスト処理プラグインで使用できるいくつかの静的メソッドを提供します。たとえば、新しいアーティファクトを追加するには、`createInfoObject` メソッドを使用します。次のコードは、テキストファイル情報オブジェクトを作成します。

```
ITxt textInfoObject = (ITxt) PostProcessingPluginHelper.createInfoObject(context,
                ITxt.PROGID, "text/plain", null, null);
```

`IPublicationPostProcessingPlugin`、`IPublicationPostProcessingContext`、`PluginTargetDestination`、および `PostProcessingPluginHelper` の詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスを参照してください。

> ⓘ 注記
>
> `pub_common.jar` および標準の SAP BusinessObjects Business Intelligence Platform SDK ライブラリの他に、`PostProcessingPluginHelper` を使用する場合は、クラスパスに `pub_processing.jar` も含める必要があります。

例

次のポスト処理プラグインクラスは、パブリケーションをパーソナライズしたそれぞれのバージョンにテキストファイルを追加します。テキストファイルには、テキストファイル自体を除く、現在のスコープ内で作成されたアーティファクトのリストが入っています。

```
package org.example.bobj.boe.sdk.publication;
import java.util.*;
import java.io.*;
import com.crystaldecisions.sdk.occa.infostore.*;
import com.crystaldecisions.sdk.plugin.desktop.txt.*;
import com.crystaldecisions.sdk.plugin.destination.managed.*;
import com.crystaldecisions.sdk.plugin.destination.smtp.*;
import
com.businessobjects.publisher.postprocessing.IPublicationPostProcessingContext;
import
com.businessobjects.publisher.postprocessing.IPublicationPostProcessingPlugin;
import com.businessobjects.publisher.postprocessing.PluginTargetDestination;
import com.businessobjects.publisher.postprocessing.PostProcessingPluginHelper;
import com.businessobjects.publisher.common.ILoggerAdapter;
public class PubExtension implements IPublicationPostProcessingPlugin
```

```
{
  public IInfoObjects handle(IPublicationPostProcessingContext context) throws
Exception
  {
          ILoggerAdapter logger = context.getAdminLogAdapter();
          IDestination destination = context.getDestination();
          int scopeID = context.getScopeID();
          logger.info("Currently processing Scope ID " + scopeID
                  + " for destination '" + destination.getName() + "'");
          int publicationID = context.getPublication().getID();
          String filename = (String) context.getOptions();
          if (filename == null || filename.trim().length() == 0)
    {
               filename = "c:¥¥manifest-publication-" + publicationID + "-scope-"
                        + scopeID + "_" + System.currentTimeMillis() + ".txt";
          }

          ArrayList documents = context.getDocuments();
          if (documents == null || documents.size() == 0)
    {
               throw new Exception("No documents to process.");
    }

          File file = new File(filename);
          FileOutputStream fileOutputStream = new FileOutputStream(file);
          PrintWriter printWriter = new PrintWriter(fileOutputStream);
          printWriter.println("PubExtension manifest for publication with SI_ID "
                  + publicationID + ", scope ID " + scopeID);

          Iterator documentIter = documents.iterator();
          while (documentIter.hasNext())
    {
               IInfoObject document = (IInfoObject) documentIter.next();
               String message = "SI_CUID: " + document.getCUID() + ", SI_ID: "
                       + document.getID() + ", " + document.getTitle();
               printWriter.println(message);
          }
          printWriter.close();

          ArrayList<File> files = new ArrayList<File>();
          files.add(file);
          ITxt textInfoObject = (ITxt)
PostProcessingPluginHelper.createInfoObject(context,
      ITxt.PROGID, "text/plain", null, null);
          textInfoObject.getFiles().addFile(file);
          IInfoObjects newInfoObjects =
context.getInfoStore().newInfoObjectCollection();
          newInfoObjects.add(textInfoObject);
          return newInfoObjects;
     }
    public Collection getTargetDestinations() throws SDKException
 {
         PluginTargetDestination inboxDestination = new PluginTargetDestination(
                 IManaged.PROGID,

IDestinationPluginArtifactFormat.CeDistributionMode.FILTER_EXCLUDE_SOURCE_DOCUMEN
TS);
         PluginTargetDestination smtpDestination = new PluginTargetDestination(
              ISMTP.PROGID,

IDestinationPluginArtifactFormat.CeDistributionMode.FILTER_EXCLUDE_SOURCE_DOCUMEN
TS);
         PluginTargetDestination[] destinations = {inboxDestination,
smtpDestination};
         return Arrays.asList(destinations);
     }
}
```

関連情報

カスタムパブリケーション拡張のビルドおよびデプロイメント [200 ページ]

5.6.8.6 配布完了プラグイン

配布完了プラグインは、公開エンジンがスコープバッチの処理を完了し、そのスコープバッチに関連付けられている受信者にパーソナライズされたパブリケーション内容を配信した後で実行されるパブリケーション拡張です。配布完了プラグインは、プログラムで、またはセントラル管理コンソール（CMC）からパブリケーションに追加できます。

> **注記**
>
> BI 起動パッドアプリケーションでは、UI を使用してパブリケーションに配布完了プラグインを指定できません。

カスタム配布完了プラグインを作成して、スコープバッチ完了後に特定のアクションを実行できます。

カスタム配布完了プラグインは、`com.businessobjects.publisher.distribution.IDistributionCompletePlugin` インターフェイスを実装する Java クラスで構成されます。このインターフェイスには 1 つのメソッドがあり、これをプラグインクラスで実装する必要があります。

メソッド	説明
`onDistributionComplete`	このメソッドは、動的受信者を含む true を返す必要があります。パラメータは、`com.businessobjects.publisher.distribution.IDistributionCompleteContext` オブジェクトです。このオブジェクトには、現在のスコープバッチや SAP BusinessObjects Business Intelligence platform セッションなど、コンテキストに関する情報が含まれています。

`IDistributionCompletePlugin` インターフェイスと `IDistributionCompleteContext` インターフェイスの詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスを参照してください。

例

次の例は、スコープバッチに対して作成されたすべてのアーティファクトを削除し、その後スコープバッチを削除します。

```
package com.businessobjects.publisher.distribution;
import com.businessobjects.publisher.common.ILoggerAdapter;
import com.businessobjects.sdk.plugin.desktop.scopebatch.IScopeBatch;
import com.crystaldecisions.sdk.exception.SDKException;
import com.crystaldecisions.sdk.occa.infostore.IInfoObject;
import com.crystaldecisions.sdk.occa.infostore.IInfoObjects;
public class CleanScopeBatch implements IDistributionCompletePlugin
{
```

```
  public boolean onDistributionComplete(IDistributionCompleteContext context)
  {
    private ILoggerAdapter logger = context.getAdminLogAdapter();
    try
    {
      IScopeBatch scopebatch = context.getScopeBatch();
      if (scopebatch == null)
      {
        logger.error("No valid ScopeBatch object in
DistributionCompleteContext");
        return false;
      }
      StringBuffer queryString = new StringBuffer("select SI_ID from
CI_INFOOBJECTS where children(¥"SI_NAME='PublicationScopeBatch-Artifact'¥",
¥"SI_ID=");
      queryString.append(scopebatch.getID());
      queryString.append("¥")");
      logger.debug("Query for artifacts to delete: "    +
queryString.toString());
      IInfoObjects artifacts =
context.getInfoStore().query(queryString.toString());
      if (artifacts == null || artifacts.size() == 0)
      {
        logger.debug("There were no artifacts to delete for ScopeBatch ["+
scopebatch.getID() + "]");
        return false;
      }
      else
      {
        logger.info("Deleting " + artifacts.size()+ " artifacts for ScopeBatch
["    + scopebatch.getID() + "]");
      }
      for (int i = 0; i < artifacts.size(); i++)
      {
        IInfoObject artifact = (IInfoObject) artifacts.get(i);
        artifact.deleteNow();
      }
      scopebatch.deleteNow();
    }
    catch (SDKException e)
    {
      return false;
    }
    return true;
  }
}
```

関連情報

カスタムパブリケーション拡張のビルドおよびデプロイメント [200 ページ]

5.7 サーバー管理

一般に、サーバー管理タスクは、セントラル管理コンソール (CMC) Web アプリケーションで実行されます。Server Intelligence API を使用して、これらのタスクをプログラムで実行することもできます。

Server Intelligence は、BusinessObjects Enterprise XI 3.x 以降で使用されるサーバー管理フレームワークです。このフレームワークにより、SAP BusinessObjects Business Intelligence platform サーバーおよびサービスの管

理およびデプロイメントが容易になります。特に、複雑なデプロイメントで、パフォーマンスおよびフォールトトレランスを簡単に最適化できます。Server Intelligence の詳細については、*SAP BusinessObjects Business Intelligence Platform* サーバー管理ガイドを参照してください。

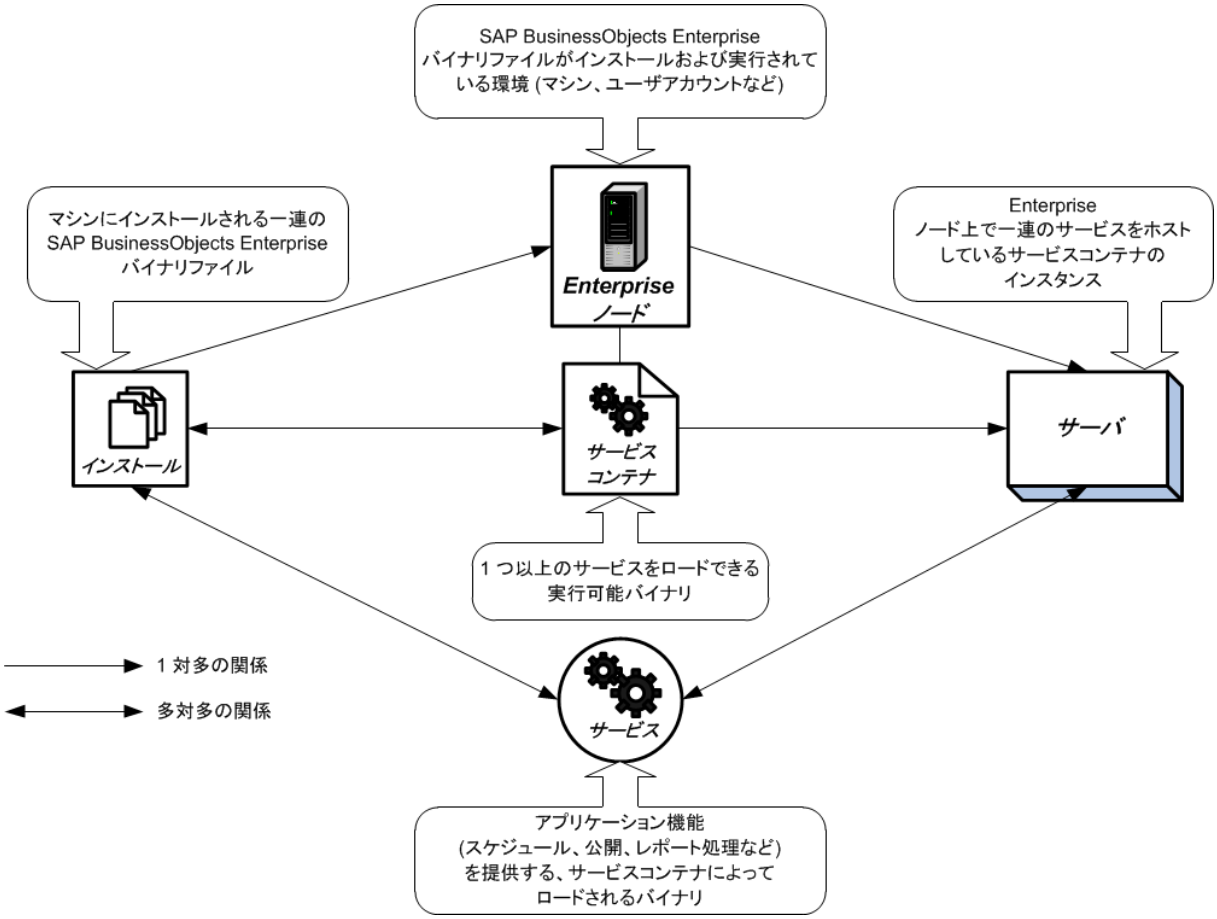
Server Intelligence フレームワークは、BI platform インストール内でサーバーを効率よく管理するための API を公開しています。このセクションでは、Server Intelligence API の使用方法を説明します。

サーバー管理に使用されるクラス

- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
  サーバーに関する一般情報を提供し、サーバーを開始、停止、再起動、有効、または無効にできます。
- `com.crystaldecisions.sdk.plugin.desktop.servergroup.IServerGroup`
  サーバーグループを管理し、そのグループに属するメンバーを指定します。

5.7.1 Server Intelligence のアーキテクチャ

次の図は、Server Intelligence アーキテクチャのコンポーネントを示します。



5.7.2 Server Intelligence のコンポーネント

BI platform インストールは、以下のコンポーネントで構成されます。

- Enterprise ノード
- インストール
- サービスコンテナ
- サービス
- サーバー

Enterprise ノード

Enterprise ノードは、BI platform インストールのコンテキストです。次の情報が含まれます。

- インストールされるバイナリの物理的な場所（ホストコンピュータとハードドライブ）。
- 基本インストールディレクトリ。
- バイナリを実行するオペレーティングシステムのユーザーアカウント。
- Server Intelligence フレームワークで使用される値のプレースホルダ。JRE `bin` ディレクトリと一時監査ログファイルが格納されるディレクトリの絶対パスが含まれます。
- Server Intelligence Agent プロセス。これは、デプロイメント内のすべてのサーバーの状態を監視します。

すべてのコンポーネントが 1 つのコンピュータにインストールされている単純なデプロイメントの場合、Enterprise ノードは 1 つだけになります。複雑なデプロイメントの場合、複数の Enterprise ノード（物理コンピュータごとに 1 つ）が存在します。

Enterprise ノードは、`com.businessobjects.sdk.plugin.desktop.enterprisenode.IEnterpriseNode` インターフェイスを使用して表されます。

インストール

インストールは、BI platform バイナリのセットです。インストールは、これらのバイナリのプラットフォーム固有の相対パスおよびファイル名のプレースホルダを定義します。

インストールは、実際のインストールを表すものではないことに注意してください。これは、バイナリとバイナリの相対パスのリストを含むマニフェストと似ています。Server Intelligence フレームワークは、この情報と Enterprise ノード内の情報を使用して、物理コンピュータ上の実際のバイナリの絶対パスを決定します。

最も単純なタイプのデプロイメントには、1 つの Enterprise ノードと、その中に 1 つのインストールだけがあります。複雑なデプロイメントは、複数の Enterprise ノードと 1 つのインストールを含むことがあります。この場合は、各ノードに同じバイナリがインストールされます。また、複雑なデプロイメントは、複数のインストールを含むこともあります。この場合は、各ノードに異なるバイナリセットがインストールされます。

BI platform インストールは、`com.businessobjects.sdk.plugin.desktop.install.IInstall` インターフェイスを使用して表されます。

サービスコンテナ

サービスコンテナは、1 つ以上のサービスを実行するように設定できる実行可能バイナリファイルです。1 つのサービスコンテナを複数のサーバーが使用できます。各サーバーは、1 つのサービスコンテナに関連付けられます。

BI platform の実行ファイルは、`com.businessobjects.sdk.plugin.desktop.servicecontainer.IServiceContainer` インターフェイスを使用して表されます。

サービス

サービスは、サービスコンテナによって読み込まれるバイナリファイルです。レポートキャッシュ、監査、レポートスケジュールなどのアプリケーション機能を提供します。サーバーは、1 つ以上のサービスをホストできます。サービスコンテナは、サーバーでホストできるサービスを決定します。

BI platform のサービスは、`com.businessobjects.sdk.plugin.desktop.service.IService` インターフェイスを使用して表されます。

サーバー

サーバーは、サービスコンテナの実行インスタンスです。サーバーには、関連付けられているサービスコンテナを起動したり、1 つ以上のサービスをホストするための構成情報が含まれます。

BI platform のサーバーは、`com.crystaldecisions.sdk.plugin.desktop.server.IServer` インターフェイスを使用して表されます。各サーバーは次の要素を保持します。

- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredContainer` オブジェクト。これは、`IServer` インターフェイスの `getConfiguredContainer` メソッドを呼び出して取得できます。`IConfiguredContainer` オブジェクトには、サーバーの起動時にサービスコンテナの実行ファイルを設定するために使用される設定情報が含まれます。
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices` コレクション。これは、`IServer` インターフェイスの `getConfiguredServices` メソッドを呼び出して取得できます。`IConfiguredServices` には、サーバーでホストされるサービスごとに 1 つの `com.businessobjects.sdk.plugin.desktop.common.IConfiguredService` オブジェクトが含まれます。各 `IConfiguredService` オブジェクトには、サーバーの起動時にサービスに渡される構成情報が含まれます。

これらのインターフェイスおよびそのメンバーの詳細については、*SAP BusinessObjects Business Intelligence* フレームワーク *Java API* リファレンスを参照してください。

5.7.3 サーバーを管理する

このセクションでは、Server Intelligence API を使用して、次のサーバー管理タスクを実行する方法について説明します。

- サーバーの起動
- サーバーの停止
- サーバーの再起動
- サーバーの複製
- サーバーのメトリックスの取得。

5.7.3.1 サーバーを開始するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

```
String serverQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. サーバーオブジェクトの setExpectedRunState を呼び出し、引数を ExpectedRunState.RUNNING に設定します。

```
server.setExpectedRunState(ExpectedRunState.RUNNING);
```

4. サーバーオブジェクトを保存します。

```
server.save();
```

これは、サーバーを実行する必要があることを示すようにサーバーの SI_EXPECTED_RUN_STATE プロパティを設定します。現在、サーバーが実行されていない場合、Server Intelligence Agent はサーバーの起動プロセスを開始します。

例

```
void startServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  server.setExpectedRunState(ExpectedRunState.RUNNING);
  server.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects

- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState

5.7.3.2 サーバーを停止するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

```
String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. サーバーオブジェクトの setExpectedRunState を呼び出し、引数を ExpectedRunState.STOPPED に設定します。

```
server.setExpectedRunState(ExpectedRunState.STOPPED);
```

4. サーバーオブジェクトを保存します。

```
server.save();
```

これは、サーバーを停止する必要があることを示すようにサーバーの SI_EXPECTED_RUN_STATE プロパティを設定します。現在、サーバーが実行されている場合、Server Intelligence Agent はサーバーの停止プロセスを開始します。

例

```
void stopServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  server.setExpectedRunState(ExpectedRunState.STOPPED);
  server.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState

5.7.3.3 サーバーを再起動するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

```
String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. サーバーオブジェクトの setExpectedRunState を呼び出し、引数を ExpectedRunState.RESTART に設定します。

```
server.setExpectedRunState(ExpectedRunState.RESTART);
```

4. サーバーオブジェクトを保存します。

```
server.save();
```

これは、サーバーを再起動する必要があることを示すようにサーバーの SI_EXPECTED_RUN_STATE プロパティを設定します。現在、サーバーが実行されている場合、Server Intelligence Agent はサーバーの停止プロセスを開始して、再起動します。現在、サーバーが実行されていない場合、Server Intelligence Agent はサーバーの起動プロセスを開始します。

例

```
void restartServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  server.setExpectedRunState(ExpectedRunState.RESTART);
  server.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState

5.7.3.4 サーバーを複製するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

```
String serverQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. IInfoStore オブジェクトの newInfoObjectCollection メソッドを呼び出して、IInfoObjects コレクションを作成します。

```
IInfoObjects cloneInfoObjects = infoStore.newInfoObjectCollection();
```

4. IInfoObjects コレクションの静的 copy メソッドを呼び出して、サーバーの複製を作成します。最初の引数を、複製するサーバーオブジェクトに設定します。2 番目の引数を定数 IInfoObjects.CopyModes.COPY_NEW_OBJECT_NEW_FILES に設定します。

```
IServer serverClone = (IServer) cloneInfoObjects.copy(server,
  IInfoObjects.CopyModes.COPY_NEW_OBJECT_NEW_FILES);
```

5. 複製オブジェクトのプロパティを設定します。

> ⓘ 注記
>
> サーバーが Input File Repository Server の場合は、フレンドリ名の先頭に「Input.」を付ける必要があります。Output File Repository Server の場合は、フレンドリ名の先頭に「Output.」を付ける必要があります。

6. 複製オブジェクトを保存します。

```
serverClone.save();
```

例

次の例は、既存のサーバーの複製を作成します。

```
void cloneServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String queryString = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
  IInfoObjects serverInfoObjects = infostore.query(queryString);
  IServer server = (IServer) serverInfoObjects.get(0);
  IInfoObjects cloneInfoObjects = infostore.newInfoObjectCollection();
  IServer serverClone = (IServer) cloneInfoObjects.copy(server,
  IInfoObjects.CopyModes.COPY_NEW_OBJECT_NEW_FILES);
```

```
  serverClone.setTitle("Clone_" + System.currentTimeMillis() + "_"
    + server.getTitle());
  serverClone.setFriendlyName("Clone_" + System.currentTimeMillis() + "_"
    + server.getFriendlyName());
  serverClone.setExpectedRunState(ExpectedRunState.RUNNING);
  serverClone.setDisabled(true);
  serverClone.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`

5.7.3.5 サーバーメトリックスを取得する

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリからサーバーオブジェクトを取得し、`IServer` にキャストします。

   クエリーには、`SI_METRICS` プロパティを入れる必要があります。

   ```
   String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
   serverID;
   IServer server = (IServer) infostore.query(serverQuery).get(0);
   ```

3. リポジトリからメトリックスの説明オブジェクトを取得し、`com.businessobjects.sdk.plugin.desktop.metricdescriptions.IMetricDescriptions` にキャストします。

   ```
   String metricDescriptionsQuery =
     "SELECT SI_ID, SI_METRIC_DESCRIPTIONS FROM CI_SYSTEMOBJECTS WHERE
   SI_KIND='MetricDescriptions'";
   IMetricDescriptions metricDescriptions = (IMetricDescriptions)
     infostore.query(metricDescriptionsQuery).get(0);
   ```

4. サーバーメトリックスコレクションを取得します。

   ```
   IServerMetrics serverMetrics = server.getMetrics();
   ```

5. サービスインターフェイスメトリックスを取得します。

> ⓘ 注記
>
> 利用可能なサービスインターフェイスの名前を取得するには、IServerMetrics オブジェクトの getServiceInterfaceNames メソッドを使用します。

```
IMetrics serviceMetrics = serverMetrics.getMetrics(serviceName);
```

6. 反復子を使用して、IServerMetrics コレクションから各メトリックスオブジェクトを取得します。

```
Iterator serviceMetricsIter = serviceMetrics.iterator();
while(serviceMetricsIter.hasNext())
{
  IMetric metric = (IMetric) serviceMetricsIter.next();
  String metricName = metric.getName();
  ...
}
```

IMetric の getName メソッドは、メトリックスの内部名を返すことに注意してください。次のコードは、メトリックスのローカライズされた名前を取得する方法を示します。

```
IMLDescriptions descriptions =
metricDescriptions.getMetricDescriptions(serviceName);
IPropertyRenderTemplate propertyRenderTemplate =
descriptions.getPropertyRenderTemplate(metricName);
String localizedMetricName = propertyRenderTemplate.getLabel(Locale.ENGLISH);
```

例

次のコードは、サーバーにあるすべてのサービスインターフェイスのメトリックスを取得し、それらを HTML テーブルに表示します。

```
String getServerMetrics(IEnterpriseSession enterpriseSession, int serverID)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  String metricDescriptionsQuery = "SELECT SI_ID, SI_METRIC_DESCRIPTIONS FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='MetricDescriptions'";
  IMetricDescriptions metricDescriptions = (IMetricDescriptions)
infostore.query(metricDescriptionsQuery).get(0);
  IServerMetrics serverMetrics = server.getMetrics();
  Set serviceNames = serverMetrics.getServiceInterfaceNames();
  Iterator serviceNamesIter = serviceNames.iterator();
  String resultString = "";
  while(serviceNamesIter.hasNext())
  {
    String serviceName = (String) serviceNamesIter.next();
    IMetrics serviceMetrics = serverMetrics.getMetrics(serviceName);
    resultString += "<div class=¥"bordered¥"><h5>" + serviceName + " metrics</
h5>¥n<table>"
      + "<tr><th align=¥"left¥">Name</th><th align=¥"left¥">Value</th></tr>¥n";
    Iterator serviceMetricsIter = serviceMetrics.iterator();

    while(serviceMetricsIter.hasNext())
    {
      IMetric metric = (IMetric) serviceMetricsIter.next();
```

```
      String metricName = metric.getName();
      IMLDescriptions descriptions =
metricDescriptions.getMetricDescriptions(serviceName);
      IPropertyRenderTemplate propertyRenderTemplate =
descriptions.getPropertyRenderTemplate(metricName);
      String localizedMetricName =
propertyRenderTemplate.getLabel(Locale.ENGLISH);
      resultString += "<tr><td valign=¥"top¥">" + localizedMetricName + "</
td><td valign=¥"top¥">"
        + metric.getValue() + "</td></tr>¥n";
    }
    resultString += "</table></div>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.crystaldecisions.sdk.plugin.desktop.server.IServerMetrics
- com.businessobjects.sdk.plugin.desktop.common.IMetric
- com.businessobjects.sdk.plugin.desktop.common.IMetrics
- com.businessobjects.sdk.plugin.desktop.metricdescriptions.IMetricDescriptions
- com.businessobjects.sdk.plugin.desktop.metricdescriptions.IMLDescriptions
- com.businessobjects.sdk.plugin.desktop.metricdescriptions.IPropertyRenderTemplate
- java.util.Iterator
- java.util.Locale
- java.util.Set

5.7.4 サーバーの追加

デプロイメントにプログラムによってサーバーを追加するには、最初に、サーバーをホストする Enterprise ノードを決定する必要があります。次に、そのサーバーが使用するサービスコンテナを選択する必要があります。サーバーがホストできるサービスはサービスコンテナによって決定されるため、ホストするサービスがサポートされているサービスコンテナを選択してください。最後に、サーバーが提供するサービスを決定する必要があります。

このセクションでは、デプロイメントにサーバーを追加する方法、およびサーバーを作成するために使用できるさまざまなサービスとサービスコンテナについて説明します。

関連情報

すべてのサービスコンテナを取得する [221 ページ]
すべてのサービスを取得する [223 ページ]
サーバーを追加するには [224 ページ]

5.7.4.1 すべてのサービスコンテナを取得する

新しいサーバーを追加する場合は、それにサービスコンテナを関連付ける必要があります。サーバーがホストできるサービスは、サービスコンテナによって決定されます。Central Management Server（CMS）にデプロイメントのすべてのサービスコンテナを照会し、各コンテナの CUID、名前、サポートされているサービスなどの情報を取得できます。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. CMS に照会して、すべてのサービスコンテナを取得します。

   これらのサービスコンテナは、CMS のフォルダ 55 にあります。`SI_ML_DESCRIPTION` プロパティ バッグと `SI_SERVICE4SC` プロパティ バッグを取得します。前者には、各コンテナのロケール固有の説明が含まれます。後者には、コンテナでサポートされているすべてのサービスの ID が含まれます。

```
String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION,
SI_SERVICE4SC FROM CI_SYSTEMOBJECTS WHERE SI_KIND='ServiceContainer' AND
SI_PARENT_FOLDER=55";
IInfoObjects serviceContainers = (IInfoObjects)
infostore.query(servicesQuery);
```

3. 反復子を使用して、`IInfoObjects` コレクションに含まれる各サービスコンテナにアクセスします。

```
Iterator it = serviceContainers.iterator();
while (it.hasNext())
{
  IServiceContainer serviceContainer = (IServiceContainer) it.next();
  ...
}
```

4. `IServiceContainer` クラスの `getDescription` メソッドと `getCUID` メソッドを呼び出して、各コンテナの識別情報を取得します。

> ⓘ 注記
>
> `getDescription` メソッドは、パラメータとしてロケールを受け取ります。このロケール固有の説明は、セントラル管理コンソール（CMC）の UI に表示されます。

```
resultString += "<b>Service Container Description:</b> " +
serviceContainer.getDescription(Locale.ENGLISH) + " - ";
resultString += "<b>CUID: </b>" + serviceContainer.getCUID() + "<br>";
```

5. `IServiceContainer` クラスの `getServices` メソッドを呼び出して、そのコンテナでサポートされているすべてのサービスの ID を取得します。

```
Set supportedServices = serviceContainer.getServices();
```

6. 反復子を使用して、サポートされている各サービスの ID にアクセスします。

```
Iterator servIt = supportedServices.iterator();
while (servIt.hasNext())
{
  Integer serviceID = (Integer) servIt.next();
  ...
}
```

例

以下の例は、CMS にある各サービスコンテナの説明、CUID、およびサポートされているサービスの ID を取得します。

```
String getServiceContainers(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION,
SI_SERVICE4SC FROM CI_SYSTEMOBJECTS WHERE SI_KIND='ServiceContainer' AND
SI_PARENT_FOLDER=55";
  IInfoObjects serviceContainers = (IInfoObjects) infostore.query(servicesQuery);
  String resultString = "";
  Iterator it = serviceContainers.iterator();
  while (it.hasNext())
  {
    IServiceContainer serviceContainer = (IServiceContainer) it.next();
    resultString += "<b>Service Container Description:</b> " +
serviceContainer.getDescription(Locale.ENGLISH) + " - ";
    resultString += "<b>CUID: </b>" + serviceContainer.getCUID() + "<br>";
    resultString += "<b>Supported Service IDs:</b><br>";

    Set supportedServices = serviceContainer.getServices();
    Iterator servIt = supportedServices.iterator();
    while (servIt.hasNext())
    {
      Integer serviceID = (Integer) servIt.next();
      resultString += "    " + serviceID + "<br>";
    }
    resultString += "<br>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.businessobjects.sdk.plugin.desktop.servicecontainer.IServiceContainer
- java.util.Iterator
- java.util.Locale
- java.util.Set

5.7.4.2 すべてのサービスを取得する

新しいサーバーを追加する場合は、サーバーがホストするサービスを決定する必要があります。Central Management Server（CMS）にデプロイメントのすべてのサービスを照会し、説明、CUID などの情報を取得できます。

