Oracle® Data Relationship Management

Oracle® Data Relationship Steward

Oracle® Data Relationship Management for Oracle Hyperion Enterprise Planning Suite

Oracle® Data Relationship Management for Oracle Hyperion Financial Close Suite

Oracle® Data Relationship Management for Customer Hub

Oracle® Data Relationship Management Read Only Access

Administrator's Guide

リリース 11.1.2.3

ORACLE

ENTERPRISE PERFORMANCE MANAGEMENT SYSTEM

Data Relationship Management Administrator's Guide, 11.1.2.3

著者: EPM 情報開発チーム

目次

ドキュメントのアクセシビリティについて

Oracle のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc を参照してください。

Access to Oracle Support

Oracle サポート・サービスでは、My Oracle Support を通して電子支援サービスを提供しています。詳細情報は http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info か、聴覚に障害のあるお客様は http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

1 はじめに

この章の内容

Data Relationship Management アプリケーションの管理

Oracle Data Relationship Management では、アプリケーションを使用してデータを管理し、データへのアクセスや変更を行うためのユーザー要求を処理します。1 つの Data Relationship Management インストールで、1 つ以上のアプリケーションをサポートできます。アプリケーションごとに独自のシステム・メタデータおよびセキュリティ構成を使用して、データの管理とアクセスを行います。複数のデータ・セットや、共通および制限されたデータ・セットに対する様々なレベルのアクセス権を持った複数のユーザーを同じアプリケーションでサポートできます。ただし、アプリケーション内のすべてのシステム・メタデータが、同じユーザーによって共有および管理されます。システム・メタデータへの変更はすぐに有効になり、すべてのユーザーとデータが影響を受けます。別のユーザー・グループが、その他のグループにより行われたメタデータ変更の影響を受けないようにする必要がある場合は、各グループで別々のアプリケーションを使用することをお薦めします。

アプリケーションは、Data Relationship Management のプライマリ・アプリケーション・サーバーからアクセス可能な構成コンソールで作成されます。新規アプリケーション作成の詳細は、Oracle Data Relationship Management Installation Guide のアプリケーションの作成に関する項を参照してください。

新しい Data Relationship Management アプリケーションには、プロパティ定義とカテゴリ、およびデフォルトの管理ユーザーなどのコア・メタデータ・オブジェクトが含まれます。この初期構成で、デフォルト・ユーザーは、アプリケーションを構築、移入、プロビジョニングするための、次の 4 つのタスクを実行できます:

- バージョンおよび階層の作成
- クエリー、比較、インポート、ブレンダおよびエクスポートなどのユーザー・メタデータ・オブジェクトの定義
- ドメイン、プロパティ定義、検証およびノード・タイプを含むシステム・メタデータ・オブジェクトの設定および構成
- ユーザーの追加、および製品機能、オブジェクト、データにアクセスするためのセキュリティの構成

このガイドでは、Data Relationship Management アプリケーションのシステム・メタデータおよびユーザー・セキュリティに関連する管理タスクを説明しています。バージョン、階層およびユーザー・メタデータ・オブジェクトの管理の詳細は、Oracle Data Relationship Management User's Guide を参照してください。

Data Relationship Management へのアクセス

➤ Data Relationship Management クライアントを起動するには:

1 「スタート」、「プログラム」、「Oracle EPM System」、「Data Relationship Management」、「Web クライアント」の順に選択します。

2 ユーザー名とパスワードを入力します。

ユーザー名とパスワードは大文字と小文字が区別されます。

3 アプリケーションを選択し、「ログオン」をクリックします。

詳細は、パスワードの変更を参照してください。

パスワードの変更

➤ パスワードを変更するには:

1 Data Relationship Management のホーム・ページから、「プリファレンス」を選択します。

2 「パスワードの変更」をクリックします。

3 現在のパスワードを入力します。

4 新規パスワードを入力します。

注: ユーザーがネイティブに認証されており、PasswordPolicyEnabled システム・プリファレンスが True に設定されている場合、パスワードには次のうち 3 つの要素が含まれる必要があります。

- 大文字
- 小文字
- 数字
- 特殊文字

注: それ以外の場合は、ユーザーが Oracle Hyperion Shared Services を介して認証されるときに外部ディレクトリによって認証される場合を除き、パスワードは制限されません。

5 新規パスワードを再入力します。

6 「OK」をクリックします。

2 ユーザーの管理

この章の内容

ユーザー権限

Data Relationship Management は、3 レベルの権限を使用して製品機能およびデータへのユーザー・アクセスを制御します。レベルが高い権限には、それより低いレベルの権限も含まれます。ユーザーに高いレベルの権限を付与すると、それより低いレベルの権限も付与されます。たとえば、ユーザーにレベル 1 の権限を付与すると、それより低いレベル 2 とレベル 3 の権限も付与されます。

詳細は、次を参照してください:

- バージョン権限
- 要求権限
- クエリー権限
- 比較権限
- インポート権限
- ブレンダ権限
- エクスポート権限
- スクリプト権限
- 監査権限
- アプリケーション権限
- アクセス権限