> ⓘ 注記
>
> 追加する新しいサービスは、新しいサーバーに関連付けられるサービスコンテナによってサポートされている必要があります。`IService.getContainers` メソッドを使用して、特定のサービスをホストするように構成されたすべてのサービスコンテナの ID を取得します。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. CMS に照会して、すべてのサービスを取得します。

   これらのサービスは、CMS のフォルダ 52 にあります。各サービスのロケール固有の説明が入った `SI_ML_DESCRIPTION` プロパティ バッグを取得します。

```
String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION
FROM CI_SYSTEMOBJECTS WHERE SI_KIND='Service' AND SI_PARENT_FOLDER=52";
IInfoObjects services = (IInfoObjects) infostore.query(servicesQuery);
```

3. 反復子を使用して、`IInfoObjects` コレクションに含まれる各サービスにアクセスします。

```
Iterator it = services.iterator();
while (it.hasNext())
{
  IService service = (IService) it.next();
  ...
}
```

4. `IService` クラスの `getDescription` メソッドと `getCUID` メソッドを呼び出して、各サービスの識別情報を取得します。

   > ⓘ 注記
   >
   > `getDescription` メソッドは、パラメータとしてロケールを受け取ります。セントラル管理コンソール（CMC）の UI を通してサーバーにサービスを追加する際に、このロケール固有の説明が表示されます。

```
resultString += "<b>Service Description:</b> " +
service.getDescription(Locale.ENGLISH) + " - ";
resultString += "<b>CUID: </b>" + service.getCUID() + "<br>";
```

例

以下の例は、CMS にある各サービスの説明と CUID を取得します。

```
String getServices(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='Service' AND SI_PARENT_FOLDER=52";
```

```
  IInfoObjects services = (IInfoObjects) infostore.query(servicesQuery);
  String resultString = "";
  Iterator it = services.iterator();
  while (it.hasNext())
  {
    IService service = (IService) it.next();
    resultString += "<b>Service Description:</b> " +
service.getDescription(Locale.ENGLISH) + " - ";
    resultString += "<b>CUID: </b>" + service.getCUID() + "<br>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.businessobjects.sdk.plugin.desktop.service.IService`
- `java.util.Iterator`
- `java.util.Locale`

5.7.4.3 サーバーを追加するには

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. `IInfoStore` オブジェクトの `newInfoObjectCollection` メソッドを呼び出して、`IInfoObjects` コレクションを作成します。

   ```
   IInfoObjects servers = infostore.newInfoObjectCollection();
   ```

3. コレクションの `add` メソッドを呼び出して、`IServer` オブジェクトを作成します。

   ```
   IServer server = (IServer) servers.add(IServer.KIND);
   server.setTitle(newServerName);
   server.setDescription("Sample adaptive processing server");
   ```

4. `setFriendlyName` を呼び出して、サーバーのフレンドリ名を指定します。

   ```
   server.setFriendlyName(newServerName);
   ```

   > ⓘ 注記
   >
   > サーバーが Input File Repository Server の場合は、フレンドリ名の先頭に「`Input.`」を付ける必要があります。Output File Repository Server の場合は、フレンドリ名の先頭に「`Output.`」を付ける必要があります。

5. `setExpectedRunState` を呼び出して、サーバーの作成後にサーバーを自動的に起動するかどうかを指定します。

com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState クラスには、予期された実行状態の定数が含まれます。

```
server.setExpectedRunState(ExpectedRunState.RUNNING);
```

6. setDisabled を呼び出して、デフォルトでサーバーを無効にするかどうかを指定します。

```
server.setDisabled(true);
```

7. setAutoBoot を呼び出して、サーバーを Server Intelligence Agent で管理するかどうかを指定します。

```
server.setAutoBoot(true);
```

8. サーバーオブジェクトをリポジトリに保存し、再度取得します。

```
server.save();
String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
server.getID();
server = (IServer) infostore.query(serverQuery).get(0);
```

> **注記**
>
> オブジェクトへの変更を保存したら、保存後に再度取得することをお勧めします。これにより、すべてのプロパティに有効な値が含まれます。

9. サーバーにサービスコンテナを接続します。

```
String serviceContainerQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_PROGID = '"
  + IServiceContainer.KIND + "' AND SI_NAME =
'AdaptiveProcessingServiceContainer'";
IServiceContainer serviceContainer = (IServiceContainer)
infoStore.query(serviceContainerQuery).get(0);
server.setContainer(serviceContainer.getID());
```

サービスコンテナは、サーバーでホストできるサービスタイプを決定します。IServiceContainer クラスの getServices メソッドを使用して、このコンテナによってサポートされているすべてのサービスの ID を取得できます。

10. ホスト先の Enterprise ノードにサーバーを接続します。

```
String enterpriseNodeQuery = "SELECT TOP 1 SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_KIND = '"
  + IEnterpriseNode.KIND + "'";
IEnterpriseNode enterpriseNode = (IEnterpriseNode)
infostore.query(enterpriseNodeQuery).get(0);
server.setEnterpriseNode(enterpriseNode.getID());
```

11. サーバーでホストするサービスを取得します。

```
String servicesQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE "
  + "SI_NAME IN('AuditingService', 'PublishingService', 'AuditProxyService')";
IInfoObjects services = infostore.query(servicesQuery);
```

> **注記**
>
> SI_NAME 値は、サーバーのサービスを選択する際にセントラル管理コンソール (CMC) に表示されるサービスの説明ではありません。UI に表示されるサービスの説明は、SI_ML_DESCRIPTION プロパティ バッグ内のロケール固有の値になります。SDK を使用して、CMS 内のすべてのサービスのリストを照会し、追加するサービスの SI_NAME 値と SI_CUID 値を取得できます。

12. サーバーでホストされるサービスのリストに各サービスを追加します。

```
IConfiguredServices configuredServices = server.getHostedServices();
Iterator serviceIter = services.iterator();
while(serviceIter.hasNext())
{
  IService service = (IService) serviceIter.next();
  configuredServices.add(service.getID());
  ...
}
```

13. サーバーオブジェクトを保存します。

```
server.save();
```

これで、セントラル管理コンソールの［サーバー］ページからサーバーにアクセスできるようになります。

例

```
void addServer(IEnterpriseSession enterpriseSession, String newServerName)
throws SDKException
 {
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects servers = infostore.newInfoObjectCollection();
  IServer server = (IServer) servers.add(IServer.KIND);
  server.setTitle(newServerName);
  server.setDescription("Sample adaptive processing server");
  server.setFriendlyName(newServerName);
  server.setExpectedRunState(ExpectedRunState.RUNNING);
  server.setDisabled(true);
  server.setAutoBoot(true);
  server.save();
  String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
server.getID();
  server = (IServer) infostore.query(serverQuery).get(0);
  String serviceContainerQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_KIND = '"
    + IServiceContainer.KIND + "' AND SI_NAME =
'AdaptiveProcessingServiceContainer'";
  IServiceContainer serviceContainer = (IServiceContainer)
infostore.query(serviceContainerQuery).get(0);
  server.setContainer(serviceContainer.getID());

  String enterpriseNodeQuery = "SELECT TOP 1 SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_KIND = '"
    + IEnterpriseNode.KIND + "'";
  IEnterpriseNode enterpriseNode = (IEnterpriseNode)
infostore.query(enterpriseNodeQuery).get(0);
  server.setEnterpriseNode(enterpriseNode.getID());

  String servicesQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE "
    + "SI_NAME IN('AuditingService', 'PublishingService', 'AuditProxyService')";
  IInfoObjects services = infostore.query(servicesQuery);

  IConfiguredServices configuredServices = server.getHostedServices();
  Iterator serviceIter = services.iterator();
  while(serviceIter.hasNext())
  {
    IService service = (IService) serviceIter.next();
    configuredServices.add(service.getID());
    server.save();
  }
```

```
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`
- `com.businessobjects.sdk.plugin.desktop.enterprisenode.IEnterpriseNode`
- `com.businessobjects.sdk.plugin.desktop.service.IService`
- `com.businessobjects.sdk.plugin.desktop.servicecontainer.IServiceContainer`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredService`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices`

5.7.5 サービスとサービスコンテナの設定

続行する前に、サービス構成の各部を理解することが重要です。

サービス構成は、次の部分から成ります。

- 構成プロパティ
- 実行プロパティ
- ローカルファイル

関連情報

構成プロパティ [227 ページ]
実行プロパティ [231 ページ]
ローカルファイル [233 ページ]

5.7.5.1 構成プロパティ

構成プロパティは、サービスまたはサービスコンテナの動作を制御します。これらのプロパティの値は、いつでも変更できます。ただし、プロパティによっては変更がすぐに適用されないことがあります。一部のプロパティは、変更を有効にするためにサーバーを再起動する必要があります。

サービスの構成プロパティは、`com.businessobjects.sdk.plugin.desktop.common.IServiceConfigProperties` オブジェクトに保

存されます。このオブジェクトは、IConfiguredService オブジェクトの getConfigProps を呼び出して取得できます。

サービスコンテナの構成プロパティは、com.businessobjects.sdk.plugin.desktop.common.IServiceContainerConfigProperties オブジェクトに保存されます。このオブジェクトは、IConfiguredContainer オブジェクトの getConfigProps を呼び出して取得できます。

5.7.5.1.1 実際の構成プロパティと要求された構成プロパティ

サーバーがアクティブな間は、いくつかの構成プロパティの変更を適用できません。Server Intelligence フレームワークは、現在 (実際) の構成と、サーバーホストが再起動されるときに適用される構成の変更要求の両方を追跡します。

IConfiguredService および IConfiguredContainer の getActualConfigProps メソッドは、IActualConfigProperties コレクションを返します。このコレクションは、IConfigProperties から継承されます。このコレクションには、サーバーでホストされているサービスに現在適用されている構成プロパティが含まれます。

5.7.5.1.2 構成プロパティのデフォルト

サービスコンテナまたはサービスがサーバーに関連付けられている場合は、デフォルトの構成プロパティ値が IService または IServiceContainer オブジェクトの設定から自動的にコピーされます。問題の原因となる構成プロパティを変更する場合は、IService または IServiceContainer オブジェクトのデフォルトの構成プロパティ値に戻すことができます。

5.7.5.1.3 構成プロパティを取得するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

   クエリーには、SI_HOSTED_SERVICES プロパティを入れる必要があります。

   ```
   String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
     + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
   IServer server = (IServer) infostore.query(serverQuery).get(0);
   ```

3. サーバーオブジェクトの getHostedServices メソッドを呼び出します。

   これは、IConfiguredServices コレクションを返します。このコレクションには、サーバーが提供するサービスの設定情報が入っています。

   ```
   IConfiguredServices cfgServices = server.getHostedServices();
   ```

4. IConfiguredServices コレクションの get メソッドを呼び出して、1 つの設定済みサービスを取得します。

```
Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
```

   この例では、IConfiguredServices コレクションの getConfiguredServiceIDs メソッドを使用し、結果セットから最初の ID を取得します。構成情報を取得するサービスの ID を引数として設定します。これは、IConfiguredService オブジェクトを返します。このオブジェクトには、このサーバーの指定されたサービスの構成が含まれます。

5. IConfiguredService オブジェクトの getConfigProps メソッドを呼び出し、選択したサービスの構成プロパティの名前と値を反復処理します。

```
IConfigProperties configProps = (IConfigProperties)
cfgService.getConfigProps();
String[] propertyNames = configProps.getPropNames();
String resultString = "";
int size = propertyNames.length;
for(int i = 0; i < size; i++)
{
  IConfigProperty property = configProps.getProp(propertyNames[i]);
  String name = property.getDisplayName(Locale.ENGLISH);
  String value = property.getValue().toString();
  ...
}
```

   getConfigProps メソッドは、個別の構成プロパティを含む IConfigurationProperties コレクションを返します。getPropNames メソッドを使用して、サービスの構成プロパティの名前を含む配列を返します。次に、IConfigurationProperties コレクションの getProp メソッドを使用して、コレクションからプロパティを取得します。

例

次の例は、設定されているサービスの構成プロパティを取得し、ローカライズされた表示名と値を標準出力に出力します。

```
String retrieveConfigProperties(IEnterpriseSession enterpriseSession, int
serverID) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  IConfiguredServices cfgServices = server.getHostedServices();
  Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
  Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
  IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
  IConfigProperties configProps = (IConfigProperties)
cfgService.getConfigProps();
  String[] propertyNames = configProps.getPropNames();
  String resultString = "";
  int size = propertyNames.length;
  for(int i = 0; i < size; i++)
  {
    IConfigProperty property = configProps.getProp(propertyNames[i]);
    String name = property.getDisplayName(Locale.ENGLISH);
```

```
    String value = property.getValue().toString();
    resultString += name + ": " + value + "<br>";
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices
- com.businessobjects.sdk.plugin.desktop.common.IConfiguredService
- com.businessobjects.sdk.plugin.desktop.common.IConfigProperties
- com.businessobjects.sdk.plugin.desktop.common.IConfigProperty
- java.util.Locale
- java.util.Set

5.7.5.1.4 構成プロパティをリセットするには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

   クエリーには、SI_HOSTED_SERVICES プロパティを入れる必要があります。

   ```
   String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
     + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
     IServer server = (IServer) infostore.query(serverQuery).get(0);
   ```

3. サーバーオブジェクトの getHostedServices メソッドを呼び出します。

   これは、IConfiguredServices コレクションを返します。このコレクションには、サーバーが提供するサービスの設定情報が入っています。

   ```
   IConfiguredServices cfgServices = server.getHostedServices();
   ```

4. IConfiguredServices コレクションの get メソッドを呼び出して、1 つの設定済みサービスを取得します。

   ```
   Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
   Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
   IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
   ```

   この例では、IConfiguredServices コレクションの getConfiguredServiceIDs メソッドを使用し、結果セットから最初の ID を取得します。構成情報を取得するサービスの ID を引数として設定します。これは、IConfiguredService オブジェクトを返します。このオブジェクトには、このサーバーの指定されたサービスの構成が含まれます。

5. IConfiguredService オブジェクトの resetToFactoryConfigProps メソッドを呼び出して、サービス構成プロパティのすべての値をサービスで定義されているデフォルト値にリセットします。

```
cfgService.resetToFactoryConfigProps();
```

6. サーバーオブジェクトを保存します。

```
server.save();
```

> ⓘ 注記
>
> 変更は、次にサーバーが再起動された後で有効になります。

例

次の例は、設定されたサービスの構成プロパティをリセットします。

```
void resetConfigProperties(IEnterpriseSession enterpriseSession, int serverID)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  IConfiguredServices cfgServices = server.getHostedServices();
  Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
  Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
  IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
  cfgService.resetToFactoryConfigProps();
  server.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices
- com.businessobjects.sdk.plugin.desktop.common.IConfiguredService
- java.util.Set

5.7.5.2 実行プロパティ

構成プロパティと同様に、実行プロパティは、サービスまたはサービスコンテナの動作を制御します。これらのプロパティの値は、コマンド ラインの引数としてサービスコンテナのバイナリに渡されます。実行プロパティへの変更は、サーバーが再起動された場合にのみ適用されます。

サービスの実行プロパティは、com.businessobjects.sdk.plugin.desktop.common.IExecProps オブジェクトに保存されます。このオブジェクトは、IConfiguredService オブジェクトの getConfigProps を呼び出して取得できます。

サービスコンテナの実行プロパティは、com.businessobjects.sdk.plugin.desktop.common.IExecProps オブジェクトに保存されます。このオブジェクトは、IConfiguredContainer オブジェクトの getExecProps を呼び出して取得できます。

5.7.5.2.1 実行プロパティを取得するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

   クエリーには、SI_HOSTED_SERVICES プロパティを入れる必要があります。

```
String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
  + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. サーバーオブジェクトの getContainer メソッドを呼び出して、このサーバーに関連付けられているサーバーコンテナを返します。

```
IConfiguredContainer cfgContainer = server.getContainer();
```

4. IConfiguredContainer オブジェクトの getExecProps メソッドを呼び出します。

   このメソッドは、IExecProps オブジェクトを返します。このオブジェクトには、サーバーが起動したときにコンテナに渡されるコマンド ライン情報を取得するためのメソッドが含まれています。

```
IExecProps execProps = cfgContainer.getExecProps();
String resultString = "";
resultString += "File paths: " + execProps.getFilePath() + "<br>";
resultString += "Working directory: " + execProps.getWorkDir()+ "<br>";
resultString += "Execution flags: " + execProps.getExecFlags() + "<br>";
resultString += "Command line arguments: " + execProps.getArgs() + "<br>";
```

例

次の例は、設定されたサービスの実行プロパティを取得します。

```
String retrieveExecutionProperties(IEnterpriseSession enterpriseSession, int
serverID) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  IConfiguredContainer cfgContainer = server.getContainer();
  IExecProps execProps = cfgContainer.getExecProps();
  String resultString = "";
  resultString += "File paths: " + execProps.getFilePath() + "<br>";
```

```
  resultString += "Working directory: " + execProps.getWorkDir()+ "<br>";
  resultString += "Execution flags: " + execProps.getExecFlags() + "<br>";
  resultString += "Command line arguments: " + execProps.getArgs() + "<br>";
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.businessobjects.sdk.plugin.desktop.common.IConfiguredContainer
- com.businessobjects.sdk.plugin.desktop.common.IExecProps

5.7.5.3 ローカルファイル

ローカルファイルは、サービスまたはサービスコンテナをホストするエンタープライズノードに置かれているファイルです。これらのファイルには、サービスまたはサービスコンテナが必要とする追加の情報が格納されます。

サービスまたはサービスコンテナのローカルファイルは、com.businessobjects.sdk.plugin.desktop.common.IStringProps オブジェクトに保存されます。このオブジェクトは、IConfiguredService または IConfiguredContainer オブジェクトの getLocalFiles を呼び出して取得できます。

5.7.5.3.1 ローカルファイルを取得するには

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリからサーバーオブジェクトを取得し、IServer にキャストします。

   クエリーには、SI_HOSTED_SERVICES プロパティを入れる必要があります。

   ```
   String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
     + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
     IServer server = (IServer) infostore.query(serverQuery).get(0);
   ```

3. サーバーオブジェクトの getContainer メソッドを呼び出して、このサーバーに関連付けられているサーバーコンテナを返します。

   ```
   IConfiguredContainer cfgContainer = server.getContainer();
   ```

4. IConfiguredContainer オブジェクトの getLocalFiles メソッドを呼び出します。

このメソッドは、IStringProps オブジェクトを返します。このオブジェクトを反復処理して、コンテナによって使用されるファイル システム ファイルの名前を取得できます。

```
IStringProps localFiles = cfgContainer.getLocalFiles();
String resultString = "";
Iterator localFilesIter = localFiles.iterator();
while(localFilesIter.hasNext())
{
  String filename = (String) localFilesIter.next();
  resultString += filename;
}
```

IStringProps の add および remove メソッドを使用して、ローカルファイルを追加または削除します。

例

次の例では、設定されているサービスのローカルファイルを取得し、ファイル名を標準出力に出力します。

```
String retrieveLocalFiles(IEnterpriseSession enterpriseSession, int serverID)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  IConfiguredContainer cfgContainer = server.getContainer();
  IStringProps localFiles = cfgContainer.getLocalFiles();

  String resultString = "";
  Iterator localFilesIter = localFiles.iterator();
  while(localFilesIter.hasNext())
  {
    String filename = (String) localFilesIter.next();
    resultString += filename;
  }
  return resultString;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.businessobjects.sdk.plugin.desktop.common.IConfiguredContainer
- com.businessobjects.sdk.plugin.desktop.common.IStringProps
- java.util.Iterator

5.7.6 サーバーグループの管理

このセクションでは、次のサーバーグループ管理タスクをプログラムで実行する方法について説明します。

- サーバーグループの作成
- サーバーグループの削除
- サーバーのグループへの追加
- サーバーのグループからの削除

5.7.6.1 新しいサーバーグループを作成する

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. IInfoStore インターフェイスの newInfoObjectCollection メソッドを呼び出して、IInfoObjects コレクションを作成します。

```
IInfoObjects newServerGroups = infostore.newInfoObjectCollection();
```

3. コレクションの add メソッドを呼び出して InfoObject を作成し、結果を適切なプラグインインターフェイスにキャストします。

```
IServerGroup newServerGroup = (IServerGroup)
newServerGroups.add(IServerGroup.KIND);
```

add メソッドは、作成する InfoObject の種類を表す IServerGroup.KIND フィールド値を使用します。この例では、サーバーグループを作成し、結果を IServerGroup プラグインインターフェイスにキャストします。

4. IInfoObject のプロパティを設定します。

```
newServerGroup.setTitle(newServerGroupName);
newServerGroup.setDescription("New server group description");
```

5. IInfoObjects コレクションをリポジトリにコミットします。

```
infostore.commit(newServerGroups);
```

例

```
void createNewServerGroup(IEnterpriseSession enterpriseSession, String
newServerGroupName) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

  IInfoObjects newServerGroups = infostore.newInfoObjectCollection();
  IServerGroup newServerGroup = (IServerGroup)
newServerGroups.add(IServerGroup.KIND);
```

```
  newServerGroup.setTitle(newServerGroupName);
  newServerGroup.setDescription("New server group description");

  infostore.commit(newServerGroups);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.servergroup.IServerGroup

5.7.6.2 サーバーグループにサーバーを追加する

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. 指定された ServerGroup オブジェクトを含む IInfoObjects コレクションを照会して取得します。

```
String serverGroupQuery = "Select SI_GROUP_MEMBERS From CI_SYSTEMOBJECTS "
  + "Where SI_ID=" + serverGroupID;
IInfoObjects serverGroups = infostore.query(serverGroupQuery);
IServerGroup serverGroup = (IServerGroup) serverGroups.get(0);
```

3. 指定された Server オブジェクトを含む IInfoObjects コレクションを照会して取得します。

```
String serverQuery = "Select SI_ID, SI_ML_NAME From CI_SYSTEMOBJECTS "
  + "Where SI_PROGID='CrystalEnterprise.Server' " + "AND SI_ID=" + serverID;
IInfoObjects servers = infostore.query(serverQuery);
IServer server = (IServer) servers.get(0);
```

4. サーバーをサーバーグループに追加します。

```
serverGroup.getServers().add(server.getTitle());
```

> ⓘ 注記
>
> グループをサーバーグループに追加することもできます。グループをサーバーグループに追加するには、ServerGroup.getSubGroups().add メソッドを使用します。

5. IInfoObjects コレクションをリポジトリにコミットします。

```
infostore.commit(serverGroups);
```

例

```
void addServerToServerGroup(IEnterpriseSession enterpriseSession, int serverID,
int serverGroupID) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String serverGroupQuery = "Select SI_GROUP_MEMBERS From CI_SYSTEMOBJECTS "
    + "Where SI_ID=" + serverGroupID;
  IInfoObjects serverGroups = infostore.query(serverGroupQuery);
  IServerGroup serverGroup = (IServerGroup) serverGroups.get(0);
  String serverQuery = "Select SI_ID, SI_ML_NAME From CI_SYSTEMOBJECTS "
    + "Where SI_PROGID='CrystalEnterprise.Server' " + "AND SI_ID=" + serverID;
  IInfoObjects servers = infostore.query(serverQuery);
  IServer server = (IServer) servers.get(0);
  serverGroup.getServers().add(server.getTitle());
  infostore.commit(serverGroups);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.server.IServer
- com.crystaldecisions.sdk.plugin.desktop.servergroup.IServerGroup

5.8 フェデレーション

フェデレーションを使用すると、管理者は、複数の BI platform インストール間でコンテンツの自動レプリケーションを設定できます。1 つのデプロイメント（レプリケート元サイト）上でコンテンツを作成および管理し、スケジュールに基づく間隔で他のデプロイメント（レプリケート先サイト）にコンテンツをレプリケートするように設定できます。

このレプリケーションは、単方向にすることも、双方向にすることもできます。単方向の設定では、レプリケート先サイトがレプリケート元サイトから変更を取得します。レプリケート元サイトのコンテンツに加えた変更は、レプリケート先サイトで加えた変更を上書きします。双方向の設定では、レプリケート元サイトとレプリケート先サイトの両方で同期化が行われます。レプリケート先のコンテンツが更新されると、レプリケート元のコンテンツもそれに応じて更新され、レプリケート元のコンテンツが更新されると、レプリケート先のコンテンツも更新されます。

フェデレーションはリモートスケジュールもサポートしています。これにより、管理者は、レプリケート先サイトでレポートを設定し、レプリケート元サイトで実際のレポート処理を実行することができます。

デプロイメント間をまたいでコンテンツをレプリケートできることには、コンテンツ管理の簡略化、アプリケーションの一貫性の維持、複数のオフィスにまたがる権限ポリシーの適用など、多くの利点があります。

フェデレーションは、セントラル管理コンソール（CMC）または SDK を使用して設定できます。このセクションでは、SDK を使用してフェデレーションを設定する方法を示します。

> ⓘ 注記
>
> フェデレーション、および CMC でフェデレーションを設定する方法の詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

5.8.1 フェデレーションの設定

フェデレーションを設定する基本的な手順は次のとおりです。

- レプリケート元サイトで、レプリケーション一覧を作成します。レプリケーション一覧は、レプリケート先サイトに複製する内容を指定します。
- 各レプリケート先サイトで、リモート接続を作成します。リモート接続は、CMS ホスト名、接続の確立に使用するユーザー名とパスワードなど、レプリケート元サイトに関する情報を含みます。
- 各レプリケート先サイトで、レプリケーションジョブを作成します。レプリケーションジョブは、リモート接続内の情報を使用してレプリケート元サイトとの接続を確立し、レプリケーション一覧を取得し、該当する内容と同期します。

関連情報

レプリケーション一覧を作成する [239 ページ]
リモート接続を作成する [241 ページ]
レプリケーションジョブを作成およびスケジュールする [243 ページ]

5.8.2 フェデレーションに使用されるクラス

次のパッケージには、フェデレーション設定用のクラスが入っています。

- `com.businessobjects.sdk.plugin.desktop.manifest`。レプリケーション一覧作成用のクラスが入っています。
- `com.businessobjects.sdk.plugin.desktop.remotecluster`。リモート接続設定用のクラスが入っています。
- `com.businessobjects.sdk.plugin.desktop.replication`。レプリケーションジョブ作成用のクラスが入っています。
- `com.crystaldecisions.sdk.occa.infostore`。リポジトリにアクセスするためのクラスが入っています。レプリケーションジョブのスケジュールオプションの設定に使用される `ISchedulable` インターフェイスも入っています。

このセクションで説明する API の詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスを参照してください。

5.8.3 レプリケーション一覧を作成する

レプリケーション一覧は、レプリケート元サイトで作成する必要があります。この一覧は、レプリケート先サイトにレプリケートされるコンテンツを指定します。

1. レプリケート元サイトにログオンし、InfoStore サービスを取得します。

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession originEnterpriseSession =
sessionMgr.logon("originUsername", "originPassword", "originCMS",
"secEnterprise");
IInfoStore infostore = (IInfoStore)
originEnterpriseSession.getService("InfoStore");
```

2. `IInfoObjects` コレクションを作成し、それに `IManifest` オブジェクトを追加します。

   `com.businessobjects.sdk.plugin.desktop.manifest.IManifest` インターフェイスは、Central Management Server (CMS) のレプリケーション一覧オブジェクトを表すために使用されます。対応するプラグインの種類は `IManifest.KIND` です。

```
IInfoObjects replicationLists = infostore.newInfoObjectCollection();
IManifest replicationList = (IManifest) replicationLists.add(IManifest.KIND);
replicationList.setTitle(replicationListName);
```

3. レプリケートされるオブジェクトの ID をレプリケート可能なオブジェクトのコレクションに追加します。

```
Collection replicableObjects = replicationList.getReplicableObjects();
replicableObjects.add(folderID);
```

   `getReplicableObjects` メソッドは、レプリケート元サイトにあるレプリケートされるオブジェクトの ID を含むコレクションを返します。

> ⓘ 注記
>
> レプリケートできるオブジェクトの種類には制限があります。たとえば、最上位のフォルダとパブリケーションオブジェクトはレプリケートできません。詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

4. レプリケートする依存タイプを、含まれている依存コレクションに追加します。

```
IIncludedDependencies includedDependencies =
replicationList.getIncludedDependencies();
includedDependencies.addDependency("Profile-Principal",
CeRelationshipDirection.PARENT_CHILDREN);
includedDependencies.addDependency("DataConnection-Universe",
CeRelationshipDirection.CHILD_PARENTS);
```

   `getIncludedDependencies` メソッドは、`IIncludedDependencies` コレクションを返します。`addDependency` メソッドを使用して、レプリケートする依存タイプを追加します。最初のパラメータは、リレーション名を指定します。2 番目のパラメータは方向 (親から子、または子から親) で、`com.businessobjects.sdk.plugin.desktop.common.CeRelationshipDirection` インターフェイスのいずれかの定数に設定する必要があります。

5. レプリケーション一覧をリポジトリに保存します。

> ⓘ 注記
>
> プログラムでレプリケーションジョブを作成する場合は、レプリケーション一覧の CUID を指定する必要があります。レプリケーション一覧の CUID は、リポジトリにレプリケーション一覧を保存した後に取得できます。

```
replicationList.save();
```

例

次のコードは、フォルダ (変数 `folderID` の ID で識別される) を含み、レプリケート時にすべてのオブジェクト依存関係を含めるように設定されたレプリケーション一覧を作成します。

```
public void createReplicationList(String replicationListName, int folderID)
throws SDKException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession originEnterpriseSession =
sessionMgr.logon("originUsername", "originPassword", "originCMS",
"secEnterprise");

  IInfoStore infostore = (IInfoStore)
originEnterpriseSession.getService("InfoStore");

  IInfoObjects replicationLists = infostore.newInfoObjectCollection();
  IManifest replicationList = (IManifest) replicationLists.add(IManifest.KIND);
  replicationList.setTitle(replicationListName);

  Collection replicableObjects = replicationList.getReplicableObjects();
  replicableObjects.add(folderID);

  IIncludedDependencies includedDependencies =
replicationList.getIncludedDependencies();
  includedDependencies.addDependency("Profile-Principal",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("DataConnection-
Universe",CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("Category-Document",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("User-Inbox",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("Webi-Universe",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("User-Favorites",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("UserGroup-User",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("Universe(Core)-
Universe",CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("EnterpriseData-
Flash",CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("Universe-
UserGroup",CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("CustomRole-Object",
CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("User-
PersonalCategory",CeRelationshipDirection.PARENT_CHILDREN);
  replicationList.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.common.CeRelationshipDirection`
- `com.businessobjects.sdk.plugin.desktop.manifest.IIncludedDependencies`
- `com.businessobjects.sdk.plugin.desktop.manifest.IManifest`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `java.util.Collection`

関連情報

依存タイプ [247 ページ]

5.8.4 リモート接続を作成する

リモート接続を作成する前に、次の情報を確認してください。

- レプリケート元サイトの CMS ホスト名とポート。
- 接続の確立に使用されるレプリケート元サイトの BI platform ユーザーアカウントのユーザー名とパスワード。
- レプリケート元サイトのクラスタ（Web サービス）URI。

リモート接続は、レプリケート先サイトがレプリケート元サイトに接続するために使用します。レプリケーションジョブを作成する前に、レプリケート先サイトでリモート接続を作成する必要があります。

1. レプリケート先サイトにログオンし、InfoStore サービスを取得します。

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession destinationEnterpriseSession =
sessionMgr.logon("destinationUsername", "destinationPassword",
"destinationCMS", "secEnterprise");
IInfoStore infostore = (IInfoStore)
destinationEnterpriseSession.getService("InfoStore");
```

2. `IInfoObjects` コレクションを作成し、それに `IRemoteCluster` オブジェクトを追加します。

   `com.businessobjects.sdk.plugin.desktop.remotecluster.IRemoteCluster` インターフェイスは、Central Management Server (CMS) のリモート接続オブジェクトを表すために使用されます。対応するプラグインの種類は `IRemoteCluster.KIND` です。

```
IInfoObjects remoteClusters = infostore.newInfoObjectCollection();
IRemoteCluster remoteCluster = (IRemoteCluster)
remoteClusters.add(IRemoteCluster.KIND);
remoteCluster.setTitle(remoteClusterName);
```

3. リモート接続オブジェクトで必要な設定を行います。

   ユーザー名、パスワード、認証の種類は、レプリケート元サイトへのログインに使用されます。クラスタ URI は、レプリケート元サイトで Web サービスを呼び出すためのベース URI を指定します。デフォルトは、`http://<hostname>:<port>/dswsbobje` です。`<hostname>` はレプリケート元アプリケーションサーバー (Web サービスプロバイダーをホストするサーバー) のホスト名に置き換え、`<port>` はポート番号に置き換えてください。

```
remoteConnection.setCMS("originCMS");
remoteConnection.setUserName("originUsername");
remoteConnection.setPassword("originPassword");
remoteConnection.setAuthType("secEnterprise");
remoteConnection.setClusterURI("http://originCMS:8080/dswsbobje");
```

4. リモート接続をリポジトリに保存します。

   > **注記**
   >
   > プログラムでレプリケーションジョブを作成する場合は、リモート接続の ID を指定する必要があります。リモート接続の ID は、リポジトリにリモート接続を保存した後に取得できます。

```
remoteConnection.save();
```

例

次のコードは、リモート接続を作成します。

```
public void createRemoteConnection(String connectionName) throws SDKException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession destinationEnterpriseSession =
sessionMgr.logon("destinationUsername", "destinationPassword", "destinationCMS",
"secEnterprise");
  IInfoStore infostore = (IInfoStore)
destinationEnterpriseSession.getService("InfoStore");
  IInfoObjects remoteConnections = infostore.newInfoObjectCollection();

  IRemoteCluster remoteConnection = (IRemoteCluster)
remoteConnections.add(IRemoteCluster.KIND);
  remoteConnection.setTitle(connectionName);

  remoteConnection.setCMS("originCMS");
  remoteConnection.setUserName("originUsername");
  remoteConnection.setPassword("originPassword");
  remoteConnection.setAuthType("secEnterprise");
  remoteConnection.setClusterURI("http://originCMS:8080/dswsbobje");

  remoteConnection.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.remotecluster.IRemoteCluster`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`

- com.crystaldecisions.sdk.framework.ISessionMgr
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

5.8.5 レプリケーションジョブを作成およびスケジュールする

レプリケーションジョブを作成するには、以下の条件が必要です。

- レプリケート先サイトのリポジトリにリモート接続が存在する必要があります。リモート接続には、レプリケート元サイトに接続するために必要なすべての情報が含まれている必要があります。
- レプリケート元サイトのリポジトリにレプリケーション一覧が存在する必要があります。一覧には、レプリケートされる項目の ID が含まれている必要があります。

レプリケーションジョブは、レプリケート元サイトとレプリケート先サイトの間でオブジェクトの複製のスケジュール管理を行います。

1. レプリケート先サイトにログオンし、InfoStore サービスを取得します。

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();

IEnterpriseSession destinationSession =
sessionMgr.logon("destinationUserName", "destinationPassword",
"destinationCMS", "secEnterprise");
IInfoStore destinationInfoStore = (IInfoStore)
destinationSession.getService("InfoStore");
```

2. IInfoObjects コレクションを作成し、それに IReplication オブジェクトを追加します。

   com.businessobjects.sdk.plugin.desktop.replication.IReplication インターフェイスは、Central Management Server (CMS) のレプリケーションジョブオブジェクトを表すために使用されます。対応するプラグインの種類は IReplication.KIND です。

```
IInfoObjects replicationJobs = destinationInfoStore.newInfoObjectCollection();
IReplication replicationJob = (IReplication)
replicationJobs.add(IReplication.KIND);
replicationJob.setTitle("replicationJobName");
```

3. レプリケーションオブジェクトの親 ID を、レプリケート元サイトへの接続に使用するリモート接続の ID に設定します。

```
replicationJob.setParentID(remoteConnectionID);
```

4. レプリケーションオブジェクトのリモートマニフェストをレプリケート元サイトのレプリケーション一覧の CUID に設定します。

```
replicationJob.setRemoteManifest(replicationListCUID);
```

5. レプリケーションジョブのスケジュール情報を設定します。

   IReplication は com.crystaldecisions.sdk.occa.infostore.ISchedulable インターフェイスを実装しており、レポートとほぼ同じ方法でスケジュールできます （レプリケーションジョブの場合は、レプリケート先または形式を設定しません）。

```
ISchedulingInfo schedulingInfo = replicationJob.getSchedulingInfo();
schedulingInfo.setRightNow(true);
```

6. 単方向レプリケーションと双方向レプリケーションのどちらを実行するかを指定します。

   `setPushEnabled` メソッドを使用すると、双方向レプリケーションを有効にするかどうかを指定できます。false に設定すると、単方向レプリケーションが使用されます。true に設定すると、双方向レプリケーションが有効になります。

   双方向レプリケーションを有効にした場合は、競合の解決に使用するメソッドも指定する必要があります。競合は、レプリケート元とレプリケート先の両方で同じ項目が更新されたときに発生します。`setConflictResolutionMode` を使用して、レプリケート元に加えた変更を優先するか (`IReplication.CeConflictResolutionMode.MASTER_WINS`)、レプリケート先に加えた変更を優先するか (`IReplication.CeConflictResolutionMode.SLAVE_WINS`) を指定します。

```
replicationJob.setPushEnabled(true);
replicationJob.setConflictResolutionMode(IReplication.CeConflictResolutionMode
.MASTER_WINS);
```

7. レプリケーションジョブをリポジトリに保存し、スケジュールします。

```
replicationJob.save();
replicationJob.schedule();
```

例

次のコードは、競合が発生した場合に、常にレプリケート元の変更がレプリケート先の変更を上書きする双方向レプリケーションジョブを作成します。

```
public void createScheduleReplicationJob() throws SDKException, IOException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();

  IEnterpriseSession destinationSession =
sessionMgr.logon("destinationUserName", "destinationPassword", "destinationCMS",
"secEnterprise");
  IInfoStore destinationInfoStore = (IInfoStore)
destinationSession.getService("InfoStore");

  String remoteConnectionQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS Where
SI_KIND = '" + IRemoteCluster.KIND
    + "' AND SI_NAME = '" + "remoteConnectionName" + "' "
    + " AND SI_INSTANCE = 0";
  IInfoObjects remoteConnections =
destinationInfoStore.query(remoteConnectionQuery);
  IRemoteCluster remoteConnection = (IRemoteCluster) remoteConnections.get(0);
  int remoteConnectionID = remoteConnection.getID();
  IEnterpriseSession originEnterpriseSession =
sessionMgr.logon("originUserName", "originPassword", "originCMS",
"secEnterprise");
  IInfoStore originInfoStore = (IInfoStore)
originEnterpriseSession.getService("InfoStore");
  String replicationListQuery = "SELECT SI_ID, SI_CUID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = '" + IManifest.KIND
    + "' AND SI_NAME = '" + "replicationListName" + "' "
    + " AND SI_INSTANCE = 0";
  IInfoObjects replicationLists = originInfoStore.query(replicationListQuery);
  IManifest replicationList = (IManifest)replicationLists.get(0);
  String replicationListCUID = replicationList.getCUID();

  IInfoObjects replicationJobs = destinationInfoStore.newInfoObjectCollection();
  IReplication replicationJob = (IReplication)
replicationJobs.add(IReplication.KIND);
```

```
  replicationJob.setTitle("replicationJobName");
  replicationJob.setParentID(remoteConnectionID);
  replicationJob.setRemoteManifest(replicationListCUID);
  replicationJob.enableCleanup(true);
  replicationJob.setPushEnabled(true);

replicationJob.setConflictResolutionMode(IReplication.CeConflictResolutionMode.MA
STER_WINS);

  ISchedulingInfo schedulingInfo = replicationJob.getSchedulingInfo();
  schedulingInfo.setRightNow(true);
  replicationJob.save();
  replicationJob.schedule();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.manifest.IManifest`
- `com.businessobjects.sdk.plugin.desktop.remotecluster.IRemoteCluster`
- `com.businessobjects.sdk.plugin.desktop.replication.IReplication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `java.io.IOException`

関連情報

レプリケーション一覧を作成する [239 ページ]
リモート接続を作成する [241 ページ]

5.8.6 リモートスケジュール

リモートスケジュールを使用すると、レプリケート先サイトでレポートをスケジュールし、それをレプリケート元サイトで処理できます。

> ⓘ 注記
>
> リモートスケジュールの詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

レプリケーションジョブでリモートスケジュールを有効にするには、［リモートスケジュールを複製］および［双方向レプリケーション］オプションを有効にする必要があります。次のコードは、プログラムでこれを実行する方法を示します。

```
public void enableRemoteScheduling(IEnterpriseSession enterpriseSession, String
replicationJobName) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String replicationJobQuery = "SELECT SI_ID, SI_REPLICATE_REMOTESCHEDULES FROM
CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IReplication.KIND
    + "' AND SI_NAME = '" + replicationJobName + "' "
    + " AND SI_INSTANCE = 0";

  IInfoObjects replicationJobs = infostore.query(replicationJobQuery);
  IReplication replicationJob = (IReplication) replicationJobs.get(0);
  if(!replicationJob.getReplicateRemoteSchedules())
  {
    replicationJob.setReplicateRemoteSchedules(true);
    replicationJob.setPushEnabled(true);
    replicationJob.save();
  }
}
```

リモートで実行するレポートごとに、レポートの▶スケジュール▶サーバーグループのスケジュール▶設定にある［元のサイトで実行］オプションを有効にする必要があります。次のコードは、プログラムでこれを実行する方法を示します。

```
public void setRunAtOriginSite(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String reportQuery = "SELECT SI_ID, SI_SCHEDULEINFO FROM CI_INFOOBJECTS "
    + "WHERE SI_NAME = 'World Sales Report' AND SI_KIND = '"
    + IReport.KIND + "' AND SI_INSTANCE = 0";

  IInfoObjects reports = infostore.query(reportQuery);
  IReport report = (IReport) reports.get(0);
  ISchedulingInfo schedulingInfo = report.getSchedulingInfo();
  schedulingInfo.setRunOnOriginatingCluster(true);
  report.save();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.replication.IReplication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

関連情報

レプリケーションジョブを作成およびスケジュールする [243 ページ]

5.8.7 依存タイプ

レプリケーション一覧を作成する際、レプリケート時に含める依存のタイプを指定できます。`com.businessobjects.sdk.plugin.desktop.manifest.IManifest` の `getIncludedDependencies` メソッドは、`IIncludedDependencies` コレクションを返します。`addDependency` メソッドを使用して、レプリケートする依存タイプを追加します。最初のパラメータは、リレーション名を指定します。2 番目のパラメータは方向 (親から子、または子から親) で、`com.businessobjects.sdk.plugin.desktop.common.CeRelationshipDirection` インターフェイスのいずれかの定数に設定する必要があります。

次の表に、サポートされているリレーション/方向の組み合わせと対応する説明を示します。

リレーション名	方向	説明
Profile-Principal	PARENT_CHILDREN	選択したユーザーとユーザーグループのプロファイルを含む
DataConnection-Universe	CHILD_PARENTS	選択したユニバースによって使用される接続を含む
Category-Document	PARENT_CHILDREN	選択したカテゴリのドキュメントを含む
User-Inbox	PARENT_CHILDREN	選択したユーザーの受信ボックスを含む
Webi-Universe	PARENT_CHILDREN	選択したレポートのユニバースを含む
User-Favorites	PARENT_CHILDREN	選択したユーザーの個人用フォルダを含む
UserGroup-User	PARENT_CHILDREN	選択したユーザーグループのメンバーを含む
Universe(Core)-Universe	CHILD_PARENTS	選択したユニバースによって必要なユニバースを含む
EnterpriseData-Flash	CHILD_PARENTS	選択したフラッシュオブジェクトのサポートされている依存項目を含む
Universe-UserGroup	PARENT_CHILDREN	選択したユニバースのユーザーグループを含む
CustomRole-Object	CHILD_PARENTS	選択したオブジェクトに設定されるアクセスレベルを含む
User-PersonalCategory	PARENT_CHILDREN	選択したユーザーの個人用カテゴリを含む

関連情報

レプリケーション一覧を作成する [239 ページ]

5.9 モニタリング

モニタリングアプリケーションを使用すると、BI platform のサーバーのパフォーマンスを追跡して、重要なサービスの可用性を保証するために役立ちます。プローブと監視は、モニタリングアプリケーションの主要な 2 つの機能として、BI platform 内のサーバーとサービスのパフォーマンスと可用性のチェックに役立ちます。

プローブ

プローブは、エンドユーザーのワークフローをシミュレートするツールです。プローブを実行して、サーバーやサービスが正常に動作しているかどうかを確認できます。プローブを実行すると、プローブの結果とプローブの実行にかかった時間が返されます。プローブは、指定した間隔で実行されるようにスケジュールできます。各プローブの実行の結果と往復時間は、データベースに保存されます。このデータベース内の情報から推移グラフを作成して、能力計画に役立てることができます。

モニタリングアプリケーションには、いくつかのデフォルトのプローブが含まれています。BI platform では、カスタマイズしたプローブを作成してモニタリングアプリケーションに追加できます。提供される SDK、コマンド ライン インターフェイス、または DFO を使用して新しいプローブを作成できます。

5.9.1 新しいプローブの追加

提供される SDK を使用して、新しいプローブを作成し、それらをモニタリングアプリケーションに追加できます。モニタリング アプリケーションでは、各プローブが InfoObject として表されます。新しいプローブを作成し、その新しいプローブをモニタリング アプリケーションに統合するには、実装クラスを記述し、次に `Probe` 型の新しい InfoObject を作成する必要があります。InfoObject は、コマンド ライン インターフェイス、DFO、または SDK を使用して作成できます。

以下のセクションでは、新しいプローブを追加する方法を説明します。

実装クラスの記述

実装クラスを記述するには、以下の手順を実行します。

1. 提供されるプローブ SDK を使用して Java クラスを作成し、`.java` ファイルとして保存します。たとえば、`NewProbe.java` ファイルを作成します。
2. `NewProbe.java` ファイルをコンパイルします。
3. 次のコマンドを使用して JAR ファイルを作成します。

```
jar -cvf NewProbe.jar
                NewProbe.class
```

4. `C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib` ディレクトリに、`probes` という名前のディレクトリを作成します。
5. JAR ファイルおよび依存 JAR ファイルを `C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥probes` ディレクトリにデプロイします。

InfoObject の作成

InfoObject は、コマンド ライン インターフェイス (CLI)、DFO、または SDK を使用して作成できます。次の手順では、InfoObject を作成する方法について説明します。

SDK を使用して InfoObject を作成する

1. プローブ InfoObject プラグインを取得します。

```
IPluginInfo plugInfo =

infoStore.getPluginMgr().getPluginInfo("CrystalEnterprise.MON.Probe")
```

> ⓘ 注記
>
> `CrystalEnterprise.MON.Probe` はプラグイン名です。

2. 新しい InfoObject を作成します。

```
IInfoObject newProbe;
             newProbe = infoObjects.add(plugInfo);
             IProbeInfoObject probe = (IProbeInfoObject) newProbe;
```

3. 新しい InfoObject に以下のプロパティを追加します。

プロパティ	コマンド
プローブの名前	`String name = "Test Probe";` `probe.setTitle(name);`
プローブのクラス名	`String className = "com.test.probe" ;` `probe.setProbeClassName(className);`
タイムアウト（秒単位）	`long timeout = 10;` `probe.setTimeout(timeout);`

入力パラメータ（オプション）	`Map < String, Object > iMap = new HashMap < String, String > ();`
	`iMap.put("Test1", "Value1");`
	`iMap.put("Test2", true);`
	`iMap.put("Test3", new Integer(11));`
	`probe.setInputParameters (iMap);`

> ⓘ 注記
>
> InfoObject に指定するクラス名は、実装クラスで指定した名前と同じにする必要があります。

4. 新しいプローブを保存します。

```
probe.save();
```

CLI を使用して InfoObject を作成する

1. コマンド プロンプトで、`cd C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥win64_x64¥scripts` コマンドを入力します。
2. 次の表にある必要な属性とパラメータを使用して、`probeAdd` コマンドを実行します。

属性/パラメータ	説明
`-auth`	認証の種類
`-classname`	プローブの完全修飾クラス名
`-cms`	CMS 名
`-help`	このアプリケーションのヘルプを表示する
`-inputparam`	プローブの入力パラメータ。 たとえば、「key1:type1:value1; key2:type2:value2」。Integer/Long/Boolean/String をサポートしています。
`-name`	プローブの名前
`-password`	プローブを実行するパスワード
`-timeout`	タイムアウト間隔（秒単位）
`-username`	プローブを実行するユーザー名
`-type`	プローブのタイプを定義します。値には 1、2、3 があり、次のそれぞれに対応します。 • 1：ヘルスプローブ • 2：診断プローブ • 3:ヘルスプローブと診断プローブの両方として機能するプローブ

`-probeInputType`	commandLine または scriptUpload になります。
	if (commandLine)
	キー：CommandLine
	型：文字列
	値：<このコマンド>
	if (scriptUpload)
	`-scriptLocation` の後にスクリプトの場所を指定します

属性/パラメータ	説明
EnableVirtualMetrics	キー：EnableVirtualMetrics 型：論理値 値：true/false
Delimiter	キー：Delimiter 型：文字列 値：,
FirstRowhasColumnNames	キー：FirstRowhasColumnNames 型：論理値 値：true/false
metriccolumns	キー：Metric_Columns 型：整数 値：任意の整数
anchorcolumn	キー：AnchorColumn 型：整数 値：任意の整数

次のコマンドは、Java プローブの作成方法を示します。

```
probeAdd -username Administrator -name "Test" -password ""
             -timeout 100 -classname "com.test.java" -cms "localhost" -auth
secEnterprise -
             inputparam
"documented:Integer:1000;refresh:Boolean:false;TestParam:String:Test
             String".
```

次のコマンドは、スクリプトベースプローブの作成方法を示します。

```
probeAdd.bat -auth secEnterprise  -cms localhost -password Password1 -timeout
100 -username Administrator -type 3 -name testprobe -scriptbasedinput
scriptupload=C:¥monitoring_junit¥testScript.bat;enablevirtualmetrics=true;delimit
er=,;firstrowhascolumnnames=true;anchorcolumn=1;metriccolumns=5
```

CLI を使用して InfoObject を作成する

1. コマンド プロンプトで、cd C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥win64_x64¥scripts コマンドを入力します。
2. 次の表にある必要な属性とパラメータを使用して、probeAdd コマンドを実行します。

属性/パラメータ	説明
-auth	認証の種類
-classname	プローブの完全修飾クラス名

属性/パラメータ	説明
`-cms`	CMS 名
`-help`	このアプリケーションのヘルプを表示する
`-inputparam`	プローブの入力パラメータ。 たとえば、「key1:type1:value1; key2:type2:value2」。Integer/Long/Boolean/String をサポートしています。
`-name`	プローブの名前
`-password`	プローブを実行するパスワード
`-timeout`	タイムアウト間隔（秒単位）
`-username`	プローブを実行するユーザー名
`-type`	プローブのタイプを定義します。値には 1、2、3 があり、次のそれぞれに対応します。 • 1：ヘルスプローブ • 2：診断プローブ • 3:ヘルスプローブと診断プローブの両方として機能するプローブ

`-probeInputType`	commandLine または scriptUpload になります。
	if (commandLine)
	キー：CommandLine
	型：文字列
	値：<このコマンド>
	if (scriptUpload)
	`-scriptLocation` の後にスクリプトの場所を指定します

`EnableVirtualMetrics`	キー：EnableVirtualMetrics 型：論理値 値：true/false
`Delimiter`	キー：Delimiter 型：文字列 値：,
`FirstRowhasColumnNames`	キー：FirstRowhasColumnNames 型：論理値 値：true/false

属性/パラメータ	説明
metriccolumns	キー：Metric_Columns 型：整数 値：任意の整数
anchorcolumn	キー：AnchorColumn 型：整数 値：任意の整数

DFO を使用して InfoObject を作成する

1. 新しいプローブ InfoObject の DFO ファイルを作成し、それを `C:¥Program Files¥Business Objects¥BusinessObjects Enterprise 12.0¥Packages` ディレクトリ内に置きます。
2. CMS を再起動します。

5.9.1.1 新しいプローブの追加に使用するクラス

BI platform では、モニタリングアプリケーションに新しいプローブを追加したり既存のプローブのプロパティを変更するために、次のパッケージが公開されています。

- `com_businessobjects_sdk_monitoring_plugin_desktop_probe`。プローブ InfoObject の作成と変更に使用する次のクラスが含まれます。

  ```
  IProbeInfoObject.java
  IProbeInfoObjectBase.java
  ```

- `com_businessobjects_sdk_monitoring_probe`。プローブの実装に使用する次のクラスが含まれます。

  ```
  AbstractProbe.java
  DiagnosticProbeResult.java
  HealthProbeResult.java
  IDiagnosticProbe.java
  IDiagnosticProbeResult.java
  IHealthProbe.java
  IHealthProbeResult.java
  ```

5.10 プラットフォーム検索

プラットフォーム検索は、BI platform リポジトリ内のコンテンツを検索できるように最適化されたスケーラブルで高パフォーマンスな検索サービスです。このツールは、検索結果をいくつかのカテゴリにグループ化し、それらを関連度の順に順位付けることで、検索結果の絞り込みを行います。プラットフォーム検索は、継続的なインデックス化とスケジュールベースのインデックス化の両方をサポートします。

プラットフォーム検索は、ドキュメントがまだ存在しない場合に、ドキュメントを作成するためのクエリーを提示します。複数のコンテンツタイプを検索することができます。

5.10.1 プラットフォーム検索に使用されるクラス

次のパッケージには、検索結果を取得し、その検索結果を使用して Web Intelligence ドキュメントを作成するためのインターフェイスとクラスが含まれます。

- `com.sap.businessobjects.platform.search` には、検索要求を送信し、検索結果を取得するために使用される `IPlatformSearchService` インターフェイスが含まれます。
- `com.sap.businessobjects.platform.search.common` には、検索結果を処理し、Web Intelligence ドキュメントを作成するために使用される次のクラスが含まれます。

  `SearchIndexRequest`
  `SearchIndexResponse`
  `CreateDocumentRequest`
  `CreateDocumentResponse`

これらの API の詳細については、*SAP BusinessObjects Business Intelligence Platform Java API* リファレンスを参照してください。

5.10.2 検索結果を取得する

プラットフォーム検索 SDK を使用すると、検索結果を取得し、ユニバースのヒットから生成されるクエリーを使用して、検索結果から新しい Web Intelligence ドキュメントを作成できます。また、クエリーを編集したり、検索応答を使用してレポートを生成することもできます。

検索結果を取得して、オプションでその検索結果から Web Intelligence ドキュメントを作成するには、以下の手順を実行します。

1. 検索サービスを取得します。

```
IPlatformSearchService searchClient =
(IPlatformSearchService)
session.getService("PlatformSearchServiceOCA");
```

2. 検索要求オブジェクトのインスタンスを作成します。

```
SearchIndexRequest request = new SearchIndexRequest();
```

3. 以下のメソッドを使用して、要求オブジェクトのプロパティを設定します。

メソッド	使用法
`setQuery()`	`request.setQuery("Sales");`
`setMaxInstances()`	`request.setMaxInstances(4);`
`setCategoryTypes()`	`request.setCategoryTypes(catTypes);`

メソッド	使用法
setGroupFacetsByType()	`request.setGroupFacetsByType(true);`
setCategories()	`request.setCategories(new String[] {"FolderPath/Reports", "Country/US", "Country/India"});`
setPageSize()	`request.setPageSize(20);`
setPage()	`request.setPage(2);`
setHighLightingInfoEnabled()	`request.setHighLightingInfoEnabled(true)`
setSuggestedQueriesEnabled()	`request.setSuggestedQueriesEnabled(true)`
setCategoryInfoEnabled()	`request.setCategoryInfoEnabled(true)`
setHitLinksEnabled()	`request.setHitLinksEnabled(true)`
setHitPromptsEnabled()	`request.setHitPromptsEnabled(true)`
setMaxHitLinks()	`request.setMaxHitLinks(1)`
setMaxDocuments()	`request.setMaxDocuments(1000)`
setSearchAgents()	`request.setSearchAgents(new String[] {"AXk3RFXWBDRDtH5pfOJ6ilE","ATe1Gx2MGE5As8s0y1Z9ssE"})`
setMaxCategoriesShown()	`request.setMaxCategoriesShown(5);`
setMaxSubCategoriesShown()	`request.setMaxSubCategoriesShown(5);`

4. `IPlatformSearchService` インターフェイスの `search()`、`fullSearch()` 、または `quickSearch()` メソッドを呼び出して、検索サービスから検索結果を取得します。
   - `SearchIndexResponse response = searchClient.search(request)`
   - `SearchIndexResponse fullSearchResponse = searchClient.fullSearch(queryString, clientLocale);`

- ```
  SearchIndexResponse quickSearchResponse = searchClient.quickSearch(queryString, clientLocale, maxInstances, page, pageSize, categories);
  ```

5. 以下のコードを使用して、検索応答から検索結果を取得します。

   - ```
     IInfoObjects Hitobjs = response.getHitObjects();
     ```
   - ```
     IInfoObjects universeObjects = response.getUniverseObjects();
     ```
   - ```
     PropertyBag categories = response.getCategoryBag()
     ```
   - ```
     PropertyBag suggestions = response.getSuggestionBag()
     ```
   - ```
     int instanceCount = response.getTotalInstCount()
     ```
   - ```
     int documentCount = response.getTotalDocCount()
     ```

   > **注記**
   >
   > 検索要求から取得した応答を使用して Web Intelligence ドキュメントを作成する場合は、以下の手順を実行する必要があります。

6. ドキュメント要求のインスタンスを作成します。

   ```
   CreateDocumentRequest cdRequest = new CreateDocumentRequest();
   ```

7. 次のメソッドを使用して、ドキュメントの作成要求インスタンスのプロパティを設定します。

メソッド	使用法
`setUniverseCUID()`	`cdRequest.setUniverseCUID(universeCuid)`
`setTitle()`	`cdRequest.setTitle(title)`
`setAutoRun()`	`cdRequest.setAutoRun(true);`
`setAutoDrill()`	`cdRequest.setAutoDrill(true);`
`addResultObjectName()`	`cdRequest.addResultObjectName(name)`
`addFilter()`	`cdRequest.addFilter(String objectName, String[] values)`

8. 検索サービスを使用して、`createDocument()` メソッドを呼び出します。

   ```
   CreateDocumentResponse cdResponse = searchClient. createDocument(cdRequest);
   ```
   ```

9. Web Intelligence Desktop 一時ドキュメントが作成されるフォルダの ID を取得します。

```
String folderID = cdResponse.getTempFolderId();
```

> ⓘ 注記
>
> 新しく作成した Web Intelligence Desktop ドキュメントは、リポジトリに保存することも、破棄することもできます。

例: サンプルコード

次の例は、検索結果を取得し、Web Intelligence Desktop ドキュメントを作成します。

```
 IEnterpriseSession session = null;
 String queryString = "sales" ;
 try{
                session =
CrystalEnterprise.getSessionMgr().logon("administrator", "", "localhost",
"secEnterprise");
                IPlatformSearchService searchClient =
(IPlatformSearchService)session.getService("PlatformSearchServiceOCA");
                SearchIndexRequest searchRequest = new SearchIndexRequest();
                //Setting values for request
                searchRequest.setQuery(queryString);
                searchRequest.setCategories(new String[] {"FolderPath/Reports",
"Country/US", "Country/India"});
                searchRequest.setMaxInstances(4);
                searchRequest.setPageSize(20);
                searchRequest.setPage(2);
                searchRequest.setHighLightingInfoEnabled(true);
                searchRequest.setSuggestedQueriesEnabled(true);
                searchRequest.setCategoryInfoEnabled(true);
                searchRequest.setHitLinksEnabled(true);
                searchRequest.setHitPromptsEnabled(true);
                searchRequest.setCategoryTypes (new CategoryTypes[]
{SearchIndexRequest.CategoryTypes.METADATA_CATEGORY});
                searchRequest.setGroupFacetsByType(true);
    searchRequest.setMaxDocuments(10);
    searchRequest.setMaxHitLinks(10);
    searchRequest.setMaxCategoriesShown(5);
    searchRequest.setMaxSubCategoriesShown(5);