バージョン権限

権限レベル 1	権限レベル 2	権限レベル 3
バージョンの管理--ユーザーは「バージョン」と「階層」のメニュー・オプションにアクセスできます	**バージョンの参照**--ユーザーには、ノード・アクセス・グループで権限を付与されている任意のバージョンへのアクセス権があります	NA
	バージョンの作成--ユーザーは自身が所有者である任意のバージョンを管理(更新/削除)できます。ユーザーは「バージョン」メニュー・オプションにアクセスできます。 **注：** バージョンの管理権限を持つユーザーが所有者を変更するまで、バージョンを作成したユーザーが所有者です。	NA
	階層の管理--ユーザーには、「階層」メニュー・オプションへのアクセス権があります。	**階層の参照**--ユーザーには、ノード・アクセス・グループで権限を付与されている任意の階層へのアクセス権があります。ユーザーには、「編集」のノード・アクセス権以上があれば、「ノード」メニュー・オプションへのアクセス権があります。
		階層の作成--ユーザーは自身が所有者である任意の階層を管理(更新/削除)できます。ユーザーには、「階層」メニュー・オプションへのアクセス権があります。ユーザーは自身が所有者である階層のノード・タイプを無効にできます。 **注：** 階層の管理権限を持つユーザーが所有者を変更するまで、階層を作成したユーザーが所有者です。

要求権限

権限レベル 1	権限レベル 2	権限レベル 3
要求の管理--ユーザーはシステムでまだコミットされていない要求を削除できます。	**要求の作成**--ユーザーはシステムで要求をクエリーし、自身が所有者である要求を管理(更新/削除)できます。	NA

クエリー権限

権限レベル 1	権限レベル 2	権限レベル 3
システム・クエリーの管理--ユーザーには、システム・クエリーと「クエリー」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**ユーザー・クエリーの管理**--ユーザーには、ユーザー・クエリーと標準クエリーを表示および実行するアクセス権があります。「標準クエリー」の「クエリー」メニュー・オプションへのアクセス権はありません。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**クエリーの実行**--ユーザーは標準クエリーを表示および実行できます。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。ユーザーには、「編集」のノード・アクセス権以上があれば、「ノード」メニュー・オプションへのアクセス権があります。
	標準クエリーの管理--ユーザーには、標準クエリーの「クエリー」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA

比較権限

権限レベル 1	権限レベル 2	権限レベル 3
システム比較の管理--ユーザーには、システム比較と「比較」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**ユーザー比較の管理**--ユーザーには、ユーザー比較と標準比較を表示および実行するアクセス権があります。「標準比較」の「比較」メニュー・オプションへのアクセス権はありません。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**比較の実行**--ユーザーは標準比較を表示および実行できます。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。ユーザーには、「編集」のノード・アクセス権以上があれば、「ノード」メニュー・オプションへのアクセス権があります。
	標準比較の管理--ユーザーには、標準比較の「比較」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA

インポート権限

権限レベル 1	権限レベル 2	権限レベル 3
システム・インポートの管理--ユーザーには、システム・インポートと「インポート」メニュー・オプションへのアクセス権があります。「プロパティ・カテゴリ」のセキュリティに基づいて、「プロパティ」セレクタへの制限付きアクセス権があります。	**ユーザー・インポートの管理**--ユーザーには、ユーザー・インポートと標準インポートを表示および実行するアクセス権があります。「標準インポート」の「インポート」メニュー・オプションへのアクセス権はありません。「プロパティ・カテゴリ」のセキュリティに基づいて、「プロパティ」セレクタへの制限付きアクセス権があります。	**インポートの実行**--ユーザーは標準インポートを表示および実行できます。「プロパティ・カテゴリ」のセキュリティに基づいて、「プロパティ」セレクタへの制限付きアクセス権があります。
	標準インポートの管理--ユーザーには、標準インポートの「インポート」メニュー・オプションへのアクセス権があります。「プロパティ・カテゴリ」のセキュリティに基づいて、「プロパティ」セレクタへの制限付きアクセス権があります。	NA

ブレンダ権限

権限レベル 1	権限レベル 2	権限レベル 3
システム・ブレンダの管理--ユーザーには、システム・ブレンダと「ブレンダ」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**ユーザー・ブレンダの管理**--ユーザーには、ユーザー・ブレンダと標準ブレンダを表示および実行するアクセス権があります。「標準ブレンダ」の「ブレンダ」メニュー・オプションへのアクセス権はありません。	**ブレンダの実行**--ユーザーは標準ブレンダを表示および実行できます。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。
	標準ブレンダの管理--ユーザーには、標準ブレンダの「ブレンダ」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA

エクスポート権限

権限レベル 1	権限レベル 2	権限レベル 3
システム・エクスポートの管理--ユーザーには、システム・エクスポートと「エクスポート」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**ユーザー・エクスポートの管理**--ユーザーには、ユーザー・エクスポート、標準エクスポート、ブックを表示および実行するアクセス権があります。「標準エクスポート」と「ブック」の「エクスポート」メニュー・オプションへのアクセス権はありません。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	**エクスポートの実行**--ユーザーは標準エクスポートを表示および実行できます。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。
	標準エクスポートの管理--ユーザーには、標準エクスポートおよびブックの「エクスポート」メニュー・オプションへのアクセス権があります。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA

スクリプト権限

権限レベル 1	権限レベル 2	権限レベル 3
アクション・スクリプトの実行--ユーザーはアクション・スクリプトを実行できます。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA	NA

監査権限

権限レベル 1	権限レベル 2	権限レベル 3
ユーザー・トランザクションの監査--ユーザーは自身が実行したトランザクションをクエリーできます。トランザクションには、データおよびメタデータの変更と、ログインなどログに記録されたアクション、実行中の非同期操作が含まれます。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA	NA

権限レベル 1	権限レベル 2	権限レベル 3
データ・トランザクションの監査--ユーザーは、権限またはノード・アクセス・グループで自身にアクセス権があるデータ・オブジェクトに対するトランザクションをクエリーできます。トランザクションには、ユーザーが実行したトランザクションと、他のユーザーが行った変更が含まれます。ノード・レベルのトランザクションの場合で、ユーザーがすべての子孫に対する読取りアクセス権も持っている場合は、ノードおよびそのすべての子孫をトランザクションからクエリーすることができます(「子ノードを含める」オプション)。「ノード・アクセス・グループ」の割当てと「プロパティ・カテゴリ」のセキュリティに基づいて、「バージョン」、「階層」、「ノード」、「プロパティ」の各セレクタへの制限付きアクセス権があります。	NA	NA

権限レベル 1	権限レベル 2	権限レベル 3
システム・トランザクションの監査--ユーザーは自身が実行したトランザクションをクエリーできます。トランザクションには、データおよびメタデータの変更と、ログインなどログに記録されたアクション、実行中の非同期操作が含まれます。	NA	NA

アプリケーション権限

権限レベル 1	権限レベル 2	権限レベル 3
アプリケーションの管理	**カテゴリの管理**	**カテゴリの参照**--ユーザーには、プロパティ・カテゴリ・セキュリティで権限を付与されている任意のプロパティ・カテゴリへのアクセス権があります。
	プロパティの管理	**プロパティの参照**--ユーザーには、プロパティ・カテゴリ・セキュリティで権限を付与されているプロパティ・カテゴリのすべてのプロパティへのアクセス権があります。
		プロパティ・リストの管理--ユーザーはプロパティ定義の値リストと参照テーブルを管理できます。
	検証の管理	NA
	ノード・タイプの管理	NA
	プリファレンスの管理	NA

アクセス権限

権限レベル 1	権限レベル 2	権限レベル 3
アクセス権の管理	**ユーザーの管理**--ユーザーは自身のユーザー・プロファイルを編集または削除できません。	NA
	役割の管理--ユーザーは自身の役割割当てを編集できません。	NA
	アクセス・グループの管理--ユーザーは自身のノード・アクセス・グループ割当てを編集できません。	NA

権限レベル 1	権限レベル 2	権限レベル 3
	プロパティ・アクセス権の管理--ユーザーは自身のプロパティ・カテゴリ割当てを編集できません。	NA

ユーザー役割

Data Relationship Management の権限は、役割を使用して割り当てます。各ユーザー役割には、製品機能またはデータにアクセスできる一連の権限が関連付けられます。ユーザーには 1 つ以上の役割を割り当てることができ、それによってすべての役割から組み合せた権限が付与されます。アクセス権のレベルが競合する 2 つの役割を 1 つのユーザーに割り当てた場合は、レベルが高い方のアクセス権が付与されます。

Data Relationship Management には次のユーザー役割があり、割り当てられている権限がそれぞれマークされています。

権限			ユーザー役割							
レベル 1	レベル 2	レベル 3	アクセス・マネージャ	匿名ユーザー	アプリケーション管理者	データ作成者	データ・マネージャ	インタラクティブ・ユーザー	ワークフロー・ユーザー	ガバナンス・ユーザー
バージョンの管理							X			
	バージョンの参照		X	X	X	X		X	X	X
	バージョンの作成					X				
	階層の管理						X			
		階層の参照	X	X	X	X		X	X	X
		階層の作成				X				
要求の管理							X			
	要求の作成				X				X	
システム・クエリーの管理					X					
	ユーザー・クエリーの管理					X	X	X		

権限			ユーザー役割							
レベル 1	レベル 2	レベル 3	アクセス・マネージャ	匿名ユーザー	アプリケーション管理者	データ作成者	データ・マネージャ	インタラクティブ・ユーザー	ワークフロー・ユーザー	ガバナンス・ユーザー
		クエリーの実行		X					X	
	標準クエリーの管理						X			
システム比較の管理					X					
	ユーザー比較の管理					X	X	X		
		比較の実行		X					X	
	標準比較の管理						X			
システム・インポートの管理					X					
	ユーザー・インポートの管理					X	X			
		インポートの実行								
	標準インポートの管理						X			
システム・ブレンダの管理					X					
	ユーザー・ブレンダの管理					X	X			
		ブレンダの実行								
	標準ブレンダの管理						X			
システム・エクスポートの管理					X					

権限			ユーザー役割							
レベル 1	レベル 2	レベル 3	アクセス・マネージャ	匿名ユーザー	アプリケーション管理者	データ作成者	データ・マネージャ	インタラクティブ・ユーザー	ワークフロー・ユーザー	ガバナンス・ユーザー
	ユーザー・エクスポートの管理					X	X	X		
		エクスポートの実行		X					X	
	標準エクスポートの管理						X			
アクション・スクリプトの実行					X	X	X	X		
ユーザー・トランザクションの監査			X		X	X	X	X	X	
データ・トランザクションの監査					X	X	X	X		
システム・トランザクションの監査			X		X					
アプリケーションの管理					X					
	カテゴリの管理									
		カテゴリの参照	X	X		X	X	X	X	X
	プロパティの管理									
		プロパティの参照	X	X		X	X	X	X	X
		プロパティ・リストの管理					X			

権限			ユーザー役割							
レベル 1	レベル 2	レベル 3	アクセス・マネージャ	匿名ユーザー	アプリケーション管理者	データ作成者	データ・マネージャ	インタラクティブ・ユーザー	ワークフロー・ユーザー	ガバナンス・ユーザー
	検証の管理									
	ノード・タイプの管理				X					
	プリファレンスの管理									
アクセス権の管理										
	ユーザーの管理		X							
	役割の管理		X							
	アクセス・グループの管理		X							
	プロパティ・アクセス権の管理		X							
ワークフロー参加者										X

ユーザーの作成

ユーザーを作成する際に、一意の名前を定義し、役割を割り当てます。ユーザーにデータ・マネージャ役割を割り当てない場合は、ノード・アクセス・グループとプロパティ・カテゴリをそのユーザーに割り当てて、データへのアクセス権を制御できます。