    SearchIndexResponse searchResponse = searchClient.search(searchRequest);
    SearchIndexResponse fullSearchResponse =
searchClient.fullSearch(queryString, new Locale("en" , "US"));
    SearchIndexResponse quickSearchResponse = searchClient.quickSearch
(queryString, new Locale("en" , "US"), 4,2,20, new String[] {"FolderPath/
Reports"});
//Using Hit Objects
IInfoObjects hitObjs = searchResponse.getHitObjects();
System.out.println("====Number of objects hit: "+hitObjs.size());
for(int nHitIndex =0;nHitIndex < hitObjs.size(); nHitIndex++)
{

        IInfoObject obj = (IInfoObject) hitObjs.get(nHitIndex);
        System.out.println("Content found = " + obj.getTitle() + " ID = "+
obj.getID());
     System.out.println("Folder Path for ObjID = " +
obj.properties().getProperty(PropertyIDs.nameToID("SI_FOLDER_PATH")).getValue().t
oString()+"¥n");
```

```
}
//Using Category
PropertyBag categories = searchResponse.getCategoryBag();
int subBagCount = categories.getInt("SI_TOTAL");
for(int nBag = 1 ; nBag<=subBagCount ; ++nBag) {
        PropertyBag subBag =
categories.getPropertyBag(nBag).getPropertyBag(String.valueOf(CategoryTypes.METAD
ATA_CATEGORY));
        System.out.println("====No of Category: " + subBag.getInt("SI_TOTAL"));
 }
//Using Suggestion
PropertyBag suggestions = searchResponse.getSuggestionBag();
int nCategorySize = suggestions.getInt("SI_TOTAL");
System.out.println("¥n====Number of Suggestions: "+nCategorySize);
for (int i=0; i<nCategorySize; i++) {
     PropertyBag topicBag =
suggestions.getItem(Integer.toString(i+1)).getPropertyBag();
     System.out.println("Did you mean: " +
topicBag.getString(CePropertyID.SI_HIT_TEXT) + "?");
}
//NOTE: This property bag will be set only when there will be no hits.
//Using Universe Object
IInfoObjects unvObjs = searchResponse.getUniverseObjects();
System.out.println("¥n====Number of Universe Objects: " + unvObjs.size());
for(int i=0; i < unvObjs.size(); i++){
        IInfoObject unvObj = (IInfoObject)unvObjs.get(i);
//You need to perform the following steps if you want to create a new Web
Intelligence document from the search results.
//Using Create Document
CreateDocumentRequest cdRequest = new CreateDocumentRequest();
cdRequest.setUniverseCUID(unvObj.getCUID());
cdRequest.setTitle("Test");
cdRequest.setAutoRun(true);
//Create Document Request accepts filter name and filter values which are
returned as a part of the universe response
IProperties filterProp =
unvObj.properties().getProperties(PropertyIDs.nameToID("SI_HIT_FILTERS"));
PropertyArrayHelper filterHelper = new
PropertyArrayHelper((PropertyBag)filterProp, PropertyIDs.SI_TOTAL);
for(int j=0; j < filterHelper.size(); j++){
        IProperties filterNameProp = (IProperties)filterHelper.get(j);
        IProperty property =
(IProperty)filterNameProp.getProperty(PropertyIDs.nameToID("SI_HIT_OBJECT"));
        String filtername = (String)property.getValue();
        System.out.println("¥n====Filter Name: "+filtername);
        IProperties filterValueProp =
filterNameProp.getProperties(PropertyIDs.nameToID("SI_HIT_VALUES"));
        PropertyArrayHelper valuesHelper = new
PropertyArrayHelper((PropertyBag)filterValueProp, PropertyIDs.SI_TOTAL);
        String[] filtervalue = new String[valuesHelper.size()];
        for(int k=0; k < valuesHelper.size(); k++){
                filtervalue[k] = (String)valuesHelper.get(k);
                System.out.println("====Filter Value: "+filtervalue[k]);
}
cdRequest.addFilter(filtername, filtervalue);
}
//Create Document Response accepts name of the hit-object which is returned as a
part of the universe response
IProperties nameProp =
unvObj.properties().getProperties(PropertyIDs.nameToID("SI_HIT_OBJECTS"));
PropertyArrayHelper nameHelper = new PropertyArrayHelper((PropertyBag)nameProp,
PropertyIDs.SI_TOTAL);
for(int l=0; l < nameHelper.size(); l++){
        cdRequest.addResultObjectName((String)nameHelper.get(l));
}
CreateDocumentResponse createResponse = searchClient.createDocument(cdRequest);
}
} catch (Exception e) {
```

```
        e.printStackTrace();
}
```

5.11 監査

BI platform インストール内のサーバーで発生したイベントに関する情報をログに記録するようにサーバーを設定できます。たとえば、ユーザーがログオンすると、Central Management Server（CMS）がログオンイベントを発生させます。管理者は、監査システムの動作だけでなく、追跡するサーバーイベントを設定できます。システムの監査設定をプログラムで操作できます。

監査イベントは、`com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo` インターフェイスで一元的に設定されます。このインターフェイスには、すべての監査イベントを取得するためのメソッドが含まれます。監査イベントは、論理カテゴリまたは累積的な事前設定されたレベルに基づいて分類できます。

監査レベルは、グループとして有効または無効にすることができる監査イベントのコレクションです。すべての監査レベルは累積的です。つまり、それ以前のレベルにあるすべての監査イベントを含みます。カスタム監査レベルまたは事前設定された監査レベルを使用して監査イベントを管理できます。

- 事前設定された 4 つの監査レベルを使用して、キャプチャするイベントを設定できます。
- カスタム監査レベルを使用すると、特定のイベントのコレクションを有効または無効にすることができます。

また、イベントをキャプチャするときに収集する詳細を各システムのニーズに合わせて調整できます。

監査イベントがトリガーされると、イベントをトリガーしたユーザーの名前などのイベントの詳細をシステムがキャプチャします。キャプチャされる詳細は、監査レベルとは別に設定されます。

> **注記**
>
> 監査の詳細については、*SAP Business Objects Business Intelligence Platform* 管理者ガイドを参照してください。

監査に使用されるクラス

- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
  監査のエントリポイントインターフェイスです。他の監査オブジェクトタイプを取得するためのメソッドが含まれます。
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents` および `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent`
  監査イベントを取得および設定するためのメソッドが含まれます。
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails` および `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail`
  キャプチャされる監査詳細を取得および設定するためのメソッドが含まれます。
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditLevels` および `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditLevel`
  事前設定された監査レベルを取得およびデプロイするためのメソッドが含まれます。

- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategories` および `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategory`
  事前設定された監査カテゴリを取得および検査するためのメソッドが含まれます。
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditConnectionInfo`
  監査データベースへの接続に使用される認証情報を取得および設定するためのメソッドが含まれます。また、SQL データベースタイプに基づいて、どのドライバを使用するかを定義します。

5.11.1 監査イベントを取得する

監査イベントは、`IAuditEventInfo` オブジェクトによって集中的に格納および管理されます。この例は、特定の監査イベント、その説明、およびイベントが属している監査レベルを取得する方法を示します。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリから監査イベントコレクションを取得し、それを `IAuditEventInfo` インターフェイスにキャストします。

   `IAuditEventInfo` インターフェイスに含まれる対応する `SI_KIND` 定数を使用して、コレクションを取得します。クエリーには、`SI_AUDIT_EVENTS` プロパティへの参照も含める必要があります。このプロパティには、システム内のすべての監査イベントの一覧が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS FROM CI_SYSTEMOBJECTS WHERE " +
  "SI_KIND ='" + IAuditEventInfo.KIND + "'";
IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
```

3. `IAuditEventInfo` インターフェイスの `getAuditEvents` メソッドを呼び出します。

```
IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
```

4. 特定のイベントタイプの ID を使用して、`IAuditEvents` オブジェクトの `getEvent` メソッドを呼び出します。

   View 監査イベントのイベントタイプ ID は 1002 です。その他のイベントタイプ ID は、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドに記載されています。

```
IAuditEvent viewEvent = auditEvents.getEvent(1002);
```

5. イベントの説明およびその監査レベルを取得します。

```
String viewEventProperties = "Event Description: " +
viewEvent.getDescription() +
  "<br>Audit Level: " + viewEvent.getAuditLevel();
```

> ⓘ 注記
>
> 監査レベルの整数値は、`IAuditEventInfo.AuditEventLevel` サブインターフェイスの定数フィールドの値に対応します。

例

次の例は、*View* 監査イベントの説明と監査レベルを含む String オブジェクトを返します。

```
String getViewEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS FROM CI_SYSTEMOBJECTS WHERE " +
    "SI_KIND ='" + IAuditEventInfo.KIND + "'";
  IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);

  IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
  IAuditEvent viewEvent = auditEvents.getEvent(1002);

  String viewEventProperties = "Event Description: " +
viewEvent.getDescription() +
    "<br>Audit Level: " + viewEvent.getAuditLevel();
  return viewEventProperties;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent

関連情報

監査 [259 ページ]

5.11.2 監査イベント詳細を取得する

監査イベント詳細は、IAuditEventInfo オブジェクトによって集中的に格納および管理されます。監査イベントの詳細は、監査イベントが発生した際に収集するデータの内容を決定します。複数のイベントタイプに関連付けられる詳細もあります。この例は、特定の監査イベント詳細オブジェクトを取得する方法を示します。

1. 有効な Enterprise セッションから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリから監査イベントコレクションを取得し、それを IAuditEventInfo インターフェイスにキャストします。

IAuditEventInfo インターフェイスに含まれる対応する SI_KIND 定数を使用して、コレクションを取得します。クエリーには、SI_AUDIT_EVENT_DETAILS プロパティへの参照も含める必要があります。このプロパティには、システム内のすべての監査イベントの一覧が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS FROM CI_SYSTEMOBJECTS
WHERE " +
        "SI_KIND = '" + IAuditEventInfo.KIND + "'";
IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
```

3. IAuditEventInfo インターフェイスの getAuditDetails メソッドを呼び出します。

```
IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
```

4. IAuditDetails オブジェクトの getDetail メソッドを呼び出します。

   Query 監査詳細のイベント詳細 ID は 5 です。その他の詳細 ID は、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドに記載されています。

```
IAuditDetail queryEventDetail = auditDetails.getDetail(5);
```

5. 詳細の説明と、詳細の収集が有効になっているかどうかを示すフラグを取得します。

```
String queryEventDetailDescription = "Event Detail Description: " +
queryEventDetail.getDescription() +
  "<br>Collection of this event detail is enabled?: " +
queryEventDetail.isEnabled();
```

例

次の例は、*Query* 監査イベント詳細の説明を含む String オブジェクトを返します。

```
String getQueryEventDetail(IEnterpriseSession enterpriseSession) throws
SDKException
{
  InfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS FROM CI_SYSTEMOBJECTS WHERE
" +
    "SI_KIND ='" + IAuditEventInfo.KIND + "'";
  IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);

  IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
  IAuditDetail queryEventDetail = auditDetails.getDetail(5);

  String queryEventDetailDescription = "Event Detail Description: " +
queryEventDetail.getDescription() +
    "<br>Collection of this event detail is enabled?: " +
queryEventDetail.isEnabled();
  return queryEventDetailDescription;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail

関連情報

監査 [259 ページ]
監査イベント詳細を有効にする [266 ページ]

5.11.3 現在の監査レベルを変更する

監査レベルは、次のように IAuditEventInfo.AuditEventLevel サブインターフェイスのフィールド値によって定義されます。

フィールド	int value
COMPLETE	4
CUSTOM	0
DEFAULT	3
MINIMAL	2
OFF	1

監査イベントは、それぞれ 1 つの監査レベルに属しています。監査レベルは累積的で、前のレベルのすべての監査イベントを含みます。

AuditEventLevel.CUSTOM 監査レベルを使用すると、キャプチャする監査イベントを個別に選択できます。この例は、CMS によって定義された現在の監査レベルをこのカスタムオプションに変更します。

1. 有効な Enterprise セッションから InfoStore サービスへの接続を作成します。

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリから監査イベントコレクションを取得し、それを IInfoObjects コレクションに割り当てます。

   IAuditEventInfo インターフェイスに含まれる対応する SI_KIND 定数を使用して、コレクションを取得します。クエリーには、SI_AUDIT_EVENTS プロパティと SI_AUDIT_LEVELS プロパティへの参照を含める必要があります。これらのプロパティには、IAuditLevel インターフェイスによって使用される詳細情報が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_LEVELS FROM
CI_SYSTEMOBJECTS " +
  "WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. IInfoObjects コレクションから IAuditEventInfo オブジェクトを取得します。

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. IAuditEventInfo インターフェイスの setCurrentAuditLevel メソッドを呼び出し、IAuditEventInfo.AuditEventLevel の値を渡します。

```
auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
```

5. 変更をリポジトリにコミットします。

```
infoStore.commit(infoObjects);
```

例

次の例は、現在の監査レベルを CUSTOM に設定して、カスタマイズされた監査イベントを設定できるようにします。

```
void changeAuditLevel(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_LEVELS FROM
CI_SYSTEMOBJECTS " +
    "WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
  infoStore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo.AuditEventLevel

関連情報

監査イベントを有効にする [265 ページ]
カスタム監査イベントを復元する [268 ページ]
監査 [259 ページ]

5.11.4 監査イベントを有効にする

このタスクは、現在の監査レベルが既にカスタム (`AuditEventLevel.CUSTOM`) に設定されていることを前提にしています。

設定済み監査レベルによって設定できるイベント以外のイベントは、個々に有効または無効にできます。有効にされたイベントは、ログに記録される監査イベントを参照する一連の ID として格納されます。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリから監査イベントコレクションを取得し、それを `IInfoObjects` コレクションに割り当てます。

   `IAuditEventInfo` インターフェイスに含まれる対応する `SI_KIND` 定数を使用して、コレクションを取得します。クエリーには、`SI_AUDIT_EVENTS` と `SI_ENABLED_AUDIT_EVENTS` への参照も追加する必要があります。これらには、有効にされているイベントセットを変更するために必要な情報が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_ENABLED_AUDIT_EVENTS FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. `IInfoObjects` コレクションから `IAuditEventInfo` オブジェクトを取得します。

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. `IAuditEventInfo` インターフェイスの `getAuditEvents` メソッドを呼び出し、結果として返される `IAuditEvents` オブジェクトを保存します。

```
IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
```

5. `IAuditEvents` インターフェイスの `getEvent` メソッドを呼び出し、結果として返される `IAuditEvent` オブジェクトを保存します。

   Retrieve 監査イベントのイベント ID は 1013 です。その他のイベントタイプ ID は、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドに記載されています。

```
IAuditEvent retrieveEvent = auditEvents.getEvent(1013);
```

6. `IAuditEventInfo` オブジェクトから現在有効な監査イベントの `java.util.Set` コレクションを取得します。

```
Set enabledEvents = auditEventInfo.getEnabledEventTypeIDs();
```

7. 有効な監査イベントの `java.util.Set` コレクションにイベントを追加します。

```
enabledEvents.add(retrieveEvent.getID());
```

8. 変更をリポジトリにコミットします。

```
infoStore.commit(infoObjects);
```

例

次の例は、システムで有効にされている監査イベントセットに *Retrieve* イベントを追加します。

```
void enableRetrieveEvent(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_ENABLED_AUDIT_EVENTS FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);

  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
  IAuditEvent retrieveEvent = auditEvents.getEvent(1013);
  Set enabledEvents = auditEventInfo.getEnabledEventTypeIDs();
  enabledEvents.add(retrieveEvent.getID());

  infoStore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent`
- `java.util.Set`

関連情報

現在の監査レベルを変更する [263 ページ]
監査イベントを取得する [260 ページ]
カスタム監査イベントを復元する [268 ページ]
監査 [259 ページ]

5.11.5 監査イベント詳細を有効にする

それぞれの監査イベントは、監査イベント詳細に関連付けられます。詳細は、イベントの発生時にキャプチャする情報を決定します。詳細には必須と任意があり、それぞれ有効または無効にすることができます。詳細は、監査レベルとは別に設定されます。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリから監査イベントコレクションを取得し、それを IInfoObjects コレクションに割り当てます。

   IAuditEventInfo インターフェイスに含まれる対応する SI_KIND 定数を使用して、コレクションを取得します。クエリーには、SI_AUDIT_EVENT_DETAILS パラメータと SI_DISABLED_AUDIT_EVENT_DETAILS パラメータへの参照も追加する必要があります。これらには、有効にされている詳細セットを変更するために必要な情報が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS,
SI_DISABLED_AUDIT_EVENT_DETAILS FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. IInfoObjects コレクションから IAuditEventInfo オブジェクトを取得します。

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. IAuditEventInfo インターフェイスの getAuditDetails メソッドを呼び出し、結果として返される IAuditDetails オブジェクトを保存します。

```
IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
```

5. 特定のイベント詳細の ID を使用して、IAuditDetails インターフェイスの getDetail メソッドを呼び出します。

   Query 監査イベント詳細の詳細 ID は 25 です。その他のイベント詳細 ID は、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドに記載されています。

```
IAuditDetail queryDetail = auditDetails.getDetail(25);
```

6. IAuditEventInfo オブジェクトから現在有効な監査イベント詳細の java.util.Set コレクションを取得します。

```
Set enabledDetails = auditEventInfo.getEnabledDetailTypeIDs();
```

7. 有効な監査イベント詳細の java.util.Set コレクションに詳細を追加します。

```
enabledDetails.add(queryDetail.getID());
```

8. 変更をリポジトリにコミットします。

```
infoStore.commit(infoObjects);
```

例

次の例は、システムで有効にされている監査イベントセットに *Query* イベント詳細を追加します。

```
void enableQueryDetail(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS,
SI_DISABLED_AUDIT_EVENT_DETAILS " +
    "FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
  IAuditDetail queryDetail = auditDetails.getDetail(25);
  Set enabledDetails = auditEventInfo.getEnabledDetailTypeIDs();
```

```
    enabledDetails.add(queryDetail.getID());
    infoStore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail
- java.util.Set

関連情報

監査イベント詳細を取得する [261 ページ]
監査 [259 ページ]

5.11.6 カスタム監査イベントを復元する

現在の環境で CUSTOM 監査レベルを使用および設定している場合は、カスタム設定が保存されるため、監査レベルを誤って変更した場合は設定を復元できます。この例は、前と同じ監査イベントおよび設定を使用して監査レベルを CUSTOM に変更する方法を示します。

1. 有効な Enterprise セッションから InfoStore サービスへの接続を作成します。

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリから監査イベントコレクションを取得し、それを IInfoObjects コレクションに割り当てます。

   IAuditEventInfo インターフェイスに含まれる対応する SI_KIND 定数を使用して、コレクションを取得します。クエリーには、SI_AUDIT_EVENTS プロパティと SI_AUDIT_LEVELS プロパティへの参照も含める必要があります。これらのプロパティには、保存されているカスタムイベントに戻したり、監査レベルを変更するために必要な情報が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_LEVELS FROM " +
        "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. IInfoObjects コレクションから IAuditEventInfo オブジェクトを取得します。

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. IAuditEventInfo インターフェイスの setCurrentAuditLevel メソッドを呼び出し、IAuditEventInfo.AuditEventLevel の CUSTOM 値を渡します。

```
auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
```

5. 変更をリポジトリにコミットします。

```
infoStore.commit(infoObjects);
```

6. IInfoStore を使用して、IInfoObjects セットの更新を照会します。

   IAuditEventInfo インターフェイスに含まれる対応する SI_KIND 定数を使用して、コレクションを取得します。クエリーには、SI_AUDIT_EVENTS プロパティと SI_PREVIOUS_ENABLED_AUDIT_EVENTS プロパティへの参照も含める必要があります。これらのプロパティには、カスタムイベントを復元するために必要な情報が含まれます。

```
auditQuery = "SELECT SI_AUDIT_EVENTS, SI_PREVIOUS_ENABLED_AUDIT_EVENTS FROM "
+
        "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
infoObjects = infoStore.query(auditQuery);
```

7. IInfoObjects コレクションから更新された IAuditEventInfo オブジェクトを取得します。

```
auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

8. java.util.List コレクションを使用して、IAuditEventInfo オブジェクトから以前に有効にされたカスタムイベントの一覧を取得します。

```
List previousEventsList =
auditEventInfo.getPreviousEnabledCustomAuditEvents();
```

9. java.util.Set コレクションを使用して、IAuditEventInfo オブジェクトから現在有効なカスタムイベントのセットを取得します。

```
Set enabledEventsSet = auditEventInfo.getEnabledEventTypeIDs();
```

10. このセットから有効なイベントをクリアして、監査レベルの変更によって追加されたイベントをすべて削除します。

```
enabledEventsSet.clear();
```

11. 以前に有効にされたカスタムイベントの一覧を有効なカスタムイベントの空のセットに追加します。

```
enabledEventsSet.addAll(previousEventsList);
```

12. 変更をリポジトリにコミットします。

```
infoStore.commit(infoObjects);
```

例

次の例は、監査レベルをカスタムに変更してから、以前に設定したイベントと同じイベントを有効にします。

```
void restoreCustomEvents(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_PREVIOUS_ENABLED_AUDIT_EVENTS
FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
```

```
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
  infoStore.commit(infoObjects);
  auditQuery = "SELECT SI_AUDIT_EVENTS, SI_PREVIOUS_ENABLED_AUDIT_EVENTS FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  infoObjects = infoStore.query(auditQuery);

  auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  List previousEventsList = auditEventInfo.getPreviousEnabledCustomAuditEvents();

  Set enabledEventsSet = auditEventInfo.getEnabledEventTypeIDs();
  enabledEventsSet.clear();
  enabledEventsSet.addAll(previousEventsList);

  infoStore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo.AuditEventLevel`
- `java.util.Set`
- `java.util.List`

関連情報

現在の監査レベルを変更する [263 ページ]
監査イベントを有効にする [265 ページ]
監査 [259 ページ]

5.11.7 監査カテゴリを取得する

論理的に類似するカテゴリに基づいて監査イベントを整理できます。この例は、監査カテゴリおよびその一部のプロパティを取得する方法を示します。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリから監査イベントコレクションを取得し、それを `IAuditEventInfo` インターフェイスにキャストします。

   `IAuditEventInfo` インターフェイスに含まれる対応する `SI_KIND` 定数を使用して、コレクションを取得します。クエリーには、`SI_AUDIT_EVENTS` プロパティと `SI_AUDIT_CATEGORIES` プロパティへの参照を含

める必要があります。これらのプロパティには、IAuditCategory インターフェイスによって使用される詳細情報が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_CATEGORIES FROM " +
        "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
```

3. IAuditEventInfo オブジェクトから IAuditCategories コレクションを取得します。

```
IAuditCategories auditCategories = auditEventInfo.getAuditCategories();
```

4. IAuditCategories インターフェイスの getCategory メソッドを使用して、*Common Events* カテゴリを取得します。

   Common Events カテゴリの ID は 1 です。

```
IAuditCategory commonEvents = auditCategories.getCategory(1);
```

5. カテゴリの説明を取得します。

```
String commonEventDescription = commonEvents.getDescription();
```

6. 関連する監査イベントの ID の java.util.Set コレクションを取得して、String オブジェクトにキャストします。

```
String commonEventIDs = commonEvents.getEventIDs().toString();
```

例

次の例は、*Common Events* 監査カテゴリのローカライズされた説明を含む String オブジェクトの配列と、関連する監査イベントの ID セットを返します。

```
String[] commonEventCategoryInfo(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_CATEGORIES FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
  IAuditCategories auditCategories = auditEventInfo.getAuditCategories();
  IAuditCategory commonEvents = auditCategories.getCategory(1);

  String[] commonEventProperties = new String[2];
  commonEventProperties[0] = commonEvents.getDescription();
  commonEventProperties[1] = commonEvents.getEventIDs().toString();

  return commonEventProperties;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategories`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategory`

関連情報

監査イベントを取得する [260 ページ]
監査 [259 ページ]

5.11.8 サーバー監査メトリクスを取得する

監査に関連するメトリクスを含むさまざまなサーバーメトリクスを Central Management Server（CMS）から取得できます。この例は、まだ収集されないまま CMS からポーリングされている監査イベントの数を取得します。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. リポジトリからサーバーオブジェクトを取得し、`IServer` オブジェクトタイプにキャストします。

   結果を保証するには、`SI_DESCRIPTION` の説明を CMS で使用されている説明と照合します。このクエリーには、`SI_METRICS` プロパティへの参照も含める必要があります。このプロパティには、メトリクスデータにアクセスするために必要な情報が含まれます。

```
String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS " +
        "WHERE SI_DESCRIPTION = 'Central Management Server'";
IServer server = (IServer) infoStore.query(serverQuery).get(0);
```

3. `IServer` オブジェクトから `IServerMetrics` コレクションを取得します。

```
IServerMetrics serverMetrics = server.getMetrics();
```

4. `IServerMetrics` コレクションから `AuditAdmin` `IMetrics` オブジェクトを取得します。

   他の監査メトリクスは、`APSAdmin` `IMetrics` オブジェクトから取得できます。

```
IMetrics auditMetrics = serverMetrics.getMetrics("AuditAdmin");
```

5. `IMetrics` コレクションから *auditeeNumberOfEventsRemaining* `IMetric` オブジェクトを取得します。

   CMS では、*auditeeNumberOfEventsRemaining* `IMetric` オブジェクトは、`AuditAdmin` `IMetrics` コレクションの最初の位置を保持します。

```
IMetric numEvents = (IMetric) auditMetrics.get(0);
```

6. `IMetric` オブジェクトから名前と値を取得します。

```
numEvents.getName();
numEvents.getValue();
```

例

次の例は、監査メトリクスセットからまだ収集されないまま CMS からポーリングされている監査イベントの数を返します。

```
String getServerAuditMetric(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS " +
    "WHERE SI_DESCRIPTION = 'Central Management Server'";
  IServer server = (IServer) infoStore.query(serverQuery).get(0);

  IServerMetrics serverMetrics = server.getMetrics();
  IMetrics auditMetrics = serverMetrics.getMetrics("AuditAdmin");
  IMetric numEvents = (IMetric) auditMetrics.get(0);
  return numEvents.getName() + " " + numEvents.getValue().toString();
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerMetrics`
- `com.businessobjects.sdk.plugin.desktop.common.IMetric`
- `com.businessobjects.sdk.plugin.desktop.common.IMetrics`

関連情報

監査 [259 ページ]

5.11.9 監査データストア接続情報を取得する

監査データベース接続の多くの属性をプログラムで変更できます。次の例は、監査データベース接続のいくつかのプロパティを取得する方法を示します。適切な setter メソッドを使用してさまざまな値を設定し、変更した `IInfoObjects` コレクションを Central Management Server（CMS）にコミットすることができます。これらのメソッドの詳細については、*SAP BusinessObjects Enterprise Java API* リファレンスを参照してください。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリから監査イベントコレクションを取得し、それを `IInfoObjects` コレクションに割り当てます。

   `IAuditEventInfo` インターフェイスに含まれる対応する `SI_KIND` 定数を使用して、コレクションを取得します。また、`SI_AUDIT_EVENTS` と `SI_AUDIT_DB_CONNECTION` にはデータベース接続プロパティを取得し

て変更するために必要な情報が含まれているため、クエリーにはこれらのプロパティへの参照も含まれている必要があります。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUBIT_DB_CONNECTION FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. IInfoObjects コレクションから IAuditEventInfo オブジェクトを取得します。

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. IAuditEventInfo コレクションから IAuditConnectionInfo オブジェクトを取得します。

```
IAuditConnectionInfo connectionInfo = auditEventInfo.getAuditConnectionInfo();
```

5. IAuditConnectionInfo インターフェイスの getUserName メソッドを呼び出して、ユーザー名を取得します。

```
connectionInfo.getUserName()
```

> ⓘ 注記
>
> 監査データベースへのログオンに使用するパスワードをプログラムで設定するには、setPassword メソッドを使用します。パスワードは、SDK を使用して取得できません。

6. IAuditConnectionInfo インターフェイスの getDsnName メソッドを呼び出して、データソース名を取得します。

```
connectionInfo.getDsnName()
```

7. IAuditConnectionInfo インターフェイスの getDatabaseDriver メソッドを呼び出して、監査データベースのデータベースドライバタイプを取得します。

```
connectionInfo.getDatabaseDriver()
```

> ⓘ 注記
>
> 対応する setter メソッドでは、事前設定された 6 つのドライバ識別子を使用できます。監査イベントを格納するデータベースのタイプに適したドライバを選択してください。
>
> - IAuditConnectionInfo.CeAuditDatabaseDriver.DB2_AUDIT_DB_DRIVER
> - IAuditConnectionInfo.CeAuditDatabaseDriver.MAXDB_AUDIT_DB_DRIVER
> - IAuditConnectionInfo.CeAuditDatabaseDriver.MYSQL_AUDIT_DB_DRIVER
> - IAuditConnectionInfo.CeAuditDatabaseDriver.ORACLE_AUDIT_DB_DRIVER
> - IAuditConnectionInfo.CeAuditDatabaseDriver.SQL_SERVER_AUDIT_DB_DRIVER
> - IAuditConnectionInfo.CeAuditDatabaseDriver.SYBASE_AUDIT_DB_DRIVER
>
> CeAuditDatabaseDriver.SQL_SERVER_AUDIT_DB_DRIVER を使用する場合は、IAuditConnectionInfo インターフェイスの setUseTrustedConnection メソッドを使用して、シングルサインオン（信頼される接続）変数を設定できます。

例

次の例は、アクティブな監査データベース接続からいくつかのプロパティを取得し、それらを String オブジェクトとして返します。

```
String getAuditDBConnectionInfo(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_DB_CONNECTION FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  IAuditConnectionInfo connectionInfo = auditEventInfo.getAuditConnectionInfo();
  String adsConnectionInfo = "User name: " + connectionInfo.getUserName() +
"<br>" +
    "DSN name: " + connectionInfo.getDsnName() + "<br>" +
    "Database driver: " + connectionInfo.getDatabaseDriver();

  return adsConnectionInfo;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditConnectionInfo
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditConnectionInfo.CeAuditDatabaseDriver

関連情報

監査 [259 ページ]

5.11.10 監査イベントの格納期間を変更する

監査イベントは、監査データベースに数日間格納されてから自動的に削除されます。この値は通常、セントラル管理コンソール（CMC）で設定されます。この例は、イベントを格納する期間をプログラムによって変更する方法を示します。