➤ ユーザーを作成するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「ユーザー」を選択します。

3 ユーザーの一意の名前とフル・ネームを入力します。

注： 「部署」、「電話番号」および「電子メール・アドレス」はオプションです。データ・ガバナンス・ワークフローのユーザーは、電子メール通知を受信するために、構成済の電子メール・アドレスを持っている必要があります。

4 **「オプション」: 次のオプションから選択します。**

- 「パスワードの有効期限なし」--PasswordDuration システム・プリファレンス設定が無視されます。
- 「ログイン・セッションの有効期限なし」--IdleTime システム・プリファレンス設定が無視されます。

  注： このオプションを選択する場合、指定できる最大のアイドル時間は 24 時間です。アイドル時間が 24 時間を過ぎると、ログイン・セッションは期限切れになります。
- 「ユーザーにロックアウト手法を適用しない」--このユーザーに対してはロックアウト制限が無視されます。

5 **「役割」タブで、ユーザーに割り当てる役割を「使用可能」リストから選択します。矢印を使用して、役割を「選択済」リストに移動します。**

注： 役割の詳細は、ユーザー役割を参照してください。

6 **「ノード・アクセス・グループ」タブで、ユーザーに割り当てるグループを「使用可能」リストから選択します。矢印を使用して、グループを「選択済」リストに移動します。**

7 **「プロパティ・カテゴリ」タブで、ユーザーに割り当てるカテゴリを「使用可能」リストから選択します。矢印を使用して、カテゴリを「選択済」リストに移動します。**

8 **選択したリストの各カテゴリについて、次の手順を実行します。**

1. 「アクション」列で  をクリックし、ユーザーのアクセス権(「読取り」または「編集」)をカテゴリに設定します。
2. 「アクション」列で  を選択し、変更を保存します。

9  **をクリックします。**

「パスワードの変更」ダイアログ・ボックスが表示されます。

10 **ユーザーのパスワードを入力します。**

11 **再度パスワードを入力します。**

12 **「オプション:」ユーザーが次にログインしたときパスワードの変更を要求する場合は、「ユーザーは次回のログインでパスワードを変更する必要があります」を選択します。**

13 **「OK」をクリックします。**

ユーザー認証

Data Relationship Management では、格納されているパスワード情報を使用してアプリケーションによってネイティブで認証されるユーザー、または外部ユーザー・ディレクトリを使用して認証されるユーザーがサポートされます。Data Relationship Management の各アプリケーションは、このどちらか、または両方のタイプのユーザーをサポートするように構成されています。

アプリケーション認証は、Data Relationship Management コンソールの「認証設定」タブで設定します。詳細は、Oracle Data Relationship Management Installation Guide を参照してください。

ユーザー・パスワードの特性と、内部認証されたユーザーのパスワードの有効期限は、次のシステム・プリファレンスに定義されている値によって決まります。

- PasswordPolicyEnabled--有効にする場合は、パスワードに次の要素のうち 3 つを含める必要があります:
  - 大文字
  - 小文字
  - 数字
  - 特殊文字
- PasswordMaxLength--パスワードの最大文字長を決定します。
- PasswordMinLength--パスワードの最小文字長を決定します。
- PasswordDuration--パスワードが有効な日数を決定します。
- PasswordWarningPeriod--実際にログインできなくなるパスワード有効期限日の何日前(-)または何日後(+)に、パスワードを変更するようユーザーに警告するかを指定します。値が負、たとえば-3 の場合には、パスワードの期限が切れる前の 3 日間、ログイン時にユーザーに警告します。値が正、たとえば 5 の場合には、パスワードの期限が切れた後の 5 日間、ログイン時にユーザーに警告します。5 日間が過ぎると、ユーザーはパスワードを変更しないかぎりログインできなくなります。

  注： PasswordDuration と PasswordWarningPeriod の値を変更しても、次にパスワードを変更するまでユーザーには影響しません。たとえば、PasswordDuration が 30 日間に設定され、ユーザー 1 のパスワードが 26 日前に変更された場合、パスワードはあと 4 日で期限が切れます。PasswordDuration 値を 60 日間に変更しても、やはりユーザー 1 のパスワードはあと 4 日で期限が切れます。このユーザーがパスワードを変更すると、新しいパスワードの有効期限は 60 日間になります。

ユーザーの変更

ユーザーに対しては、パスワードの変更、ロックアウトまたはロック解除、あるいは役割、グループ、カテゴリの割当ての変更が可能です。

詳細は、次を参照してください:

- パスワードの変更
- ユーザーのロックアウト
- ユーザーのロック解除
- ユーザー役割と割当ての変更

パスワードの変更

➤ パスワードを変更するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ユーザー」を展開します。

3 ユーザーを選択して  をクリックします。

4  をクリックします。

5 ユーザーの新しいパスワードを入力します。

6 再度パスワードを入力します。

7 「オプション:」ユーザーが次にログインしたときパスワードの変更を要求する場合は、「ユーザーは次回のログインでパスワードを変更する必要があります」を選択します。

8 「OK」をクリックします。

ユーザーのロックアウト

Data Relationship Management アプリケーションにアクセスしないようにユーザーをロックアウトできます。ユーザーをロックアウトするとき、ロックアウトに独自の理由を設定できます。この理由が、アプリケーションにログインを試みたユーザーに表示されます。

➤ ユーザーをロックアウトするには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ユーザー」を展開します。

3 ユーザーを選択して  をクリックします。

4  をクリックします。

5 ロックアウトの理由を入力します。

6 「OK」をクリックします。

ユーザーのロック解除

ロックアウトされているユーザーをロック解除すると、アプリケーションへのアクセスが可能になります。

➤ ユーザーをロック解除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ユーザー」を展開します。

3 ユーザーを選択して  をクリックします。

4  をクリックします。

5 「OK」をクリックします。

ユーザー役割と割当ての変更

➤ ユーザー役割と割当てを変更するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ユーザー」を展開します。

3 ユーザーを選択して  をクリックします。

4 「役割」タブで、ユーザーに割り当てる役割を「使用可能」リストから選択します。矢印を使用して、役割を「選択済」リストに移動します。

5 「ノード・アクセス・グループ」タブで、ユーザーに割り当てるグループを「使用可能」リストから選択します。矢印を使用して、グループを「選択済」リストに移動します。

6 「プロパティ・カテゴリ」タブで、ユーザーに割り当てるカテゴリを「使用可能」リストから選択します。矢印を使用して、カテゴリを「選択済」リストに移動します。

7 選択したリストの各カテゴリについて、次の手順を実行します。

1.  をクリックし、ユーザーのアクセス権(「読取り」または「編集」)をカテゴリに設定します。
2.  を選択し、変更を保存します。

8  をクリックします。

ユーザーの削除

アクティブでなくなったユーザーは、アプリケーションから削除できます。ユーザーを削除すると、そのユーザーに関連付けられているユーザーレベルのメタデータ・オブジェクトもすべて削除されます。このメタデータ・オブジェクトには、クエリー、比較、インポート、ブレンダ、エクスポートおよびブックが含まれます。

➤ ユーザーを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ユーザー」を展開します。

3 ユーザーを選択して  をクリックします。

4 「このアイテムを削除」をクリックし、削除を確定します。

ユーザーのログイン・ステータスの表示

各ユーザーについて、ログインの統計および情報を表示できます。

- ユーザーが有効にログインした前回の日時
- 無効なログイン試行の回数
- ユーザーがロックアウトされているかどうか
- ユーザーがロックアウトされた日時
- ロックアウトの理由

➤ ユーザーのログイン・ステータスを表示するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「セキュリティ」の下で、「ユーザー」を展開します。**

3 **ユーザーを選択して  をクリックします。**

4 **「ログイン・ステータス」タブを選択します。**

3 ノード・アクセス・グループの管理

この章の内容

Data Relationship Management は、階層ノードとそのプロパティに対するユーザー・アクセスを粒度別に管理するためにノード・アクセス・グループを使用します。Data Relationship Management のバージョン内で階層のサブセットにおける特定のノードへのアクセス権がグループに付与され、ユーザーはそのグループに割り当てることができます。ノード・アクセス・グループは継承を使用して、アクセス・レベルが明示的に割り当てられている階層ノードの子孫ノードに類似のアクセス権を付与します。このアクセス・レベルは、下位レベルで上書きできますが、上書きされないようにロックすることもできます。

通常、ノード・アクセス・グループは組織の機能上の範囲に該当し、ユーザーは複数のグループへの割当てが必要になる場合もあります。割り当てられたアクセス・レベルが競合する場合は、最も高いセキュリティ・レベルが使用されます。

ノード・アクセス・グループには 2 つのタイプがあります。グループ・タイプは、そのグループのユーザーに割当て可能なデータ・アクセスのタイプを制御します。各ノード・アクセス・グループは、次のいずれかのグループ・タイプになります。

- 対話型--ユーザーには、割り当てられたアクセスのレベルに基づいて、データを参照、検索および編集する直接アクセス権があります
- ワークフロー--ユーザーには、割り当てられたアクセスのレベルに基づいて、データを参照、検索および編集する制限付きのアクセス権があります

表 1　対話型グループ・タイプ-ノード・アクセス・レベル

レベル	説明	使用例
読取り	読取り専用アクセスが可能--変更不可	表示とレポート
制限付き挿入	ユーザーがグローバルな挿入権限(以上)を持っているノードに対して挿入が可能。	挿入
編集	プロパティ値の編集が可能	編集
挿入	ノードの挿入、移動、削除が可能	編集、挿入、コピー、移動、削除

レベル	説明	使用例
非アクティブ化	ノードの非アクティブ化と再アクティブ化が可能	編集、挿入、移動、削除、非アクティブ化、再アクティブ化
追加	ノードの追加と削除が可能	編集、挿入、コピー、移動、削除、非アクティブ化、再アクティブ化、追加、削除

次の点に注意してください:

- アクセス・レベルは累積的です。「編集」アクセス・レベルを割り当てると、「読取り専用」と「制限付き挿入」のアクセス・レベルも暗黙的に付与されます。「追加」アクセス・レベルを割り当てると、他のすべてのアクセス・レベルが暗黙的に付与されます。
- ノード・アクセス・グループのセキュリティは、階層レベルでのみ適用されます。ノード・アクセス・グループは、孤立ノードなどグローバル・リストのノードへのアクセス権は制御しません。
- アクセス・レベルは、リム・ノードとリーフ・ノードには別々に割り当てられるため、それぞれに異なるレベルのアクセス権を定義できます。この機能は、階層のロールアップ構造を管理する必要はあるがリーフ・ノードのプロパティを編集しない場合、あるいは既存のロールアップ構造にリーフ・ノードを挿入できるが構造自体は再編成しない場合に便利です。
- ノード・アクセス・グループを定義できるのは、アクセス・マネージャ役割を持つユーザーのみです。
- ノード・アクセス・グループは、関連するノードにアクセス権を割り当てるとき、ローカル継承を使用します。制御階層に割り当てられるアクセス権のレベルに基づいてグローバル継承を使用するために、ノード・アクセス・グループをグローバルとして定義できます。
- グローバルなノード・アクセス・グループを作成できますが、各バージョンに制御階層を定義する必要があります。これには、制御されるノード・アクセス・グループを階層に割り当てます。詳細は、Oracle Data Relationship Management User's Guide を参照してください。
- 対話型およびワークフローのノード・アクセス・グループでは、階層におけるノードの可視性の処理が異なります。対話型のアクセス・グループでは、そのグループに階層内の任意のノードへのアクセス権がある場合、ユーザーは階層全体を表示できます。一方、ワークフローのアクセス・グループでは、ユーザーが表示できるのは、ユーザーにアクセス権が割り当てられている階層のノードのみに制限されます。どちらのグループ・タイプの場合でも、グループのメンバーは、アクセス権が割り当てられていない階層を表示することはできません。