1. 有効な Enterprise セッションから InfoStore サービスへの接続を作成します。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. リポジトリから監査イベントコレクションを取得し、それを IInfoObjects コレクションに割り当てます。

IAuditEventInfo インターフェイスに含まれる対応する SI_KIND 定数を使用して、コレクションを取得します。このクエリーには、SI_AUDIT_EVENTS と SI_AUDIT_DELETE_AFTER_N_DAYS への参照も含める必要があります。これらには、システムがログ格納期間を変更するために必要な情報が含まれます。

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_DELETE_AFTER_N_DAYS
FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. IInfoObjects コレクションから IAuditEventInfo オブジェクトを取得します。

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. setDeleteAfterNDays メソッドを呼び出し、監査イベントを格納する日数を表す正の整数を渡します。

> ⚠ 警告
>
> ここに設定されている日数より古いデータは、監査データベースから完全に削除され、復元できなくなります。記録を長期間維持するには、アーカイブデータベースに定期的に移動することを考慮する必要があります。

```
auditEventInfo.setDeleteAfterNDays(365);
```

5. 変更をリポジトリにコミットします。

```
infoStore.commit(infoObjects);
```

例

次の例は、監査イベントを 365 日間格納するように期間を変更します。

```
void setAuditEventStorageDuration(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_DELETE_AFTER_N_WEEKS
FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  auditEventInfo.setDeleteAfterNDays(365);
  infoStore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo

関連情報

監査 [259 ページ]

5.11.11 トレーサビリティを強化してカスタムアプリケーション全体を監査する

SAP BusinessObjects Business Intelligence platform 4.1 には、IEnterpriseLogonInformation オブジェクトを追加のパラメータとして受け取る新しいログオンメソッドが追加されました。このメソッドを使用した場合は、監査イベントがトリガーされたときに、IEnterpriseLogonInformation オブジェクトに提供されたカスタムアプリケーションの CUID と名前が記録されます。使用可能なすべての CUIDs とそのカスタムアプリケーション名の一覧については、カスタムアプリケーションの CUID 値と名前 [278 ページ]を参照してください。

1. staticCrystalEnterprise インタフェースの getSessionMgr メソッドを呼び出して、ISessionMgr オブジェクトを取得します。

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
```

2. ISessionMgr インタフェースの createLogonInfo メソッドを呼び出して、IEnterpriseLogonInformation オブジェクトを取得します。

```
IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
```

3. IEnterpriseLogonInformation インタフェースの setClientType メソッドを呼び出して、クライアントタイプを設定します。

   アプリケーションで使用するために予約されているカスタムアプリケーション識別子は、25 個あります。これらの識別子は、 Custom Application 1 から Custom Application 25 までの命名規則に従っており、setClientType メソッドにパラメータとして渡される固有の CUID を持っています。

```
logonInfo.setClientType("AZ4HLbS0loRMpE2twk442RU");
```

4. ISessionMgr インターフェイスの logonEx メソッドを呼び出して、IEnterpriseSession オブジェクトを取得します。

```
IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "myCMS:6400", "secEnterprise", logonInfo);
```

これで、このセッションでトリガーされるログオンイベントとそれ以降のすべてのイベントには、システムへのアクセスに使用されるカスタムアプリケーションの名前が含まれます。IEnterpriseSession は、通常どおりに使用します。

例

次の例は、logonEx メソッドを使用して CMS にアクセスする方法を示します。

```
IEnterpriseSession logonExDemonstration() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
```

```
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setClientType("AZ4HLbS0loRMpE2twk442RU");

  IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "myCMS:6400", "secEnterprise", logonInfo);
  return enterpriseSession;
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.security.IEnterpriseLogonInformation`

関連情報

監査 [259 ページ]
カスタムアプリケーションの CUID 値と名前 [278 ページ]

5.11.12 カスタムアプリケーションの CUID 値と名前

カスタムアプリケーションの CUID 値

これらの予約済みの CUID 値は、`IEnterpriseLogonInformation` オブジェクトを受け取るログオン呼び出しをアプリケーションに実装する際に `IEnterpriseLogonInformation.setClientType` メソッドに渡すことができます。記録された監査イベントには、監査イベントの発生元になったセッションに対応するカスタムアプリケーションの CUID と名前が示されます。

CUID	アプリケーション名
AZ4HLbS0loRMpE2twk442RU	カスタムアプリケーション 1
AR6QLgQwJjxElm5O2FT_b4A	カスタムアプリケーション 2
AWYcIdOex_tAodjJRrlogIk	カスタムアプリケーション 3
ASahxem3v_xGjXsQKQaskyk	カスタムアプリケーション 4
AbKQQkLhQ9RKpbMEV7DECkc	カスタムアプリケーション 5
AcBrUi5J0rVCiFZ.M35hCYM	カスタムアプリケーション 6
Afbw5UJ0UYdJlNbhDusG1mc	カスタムアプリケーション 7
AT084MQMNHIIs1fUB9Y4x8k	カスタムアプリケーション 8
AT0F5dLN5rZJq9IuT7pILTU	カスタムアプリケーション 9
AXKjeUbzVq1Gm9XS4kIF1Ho	カスタムアプリケーション 10
ARc7Bcof_V1JpaH.LaBmKMM	カスタムアプリケーション 11

CUID	アプリケーション名
AZFkpRO4waBGvZNBt4R85YY	カスタムアプリケーション 12
AZETCpaYYmRIingE_hCnWFw	カスタムアプリケーション 13
ARAGSiINIAtMvkpuANjWU6I	カスタムアプリケーション 14
AcfWGfuiPeREmYs3drOx0sQ	カスタムアプリケーション 15
AUKOYrZfNBpNimTvy.m_WxU	カスタムアプリケーション 16
Acdt0d02A6VNq3o8DWHWpH8	カスタムアプリケーション 17
ASRvIZ.mkeVJrhmSxnCqFb8	カスタムアプリケーション 18
AXydQKHjGVROt_8czecFvxE	カスタムアプリケーション 19
Ae25eOj9rcFMj31xm.i_5U8	カスタムアプリケーション 20
AU0U6I7_vcJKsSBk8eZBQBM	カスタムアプリケーション 21
AUPgOLoPI9JGI5M19R1SeXw	カスタムアプリケーション 22
Aa7yGnOcE1pPmfkDuxznG7U	カスタムアプリケーション 23
AQ8m0QHV9LNArZcRo6a2xuM	カスタムアプリケーション 24
ASyT3vZzFuZKkE7iXkp_tuI	カスタムアプリケーション 25

関連情報

監査 [259 ページ]
トレーサビリティを強化してカスタムアプリケーション全体を監査する [277 ページ]

5.12 ローカライズと言語パック

優先表示ロケールの管理

現在のユーザーの優先表示ロケールを保存し、後でそれを取得できます。これは、レポート内の数値と通貨の書式など、アプリケーションのロケール固有の要素をパーソナライズする必要がある場合に役立ちます。

優先表示ロケールは、ユーザーの `IEnterpriseSession` オブジェクトに保存されます。デフォルトでは、この値は設定されず、Central Management Server（CMS）に保存されないため、必要に応じてアプリケーションで設定する必要があります。

インストールおよびサポートされている言語パックの表示

SAP BusinessObjects Business Intelligence platform SDK には、インストールされている言語パックとサポートされている言語パックを取得するためのメソッドも用意されています。

優先表示ロケールおよび言語パックの管理に使用されるクラス

- `com.crystaldecisions.sdk.occa.security.IUserInfo`
  現在ログオン中のユーザーを識別し、そのユーザーの優先ロケールを取得または設定するための情報が入っています。
- `com.crystaldecisions.sdk.framework.ILanguageMgr`
  サポートされているロケール、およびこのインストール環境にインストールされているロケール（言語パック）に関する情報へのアクセスを提供します。

5.12.1 優先表示ロケールを設定する

1. 有効な `IEnterpriseSession` セッションオブジェクトから現在のユーザーを取得します。

```
IUserInfo userInfo = enterpriseSession.getUserInfo();
```

2. 現在のユーザーの優先表示ロケールを設定します。

```
userInfo.setPreferredViewingLocale(Locale.SIMPLIFIED_CHINESE);
```

例

```
public void setPreferredViewingLocale(IEnterpriseSession enterpriseSession)
throws SDKException
{
  IUserInfo userInfo = enterpriseSession.getUserInfo();
  userInfo.setPreferredViewingLocale(Locale.SIMPLIFIED_CHINESE);
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.security.IUserInfo`
- `java.util.Locale`

5.12.2 現在の優先表示ロケールを表示する

1. 有効な IEnterpriseSession セッションオブジェクトから現在のユーザーを取得します。

```
IUserInfo userInfo = enterpriseSession.getUserInfo();
```

2. 現在のユーザーの優先表示ロケールを取得します。

```
Locale preferredViewingLocale = userInfo.getPreferredViewingLocale();
```

3. 優先表示ロケールの表示名を取得します。

```
String localeName = preferredViewingLocale.getDisplayName();
```

例

```
public String getPreferredViewingLocale(IEnterpriseSession
enterpriseSession)throws SDKException
{
  IUserInfo userInfo = enterpriseSession.getUserInfo();
  Locale preferredViewingLocale = userInfo.getPreferredViewingLocale();
  String localeName = preferredViewingLocale.getDisplayName();
  return localeName;
}
```

サンプルコードで使用されるクラスを次に示します。

- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.security.IUserInfo
- java.util.Locale

5.12.3 インストールされている言語パックを表示する

1. ISessionMgr オブジェクトのインスタンスを取得します。

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
```

2. ILanguageMgr オブジェクトのインスタンスを取得します。

```
ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
```

3. ILanguageMgr オブジェクトを使用して、インストールされている言語パックを取得します。

```
 List installedLocales = languageMgr.getInstalledLocales();
```

4. インストールされている言語パックを反復処理し、その表示名を取得します。

```
String list = new String("");
Iterator installedLocalesIter = installedLocales.iterator();
```

```
while (installedLocalesIter.hasNext())
{
  Locale locale = (Locale) installedLocalesIter.next();
  list += locale.getDisplayName() + "¥n";
}
```

例

次のコードは、この SAP BusinessObjects Business Intelligence platform のインストールにインストールされている言語パックのリストを含む String オブジェクトを返します。

```
public String getInstalledLanguagePacks() throws SDKException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
  ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
  List installedLocales = languageMgr.getInstalledLocales();
  Iterator installedLocalesIter = installedLocales.iterator();

  String list = new String("");
  while (installedLocalesIter.hasNext())
  {
    Locale locale = (Locale) installedLocalesIter.next();
    list += locale.getDisplayName() + "<p>";
  }
  return list;
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.ILanguageMgr`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `java.util.Iterator`
- `java.util.List`
- `java.util.Locale`

5.12.4 サポートされているすべての言語を表示する

1. `ISessionMgr` オブジェクトのインスタンスを取得します。

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
```

2. `ILanguageMgr` オブジェクトのインスタンスを取得します。

```
ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
```

3. `ILanguageMgr` オブジェクトを使用して、サポートされている言語を取得します。

```
List supportedLocales = languageMgr.getSupportedLocales();
```

4. サポートされている言語を反復処理し、それらの表示名を取得します。

```
String list = new String("");
Iterator it = supportedLocales.iterator();
while (it.hasNext())
{
  Locale locale = (Locale) it.next();
  list += locale.getDisplayName() + "<p>";
}
```

例

次のコードは、この SAP BusinessObjects Business Intelligence platform インストールでサポートされている言語のリストを含む String オブジェクトを返します。

```
public String displaySupportedLanguagesDescription() throws SDKException
{
     ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
     ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
     List supportedLocales = languageMgr.getSupportedLocales();

  String list = new String("");
     Iterator it = supportedLocales.iterator();
     while (it.hasNext())
     {
          Locale locale = (Locale) it.next();
          list += locale.getDisplayName() + "<p>";
     }
     return list;
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.ILanguageMgr`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `java.util.Iterator`
- `java.util.List`
- `java.util.Locale`

5.13 暗号化キーの管理

暗号化キーは、BI platform に格納されている InfoObject を暗号化するために使用します。暗号化キーの使用方法については、*SAP Business Objects Business Intelligence Platform* 管理者ガイドを参照してください。

この SDK を使用して暗号化キーを管理できます。ほとんどのキー管理タスクを実行するには、暗号管理者グループのメンバーとして認証されている必要があります。暗号化キーの状態は、任意のユーザーが確認できます。ただし、暗号化キーの作成、変更、および削除を実行できるのは、暗号管理者ユーザーのみです。

暗号管理者グループのメンバーになっているユーザーは、次のような暗号化キーの管理タスクを実行できます。

- 新しいキーを作成します。新しいキーがアクティブキーになります。
- 非アクティブキーを無効にします。キーを無効にすると、それを使用する InfoObject はアクティブキーによって再暗号化されます。
- 無効にしたキーを削除します。

> ⓘ 注記
>
> 暗号化されたデータを BI platform の外部に格納している場合は、その暗号化キーを削除する前に、データを手動で解読する必要があります。

関連情報

暗号化キーの状態 [284 ページ]

5.13.1 暗号化キーの状態

各 Central Management Server（CMS）クラスタには、アクティブな暗号化キーが 1 つと、非アクティブで無効化されたキーが任意の数だけ含まれます。

アクティブキーを使用して、BI platform インストール内に格納されている InfoObject を暗号化および解読します。セキュリティを強化するには、アクティブキーを新しいキーに定期的に置き換えます。アクティブキーを置き換えた場合、そのキーは無効になります。つまり、そのキーはデータの暗号化に使用されなくなりますが、そのキーで暗号化されたオブジェクトの解読には使用されます。

リポジトリからキーを削除するには、そのキーで暗号化された InfoObject が解読され、さらに新しいキーで再暗号化されていることを確認する必要があります。そうしないと、暗号化キーを削除した後に、それらの InfoObject を解読したりアクセスすることができなくなります。`ICryptographicKey` クラスの `startRekey` メソッドを使用して、BI platform インストールに格納されている InfoObject を自動的に再暗号化することができます。キーの交換プロセスが完了したら、古いキーをリポジトリから削除できます。

> ⓘ 注記
>
> 暗号化されたデータを BI platform インストールの外部に格納している場合は、その暗号化キーを削除する前に、データを手動で解読する必要があります。

暗号化キーの状態を確認するには、`IClusterKey.KeyState` クラスの `getKeyState` メソッドを呼び出します。

状態	使用法	説明
`ACTIVE`	暗号化 解読	このキーは、データの暗号化および解読に使用されます。

状態	使用法	説明
DEACTIVATED	解読	このキーは暗号化には使用されません。ただし、このキーがアクティブキーであったときにこのキーで暗号化されたデータを解読する際は使用されます。
REKEY_IN_PROGRESS	解読	キーの交換処理が進行中です。このキーで暗号化されたオブジェクトは、アクティブなクラスタキーで再暗号化中です。
REKEY_SUSPENDED	解読	キーの交換処理が開始されましたが、一時停止されました。
REVOKED	なし	このキーで暗号化されたオブジェクトはリポジトリに格納されていません。このキーは削除できます。 **注記** 暗号化されたオブジェクトを BI platform インストールの外部に格納している場合は、その暗号化キーを削除する前に、オブジェクトを手動で解読する必要があります。

5.13.2 新しい暗号化キーを作成する

暗号化キーを作成するには、暗号管理者グループのメンバーであるアカウントを使用して、BI platform にログオンする必要があります。

セキュリティを強化するには、アクティブな暗号化キーを新しいキーに定期的に置き換えます。アクティブな暗号化キーは、置き換えた時点で自動的に非アクティブになります。

1. 有効な `IEnterpriseSession` セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. 暗号化キーを格納するための新しい `InfoObjects` コレクションを作成します。

```
IInfoObjects infoObjects = infostore.newInfoObjectCollection();
```

3. `InfoObjects` コレクションにプラグイン情報を追加することで、新しい暗号化キーを作成します。

```
ICryptographicKey cryptographicKey = (ICryptographicKey)
infoObjects.add(ICryptographicKey.KIND);
```

4. 暗号化キーの名前を設定します。

   この例では、暗号化キーの名前を `CryptographicKey` に設定しています。

```
cryptographicKey.setTitle("CryptographicKey");
```

5. `Cryptographic Keys` フォルダを取得します。

```
IInfoObjects folders = infostore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND='" + IFolder.FOLDER_KIND +
  "' AND SI_CUID='" + CeSecurityCUID.RootFolder.CRYPTOGRAPHIC_KEYS + "'");
IInfoObject folder = (IInfoObject) folders.get(0);
```

6. Cryptographic Keys フォルダを暗号化キーの親フォルダに設定します。

```
cryptographicKey.setParentID(folder.getID());
```

7. 暗号化キーを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(infoObjects);
```

アクティブな暗号化キーが非アクティブになり、新しい暗号化キーに置き換えられます。

例: 新しい暗号化キーの作成

```
void createKey(IEnterpriseSession enterpriseSession)throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  ICryptographicKey cryptographicKey =
(ICryptographicKey)infoObjects.add(ICryptographicKey.KIND);
  cryptographicKey.setTitle("CryptographicKey");

  IInfoObjects folders = infostore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND='" + IFolder.FOLDER_KIND +
    "' AND SI_CUID='" + CeSecurityCUID.RootFolder.CRYPTOGRAPHIC_KEYS + "'");
  IInfoObject folder = (IInfoObject) folders.get(0);

  cryptographicKey.setParentID(folder.getID());
  infostore.commit(infoObjects);
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.cryptographickey.ICryptographicKey
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.CeSecurityCUID
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.plugin.desktop.folder.IFolder

5.13.3 新しいキーでデータを再暗号化する

新しいキーでデータを再暗号化するには、暗号管理者グループのメンバーであるアカウントを使用して、BI platform にログオンする必要があります。再暗号化プロセスを開始するには、キーを無効にしておく必要があります。

アクティブキーで InfoObject を再暗号化するには、ICryptographicKey クラスの startRekey メソッドを呼び出します。再暗号化プロセスが完了したら、リポジトリからキーを削除できます。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. InfoStore から暗号化キーを取得します。

```
IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + masterKeyName + "'");
ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);
```

3. キーが DEACTIVATED 状態にあることを確認します。

> ⓘ 注記
>
> キーがまだ ACTIVE 状態にある場合は、アクティブな暗号化キーを新しく作成することによって無効にできます。

```
ICryptographicKey.KeyState state = key.getKeyState();
if (state == ICryptographicKey.KeyState.DEACTIVATED)
  ...
```

4. ICryptographicKey クラスの startRekey メソッドを呼び出して、再暗号化プロセスを開始します。

```
key.startRekey();
```

> ⓘ 注記
>
> suspendRekey メソッドと resumeRekey メソッドを使用すると、再暗号化プロセスを中断および再開できます。

5. レポートを保存します。

```
key.save();
```

このキーで暗号化されている InfoObject は、アクティブキーによって再暗号化され、キーの状態は KeyState.REVOKED に変更されます。

例

```
void startRekey(IEnterpriseSession enterpriseSession, String masterKeyName)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + masterKeyName + "'");
  ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);

  ICryptographicKey.KeyState state = key.getKeyState();
  if (state == ICryptographicKey.KeyState.DEACTIVATED)
  {
    key.startRekey();
    key.save();
  }
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.plugin.desktop.cryptographickey.ICryptographicKey
- com.crystaldecisions.sdk.exception.SDKException

- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoStore
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects

5.13.4 暗号化キーを削除する

暗号化キーをリポジトリから削除するには、その状態が REVOKED である必要があります。ICryptographicKey クラスの startRekey メソッドを使用して、キーを無効にします。

無効にされたキーは使用されなくなり、リポジトリから削除することができます。

> **注記**
>
> 暗号化されたデータを BI platform の外部に格納している場合は、その暗号化キーを削除する前に、データを手動で解読する必要があります。

1. 有効な IEnterpriseSession セッションオブジェクトから InfoStore サービスへの接続を作成します。

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. InfoStore から暗号化キーを取得します。

```
IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + keyName + "'");
ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);
```

3. キーが無効になっていることを確認します。

```
ICryptographicKey.KeyState state = key.getKeyState();
if (state == ICryptographicKey.KeyState.REVOKED)
  ...
```

4. IInfoObjects クラスの delete メソッドを呼び出して、キーを削除します。

```
infoObjects.delete(key);
```

5. 変更されたキーコレクションを Central Management Server（CMS）にコミットし直し、変更を保存します。

```
infostore.commit(infoObjects);
```

例: 使用されていないキーを削除する

```
void deleteKey(IEnterpriseSession enterpriseSession, String keyName) throws
SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + keyName + "'");
  ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);
  ICryptographicKey.KeyState state = key.getKeyState();
  if (state == ICryptographicKey.KeyState.REVOKED)
  {
```

```
    infoObjects.delete(key);
    infostore.commit(infoObjects);
  }
}
```

サンプルコードで使用されるクラスを次に示します。

- `com.businessobjects.sdk.plugin.desktop.cryptographickey.ICryptographicKey`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`

関連情報

新しいキーでデータを再暗号化する [286 ページ]

5.14 Business Intelligence Archive (BIAR)

SAP BusinessObjects Business Intelligence プラットフォーム BIAR エンジンは、管理者や委任管理者がユーザ、グループ、セキュリティ設定、レポートなどのビジネスインテリジェンスコンテンツを `BIAR` ファイルにバックアップするためのツールです。このファイルを使用して、いつでもコンテンツを復元できます。また、このツールを使用すると、クラスタ間でコンテンツを直接エクスポートすることもできます。つまり、最初にコンテンツを `BIAR` ファイルにエクスポートする必要がありません。

BIAR エンジンは、次の方法で使用できます。

- SAP BusinessObjects アップグレードマネジメントツールを使用する。
  SAP BusinessObjects アップグレードマネジメントツールは、SAP BusinessObjects Business Intelligence プラットフォームデプロイメントのアップグレード処理中に、以前のバージョンの SAP BusinessObjects Business Intelligence プラットフォームからビジネスインテリジェンスをインポートして、現在のバージョンにアップグレードします。

  > **注記**
  >
  > 詳細については、*SAP Business Objects Business Intelligence* プラットフォームアップグレードガイドを参照してください。

- SAP BusinessObjects Business Intelligence プラットフォーム 4.1 向けライフサイクルマネジメントコンソールを使用する。
  SAP BusinessObjects Business Intelligence プラットフォーム 4.1 向けライフサイクルマネジメントコンソールは、ライフサイクル管理プロセスの中で、1 つの SAP BusinessObjects Business Intelligence プラットフォームデプロイメントから同じバージョンの別の SAP BusinessObjects Business Intelligence プラットフォームデプロイメントにコンテンツをアップグレードします。

> **注記**
>
> 詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム *4.1* 向けライフサイクルマネジメントコンソールユーザガイドを参照してください。

- BIAR エンジンコマンドラインユーティリティを使用する。
  BIAR コマンドラインツールは、リポジトリ内で `BIAR` ファイルをインポートおよびエクスポートします。BIAR エンジン (`biarengine.jar`) は、SAP BusinessObjects Business Intelligence プラットフォームデプロイメントの以下のデフォルトの場所にあります。`C:¥Program Files (x86)¥SAP BusinessObjects¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥`。

  > **注記**
  >
  > 詳細については、*SAP BusinessObjects Business Intelligence* プラットフォーム管理者ガイドを参照してください。

- この SDK を使用する。
  SAP BusinessObjects Business Intelligence プラットフォーム Java SDK を使用して、ローカルファイルシステムにある `BIAR` ファイルから Central Management Server (CMS) に InfoObject をインポートしたり、CMS からローカルディレクトリにある `BIAR` ファイルに InfoObject をエクスポートしたり、CMS から別の CMS に InfoObject を直接インポートすることができます。

> **注記**
>
> BIAR コマンドラインツールとこの SDK は、同じバージョンの SAP BusinessObjects Business Intelligence プラットフォーム間のインポートとエクスポートをサポートします。以前のバージョンの SAP BusinessObjects Business Intelligence プラットフォームから現在のバージョンにオブジェクトをインポートする場合は、SAP BusinessObjects Enterprise アップグレードマネジメントツールを使用する必要があります。

> **注記**
>
> 転送元環境と転送先環境の Service Pack と FixPack のレベルは、同じである必要があります。オブジェクトをインポートまたはエクスポートする前に、両方の環境の更新レベルが同じであることを確認してください。

BIAR エンジンの操作時に使用されるクラス

- `com.businessobjects.sdk.biar.BIARFactory`
  `IObjectManager` インスタンスを作成するために使用されます。
- `com.businessobjects.sdk.biar.ILiveToLivePipe`
  `BIAR` ファイルを作成する必要なく、デプロイメント間でオブジェクトを転送するために使用されます。
- `com.businessobjects.sdk.biar.IExportOptions`
  InfoObject のエクスポート時にオプションを定義するために使用されます。
- `com.businessobjects.sdk.biar.IImportOptions`
  InfoObject のインポート時にオプションを定義するために使用されます。
- `com.businessobjects.sdk.biar.om.IManagedObjectIterator`
  `IObjectManager` インスタンスに保持されている InfoObject を反復処理するために使用されます。
- `com.businessobjects.sdk.biar.om.IObjectManager`

InfoObject のコレクションを管理するために使用されます。

5.14.1 インポートに関する考慮事項

コミットテストの実行

コミットテストを使用すると、ある InfoObject のコレクションを CMS にコミットした場合に発生するエラーを判別できます。SDK は、コレクションをコミットし、エラーがあればそれらを収集し、変更を自動的にロールバックするように CMS に指示します。

> **注記**
>
> コミットテストは、CMS へのオブジェクトのコレクションの追加をテストするだけです。コミットテストを使用して、File Repository Server へのファイルのコピーをテストすることはできません。

コミットテストを実行するには、`IObjectManager` オブジェクトの `testCommit` メソッドを呼び出します。`testCommit` メソッドは `ICommitResult` オブジェクトを返し、コミット中にエラーが発生した場合は、このコレクションにエラーが格納されます。

```
private ICommitResult performTestCommit(IObjectManager objectManager) {
  ICommitOptions commitOptions = BIARFactory.getCommitOptions();
  ICommitResult commitResult = objectManager.testCommit(commitOptions);
  return ICommitResult;
}
```

オブジェクトとセキュリティのインポート

BIAR エンジンでは、コンテンツをインポートまたはエクスポートする際にセキュリティ設定もインポートするかどうかを指定できます。

セキュリティ設定をインポートせずにコンテンツをインポートまたはエクスポートするには、`IImportOptions` クラスまたは `IExportOptions` クラスの `setIncludeSecurity` メソッドを呼び出します。

```
BIARFactory biarFactory;
IImportOptions importOptions = biarFactory.createImportOptions();
importOptions.setIncludeSecurity(false);
```

5.14.2 BIAR ファイルに InfoObject をエクスポートする

この例では、プログラムで BIAR ファイルを作成し、それをデータベースからローカルディレクトリにエクスポートする方法を示します。

1. 有効な Enterprise セッションから `InfoStore` サービスへの接続を作成します。

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. エクスポートするオブジェクトをデータベースから取得します。

```
String query = "select * from ci_Infoobjects";
IInfoObjects infoObjects = infoStore.query(query);
```

3. BIARFactory オブジェクトと IObjectManager オブジェクトを作成します。

```
BIARFactory biarFactory = BIARFactory.getFactory();
IObjectManager objectManager = biarFactory.createOM(enterpriseSession);
```

4. エクスポートするオブジェクトを IObjectManager オブジェクトに追加します。

```
objectManager.insertAll(infoObjects.iterator());
```

5. エクスポートオプションを構成します。

```
IExportOptions exportOptions = biarFactory.createExportOptions();
exportOptions.setIncludeSecurity(true);
exportOptions.setFailUnresolvedIDs(true);
```

6. setCallback メソッドを呼び出します。

   コールバックメソッドはイベント処理の手段です。この SDK では、エクスポートに関するイベントは IExportCallback インターフェイスによって処理されます。エクスポートイベントの通知を受けるオブジェクトは、IExportCallback インターフェイスを実装する必要があります。したがって、この例では、MyExportCallback クラスを使用して、IExportCallback インターフェイスメソッドを実装します。後で、ユーザーにフィードバックを提供するテキスト文字列を出力するための onSuccess メソッドを実装できます。

```
exportOptions.setCallback(new MyExportCallback());
```

7. IObjectManager クラスの exportToArchive メソッドを呼び出して、BIAR ファイルにオブジェクトをエクスポートします。このメソッドには、コンテンツのエクスポート先の BIAR ファイルの名前とパスを表す文字列引数が必要です。

```
try
{
 objectManager.exportToArchive(biarFile, exportOptions);
}
catch(BIARException e)
{
 e.printStackTrace();
}
System.out.println(objectManager.size() + " objects processed.");
```

> ⓘ 注記
>
> 例外処理の詳細については、SDK 例外のキャッチおよび処理 [69 ページ]を参照してください。

8. IObjectManager オブジェクトを破棄します。

```
objectManager.dispose();
}
```

このコードを実行すると、選択した InfoObject が CMS システムデータベースからローカルディレクトリの BIAR ファイルにエクスポートされます。実行時にローカルディレクトリを指定する必要があります。BIAR ファイルは、後で設定を復元するためのバックアップとして使用することもできます。

例

```
public void exportInfoObjects(IEnterpriseSession enterpriseSession, String
biarFile) throws SDKException, OMException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String query = "select * from ci_Infoobjects";
  IInfoObjects infoObjects = infoStore.query(query);
  BIARFactory biarFactory = BIARFactory.getFactory();
  IObjectManager objectManager = biarFactory.createOM(enterpriseSession);
  objectManager.insertAll(infoObjects.iterator());

  IExportOptions exportOptions = biarFactory.createExportOptions();
  exportOptions.setIncludeSecurity(true);
  exportOptions.setFailUnresolvedIDs(true);

  exportOptions.setCallback(new MyExportCallback());

  try
  {
    objectManager.exportToArchive(biarFile, exportOptions);
  }
  catch (BIARException e)
  {
    e.printStackTrace();
  }
  finally
  {
    System.out.println(objectManager.size()+ "objects processed.");
          objectManager.dispose();
  }