ワークフロー・グループ・タイプのノード・アクセス・レベル

ワークフロー・ユーザーの役割を持つユーザーは、ワークフロー・ノード・アクセス・レベルを使用して、データへのアクセス権を決定します。

表 2　ワークフロー・グループ・タイプ-ノード・アクセス・レベル

レベル	説明
通知	ノードに対する変更要求の通知を有効にします
送信	変更要求の一部としてノードを送信できるようにします
承認	変更要求の一部としてノードを承認できるようにします
エンリッチ	変更要求の一部としてノードをエンリッチできるようにします
コミット	ノードに対する変更を Data Relationship Management にコミットできるようにします

注： ワークフロー・ノード・アクセス・レベルは累積的です; エンリッチ・アクセス・レベルを割り当てると、ユーザーは承認、送信および通知の受信が可能になります。

ノード・アクセス・グループの作成

➤ ノード・アクセス・グループを作成するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「新規」から「ノード・アクセス・グループ」を選択します。**

3 **グループの名前、ラベル、説明を入力します。**

注： ノード・アクセス・グループには、Custom ネームスペースが割り当てられます。グループの「完全修飾名」は一意である必要があります。「ラベル」フィールドは、名前を入力すると自動的に入力されます。ノード・アクセス・グループのラベルは、ユーザーにわかりやすい記述子です。アプリケーション管理を除くすべての機能で表示されます。便宜上、複数のノード・アクセス・グループが同じラベルを持つことは可能です。

4 **ノード・アクセス・グループの「グループのタイプ」を選択します。**

- 「対話型」--対話型アクセス・レベルを使用する場合; 「対話型ノード・アクセス・レベル」を参照してください。
- 「ワークフロー」--要求の送信、エンリッチメント、承認、コミットおよび通知のコンテキストで、バージョン、階層およびノードへのワークフロー指向のアクセスを使用する場合。ワークフロー・ノード・アクセス・レベルを参照してください。

5 **「オプション」: グループをグローバルなノード・アクセス・グループにする場合は「グローバル」を選択します。**

注： グローバルなノード・アクセス・グループには、そのグループが使用される各バージョンで、制御階層を定義する必要があります。作成したグループは、制御されるノード・アクセス・グループとして、バージョンごとの単一の階層に割り当てることができます。

6 グループに割り当てるユーザーを「使用可能」リストから選択します。矢印を使用して、ユーザーを「選択済」リストに移動します。

7 をクリックします。

ノード・アクセス・グループの編集

➤ ノード・アクセス・グループを編集するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ノード・アクセス・グループ」を展開します。

3 グループを選択してをクリックします。

4 グループに割り当てるユーザーを「使用可能」リストから選択します。矢印を使用して、ユーザーを「選択済」リストに移動します。

5 をクリックします。

ノード・アクセス・グループの削除

➤ ノード・アクセス・グループを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「ノード・アクセス・グループ」を展開します。

3 グループを選択してをクリックします。

4 「このアイテムを削除」をクリックし、削除を確定します。

注: ノード・アクセス・グループを削除すると、グループの割当てがユーザーからも階層ノードからも削除されます。

ノード・アクセス・グループのセキュリティの割当て

ノード・アクセス・グループのセキュリティは、データ・マネージャ役割を持つユーザーによってデータに適用されます。

注: ノード・アクセス・グループのセキュリティを割り当てる前に、適切なノード・アクセス・グループが作成され、適切なユーザーがそのグループに割り当てられていることを確認してください。

➤ ノード・アクセス・グループのセキュリティを設定するには:

1 バージョンと階層を開き、ノードを選択します。

2 「ノード」から「割当て」、「ノード・アクセス」の順に選択します。

3 「プロパティ・グリッド」で、「リーフ・アクセス」または「リム・アクセス」のカテゴリを選択します。

4 アクセス・レベルを、各ノード・アクセス・グループに割り当てます。

注： 各ノード・アクセス・グループに対する割当ての使用可能なアクセス権のレベルは、グループ・タイプ(対話側またはワークフロー)に基づきます。

5 「保存」をクリックします。

4 オブジェクト・アクセス・グループの管理

この章の内容

Data Relationship Management のオブジェクト・アクセス・グループは、ユーザーがアクセスできるメタデータ・オブジェクト(エクスポート、ブック、インポート、ブレンダ、比較、クエリー、バージョン変数および外部接続を含む)を決定します。

表 3　オブジェクト・アクセス・グループのタイプ

オブジェクト・アクセス・グループ・タイプ	説明	権限
ユーザー	各ユーザーには、個人メタデータ・オブジェクト用のコア・オブジェクト・アクセス・グループがあります。	ユーザーには、自身のオブジェクト・アクセス・グループに対する実行および管理権限があります。
標準	「標準」のコア・オブジェクト・アクセス・グループは、すべてのパブリック・オブジェクトで使用できます。	すべてのユーザーには、「標準」オブジェクト・アクセス・グループ内のオブジェクトに対する暗黙的な実行権限があります。 標準[オブジェクト]の管理役割の権限を持つユーザーのみに、「標準」オブジェクト・アクセス・グループの管理権限があります。
システム	「システム」のコア・オブジェクト・アクセス・グループは、システムの操作/統合のすべてのオブジェクトで使用できます。	データ・マネージャまたはアプリケーション管理者の役割を持つユーザーにのみ、システム・オブジェクト・アクセス・グループに対する管理権限があります。
カスタム	カスタム・オブジェクト・アクセス・グループ	カスタム・オブジェクト・アクセス・グループを作成、編集または削除できるのは、アクセス・マネージャの役割を持つユーザーのみです。実行権限を持つユーザーは、グループ内のオブジェクトを実行できます。

カスタム・オブジェクト・アクセス・グループを使用すると、特定のユーザー・グループに、ユーザー・メタデータ・オブジェクト(クエリー、比較、インポート、ブレンダ、エクスポートおよびブック)のサブネットへのアクセス権を付与できます。オブジェクト・アクセス・グループは、ユーザーおよびノード・アクセ

ス・グループのリストを定義し、ユーザーおよびノード・アクセス・グループごとに権限レベル(実行または管理)を設定します。メタデータ・オブジェクトは作成時にオブジェクト・アクセス・グループに割り当てられ、その後で別のグループにコピーまたは移動できます。

- 実行--ユーザーはグループ内のオブジェクトを実行できますが、オブジェクトを編集して、変更を保存することはできません
- 管理--ユーザーはグループ内でオブジェクトの作成、編集または削除を行い、それらを実行できます

オブジェクト・アクセス・グループを使用するためのガイドラインは次のとおりです:

- オブジェクト・アクセス・グループを使用すると、直接またはノード・アクセス・グループの割当てを介して、ユーザーがグループのメンバーになることができます。どちらも必要ではありません。
- ユーザーおよびノード・アクセス・グループは、複数のオブジェクト・アクセス・グループに割り当てることができます。
- オブジェクト・アクセス・グループの各ユーザーに、管理または実行権限のいずれかを割り当てます。
- オブジェクト・アクセス・グループでのユーザーの権限割当てによって、ユーザーの役割セキュリティを上書きできます。たとえば、オブジェクト・アクセス・グループで管理権限を持つ「インタラクティブ・ユーザー」役割は、オブジェクト・アクセス・グループ内のオブジェクトを作成または変更できます。
- ユーザー、標準、システムなどのコア・オブジェクト・アクセス・グループは、ユーザーの存在とその役割の割当てに基づいて暗黙的に管理されます。
- ユーザー・メタデータ・オブジェクトを保存またはコピーする場合、ユーザーは、自身が管理権限を持つオブジェクト・アクセス・グループにオブジェクトを割り当てる必要があります。
- ユーザー・メタデータ・オブジェクトは、1 つのオブジェクト・アクセス・グループにのみ割り当てることができます。
- データ・マネージャの役割を持つユーザーには、コア標準オブジェクト・アクセス・グループに対する暗黙的な管理権限があり、カスタム・オブジェクト・アクセス・グループに明示的に割り当てることができます。
- アプリケーション管理者の役割を持つユーザーには、標準、システムおよびカスタムのすべてのオブジェクト・アクセス・グループに対する暗黙的な管理権限があります。これらのユーザーは、任意のオブジェクト・アクセス・グループのメタデータ・オブジェクトを移行できる必要があります。

オブジェクト・アクセス・グループの作成

➤ カスタム・オブジェクト・アクセス・グループを作成するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「新規」から、「オブジェクト・アクセス・グループ」を選択します。**

3 グループの名前を入力します。説明はオプションです。

4 「ユーザー」タブで、「使用可能」リストからユーザーを選択して、グループに割り当てます。矢印を使用して、ユーザーを「選択済」リストに移動します。

注： デフォルトでは、各ユーザーには実行アクセス権が付与されます。ユーザーのアクセス権を変更するには、をクリックします。その後、「アクセス」から「管理」を選択します。

5 「ノード・アクセス・グループ」タブで、「使用可能」リストからノード・アクセス・グループを選択して、グループに割り当てます。矢印を使用して、ユーザーを「選択済」リストに移動します。

注： デフォルトでは、各ノード・アクセス・グループには実行アクセス権が付与されます。グループのアクセス権を変更するには、をクリックします。その後、「アクセス」から「管理」を選択します。

6 をクリックします。

オブジェクト・アクセス・グループの編集

➤ カスタム・オブジェクト・アクセス・グループを編集するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「オブジェクト・アクセス・グループ」を展開します。

3 グループを選択し、をクリックします。

4 「ユーザー」タブおよび「ノード・アクセス・グループ」タブで、選択したユーザーとグループ、およびアクセス権限を変更します。

5 をクリックします。

オブジェクト・アクセス・グループの削除

➤ オブジェクト・アクセス・グループを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「セキュリティ」の下で、「オブジェクト・アクセス・グループ」を展開します。

3 グループを選択し、をクリックします。

4 「このオブジェクト・アクセス・グループを削除」をクリックして、削除を確認します。

注意 オブジェクト・アクセス・グループを削除すると、それに割り当てられたすべてのメタデータ・オブジェクトも削除されます。この操作を元に戻すことはできません

5 ドメインの管理

この章の内容

ドメインは、同じ Data Relationship Management アプリケーション内で、ソースの異なる複数のノード・セットの参照整合性を管理するために使用されます。ドメインは共通タイプの登録済のノード・リストで、同じアプリケーション内でバージョンが異なるそれらのノードの一貫した管理が可能です。ドメインを利用すると、次のことが簡単になります。