}
```

この一覧には、サンプルコードで使用されるクラスが含まれます。

- `com.businessobjects.sdk.biar.BIARException`
- `com.businessobjects.sdk.biar.BIARFactory`
- `com.businessobjects.sdk.biar.IExportCallback`
- `com.businessobjects.sdk.biar.IExportOptions`
- `com.businessobjects.sdk.biar.om.IObjectManager`
- `com.businessobjects.sdk.biar.om.OMException`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

関連情報

BIAR ファイルからインポートする [294 ページ]
コールバックを実行する [298 ページ]

5.14.3 BIAR ファイルからインポートする

このタスクでは、ローカル ファイル システムにある BIAR ファイルをデータベースにインポートする方法を示します。データベースにインポートする BIAR ファイルは、ローカルディレクトリになければなりません。

1. 有効な Enterprise セッションから InfoStore サービスへの接続を作成します。

```
IInfoStore infoStore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. BIARFactory オブジェクトと IObjectManager オブジェクトを作成します。

```
BIARFactory biarFactory = BIARFactory.getFactory();
IObjectManager objectManager = biarFactory.createOM(enterpriseSession);
```

3. インポートオプションを構成します。

```
IImportOptions importOptions = biarFactory.createImportOptions();
importOptions.setIncludeSecurity(true);
importOptions.setFailUnresolvedCUIDs(true);
```

4. setCallback メソッドを呼び出します。

```
importOptions.setCallback(new MyImportCallback());
```

> ⓘ 注記
>
> コールバックメソッドはイベント処理の手段です。この SDK では、インポートに関するイベントは IImportCallback インターフェイスによって処理されます。インポートイベントの通知を受けるオブジェクトは、IImportCallback インターフェイスを実装する必要があります。したがって、この例では、MyImportCallback クラスを使用して、IImportCallback インターフェイスメソッドを実装します。

5. 変更をデータベースにコミットします。

```
objectManager.commit();
System.out.println(objectManager.size() + " objects processed.");
```

6. BIAR ファイルをインポートします。

```
try
{
 objectManager.importArchive(biarFile, importOptions);
}
catch(BIARException e)
{
 e.printlnStackTrace();
}
```

> ⓘ 注記
>
> インポートしようとしている InfoObject が既にデータベースに存在する場合、既存のオブジェクトは新しいオブジェクトとマージされます。マージ後のオブジェクトには、両方のオブジェクトのプロパティが入りますが、それらが異なる場合は、インポート先のオブジェクトのプロパティよりインポート元のオブジェクトのプロパティが優先します。

7. 各オブジェクトに関する情報を表示し、反復子を閉じます。

```
IManagedObjectIterator managedObjectIter;
```

```
managedObjectIter = objectManager.readAll();
while(managedObjectIter.hasNext())
  {
    IInfoObject infoObject = managedObjectIter.next();
    System.out.println("SI_CUID: " + infoObject.getCUID()
      + ", SI_NAME: " + infoObject.getTitle() ", SI_KIND: " +
infoObject.getKind());
  }
managedObjectIter.close();
```

この手順は省略可能ですが、情報通知のために実行することをお勧めします。

8. オブジェクトマネージャを破棄します。

```
objectManager.dispose();
}
```

ローカルディレクトリの BIAR ファイルは、実行時に指定したデータベースにインポートされます。このインポートを確認するには、データベースをチェックします。これで、インポートされた BIAR ファイルを使用したり、このファイルを別の場所にエクスポートすることができます。

例

```
public void importInfoObjects(IEnterpriseSession enterpriseSession, String
biarFile) throws SDKException, OMException
{
  BIARFactory biarFactory = BIARFactory.getFactory();
  IObjectManager objectManager = biarFactory.createOM(enterpriseSession);

  IImportOptions importOptions = biarFactory.createImportOptions();
  importOptions.setIncludeSecurity(true);
  importOptions.setFailUnresolvedCUIDs(true);

  importOptions.setCallback(new MyImportCallback());
  try
  {
    objectManager.importArchive(biarFile,importOptions);
                objectManager.commit();
    IManagedObjectIterator managedObjectIter;
    managedObjectIter = objectManager.readAll();

    while(managedObjectIter.hasNext())
    {
      IInfoObject infoObject = managedObjectIter.next();
      System.out.println("SI_CUID:" +infoObject.getCUID()+ ", SI_NAME:" +
infoObject.getTitle()+ ",SI_KIND:"+infoObject.getKind());
    }
  }
  catch (BIARException e)
  {
    e.printStackTrace();
  }
  finally
  {
    managedObjectIter.close();
       System.out.println(objectManager.size() + "objects processed.");
       objectManager.dispose();
  }
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.biar.BIARException
- com.businessobjects.sdk.biar.BIARFactory
- com.businessobjects.sdk.biar.IImportCallback
- com.businessobjects.sdk.biar.IImportOptions
- com.businessobjects.sdk.biar.om.IManagedObjectIterator
- com.businessobjects.sdk.biar.om.IObjectManager
- com.businessobjects.sdk.biar.om.OMException
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- java.util.Iterator

関連情報

BIAR ファイルに InfoObject をエクスポートする [291 ページ]
コールバックを実行する [298 ページ]

5.14.4 live-to-live 転送を実行する

この例では、BIAR ファイルを使用しなくても、プログラムでオブジェクトを転送元環境からエクスポートし、そのオブジェクトを転送先環境にインポートする方法を示します。

この手順では、転送元 CMS と転送先 CMS にそれぞれ Enterprise セッションが必要です。

1. 有効な Enterprise セッションから InfoStore サービスへの接続を作成します。

```
        IInfoStore infoStore = (IInfoStore)
sourceSession.getService("InfoStore");
```

2. エクスポートするオブジェクトをデータベースから取得します。

```
 String query = "select * from ci_Infoobjects";
 IInfoObjects infoObjects = infoStore.query(query);
```

3. BIARFactory オブジェクトと IObjectManager オブジェクトを作成します。

   IObjectManager オブジェクトをインスタンス化する際は、転送元の Enterprise セッションを使用します。

```
BIARFactory biarFactory = BIARFactory.getFactory();
IObjectManager objectManager = biarFactory.createOM(destinationSession);
```

4. 転送元 CMS から IObjectManager オブジェクトにオブジェクトを追加します。

```
objectManager.insertAll(infoObjects.iterator());
```

5. エクスポートオプションを構成します。

```
IExportOptions exportOptions = biarFactory.createExportOptions();
exportOptions.setIncludeSecurity(true);
```

```
exportOptions.setFailUnresolvedIDs(true);
exportOptions.setIncludeDependencies(true);
```

6. インポートオプションを構成します。

```
IImportOptions importOptions = biarFactory.createImportOptions();
importOptions.setIncludeSecurity(true);
importOptions.setFailUnresolvedCUIDs(true);
```

7. `IObjectManager` クラスの `liveImport` メソッドを呼び出します。このメソッドは、`ILiveToLivePipe` オブジェクトを返します。

   パラメータとして転送元環境の Enterprise セッションを渡す必要があります。

```
ILiveToLivePipe pipe = objectManager.liveImport(sourceSession, exportOptions,
importOptions);
```

8. `ILiveToLivePipe` オブジェクトの `getExporter` メソッドを呼び出して、`IExporter` オブジェクトを作成します。IExporter オブジェクトを使用して、ソースシステムから出力先システムにオブジェクトをエクスポートします。

```
try{
     IExporter exporter = pipe.getExporter();
     try
     {
          exporter.exportAll(infoObjects);
          exporter.finish();
     }
     finally{
          exporter.close();
     }
     pipe.getResult().get();
}
finally
{
     dispose();
}
```

9. `IObjectManager` をコミットしてから破棄します。

```
objectManager.commit();
objectManager.dispose();
```

例

```
public String liveTransfer(IEnterpriseSession destinationSession,
IEnterpriseSession sourceSession) throws SDKException, OMException, BIARException
{
  IInfoStore infoStore = (IInfoStore)sourceSession.getService("InfoStore");
  String query = "select * from ci_Infoobjects";
  IInfoObjects infoObjects = infoStore.query(query);

  BIARFactory biarFactory = BIARFactory.getFactory();
  IObjectManager objectManager = biarFactory.createOM(destinationSession);
  objectManager.insertAll(infoObjects.iterator());
  IExportOptions exportOptions = biarFactory.createExportOptions();
  exportOptions.setIncludeSecurity(true);
  exportOptions.setFailUnresolvedIDs(true);
  exportOptions.setIncludeDependencies(true);
```

```
  IImportOptions importOptions = biarFactory.createImportOptions();
  importOptions.setIncludeSecurity(true);
  importOptions.setFailUnresolvedCUIDs(true);

  ILiveToLivePipe pipe = objectManager.liveImport(sourceSession, exportOptions,
importOptions);

  try{
     IExporter exporter = pipe.getExporter();
     try
     {
          exporter.exportAll(infoObjects);
          exporter.finish();
     }
     finally{
          exporter.close();
     }
     pipe.getResult().get();
  }
  finally
  {
     dispose();
  }

  objectManager.commit();
  objectManager.dispose();
}
```

サンプルコードで使用されるクラスを次に示します。

- com.businessobjects.sdk.biar.BIARException
- com.businessobjects.sdk.biar.BIARFactory
- com.businessobjects.sdk.biar.IExporter
- com.businessobjects.sdk.biar.IExportOptions
- com.businessobjects.sdk.biar.IImportCallback
- com.businessobjects.sdk.biar.IImportOptions
- com.businessobjects.sdk.biar.ILiveToLivePipe
- com.businessobjects.sdk.biar.om.IObjectManager
- com.businessobjects.sdk.biar.om.OMException
- com.crystaldecisions.sdk.exception.SDKException
- com.crystaldecisions.sdk.framework.IEnterpriseSession
- com.crystaldecisions.sdk.occa.infostore.IInfoObject
- com.crystaldecisions.sdk.occa.infostore.IInfoObjects
- com.crystaldecisions.sdk.occa.infostore.IInfoStore

5.14.5 コールバックを実行する

コールバックを実行するには、BIAR ファイルをデータベースにインポートしたり、データベースからエクスポートする必要があります。

コールバックを使用すると、オブジェクトがインポートまたはエクスポート中に発生するイベントの通知を受け取ることができます。コールバックを実行するには、エクスポート時には IExportCallback インターフェイス、インポート時には IImportCallback インターフェイスをそれぞれ実装する必要があります。この例では、

IExportCallback インターフェイスを実装する MyExportCallback クラスを使用してエクスポートコールバックを実行する方法を示しています。

> **注記**
>
> この例では、MyExportCallback クラスが exportInfoObjects メソッド内で setCallback メソッドから呼び出されます。このメソッドは、BIAR ファイルに exportOptions を設定する際に実装します。

1. onSuccess メソッドを実装します。

```
public class MyExportCallback implements IExportCallback
{
  public void onSuccess(int id)
  {
    System.out.println("Object with ID " + id + " was exported
successfully.");
  }
...
```

2. onFailure メソッドを実装します。

```
...
  public void onFailure(int id, BIARException cause)
  {
    System.out.println("Object with ID " + id + " failed to export: " +
cause.getMessage());
  }
}
```

> **注記**
>
> インポートアクションのコールバックを実行するには、onSuccess メソッドおよび onFailure メソッドの代わりに onProgressUpdate メソッドを使用します。onProgressUpdate メソッドの詳細については、*SAP BusinessObjects Business Intelligence Platform Java SDK* を参照してください。

例

```
public class MyExportCallback implements IExportCallback
{
  public void onSuccess(int id) {
    System.out.println("Object with ID " + id
      + " was exported successfully.");
  }
  public void onFailure(int id, BIARException cause) {
    System.out.println("Object with ID " + id + " failed to export: "
      + cause.getMessage());
  }
}
```

関連情報

BIAR ファイルに InfoObject をエクスポートする [291 ページ]

BIAR ファイルからインポートする [294 ページ]

6 参照

6.1 処理拡張機能 API リファレンス

処理拡張機能 API を使用すると、独自の処理拡張機能を作成できます。処理拡張機能はプラグイン DLL（Windows システムの場合）または共有ライブラリ（UNIX システムの場合）で、表示やスケジュールのリクエストが行われたレポートに対して操作を加えることができます。たとえば、レポートに関する情報を収集したり、必要に応じて、Crystal Reports Cache Server、Job Server、または Report Application Server（RAS）で表示またはスケジュールリクエストが処理される前に、レポートのレコード選択式やレポートパラメータを変更することができます。

処理拡張機能の典型的な例として、行レベルのセキュリティを組み込むレポート処理プラグインがあります。このプラグインでは、最初にその処理ジョブを所有するユーザーを識別し、サードパーティシステム内のそのユーザーのデータアクセス権を調べます。次に、レコード選択式を生成してレポートに追加し、データベースから返されるデータを絞り込みます。この処理拡張機能は、カスタマイズされた行レベルのセキュリティを実装する手段として動作します。

6.1.1 拡張機能の作成

プラグインにはマルチスレッドのサポートが必要なため、C または C++ を使用して、DLL または共有ライブラリを構築することをお勧めします。

- このバージョンでは、Java で記述された処理拡張機能はサポートされません。
- Windows システムでは、動的にロードされるライブラリを「ダイナミックリンクライブラリ」（ファイル拡張子は .dll）と呼びます。UNIX システムでは、動的にロードされるライブラリは通常 "共有ライブラリ"（ファイル拡張子は .so）と呼ばれます。処理拡張機能に名前を付ける際には、.dll または .so ファイル拡張子を含めることが必要です。また、ファイル名に"¥"と"/"の文字を含むことはできません。

6.1.2 拡張機能のロード

セントラル管理コンソール（CMC）と SDK は、ユーザーの処理拡張機能を有効にし、各拡張機能が適用されるオブジェクトを指定する方法を提供します。処理拡張機能を有効にすると、適切な BI platform サーバーコンポーネントが実行時に動的に処理拡張機能をロードするように設定できます。

- 現行バージョンでは、処理拡張機能を適用できるのは Crystal レポートオブジェクトだけです。処理拡張機能を特定のレポートに適用するには、CMC を使用するか、SDK の `IReportGlobal` を使用して `IProcessingExtension` を取得します。
- スケジュールするレポートが処理拡張機能を使用する場合、作成されるインスタンスは同じ処理拡張機能を使うことが必要となります。

すべての Job Server、Report Application Server (RAS)、および Crystal Reports Cache Server には、それぞれにインストールされた処理拡張機能の一覧があります。各レポートも適用される処理拡張機能の一覧を持っていま

す。レポートの表示またはスケジュールのリクエストを出すと、BI platform はそのレポートの処理拡張機能のロードを試みます。拡張機能がロードできないと（たとえば、必要な拡張機能がそのサーバーにインストールされていない場合）、エラーが送られ、その操作はセキュリティ違反として中断されます。

拡張機能のロードに成功すると、以下の操作の実行に使用できます。

- サーバーがユーザー ID やレポートタイトルなどの情報を取得するよう要求します。
- 新規の選択式を作成し、既にレポート内にある選択式に追加します。こうしてできる選択式は、元の選択式と処理拡張機能が作成した新規の選択式を（AND 論理演算子を使用して）組み合わせたものです。
- レポートとサブレポートのプロンプト値を変更します。

処理拡張機能を使用すると、データとスレッド処理が共有から非共有状態に変わります。

処理拡張機能をロードするには、該当のスケジュールまたは表示リクエストを処理する各マシンで、あらかじめその機能を使用可能にしておく必要があります。処理拡張機能は、各サーバーにコピーするか、複数のサーバーで共有することにより、マシンに登録できます。

6.1.2.1 処理拡張機能のサーバーへのコピー

BI platform インストールでは、Crystal Reports Cache Server、Job Server、Report Application Server（RAS）に処理拡張機能用のデフォルトディレクトリが作成されます。処理拡張機能は、各サーバーのデフォルトディレクトリにコピーすることをお勧めします。デフォルトの処理拡張機能ディレクトリの場所については、ディレクトリ [95 ページ]を参照してください。

> → ヒント
>
> 処理拡張機能ファイルは共有が可能です。詳細については、複数サーバーでの処理拡張機能の共有 [302 ページ]を参照してください。

拡張機能に作成した機能に応じて、次のようにライブラリをマシンにコピーします。

- 処理拡張機能をスケジュールリクエストだけに割り込ませる場合、Crystal Reports Job Server が動作する各マシンにライブラリをコピーします。
- 処理拡張機能を表示リクエストだけに割り込ませる場合、Crystal Reports Cache Server として動作する各マシンにライブラリをコピーします。
- 処理拡張機能をスケジュールリクエストと表示リクエストの両方に割り込ませる場合は、Crystal Reports Job Server と Crystal Reports Cache Server の両方または一方が動作するすべてのマシンにライブラリをコピーします。

> ⓘ 注記
>
> 特定のサーバーグループへ送信されるスケジュール/表示リクエストだけに処理拡張機能が必要な場合は、そのグループのすべての Crystal Reports Cache Server、Job Server、RAS サーバーにライブラリをコピーします。

6.1.2.2 複数サーバーでの処理拡張機能の共有

すべての処理拡張機能を 1 か所に格納する場合は、各 Crystal Reports Cache Server、Job Server、または Report Application Server のデフォルトの処理拡張機能ディレクトリを上書きします。最初に、すべてのサーバーからア

クセス可能なネットワークドライブ上の共有ディレクトリに処理拡張機能をコピーします。これらのサーバーを含む各マシンからネットワークドライブをマップ（またはマウント）します。

> ⓘ 注記
>
> サーバーを Windows と UNIX の両方で動作させている場合は、すべての処理拡張機能の `.dll` バージョンと `.so` バージョンを共有ディレクトリにコピーする必要があります。さらに、共有ネットワークドライブが（Samba などのファイル共有システムによって）Windows と UNIX から見えることが必要です。

最後に、サーバーのコマンド ラインを変更して、デフォルトの処理拡張機能ディレクトリを変更します。これは、コマンド ラインに「`-report_ProcessExtPath` <絶対パス>」を追加して行います。サーバーが動作するオペレーティングシステムのパス規則に従って、<絶対パス> を新しいフォルダのパスに置き換えます (たとえば、`M:¥code¥extensions` や `/home/shared/code/extensions`)。

この変更手順はオペレーティングシステムによって異なります。

- Windows では、セントラル管理コンソールを使用して、Crystal Reports Cache Server/Job Server/RAS サーバーを停止します。そして、サーバーのプロパティを開いてコマンド ラインを変更します。完了したら、サーバーを再起動します。
- UNIX では、Crystal Reports Cache Server/Job Server/RAS サーバーで `ccm.sh` を実行します。そして、`ccm.config` を編集し、サーバーのコマンド ラインを変更します。完了したら、サーバーを再起動します。詳細については、*SAP BusinessObjects Business Intelligence Platform* 管理者ガイドを参照してください。

6.1.3 処理拡張機能 API

このリファレンスでは、処理拡張機能の作成に使用可能な API 呼び出しを説明します。拡張機能の作成方法の詳細については、拡張機能の作成 [301 ページ]を参照してください。

処理拡張機能 API リファレンスには以下の API 呼び出しに関する情報が載っています。

API の内容	説明
エクスポート関数 [304 ページ]	拡張機能をロード、初期化、終了します。
IPROCESSEXTRequest インターフェイス [307 ページ]	ほかのサポートされるインターフェイスをすべて取得します。
IReportInfo インターフェイス [310 ページ]	処理中のレポートまたはサブレポートについての情報を取得します。読み取り専用。
IUserInfo インターフェイス [313 ページ]	レポートの処理を要求したユーザーについての情報を取得します。読み取り専用。
ITableSet インターフェイス [314 ページ]	特定のレポートで使用されるテーブル群についての情報を取得します。読み取り専用。
IPromptSet インターフェイス [320 ページ]	メインレポートとサブレポートに設定済みのプロンプト情報を変更します。

API の内容	説明
ISelectionInfoFormula インターフェイス [329 ページ]	現在のレコード選択式についての情報を取得し、選択基準を追加できるようにします。
ソ [333 ページ]	インターフェイスを取得する、最も簡単な方法です。

6.1.3.1 エクスポート関数

BI platform が拡張機能を適切にロードして初期化し、終了するためには、標準の C 呼び出し規約を使用して、以下の 3 つの関数をエクスポートする必要があります。これらは、次の順序で呼び出されます。

- IInitializePlugin 関数 [305 ページ]
- IProcessRequest 関数 [306 ページ]
- ITerminatePlugin 関数 [306 ページ]

> ⓘ 注記
>
> 使用プログラミング言語が C++ の場合は、これらの関数を .cpp ファイルのコード内に定義することが必要です。UNIX プラットフォームで作業を行っている場合は、さらに .def ファイルなどを使用してエクスポートする必要があります。例については、`simpleclient.def`（Windows の場合）または `simpleclient.def.unix`（UNIX の場合）を参照してください。これらのファイルは SAP BusinessObjects Business Intelligence platform CD の `¥samples¥processext¥simpleclient` ディレクトリにあります。

標準の呼び出し動作

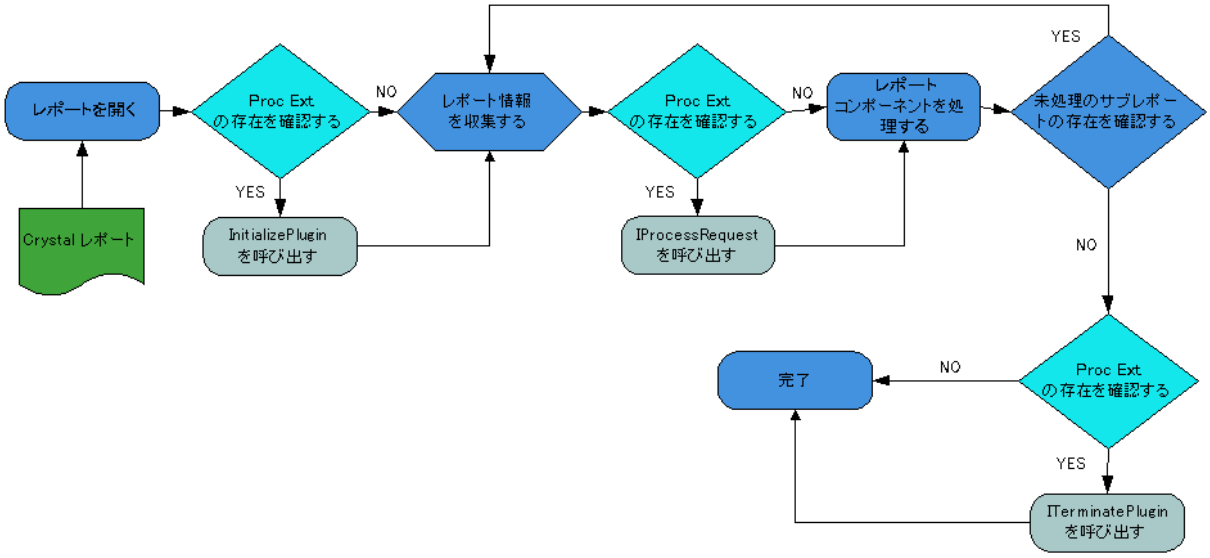
IInitializePlugin 関数（プラグインを初期化する）と ITerminatePlugin 関数（プラグインを解放する）は 1 つのリクエストにつき一度だけ呼び出され、IProcessRequest（レポートにアクセスし変更を行う）は、表示や処理の行われるサブレポートの数によって複数回呼び出される可能性があります。たとえば、ユーザーが処理拡張機能を使用するレポートを表示し、subreport1 と subreport2 の 2 つのレポートをドリルダウンする場合、呼び出しの順番は次のようになります。

- IInitializePlugin
- メインレポートの IProcessRequest
- subreport1 の IProcessRequest
- subreport2 の IProcessRequest
- ITerminatePlugin

> → ヒント
>
> 多数の別個のリクエストが処理拡張機能に同時に送信される場合があるため、DLL または共有ライブラリはマルチスレッドをサポートする必要があります。つまり、スレッドセーフで再入可能であることが必要です。処理拡張機能は複数のリクエストを処理しなくてはならない可能性があるため、それがいつまでメモリ上に留まるかはわかりません。

呼び出しフローの図解

処理拡張機能はリクエストごとにロードされます。次の図は、Job Server 上で処理されるスケジュールリクエストに対して処理拡張機能がどのように呼び出されるかを示します。



Crystal Reports Cache Server で処理される表示リクエストの場合も呼び出しの動作は同様です。IProcessRequest はイベント対応型で、ユーザーがレポートを表示したり、サブレポートをドリルダウンすると呼び出されます。

6.1.3.1.1 IInitializePlugin 関数

処理拡張機能がロードされると呼び出されます。

構文

```
bool STD_CALL IProcessRequest(
    void* ptr,
    ConstRequestHandle i_reqHandle)
```

パラメータ

ptr

エクスポート関数呼び出しの間で構造体へのハンドルを保持するのに使用します。この値は IProcessRequest と ITerminatePlugin に渡されます。

備考

この関数はリクエストごとに一度ずつ呼び出されます。void ポインタを使用して、3 つの関数の呼び出しの間で構造体の制御を維持することができます。これは、そのユーザーだけに対して使用されます。この時点でレポートへアクセスすることはできません。

関数呼び出し間で共有するメモリを作成して、ここに関数呼び出しを設定することができます。

6.1.3.1.2 IProcessRequest 関数

レポートの操作を行うために呼び出されます。そのためのハンドルが関数に渡されます。

構文

```
bool STD_CALL IProcessRequest(
    void* ptr,
    ConstRequestHandle i_reqHandle)
```

パラメータ

ptr

3 つのエクスポート関数間で値を渡すのに使用する void ポインタ。これは IInitializePlugin 関数 [305 ページ]で割り当てられた値です。

i_reqHandle

レポートを操作するために使用するハンドル。処理拡張機能 API [303 ページ] のすべての関数を使用するのにこのハンドルが必要です。

備考

この関数では、API を使用して任意のロジックでレポートの操作が可能です。たとえば、ユーザー、レポート、テーブルの情報にアクセスしたり、プロンプト値や選択式を変更したりするコードが追加できます。

6.1.3.1.3 ITerminatePlugin 関数

DLL または共有ライブラリのアンロード直前に呼び出されます。リクエストは、通常のリクエストとして実行、処理されます。

構文

```
bool STD_CALL ITerminatePlugin(
    void* ptr)
```

パラメータ

ptr

エクスポート関数呼び出しの間で構造体へのハンドルを保持するのに使用します。この時点で、この値は IInitializePlugin から IProcessRequest に渡されています。

備考

レポートへの変更はすべてこの時点で完了していなければなりません。ここでは単にクリーンアップを行います。最後に、ptr パラメータに対応したメモリ内のデータを解放します。

6.1.3.2 IPROCESSEXTRequest インターフェイス

このインターフェイスは、サポートされる他のすべてのインターフェイスを取得するのに使用します。これは、IProcessRequest 関数 [306 ページ]の i_reqHandle パラメータにポインタとしてユーザーに渡される構造体です。すべての関数に、IProcessRequest で受け取った元のハンドルを渡す必要があります。

このインターフェイスにより、処理中のリクエストのタイプに関する情報も取得できます。リクエストのタイプがわかると、スケジュールや表示を行うレポートにレコード選択式を適用する方法を決めるのに便利です。SetSelectionFormula 関数 [332 ページ]を参照してください。

変数と関数

変数と関数	説明
hIRequest 変数 [308 ページ]	インターフェイスへのハンドル。インターフェイスがサポートする各関数にこのハンドルを渡す必要があります。読み取り専用。
GetInterface 関数 [308 ページ]	この API リファレンスで説明されているインターフェイスの 1 つを取得します。
GetRequestType 関数 [309 ページ]	処理中のリクエストのタイプを取得します。

6.1.3.2.1 hIRequest 変数

インターフェイスへのハンドル。インターフェイスがサポートする各関数にこのハンドルを渡す必要があります。読み取り専用。

6.1.3.2.2 GetInterface 関数

この API リファレンスで説明されているインターフェイスの 1 つを取得します。残りの API にアクセスするには、いずれかのソ [333 ページ]（推奨）またはこの関数を使用します。

構文

```
GetInterface(
    IPROCESSEXTREQ_HANDLE i_hIRequest,
    INTERFACE_ID i_id,
    void** o_pIRequest)
```

パラメータ

i_hIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

i_id [In]

要求しているインターフェイスの ID。この ID はヘッダーファイルに定義されています。

o_pIRequest [Out]

要求したインターフェイスを返します。まず、必要とするインターフェイス型へのポインタを作成する必要があります。この関数を呼び出した後、このポインタは実際に要求したインターフェイスを指します。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

備考

インターフェイスを取得するには、ソ [333 ページ]のいずれかを使用できます。これはインターフェイスを取得する、最も簡単な方法です。たとえば、次のコードは GET_USERINFO_INTERFACE マクロを使用して、UserInfo インターフェイスを取得します。

```
IUserInfo* pUserInfoInterface = NULL;
GET_USERINFO_INTERFACE(i_reqHandle, pUserInfoInterface)
    unsigned short userNameSize=2;
    Char* userName = new Char[userNameSize];
    ret = pUserInfoInterface->GetUserName(pUserInfoInterface->hIUserInfo,
&userNameSize, userName) ;
```

GetInterface 関数を使用して、UserInfo インターフェイスを取得することもできます。

```
IUserInfo* pUserInfoInterface = NULL;
i_reqHandle->GetInterface(i_reqHandle, pUserInfoInterface);
     unsigned short userNameSize=2;
    Char* userName = new Char[userNameSize];
    ret = pUserInfoInterface->GetUserName(pUserInfoInterface->hIUserInfo,
&userNameSize, userName) ;
```

6.1.3.2.3 GetRequestType 関数

処理中のリクエストのタイプを取得します。

構文

```
GetRequestType(
    IPROCESSEXTREQ_HANDLE i_hIRequest,
    RequestType** o_requestType)
```

パラメータ

i_hIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

o_requestType [Out]

リクエストのタイプ。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

備考

返されるリクエストのタイプは、次のいずれかの値です。

定数	値
view	0
schedule	1
both	2
none	3

6.1.3.3 IReportInfo インターフェイス

これは読み取り専用インターフェイスで、処理中のレポートまたはサブレポートの情報を取得します。このインターフェイスを取得するには、GetInterface 関数 [308 ページ]または GET_REPORTINFO_INTERFACE マクロ [334 ページ]を使用します。

サブレポートはそれぞれメインレポートから呼び出されて処理されるため、処理拡張機能は同じレポートオブジェクト ID の呼び出しを複数受ける場合があります。ただし、レポートコンテキストは呼び出しごとに変わることがあります。たとえば、ユーザーがメインレポートを表示している場合は、メインレポートのレポートコンテキストで IProcessRequest が呼び出されます。レポート ID、タイトル、テーブルへのリクエストに対してはすべてメインレポートの情報を返します。ユーザーがサブレポートをドリルダウンした場合は、サブレポートのコンテキストで IProcessRequest が呼び出され、サブレポートの情報が返されます。処理中のレポートがメインレポートなのかサブレポートなのかを判別するには、GetReportTitle 関数 [311 ページ]を使用します。

変数と関数

変数と関数	説明
hIReportInfo 変数 [311 ページ]	インターフェイスへのハンドル。インターフェイスがサポートする各関数にこのハンドルを渡す必要があります。読み取り専用。

変数と関数	説明
GetReportTitle 関数 [311 ページ]	レポートまたはサブレポートのタイトル（または名前）を取得します。
GetReportID 関数 [312 ページ]	現在処理中のレポートの ID を取得します。

6.1.3.3.1 hIReportInfo 変数

インターフェイスへのハンドル。インターフェイスがサポートする各関数にこのハンドルを渡す必要があります。読み取り専用。

6.1.3.3.2 GetReportTitle 関数

レポートまたはサブレポートのタイトル（または名前）を取得します。

構文

```
GetReportID(
    IREPORTINFO_HANDLE i_hIReportInfo,
    long* o_reportId)
```

パラメータ

i_hIReportInfo [In]

インターフェイスのハンドル。hIReportInfo 変数 [311 ページ]を参照してください。

io_reportTitleSize [In, Out]

バッファのサイズ。

o_reportTitle

レポートまたはサブレポートのタイトル。o_isSubReport が False ならばメインレポートのタイトルが、o_isSubReport が True ならばサブレポートのタイトルが取得されます。

o_isSubReport

処理中のレポートがサブレポートの場合は True を返し、メインレポートの場合は False を返します。レポートがサブレポートかメインレポートかによって、GetOldSelectionFormula 関数 [331 ページ]が取得する選択式も決まります。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

6.1.3.3.3 GetReportID 関数

現在処理中のレポートの ID を取得します。

構文

```
GetReportID(
    IREPORTINFO_HANDLE i_hIReportInfo,
    long* o_reportId)
```

パラメータ

i_hIReportInfo [In]

インターフェイスのハンドル。hIReportInfo 変数 [311 ページ]を参照してください。

o_reportId [Out]

レポートまたはサブレポートオブジェクトの ID。

戻り値

成功した場合はゼロより大きく、失敗した場合はゼロ以下。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

備考

この関数はレポートまたはサブレポートのオブジェクト ID を取得します。割り込まれたリクエストが表示リクエストの場合、表示中のレポートまたはインスタンスの ID を取得します。割り込まれたリクエストがスケジュー

ルリクエストの場合、GetReportID 関数はスケジュールされたオブジェクトの ID を返します。これは、実際は生成されたインスタンスの親です。たとえば、ReportA に基づくインスタンスをスケジュールする場合、GetReportID は ReportA のオブジェクト ID を返します。InstanceB に基づくインスタンスをスケジュールする場合、GetReportID は InstanceB のオブジェクト ID を返します。

6.1.3.4 IUserInfo インターフェイス

これは読み取り専用インターフェイスで、レポートの処理を要求したユーザーについての情報を取得します。このインターフェイスを取得するには、GetInterface 関数 [308 ページ]またはGET_USERINFO_INTERFACE マクロ [334 ページ]を使用します。

変数と関数

変数と関数	説明
hIUserInfo 変数 [313 ページ]	IUserInfo インターフェイスへのハンドル。IUserInfo がサポートする関数すべてにこのハンドルを渡す必要があります。読み取り専用。
GetUserName 関数 [313 ページ]	リクエストの処理を要求したユーザーの名前を取得します。

6.1.3.4.1 hIUserInfo 変数

IUserInfo インターフェイスへのハンドル。IUserInfo がサポートする関数すべてにこのハンドルを渡す必要があります。読み取り専用。

6.1.3.4.2 GetUserName 関数

リクエストの処理を要求したユーザーの名前を取得します。

構文

```
GetUserName(
    IUSERINFO_HANDLE i_hIUserInfo ,
    unsigned short* io_userNameSize,
    Char* o_userName)
```

パラメータ

i_hIUserInfo [In]

インターフェイスへのハンドル。hIUserInfo 変数 [313 ページ]を参照してください。

io_userNameSize [In, Out]

バッファのサイズ。

o_userName [Out]

ユーザーの名前。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

備考

表示リクエスト中、GetUsername はシステムにログオンしてレポートを表示しているユーザーの名前を取得します。スケジュールリクエスト中、この関数は、送信者やレポートのオーナーではなく、スケジュール中のインスタンスのオーナーの名前を取得します。

6.1.3.5 ITableSet インターフェイス

これは読み取り専用インターフェイスで、特定のレポートで使用されるテーブル群についての情報を取得します。各テーブルの情報（たとえば、テーブル名とエイリアス名）は配列として保存され、その配列はテーブルセットと見なされます。このインターフェイスを取得するには、GetInterface 関数 [308 ページ]または GET_TABLESET_INTERFACE マクロ [335 ページ]を使用します。

サブレポートはそれぞれメインレポートから呼び出されて処理されるため、処理拡張機能は同じレポートオブジェクト ID の呼び出しを複数受ける場合があります。ただし、レポートコンテキストは呼び出しごとに変わることがあります。たとえば、ユーザーがメインレポートを表示している場合は、メインレポートのレポートコンテキストで IProcessRequest が呼び出されます。レポート ID、タイトル、テーブルへのリクエストに対してはすべてメインレポートの情報を返します。ユーザーがサブレポートをドリルダウンした場合は、サブレポートのコンテキストで IProcessRequest が呼び出され、サブレポートの情報が返されます。処理中のレポートがメインレポートなのかサブレポートなのかを判別するには、GetReportTitle 関数 [311 ページ]を使用します。

変数と関数

変数と関数	説明
hITableSet 変数 [315 ページ]	ITableSet インターフェイスへのハンドル。使用するインターフェイスの各関数にこのハンドルを渡す必要があります。読み取り専用。
NumOfTables 関数 [315 ページ]	処理中のレポートが現在使用しているテーブルの個数を取得します。
GetTableName 関数 [316 ページ]	処理中のレポートで現在使用しているテーブルの名前を取得します。
GetAliasName 関数 [317 ページ]	処理中のレポートで現在使用しているテーブルのエイリアス名を取得します。
SearchByName 関数 [318 ページ]	レポート内のテーブルの検索にテーブル名を使用します。
SearchByAlias 関数 [319 ページ]	レポート内のテーブルの検索にエイリアス名を使用します。

6.1.3.5.1 hITableSet 変数

ITableSet インターフェイスへのハンドル。使用するインターフェイスの各関数にこのハンドルを渡す必要があります。読み取り専用。

6.1.3.5.2 NumOfTables 関数

処理中のレポートが現在使用しているテーブルの個数を取得します。

構文

```
NumOfTables(
    ITABLESET_HANDLE i_hITableSet,
    int* o_iIndex)
```

パラメータ

i_hITableSet [In]

ITableSet インターフェイスへのハンドル。hITableSet 変数 [315 ページ]を参照してください。

o_iIndex [Out]

テーブルセットの配列のサイズ。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

6.1.3.5.3 GetTableName 関数

処理中のレポートで現在使用しているテーブルの名前を取得します。

構文

```
GetTableName(
    ITABLESET_HANDLE i_hITableSet,
    int i_iIndex,
    unsigned short* io_tableNameSize,
    Char* o_tableName)
```

パラメータ

i_hITableSet [In]

ITableSet インターフェイスへのハンドル。hITableSet 変数 [315 ページ]を参照してください。

i_iIndex [In]

テーブルのインデックス番号。

io_tableNameSize [In, Out]

バッファのサイズ。

o_tableName [Out]

テーブルの名前。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

6.1.3.5.4 GetAliasName 関数

処理中のレポートで現在使用しているテーブルのエイリアス名を取得します。

構文

```
GetAliasName(
    ITABLESET_HANDLE i_hITableSet,
    int i_iIndex,
    unsigned short* io_aliasNameSize,
    Char* o_aliasName)
```

パラメータ

i_hITableSet [In]

ITableSet インターフェイスへのハンドル。hITableSet 変数 [315 ページ]を参照してください。

i_iIndex [In]

テーブルのインデックス番号。

io_aliasNameSize [In, Out]

バッファのサイズ。

o_aliasName [Out]

エイリアスの名前。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

備考

エイリアスは、テーブルに固有の識別子として働く文字列です。データベースの名前や保存場所を変更しても、エイリアスの名前は変わりません。エイリアス名は、選択式が常に適切に実行されるようにするのに特に便利です。たとえば、式中で使用しているテーブルの名前が変更または移動された場合、エイリアスを使用して、テーブルの新しい名前や位置を検索することができます。

6.1.3.5.5 SearchByName 関数

レポート内のテーブルの検索にテーブル名を使用します。

構文

```
SearchByName(
    ITABLESET_HANDLE i_hITableSet,
    ConstChar* i_tableName,
    bool* o_bIsExists,
    int* o_iIndex)
```

パラメータ

i_hITableSet [In]

ITableSet インターフェイスへのハンドル。hITableSet 変数 [315 ページ]を参照してください。

i_tableName [In]

テーブルの名前。

o_bIsExists [Out]

テーブルが存在すれば True を、存在しなければ False を返します。

o_iIndex [Out]

テーブルのインデックス番号。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

6.1.3.5.6 SearchByAlias 関数

レポート内のテーブルの検索にエイリアス名を使用します。

構文

```
SearchByAlias(
    ITABLESET_HANDLE i_hITableSet,
    ConstChar* i_aliasName,
    bool* o_bIsExists,
    int* o_iIndex)
```

パラメータ

>

i_hITableSet [In]

ITableSet インターフェイスへのハンドル。hITableSet 変数 [315 ページ]を参照してください。

i_aliasName [In]

テーブルのエイリアス名。

o_bIsExists [Out]

テーブルが存在すれば True を、存在しなければ False を返します。

o_iIndex [Out]

テーブルのインデックス番号。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

6.1.3.6 IPromptSet インターフェイス

このインターフェイスを使うと、メインレポートとそのサブレポートに設定済みのプロンプト情報を変更することができます。各プロンプトに関する情報 (たとえば、プロンプト名と値一式) はサーバー上に配列として保存され、その配列はプロンプトセットと見なされます。取得されたプロンプトセットはすべてのレポートコンテキストの和で、メインレポートと各サブレポートのプロンプトを含みます。

プロンプトセットは次のように図式化できます。

<table>
<tr><td rowspan="6">PromptNum 1</td><td rowspan="2">ValueIndex 1</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="2">ValueIndex 2</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="2">ValueIndex 3</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="4">PromptNum 2</td><td rowspan="2">ValueIndex 1</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="2">ValueIndex 2</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
</table>

レポートは複数のプロンプトを持つことができます。各プロンプトは 1 つまたは複数の値を持つことができ、値は離散値か範囲値にすることができます。離散値は単一の値で、PromptValue 1 で表され、範囲値は 2 つの値（範囲の開始と範囲の終了）を持ち、それぞれ PromptValue 1 と PromptValue 2 で表されます。

一般的には、プロンプトの総数を調べ、それぞれの情報を取得し、値を変更します。このインターフェイスを取得するには、GetInterface 関数 [308 ページ]または GET_PROMPTSET_INTERFACE マクロ [335 ページ]を使用します。

- このインターフェイスの関数は、必要のない値を持つプロンプトは無視します。
- 存在するプロンプト値のみを変更することができます。値は NULL には変更できません。また NULL を値に変更することもできません。

変数と関数

変数と関数	説明
hIPromptSet 変数 [321 ページ]	IPromptSet インターフェイスへのハンドル。使用するインターフェイスの各関数にこのハンドルを渡す必要があります。読み取り専用。
NumOfPrompts 関数 [321 ページ]	処理中のレポートで現在使用しているプロンプトの個数を取得します。
GetName 関数 [322 ページ]	処理中のレポートで現在使用しているプロンプトの名前を取得します。
GetSubReportName 関数 [323 ページ]	特定のプロンプトが使用されているサブレポートの名前を取得します。
GetType 関数 [324 ページ]	特定のプロンプトのタイプを取得します。
NumOfPromptValues 関数 [325 ページ]	特定のプロンプトに使用可能な値の個数を取得します。
GetPromptValue 関数 [326 ページ]	特定のプロンプトの値を取得します。
SetPromptValue 関数 [328 ページ]	特定のプロンプト値の 1 つを上書きします。

6.1.3.6.1 hIPromptSet 変数

IPromptSet インターフェイスへのハンドル。使用するインターフェイスの各関数にこのハンドルを渡す必要があります。読み取り専用。

6.1.3.6.2 NumOfPrompts 関数

処理中のレポートで現在使用しているプロンプトの個数を取得します。

構文

```
NumOfPrompts(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int* o_iIndex)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

o_iIndex [Out]

プロンプトセットの配列のサイズ。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

備考

この関数は、メインレポートとサブレポート間で共有されるプロンプトの個数を取得します。ほかのインターフェイスと違い、IPromptSet インターフェイスはレポートとサブレポートを区別しないため、プロンプト関連の関数を呼び出す際は、GetReportTitle 関数 [311 ページ]の o_isSubReport パラメータは常に False です。

6.1.3.6.3 GetName 関数

処理中のレポートで現在使用しているプロンプトの名前を取得します。

構文

```
GetName(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iIndex,
    unsigned short* io_nameSize,
    Char* o_name)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

i_iIndex [In]

プロンプトのインデックス 番号。

io_nameSize [In, Out]

バッファのサイズ。

o_name [Out]

プロンプトの名前。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

6.1.3.6.4 GetSubReportName 関数

特定のプロンプトが使用されているサブレポートの名前を取得します。

構文

```
GetName(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iIndex,
    unsigned short* io_nameSize,
    Char* o_name)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

i_iIndex [In]

プロンプトのインデックス 番号。

io_subreportSize [In, Out]

バッファのサイズ。

o_subreportName [Out]

サブレポートの名前。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

6.1.3.6.5 GetType 関数

特定のプロンプトのタイプを取得します。

構文

```
GetName(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iIndex,
    unsigned short* io_nameSize,
    Char* o_name)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

i_iIndex [In]

プロンプトのインデックス 番号。

o_type [Out]

プロンプトのタイプ。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

備考

プロンプトのタイプとして、次のいずれかの値を指定できます。

定数	値
PROMPTTYPE_INVALID	0
PROMPTTYPE_NUMBER	1
PROMPTTYPE_CURRENCY	2
PROMPTTYPE_BOOL	3
PROMPTTYPE_DATE	4
PROMPTTYPE_TIME	5
PROMPTTYPE_DATETIME	6
PROMPTTYPE_STRING	7

6.1.3.6.6 NumOfPromptValues 関数

特定のプロンプトに使用可能な値の個数を取得します。

構文

```
NumOfPromptValues(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iPromptNum,
    int* o_iValueSize)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

i_iPromptNum [In]

プロンプトのインデックス 番号。

o_iValueSize [Out]

プロンプト値の個数。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

6.1.3.6.7 GetPromptValue 関数

特定のプロンプトの値を取得します。

構文

```
GetPromptValue(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iPromptNum,
    int i_iValueIndex,
    PromptValuesType* o_valueType,
    unsigned short* io_promptValueSize1,
    Char* o_promptValue1,
    unsigned short* io_promptValueSize2,
    Char* o_promptValue2)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

i_iPromptNum [In]

プロンプトのインデックス 番号。

i_iValueIndex [In]

プロンプト値のインデックス番号。

o_valueType [Out]

プロンプト値のタイプ。

io_promptValueSize1 [In, Out]

最初のプロンプト値のバッファサイズ。

o_promptValue1 [Out]

最初のプロンプト値。

io_promptValueSize2 [In, Out]

2 番目のプロンプト値のバッファサイズ。

o_promptValue2 [Out]

2 番目のプロンプト値。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

備考

プロンプト値のタイプは、次のいずれかの値になります。

定数	値	説明
pvalue_single	0	離散値。
pvalue_range	1	範囲値。

プロンプトのタイプが範囲の場合、両方の値に十分なスペースが与えられていないと、GetPromptValue を 2 度呼び出す必要があります。1 度は io_promptValueSize1 のより大きな値の追加、そして io_promptValueSize2 のより大きな値の追加にもう 1 度です。

6.1.3.6.8 SetPromptValue 関数

特定のプロンプト値の 1 つを上書きします。

構文

```
SetPromptValue(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iPromptNum,
    int i_iValueIndex,
    PromptValuesType i_valueType,
    ConstChar* i_promptValueInfo1,
    ConstChar* i_promptValueInfo2)
```

パラメータ

i_hIPromptSet [In]

IPromptSet インターフェイスへのハンドル。hIPromptSet 変数 [321 ページ]を参照してください。

i_iPromptNum [In]

プロンプトのインデックス 番号。

i_iValueIndex [In]

プロンプト値のインデックス番号。

i_valueType [In]

プロンプト値のタイプ。これは、現在のプロンプト値のタイプと同じである必要があります。変更はできません。

i_promptValueInfo1 [In]

最初のバッファに設定するプロンプト値。

i_promptValueInfo2 [In]

2 番目のバッファに設定するプロンプト値。GetPromptValue 関数 [326 ページ]を参照してください。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

備考

プロンプト値のタイプは、次のいずれかの値になります。

定数	値	説明
pvalue_single	0	離散値。
pvalue_range	1	範囲値。

プロンプト値のタイプが離散値ならば、パラメータとして必要なのは 1 個のプロンプト値です。プロンプト値のタイプが範囲値ならば、2 個のプロンプト値が必要です。

> ⓘ 注記
>
> インスタンスのプロンプトは、そのインスタンスをリフレッシュしないかぎり、変更できません。プロンプト値を変更する場合、レポートをスケジュールする前に処理拡張機能を適用します。

6.1.3.7 ISelectionInfoFormula インターフェイス

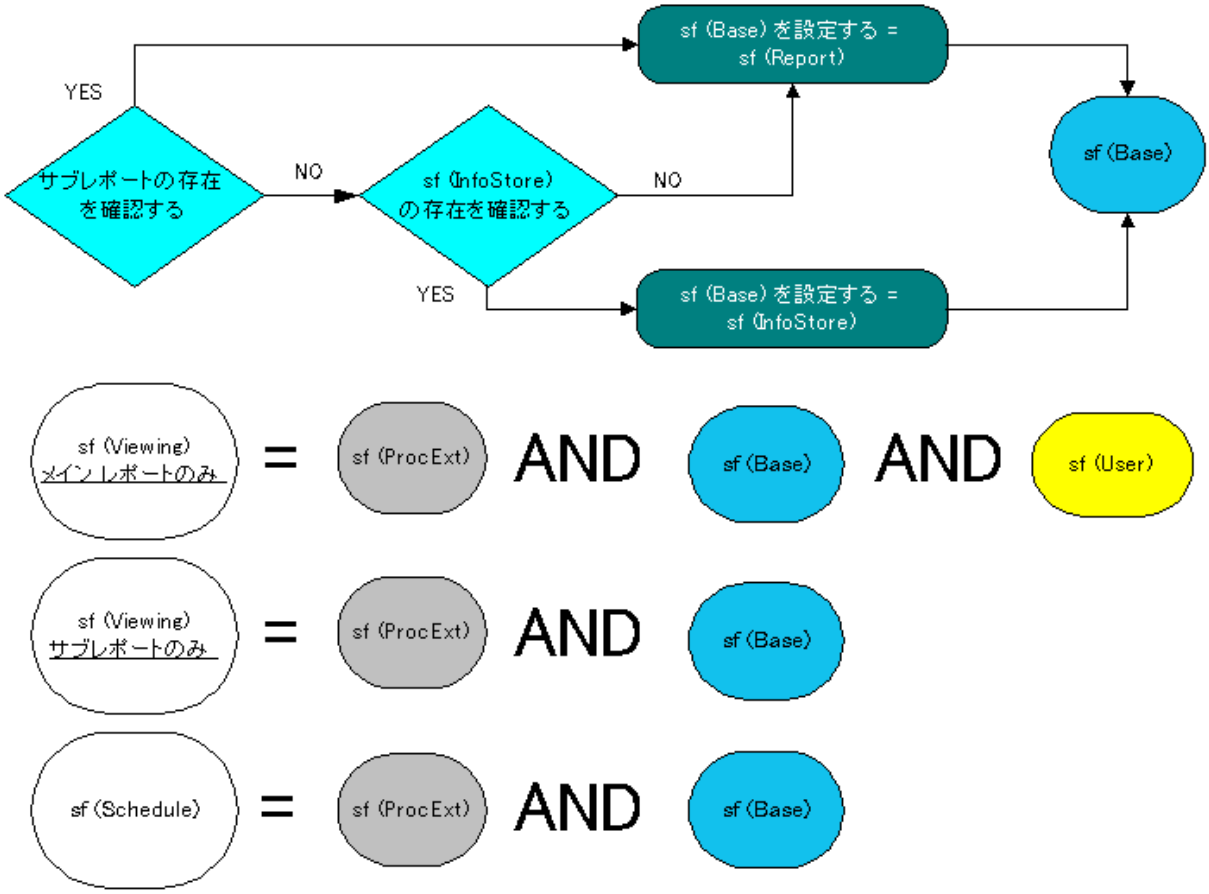
現在のレコード選択式についての情報を取得し、選択基準を追加できるようにします。このインターフェイスを取得するには、GetInterface 関数 [308 ページ]または GET_SELECTIONFORMULA_INTERFACE マクロ [336 ページ]を使用します。

> ⓘ 注記
>
> このインターフェイスで変更できるのは、レコード選択式だけです。グループ選択式には適用されません。

表示またはスケジュールされるのがレポートかサブレポートかによって、現在の選択式は次の方法で設定されたフィルタ文字列の組み合わせとなります。

- 既存の Crystal レポートに保存された文字列
- ユーザーがクライアントアプリケーションで指定した文字列
- セントラル管理コンソール（CMC）で管理者が指定した文字列

処理拡張機能を使用して現在の選択式を拡張できます。次の図のように、既存の選択式と処理拡張機能が設定する式との論理積（AND）が取られます。



sf (Viewing):

Crystal Reports Cache Server を通じて表示リクエストが行われると生成される選択式。sf（User）選択式は、メインレポートの表示の際にのみ適用され、サブレポートの表示では適用されません。

sf (Schedule):

スケジュールリクエストが Job Server 上に渡され実行されると生成される選択式。

sf (InfoStore):

管理者は、CMS に格納される選択式にセントラル管理コンソール（CMC）からアクセスできます。これはメインレポートの選択式で、選択式はサブレポートには保存されません。レポートに選択式が含まれる場合、レポートが CMS に追加される際に、この式がレポートから CMS にコピーされます。

sf (Report):

既存のレポートまたはサブレポートから取り出された選択式。処理拡張機能はメインレポートと各サブレポートに対して呼び出されるため、選択式は現在のレポートコンテキストによって変化します。

sf (User):

ユーザーのクライアントアプリケーションで指定された選択式。たとえば、"&sf=<選択式>"のように、URL の一部としてクライアントからサーバーに渡すことができます。

sf (ProcExt):

レポートに割り当てられたユーザー定義の処理拡張機能で設定された選択式。レポートに複数の処理拡張機能が割り当てられている場合は、各機能の選択式が現在のレポートコンテキストに同時に追加されます。

> ⓘ 注記
>
> 拡張処理機能がレポートのインスタンスに含まれるパラメータに影響しなくても、選択式には影響します。

変数と関数

変数と関数	説明
hISelectionFormula 変数 [331 ページ]	ISelectionInfoFormula インターフェイスへのハンドル。使用するインターフェイスの各関数にこのハンドルを渡す必要があります。読み取り専用。
GetOldSelectionFormula 関数 [331 ページ]	処理中のレポートが現在使用している選択式を取得します。
SetSelectionFormula 関数 [332 ページ]	レポートのレコード選択式を設定します。

6.1.3.7.1 hISelectionFormula 変数

ISelectionInfoFormula インターフェイスへのハンドル。使用するインターフェイスの各関数にこのハンドルを渡す必要があります。読み取り専用。

6.1.3.7.2 GetOldSelectionFormula 関数

処理中のレポートが現在使用している選択式を取得します。

構文

```
GetOldSelectionFormula(
    ISELECTIONFORMULA_HANDLE i_hISelectionFormula,
    int* io_selectionFormulaSize,
    Char* o_selectionFormula)
```

パラメータ

i_hISelectionFormula [In]

ISelectionInfoFormula インターフェイスへのハンドル。hISelectionFormula 変数 [331 ページ]を参照してください。

io_selectionFormulaSize [In, Out]

バッファのサイズ。サイズは、ユーザー（レポートがオンデマンド表示されている場合）またはセントラル管理コンソール（CMC）がレポートに設定した選択式の合計です。

o_selectionFormula [Out]

レポートまたはサブレポートで現在使用しているレコード選択式。現在使用されているレコード選択式は、"old"な選択式とみなされます。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

備考

GetOldSelectionFormula 関数で何が返されるかは、レポートがメインレポートかサブレポートかによって決まります。o_isSubReport パラメータ (GetReportTitle 関数 [311 ページ]を参照) が False なら、関数はメインレポートに設定済みの選択式を取得します。o_isSubReport が True ならば、サブレポートの設定済み選択式を取得します。

新規の選択式は SetSelectionFormula 関数 [332 ページ]で指定できます。この関数は新しいフィルタ文字列を設定済みの選択式に追加します。

6.1.3.7.3 SetSelectionFormula 関数

レポートのレコード選択式を設定します。

構文

```
SetSelectionFormula(
    ISELECTIONFORMULA_HANDLE i_hISelectionFormula,
    ConstChar* i_selectionFormula)
```

パラメータ

i_hISelectionFormula [In]

ISelectionInfoFormula インターフェイスへのハンドル。hISelectionFormula 変数 [331 ページ]を参照してください。

i_selectionFormula [In]

このフィルタ文字列は現在の選択式に"追加"されます。サーバーは i_selectionFormula で渡された新しいフィルタ文字列と古い選択式を連結します。そして、この変更された選択式がレポートに送られます。

戻り値

各関数は、Long 型のエラーコードの RET_VALUE を返します。戻り値の値が 0 以上の場合は呼び出しが成功、0 またはそれ以下の場合は失敗したことを示します。失敗の有無をチェックするには、RET_OK(x) および RET_ERROR(x) を使用します。

関数呼び出しが失敗した場合は、バッファサイズパラメータの値をチェックしてください。関数呼び出しが成功すると、このパラメータには、呼び出し時に渡したバッファのサイズが入ります。しかし、バッファサイズが小さすぎると、関数は PROCESSEXTERR_BUFFERSIZETOOSMALL を返し、必要なバッファサイズがパラメータに指定されます。

備考

レコード選択式は、すべてのリクエストに対して適用したり、レポートのスケジュール時または表示時のみに適用したりできます。行われたリクエストのタイプを調べるには、GetRequestType 関数 [309 ページ]を使用します。

行レベルセキュリティを常に確保するためには、処理拡張機能を表示リクエストとスケジュールリクエストの両方に適用することが必要です。たとえば、社員名簿のレポートで sf = {Employee.Last Name}<"M"という選択式を使用して、レポートのスケジュール時に処理拡張機能を適用すると、返されるデータは M までの社員の姓（M は含まない）になります。しかし、表示時に処理拡張機能が適用されなければ、レポートが最新表示される際、データベースからすべてのデータ（A から Z までの姓）が取得されることになります。つまり、処理拡張機能を使用してスケジュールされたレポートをリフレッシュして表示する際に同じアクセス権限を適用するには、スケジュール時だけでなく表示時にも同じ処理拡張機能を適用する必要があります。

6.1.3.8 ソ

特定のインターフェイスを取得するには、IPROCESSEXTRequest インターフェイスの GetInterface 関数 [308 ページ]かマクロを使用します。マクロの使用はインターフェイスを取得する、最も簡単な方法です。この 2 つの方法の例については、GetInterface 関数 [308 ページ]を参照してください。

マクロ	説明
GET_REPORTINFO_INTERFACE マクロ [334 ページ]	IReportInfo インターフェイス [310 ページ]を取得します。
GET_USERINFO_INTERFACE マクロ [334 ページ]	IUserInfo インターフェイス [313 ページ]を取得します。
GET_TABLESET_INTERFACE マクロ [335 ページ]	IUserInfo インターフェイス [313 ページ]を取得します。
GET_PROMPTSET_INTERFACE マクロ [335 ページ]	IPromptSet インターフェイス [320 ページ]を取得します。
GET_SELECTIONFORMULA_INTERFACE マクロ [336 ページ]	ISelectionInfoFormula インターフェイス [329 ページ]を取得します。

6.1.3.8.1 GET_REPORTINFO_INTERFACE マクロ

IReportInfo インターフェイス [310 ページ]を取得します。

構文

```
GET_REPORTINFO_INTERFACE(
    i_pIRequest,
    o_pIReportInfo);
```

パラメータ

i_pIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

o_pIReportInfo [Out]

IReportInfo インターフェイス [310 ページ]へのポインタ。

6.1.3.8.2 GET_USERINFO_INTERFACE マクロ

IUserInfo インターフェイス [313 ページ]を取得します。

構文

```
GET_USERINFO_INTERFACE(
    i_pIRequest,
    o_pIUserInfo);
```

パラメータ

i_pIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

o_pIUserInfo [Out]

IUserInfo インターフェイス [313 ページ]へのポインタ。

6.1.3.8.3 GET_TABLESET_INTERFACE マクロ

ITableSet インターフェイス [314 ページ]を取得します。

構文

```
GET_TABLESET_INTERFACE(
    i_pIRequest,
    o_pITableSet);
```

パラメータ

i_pIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

o_pITableSet [Out]

ITableSet インターフェイス [314 ページ]へのポインタ。

6.1.3.8.4 GET_PROMPTSET_INTERFACE マクロ

IPromptSet インターフェイス [320 ページ]を取得します。

構文

```
GET_PROMPTSET_INTERFACE(
    i_pIRequest,
    o_pIPromptSet);
```

パラメータ

i_pIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

o_pIPrompt [Out]

IPromptSet インターフェイス [320 ページ]へのポインタ。

6.1.3.8.5 GET_SELECTIONFORMULA_INTERFACE マクロ

ISelectionInfoFormula インターフェイス [329 ページ]を取得します。

構文

```
GET_SELECTIONFORMULA_INTERFACE(
    i_pIRequest,
    o_pIUserInfo);
```

パラメータ

i_pIRequest [In]

IProcessRequest で受け取ったハンドル。hIRequest 変数 [308 ページ]を参照してください。

o_pIUserInfo [Out]

ISelectionInfoFormula インターフェイス [329 ページ]へのポインタ。

6.1.4 例

次の例で、処理拡張機能 API のさまざまな側面がわかります。

詳細については、拡張機能の作成 [301 ページ]を参照してください。

6.1.4.1 リクエストタイプの取得

```
    // Request Demo
    // Accesses the request handle and retrieves the request type.
    std::cout<<"-------------- Request Demo --------------¥n";

    if(! i_reqHandle->GetRequestType(i_reqHandle,&rtype))
    {
        std::cout<<"Error getting RequestType!¥n";
        return false;
    }
    else
        std::cout<<"Client: request type="<<rtype<<std::endl;
```

6.1.4.2 レポートタイトルと ID の取得

```
// Report Info Demo
// Retrieve the report title and report ID. (The ID that identifies the report
inside CMS)
std::cout<<"-------------- Report Info Demo --------------¥n";
IReportInfo* pReportInfoInterface = NULL;
if( RET_OK( GET_REPORTINFO_INTERFACE(i_reqHandle, pReportInfoInterface) ) )
{
    // Try to get the report info interface.
    if( pReportInfoInterface == NULL )
    {
        std::cout<<"Error: no ReportInfo interface!¥n";
        return false;
    }

    // Retrieve the report ID.
    long reportid=0;
    if(! RET_OK(pReportInfoInterface->GetReportID(pReportInfoInterface-
>hIReportInfo,&reportid)  ))
    {
        std::cout<<"Error getting the Report ID!¥n";
        return false;
    }
    else
        std::cout<<"Client: report id="<<reportid<<std::endl;

    // Retrieve the report title.

    // Because the size of the report title is unknown, try 2 first.
    unsigned short ReportTitleSize=2;
    Char* ReportTitle = new Char[ReportTitleSize];
    // The following is a Boolean flag to indicate whether the report interface
    // is that of a subreport.
    bool* isSubreport = false;
    ret = pReportInfoInterface->GetReportTitle(pReportInfoInterface-
>hIReportInfo, &ReportTitleSize,ReportTitle, isSubreport) ;
    // If 2 is too small, get the real size and call the GetReportTitle function
again.
    if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
    {
```

```
        std::cout<<"ReportTitle size too small, should be:"
<<ReportTitleSize<<"¥n";
        // Re-create the variable with the actual size.
        delete ReportTitle;
        ReportTitle = new Char[ReportTitleSize];
        // Call the function again.
           if(! RET_OK( pReportInfoInterface-
>GetReportTitle(pReportInfoInterface->hIReportInfo,
&ReportTitleSize,ReportTitle, isSubreport) ))
           {
            std::cout<<"Error getting ReportTitle!"<<ret<<"¥n";
           }
        else
           {
            char* myReportTitle = new char[ReportTitleSize];
                 std::cout<<"Client: ReportTitle ="<<w2ac(ReportTitle,
ReportTitleSize, myReportTitle)<<std::endl;

                // An example using the reportTitle:
                // Compare the reportTitle to the title hard-coded in the
tochangeSF
                // function. If a match is found, then change the selection
formula
                // later in the program (see the selection formula demo, below).
                if( tochangeSF(myReportTitle) )
                     changeSF=true;
                delete myReportTitle;
            }
        }
        // If 2 chars is large enough, check if the funtion returns correctly.
        // If it did not return correctly...
        else if (!RET_OK(ret))
        {
            std::cout<<"ERROR at getting ReportTitle!"<<ret<<"¥n";
            return false;
        }
        // Otherwise, if it returned correctly...
        else
        {
            char* myReportTitle = new char[ReportTitleSize];
            std::cout<<"Client: ReportTitle ="<<w2ac(ReportTitle,
ReportTitleSize, myReportTitle)<<std::endl;
            if ( tochangeSF(myReportTitle) )
             changeSF=true;
            delete myReportTitle;
        }
        // Clean up.
        delete ReportTitle;
    }
    // If the reportinfo interface could not be retrieved...
    else
    {
        std::cout<<"Error: get reportinfo interface return false!¥n";
        return false;
    }
```

6.1.4.3 ユーザー名の取得

```
// UserInfo Demo
// Retrieves the user's name.
std::cout<<"-------------- UserInfo Demo --------------¥n";
IUserInfo* pUserInfoInterface = NULL;
if( RET_OK( GET_USERINFO_INTERFACE(i_reqHandle, pUserInfoInterface) ) )
```

```
{
    // Test for the UserInfo interface.
    if( pUserInfoInterface == NULL )
    {
        std::cout<<"Error: no userInfo interface!¥n";
        return false;
    }
    if( pUserInfoInterface != NULL )
    {
        // The interface is available, so retrieve the user name.
        // The length of the user name is unknown, so try a length of 2 first.
        unsigned short userNameSize=2;
        // A variable for the user name (2 chars in length).
        Char* userName = new Char[userNameSize];
        // Try to retrieve the user name.
        ret = pUserInfoInterface->GetUserName(pUserInfoInterface-
>hIUserInfo,&userNameSize,userName) ;
        // 2 is most likely too short, but the actual size of the user name
        // is returned by the function, so now the real size can be used.
        if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
        {
            std::cout<<"UserName size too small, should be:"
<<userNameSize<<"¥n";
            // Re-create the variable.
            delete userName;
            userName = new Char[userNameSize];
            // Retrieve the user name.
            if(! RET_OK( pUserInfoInterface->GetUserName(pUserInfoInterface-
>hIUserInfo,&userNameSize,userName) ))
            {
                std::cout<<"Error getting user name!"<<ret<<"¥n";
            }
            else
            {
                // A character buffer for the w2ac function. (See note in
function declaration, below.)
                char* myuserName = new char[userNameSize];
                std::cout<<"Client: userName ="<<w2ac(userName, userNameSize,
myuserName)<<std::endl;
                delete myuserName;
            }
        }
        // 2 chars was big enough. Check if the function succeeded.
        // If the function did not succeed...
        else if (!RET_OK(ret))
        {
            std::cout<<"ERROR getting user name!"<<ret<<"¥n";
            return false;
        }
        // If the function succeeded...
        else
        {
            char* myuserName = new char[userNameSize];
            std::cout<<"Client: userName ="<<w2ac(userName, userNameSize,
myuserName)<<std::endl;
            delete myuserName;
        }
        // Clean up the variable.
        delete userName;
    }
}
// Could not retrieve the user info interface.
else
{
    std::cout<<"Error: get userinfo interface return false!¥n";
    return false;
}
```

6.1.4.4 レポートのテーブル情報の取得

```
    // TableSet Demo:
    // Retrieves the number of tables used in this report
    // as well as the alias name and table name for each table.

    std::cout<<"-------------- TableSet Demo --------------¥n";
    ITableSet* pTableSetInterface = NULL;
    if( RET_OK( GET_TABLESET_INTERFACE(i_reqHandle, pTableSetInterface) ) )
    {
        // Retrieve the TableSet interface.
        if( pTableSetInterface == NULL )
        {
            std::cout<<"Error: no TableSet interface!¥n";
            return false;
        }
        // If the interface is retrieved...
        if( pTableSetInterface != NULL )
        {
            // Get the number of tables.
            int numOfTables;
            short userid=0;
            if(! RET_OK(pTableSetInterface->NumOfTables(pTableSetInterface-
>hITableSet,&numOfTables)  ))
            {
                std::cout<<"Error getting number of tables!¥n";
                return false;
            }
            else
            std::cout<<"Client: num of tables="<<numOfTables<<std::endl;
            // Loop through the tables.
            for(int i=0; i<numOfTables; i++)
            {
                // Retrieve the alias name.
                // Since the length of the alias name is unknown, try a length
                // of 2 first.
                unsigned short aliasNameSize=2;
                Char* aliasName = new Char[aliasNameSize];
                ret = pTableSetInterface->GetAliasName(pTableSetInterface-
>hITableSet,i,&aliasNameSize,aliasName) ;
                // Check if the 2 char buffer is large enough.
                // If the buffer is not large enough...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"aliasName size is too small, should be:"
<<aliasNameSize<<"¥n";
                    // Re-create the variable with the real size.
                    delete aliasName;
                    aliasName = new Char[aliasNameSize];
                    // Call GetAliasName again.
                    if(! RET_OK( pTableSetInterface-
>GetAliasName(pTableSetInterface->hITableSet,i,

&aliasNameSize,aliasName) ))
                    {
                        std::cout<<"Error getting aliasName!"<<ret<<"¥n";
                    }
                    // Convert the alias name using the w2ac function. (See note
in w2ac function, below.)
                    else
                    {
                        char* myaliasName = new char[aliasNameSize];
                        std::cout<<"Client: aliasName ="<<w2ac(aliasName,
aliasNameSize, myaliasName)<<std::endl;
                        delete myaliasName;
                    }
```

```
                }
                // The buffer size is OK. Check if an error was returned.
                // If an error was returned...
                else if (!RET_OK(ret))
                {
                    std::cout<<"Error getting aliasName!"<<ret<<"¥n";
                    return false;
                }
                // If no error was returned...
                else
                {
                    char* myaliasName = new char[aliasNameSize];
                    std::cout<<"Client: aliasName ="<<w2ac(aliasName,
aliasNameSize, myaliasName)<<std::endl;
                    delete myaliasName;
                }
                // Clean up.
                delete aliasName;


                // Retrieve the table name.
                // Since the length of the table name is unknown, try a length
                // of 2 first.
                unsigned short tableNameSize=2;
                Char* tableName = new Char[tableNameSize];
                ret = pTableSetInterface->GetTableName(pTableSetInterface-
>hITableSet,i,

&tableNameSize,tableName) ;
                // Chek if 2 chars is big enough.
                // If it is not big enough ...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"tableName size is too small, should be:"
<<tableNameSize<<"¥n";
                    // Re-create the variable and try again with the
                    // real size.
                    delete tableName;
                    tableName = new Char[tableNameSize];
                    if(! RET_OK( pTableSetInterface-
>GetTableName(pTableSetInterface->hITableSet,i,

&tableNameSize,tableName) ))
                    {
                        std::cout<<"Error getting tableName!"<<ret<<"¥n";
                    }
                    else
                    {
                        char* mytableName = new char[tableNameSize];
                        std::cout<<"Client: tableName ="<<w2ac(tableName,
tableNameSize, mytableName)<<std::endl;
                        delete mytableName;
                    }
                }
                // Check if the buffer size is OK.
                // If an error occurred...
                else if (!RET_OK(ret))
                {
                    std::cout<<"Error getting tableName!"<<ret<<"¥n";
                    return false;
                }
                // If no error occurred...
                else
                {
                    char* mytableName = new char[tableNameSize];
                    std::cout<<"Client: tableName ="<<w2ac(tableName,
tableNameSize, mytableName)<<std::endl;
                    delete mytableName;
```

```
                }
                // Clean up.
                delete tableName;
            }
        }
    }
    // The tableset interface could not be retrieved.
    else
    {
        std::cout<<"Error: get table set interface return false!¥n";
        return false;
    }
```

6.1.4.5 プロンプト情報の取得

```
    // Prompt Demo:
    // Retrieve the number of prompts in the report.
    // For each prompt, get its subreport name, prompt type, number of values
for that prompt
    // and what those values are. Note that there will be no subreport name if
it is the main report.
    // Allow the oportunity for the program to translate and reset the values
for a particular prompt.
    std::cout<<"-------------- PromptSet Demo --------------¥n";

    // Initiliazing the PromptSet interface.
    IPromptSet* pPromptSetInterface = NULL;
    // Retrieve the PromptSet interface.
    if( RET_OK( GET_PROMPTSET_INTERFACE(i_reqHandle, pPromptSetInterface) ) )
    {
        if( pPromptSetInterface == NULL )
        {
            std::cout<<"Error: No Prompt interface!¥n";
            return false;
        }
        // If the promptset interface is retrieved, get the number of prompts.
        if( pPromptSetInterface != NULL )
        {
            int numOfPrompt;
            if(! RET_OK(pPromptSetInterface->NumOfPrompts(pPromptSetInterface-
>hIPromptSet,&numOfPrompt)  ))
            {
                std::cout<<"Error getting the number of prompts!¥n";
                return false;
            }
            else
            std::cout<<"Client: num of Prompts="<<numOfPrompt<<std::endl;
            // Loop through the promptset.
            for(int i=0; i<numOfPrompt; i++)
            {
                // Get prompt name.
                // Since the length of the prompt name is unknown,
                // try 2 first.
                unsigned short nameSize=2;
                Char* name = new Char[nameSize];
                ret = pPromptSetInterface->GetName(pPromptSetInterface-
>hIPromptSet,i,&nameSize, name);
                // If 2 is too short...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"Name size is too small, should be:"
<<nameSize<<"¥n";
                    // ...recreate the variable and try again with the
```

```
                    // real size.
                    delete name;
                    name = new Char[nameSize];
                    if(! RET_OK( pPromptSetInterface-
>GetName(pPromptSetInterface->hIPromptSet,i,
                                                                    &nameSize,
name)))
                    {
                        std::cout<<"Error getting name!"<<ret<<"¥n";
                    }
                    else
                    {
                        char* myname = new char[nameSize];
                        std::cout<<"Client: name ="<<w2ac(name, nameSize,
myname)<<std::endl;
                        delete myname;
                    }
                }
                // If a buffer length of 2 is ok, check if there were errors.
                // If there were errors ...
                else if (!RET_OK(ret))
                {
                    std::cout<<"ERROR at getting name!"<<ret<<"¥n";
                    return false;
                }
                //  If there were no errors...
                else
                {
                    char* myname = new char[nameSize];
                    std::cout<<"Client: name ="<<w2ac(name, nameSize,
myname)<<std::endl;
                    delete myname;
                }
                // Clean up.
                delete name;

                // Get the sub-reportname of the current prompt.
                // Since the length of the sub-reportname is unknown, try 2
first.
                unsigned short subrptnameSize=2;
                Char* subrptname = new Char[subrptnameSize];
                ret = pPromptSetInterface->GetSubReportName(pPromptSetInterface-
>hIPromptSet,i,

&subrptnameSize, subrptname);
                // If the buffer size is too small...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"Subreport name size is too small, should be:"
<<subrptnameSize<<"¥n";
                    // ...recreate the variable and try again with the real size
                    delete subrptname;
                    subrptname = new Char[subrptnameSize];
                    if(! RET_OK( pPromptSetInterface-
>GetSubReportName(pPromptSetInterface->hIPromptSet,i,&subrptnameSize,
subrptname)))
                    {
                        std::cout<<"Error getting the name!"<<ret<<"¥n";
                    }
                    else
                    {
                        char* mysubrptname = new char[subrptnameSize];
                        std::cout<<"Client: subrptname ="<<w2ac(subrptname,
subrptnameSize, mysubrptname)<<std::endl;
                        delete mysubrptname;
                    }
                }
                //  If the buffer size is OK...
```

```
                else if (!RET_OK(ret))
                {
                    std::cout<<"ERROR at getting subrptname!"<<ret<<"¥n";
                    return false;
                }
                //  If the buffer size is not OK...
                else
                {
                    char* mysubrptname = new char[subrptnameSize];
                    std::cout<<"Client: subrptname ="<<w2ac(subrptname,
subrptnameSize, mysubrptname)<<std::endl;
                    delete mysubrptname;
                }
                // Cleanup.
                delete subrptname;

                // Get the number of values.
                int numOfValues;
                 if(! RET_OK(pPromptSetInterface-
>NumOfPromptValues(pPromptSetInterface->hIPromptSet, i, &numOfValues)  ))
                {
                    std::cout<<"Error getting the number of prompt Values!¥n";
                    return false;
                }
                else
                    std::cout<<"Client: num of Prompt values
="<<numOfValues<<std::endl;
                // Loop through the prompt set.
                for(int j=0; j<numOfValues; j++)
                {
                    // Get the individual prompt values.
                    // Since the length of the prompt values is unknown, try 10
first.
                    unsigned short valueSize1=10;
                    Char* value1 = new Char[valueSize1];
                    unsigned short valueSize2=10;
                    Char* value2 = new Char[valueSize2];
                    PromptValuesType ptype;
                    ret = pPromptSetInterface-
>GetPromptValue(pPromptSetInterface->hIPromptSet,i, j, &ptype,
                                                                &valueSize1,
value1, &valueSize2, value2);

                    // The buffer is too small so...
                    if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                    {
                        std::cout<<"Value size is too small, should be:"
<<valueSize1<<","<<valueSize2<<"¥n";
                        // ...recreate the variables and try again with the real
size.
                        delete value1;
                        value1 = new Char[valueSize1];
                        delete value2;
                        value2 = new Char[valueSize2];
                        if(! RET_OK( pPromptSetInterface-
>GetPromptValue(pPromptSetInterface->hIPromptSet,i,j, &ptype,
                                                        &valueSize1,
value1,&valueSize2, value2)))
                        {
                            std::cout<<"Error getting value."<<ret<<"¥n";
                        }
                        else
                        {
                            char* myvalue1 = new char[valueSize1];

                            std::cout<<"Client: prompt value1 ="<<w2ac(value1,
valueSize1, myvalue1)<<std::endl;
                            delete myvalue1;
```

```
                            // If the prompt type is a range, there will
                            // be a second value.
                            if(ptype == pvalue_range)
                            {
                                char* myvalue2 = new char[valueSize2];
                                std::cout<<"Client: prompt value2
="<<w2ac(value2, valueSize2, myvalue2)<<std::endl;
                                delete myvalue2;
                            }

                        }
                    }
                    // If there are any errors...
                    else if (!RET_OK(ret))
                    {
                        std::cout<<"Error getting prompt value."<<ret<<"¥n";
                        return false;
                    }
                    // If there are no errors...
                    else
                    {
                        char* myvalue1 = new char[valueSize1];

                        std::cout<<"Client: prompt value1 ="<<w2ac(value1,
valueSize1, myvalue1)<<std::endl;
                        delete myvalue1;
                        // If and only if the prompt type is range, then will
there
                        // be a second value.
                        if(ptype == pvalue_range)
                        {
                            char* myvalue2 = new char[valueSize2];
                            std::cout<<"Client: prompt value2 ="<<w2ac(value2,
valueSize2, myvalue2)<<std::endl;
                            delete myvalue2;
                        }
                    }
                    // Cleanup.
                    delete value1;
                    delete value2;

                    // Reset value for the individual prompt.
                    int vSize=3;
                    Char* v = new Char[vSize];
                    v[0] = '5';
                    v[1] = '5';
                    v[2] = '¥0';
                    if(! RET_OK(pPromptSetInterface-
>SetPromptValue(pPromptSetInterface->hIPromptSet, i,j, ptype,v, v)  ))
                    {
                        std::cout<<"Error setting num of prompt values.¥n";
                        return false;
                    }
                    else
                        std::cout<<"changed values!¥n";
                    // Cleanup.
                    delete v;
                }
            }
        }
    }
    // Not able to retrieve the interface.
    else
    {
        std::cout<<"Error: Get prompt interface return false!¥n";
        return false;
    }
```

6.1.4.6 選択式の変更

```
    //  Selection Formula Demo
    //  In the report info demo above, the report is checked to see if its
    //  name matches the one in the tochangeSF() function below.
    //  If a match of report is found, modify the selection formula as follows:
    //  add the selection formula to the existing selection formula.
    //  That is, after calling
    //  SetSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
mySF),
    //  the selection formula becomes: existingSF "AND" newSF.
    //  Also demonstrates retrieving the existing selection formula
    //  (GetOldSelectionFormula) for reference.
    std::cout<<"-------------- Request Demo --------------¥n";

    if(! i_reqHandle->GetRequestType(i_reqHandle,&rtype))
    {
        std::cout<<"Error getting RequestType!¥n";
        return false;
    }
    else
        std::cout<<"Client: request type="<<rtype<<std::endl;

    // If this is the required report, get the SelectionFormula interface.
    if(changeSF )
    {
        ISelectionFormula* pSelectionFormulaInterface = NULL;
        // Check to see if the SelectionFormula interface was retrieved.
        if( RET_OK( GET_SELECTIONFORMULA_INTERFACE(i_reqHandle,
pSelectionFormulaInterface) ) )
        {
            if( pSelectionFormulaInterface == NULL )
            {
                std::cout<<"Error: no SelectionFormula interface!¥n";
                return false;
            }
            std::cout<<"-------------- sf Demo --------------¥n";
            // If the request type equals schedule...
            if(rtype == schedule)
            {

                if( pSelectionFormulaInterface != NULL )
                {
                    // Create a char* with a selection
                    // condition "{Employee.Last Name}>"H".
                    int strSize=25;
                    Char* str = new Char[strSize];

                    str[0]='{' ;
                    str[1]='E';
                    str[2]='m';
                    str[3]='p';
                    str[4]='l';
                    str[5]='o';
                    str[6]='y';
                    str[7]='e';
                    str[8]='e';
                    str[9]='.';
                    str[10]='L';
                    str[11]='a';
                    str[12]='s';
                    str[13]='t';
                    str[14]=' ';
                    str[15]='N';
                    str[16]='a';
                    str[17]='m';
```

```
                    str[18]='e';
                    str[19]='}';
                    str[20]='>';
                    str[21]='"';
                    str[22]='H';
                    str[23]='"';
                    str[24]='¥0';

                    // SetSelectionFormula ***APPENDS*** the condition to the
existing SelectionFormula.
                    ret = pSelectionFormulaInterface-
>SetSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                        str) ;
                    // Cleanup.
                    delete str;
                    std::cout<<"sf ret:"<<ret<<std::endl;
                    // Retrieve the existing selection formula.
                    // Since the length of the selection formula is unknown, try
2 first.
                    int oldsfSize=2;
                    Char* oldsf = new Char[oldsfSize];
                    ret = pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                  &oldsfSize,
oldsf);

                    // If the buffer is too small...
                    if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                    {
                        std::cout<<"oldsf size too small, should be:"
<<oldsfSize<<"¥n";
                        // Re-create the variable and delete oldsf.
                        delete oldsf;
                        oldsf = new Char[oldsfSize];
                        if(! RET_OK( pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                 &oldsfSize,
oldsf)))
                        {
                            std::cout<<"Error getting oldsf!"<<ret<<"¥n";
                        }
                        else
                        {
                            char* myoldsf = new char[oldsfSize];
                            std::cout<<"Client: oldsf ="<<w2ac(oldsf, oldsfSize,
myoldsf)<<std::endl;
                            delete myoldsf;
                        }
                    }
                    // Check if the buffer size is OK.
                    // If there were any errors...
                    else if (!RET_OK(ret))
                    {
                        std::cout<<"ERROR at getting oldsf!"<<ret<<"¥n";
                        return false;
                    }
                    //  If there were no errors...
                    else
                    {
                        char* myoldsf = new char[oldsfSize];
                        std::cout<<"Client: oldsf ="<<w2ac(oldsf, oldsfSize,
myoldsf)<<std::endl;
                        delete myoldsf;
                    }
                    // Clean up.
                    delete oldsf;
                }
            }
```

```
            // If the request type is view...
            if(rtype == view)
            {
                // Set up a select criterion of:
                // {Employee.Last Name} < "M"
                int str2Size=25;
                Char* str2 = new Char[str2Size];

                str2[0]='{' ;
                str2[1]='E';
                str2[2]='m';
                str2[3]='p';
                str2[4]='l';
                str2[5]='o';
                str2[6]='y';
                str2[7]='e';
                str2[8]='e';
                str2[9]='.';
                str2[10]='L';
                str2[11]='a';
                str2[12]='s';
                str2[13]='t';
                str2[14]=' ';
                str2[15]='N';
                str2[16]='a';
                str2[17]='m';
                str2[18]='e';
                str2[19]='}';
                str2[20]='<';
                str2[21]='"';
                str2[22]='M';
                str2[23]='"';
                str2[24]='¥0';

                // SetSelectionFormula ***APPENDS*** the condition to the
existing selection formula.
                ret = pSelectionFormulaInterface-
>SetSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                  str2) ;

                // Cleanup.
                delete str2;
                std::cout<<"sf2 ret:"<<ret<<std::endl;
                // Retrieve the existing selection formula.
                // Since the length of the selection formula is unknown,
                // try 2 first.
                int oldsf2Size=2;
                Char* oldsf2 = new Char[oldsf2Size];
                ret = pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,

&oldsf2Size, oldsf2);

                // If the buffer is too small...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"oldsf size too small, should be:"
<<oldsf2Size<<"¥n";
                    // ... Re-create the variable with the real size.
                    delete oldsf2;
                    oldsf2 = new Char[oldsf2Size];
                    if(! RET_OK( pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                  &oldsf2Size,
oldsf2)))
                    {
                        std::cout<<"Error getting oldsf2!"<<ret<<"¥n";
                    }
```

```
                    else
                    {
                        char* myoldsf2 = new char[oldsf2Size];
                        std::cout<<"Client: oldsf2 ="<<w2ac(oldsf2, oldsf2Size,
myoldsf2)<<std::endl;
                        delete myoldsf2;
                    }
                }
                // If the buffer size is OK, check for any errors?
                // If there are errors...
                else if (!RET_OK(ret))
                {
                    std::cout<<"Error getting oldsf2!"<<ret<<"¥n";
                    return false;
                }
                // If there are no errors...
                else
                {
                    char* myoldsf2 = new char[oldsf2Size];
                    std::cout<<"Client: oldsf2 ="<<w2ac(oldsf2, oldsf2Size,
myoldsf2)<<std::endl;
                    delete myoldsf2;
                }
                // Cleanup.
                delete oldsf2;
            }
        }
    }
```

重要免責事項および法的情報

ハイパーリンク

リンクの一部は、アイコンやマウスオーバーテキストで分類されています。これらのリンクから、追加の情報を得ることができます。
アイコンについて。

- このアイコンが付いたリンク: SAP がホストしているものではない Web サイトに移動します。これらのリンクを使用することで、お客様は (お客様と SAP との契約書に別段の明示的な記載がない限り) 以下のことに同意することになります。
  - リンク先のサイトのコンテンツが SAP のドキュメンテーションではないこと。お客様は、この情報に基づいて SAP に対する製品クレームを推断することはできません。
  - SAP が、リンク先のサイトのコンテンツについて同意することも反対することもなく、また SAP がその利用可能性や正確性について保証しないこと。SAP は、かかるコンテンツの使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。
- このアイコンが付いたリンク: 当該の特定の SAP 製品又はサービスのドキュメンテーションから離れ、SAP がホストしている Web サイトに移動します。これらのリンクを使用することで、お客様は (お客様と SAP との契約書に別段の明示的な記載がない限り)、この情報に基づいて SAP に対する製品クレームを推断することはできないことに同意します。

外部プラットフォームでホストされているビデオ

一部のビデオは、サードパーティのビデオホスティングプラットフォームに置かれている場合があります。SAP では、これらのプラットフォームに保存されているビデオが将来にわたって利用できると保証することはできません。また、これらのプラットフォームにホストされている、いかなる広告またはその他のコンテンツ (関連ビデオまたは同じサイトでホストされている別のビデオに移動する場合など) については、SAP の管理外であり責任を負いません。

ベータおよびその他の試験的機能

試験的機能は、SAP が将来のリリースを保証する正式に提供される機能の範囲外です。これは、試験的機能は、SAP により通知なく理由の如何を問わず随時変更される場合があることを意味します。試験的機能は、本稼働使用のためのものではありません。お客様は、試験的機能を実際の運用環境で、又は十分なバックアップがとられていないデータとともに、デモンストレーション、テスト、試験、評価その他の方法で使用してはなりません。
試験的機能の目的は、早期にフィードバックを得ることで、それに応じて顧客の皆様やパートナーが将来の製品に影響を与えることを可能にすることです。SAP コミュニティなどにおいてフィードバックを提供することで、お客様は、投稿物や二次的著作物の知的財産権が SAP の独占的所有物であり続けることを承認することになります。

コード例

ソフトウェアのコーディングやコードスニペットはすべて、例です。それらは、本稼働使用のためのものではありません。コード例は、構文や表現規則を分かりやすく説明し視覚化することのみを目的としています。SAP は、コード例の正確性や完全性について保証しません。SAP は、コード例の使用により発生した過誤や損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、損害に対して一切責任を負いません。

偏見のない表現

SAP は、ダイバーシティ & インクルージョンの文化を支持しています。SAP の文書では、可能な限り、文化、民族性、ジェンダー、および障がいの有無を問わず、すべての人々に対する偏見を伴わない表現を採用します。

www.sap.com/contactsap

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

本書のいかなる部分も、SAP SE 又は SAP の関連会社の明示的な許可なくして、いかなる形式でも、いかなる目的にも複製又は伝送することはできません。本書に記載された情報は、予告なしに変更されることがあります。

SAP SE 及びその頒布業者によって販売される一部のソフトウェア製品には、他のソフトウェアベンダーの専有ソフトウェアコンポーネントが含まれています。製品仕様は、国ごとに変わる場合があります。

これらの文書は、いかなる種類の表明又は保証もなしで、情報提供のみを目的として、SAP SE 又はその関連会社によって提供され、SAP 又はその関連会社は、これら文書に関する誤記脱落等の過失に対する責任を負うものではありません。SAP 又はその関連会社の製品及びサービスに対する唯一の保証は、当該製品及びサービスに伴う明示的保証がある場合に、これに規定されたものに限られます。本書のいかなる記述も、追加の保証となるものではありません。

本書に記載される SAP 及びその他の SAP の製品やサービス、並びにそれらの個々のロゴは、ドイツ及びその他の国における SAP SE （又は SAP の関連会社）の商標若しくは登録商標です。本書に記載されたその他すべての製品およびサービス名は、それぞれの企業の商標です。

商標に関する詳細の情報や通知については、https://www.sap.com/japan/about/legal/trademark.html をご覧ください。

THE BEST RUN SAP