- 一意性を保証するためにノード名を修飾
- バージョン間で識別プロパティを共有
- ノードに関する名前の変更、上位への移動、下位への移動、削除など特定タイプの変更を制限
- バージョンにかかわらずビジネス・ルールの一貫性を保つために検証を割当て

ドメイン・ノードは、ドメインのメンバーシップを持つ 1 つのバージョンにおけるグローバル・ノードです。ドメイン・ノードは名前を変更できますが、メンバーとして割り当てた後でドメインから削除することはできません。同じバージョンでドメインの異なるノードと併用するときの参照整合性を保証するために、ドメイン・ノードの名前は、ノードの自然な識別子でも、接頭辞または接尾辞で修飾されていてもかまいません。ドメイン・ノードの説明と、非アクティブのステータス/日付は、それが存在する任意のバージョンのドメイン・ノードによって共有されます。

ドメインの作成

➤ ドメインを作成するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「新規」から「ドメイン」を選択します。**

3 **次の情報を入力します:**

- 「名前」
- 「説明」(オプション)

- 「修飾子」(オプション)--ノード名の完全修飾に使用するテキスト。複数のドメインが同じ修飾子テキストを使用することはできません。修飾子の場所を示す「接頭辞」または「接尾辞」を選択します。

  注： ドメインにノードを割り当てた後で、修飾子テキストを変更することはできません。
- 「区切り文字」(オプション)--ドメイン修飾子テキストをノード名と区切るために使用する、オプションの 1 文字。

  注： ドメインにノードを割り当てた後で、区切り文字を変更することはできません。
- 「ノードの削除の許可」--ユーザーがバージョンからノードを削除できるようにする場合は、これを選択します。
- 「リーフの編集可能」--ドメインのノードについてリーフのシステム・プロパティ値をユーザーが変更できるようにする場合は、これを選択します。

4 **「使用可能な検証」リストから、ドメインのメンバーに適用されるノード・レベルの検証を選択し、「選択した検証」リストに移動します。**

注： ドメインレベルの検証を割り当てると、ノードに設定されていた、あるいは祖先ノード、階層またはバージョンのレベルの割当てから継承された同じ検証の割当て値は上書きされます。

5 **  をクリックします。**

ドメインの編集

ドメインは作成後に編集できますが、次の 2 つの例外があります。

- 名前は変更できません。
- ノードをドメインに割り当てた後で、修飾子と区切り文字を変更することはできません。

➤ ドメインを編集するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **ドメインを選択して  をクリックします。**

3 **ドメインを変更し、  をクリックします。**

ドメインの削除

ドメインは削除できます。ドメインを削除すると、ドメインのノード・レコードも削除されます。

注： ノードが割り当てられているドメインを削除すると、そのドメインに割り当てられているすべてのノードが非ドメイン・ノードに戻ります。

➤ ドメインを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 ドメインを選択し、  をクリックします。

3 「このドメインの削除」を選択します。

6 プロパティ・カテゴリの管理

この章の内容

プロパティ・カテゴリ

プロパティ・カテゴリは、Data Relationship Management のプロパティをグループ化する機能で、セキュリティ権限をプロパティのセットに割り当てる際に使用します。デフォルトで使用できるコアのプロパティは、1 つのプロパティ・カテゴリのみに収められています。アプリケーション管理者によって作成されるカスタム・プロパティには、複数のプロパティ・カテゴリを関連付けることができます。

Data Relationship Management にはコアのプロパティ・カテゴリが用意されています。次の表で説明します。

表 4　プロパティ・カテゴリ

カテゴリ	説明
システム	ID、名前、説明など、ノードの基本的な識別特性に関連するプロパティ。 このカテゴリで変更できるのは、個々のユーザーに対する読取り専用フラグの割当てのみです。読取りアクセス権しか持たないユーザーは、値を表示できるだけで編集はできません。このカテゴリにプロパティを割り当てることはできません。
共有情報	どのノードがプライマリか共有かに関する情報と、関連する共有ノードのリストを示し、プライマリ・ノードが欠落しているかどうかを示します。 システム・プリファレンスで「共有ノード」が有効な場合、このカテゴリは表示専用です。 **注：** このカテゴリのプロパティはすべて読取り専用です。
統計	子の数や兄弟の数などノードの統計情報を示すプロパティです。 **注：** このカテゴリのプロパティはすべて読取り専用です。
検証	ノードに割り当てられる検証。検証ごとに 1 つのプロパティです。
リーフ・アクセス	ノードのセキュリティ・グループと、ノードのリーフ・アクセス・レベル。グループごとに 1 つのプロパティです。
リム・アクセス	ノードのセキュリティ・グループと、ノードのリム・アクセス・レベル。グループごとに 1 つのプロパティです。

注： すべてのユーザーにすべてのプロパティ・カテゴリが表示されるとはかぎりません。ユーザー・アクセス権が特定のカテゴリに制限されている場合や、ノード・タイプがフィルタされている場合があるためです。「検証」、「リーフ・アクセス」および「リム・アクセス」のカテゴリは、データ・マネージャ役割を割り当てられているユーザー専用であり、検証またはノード・アクセス・グループのセキュリティを割り当てるときにしか利用できません。

プロパティ・カテゴリの作成

➤ プロパティ・カテゴリを作成するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「プロパティ・カテゴリ」を選択します。

3 プロパティ・カテゴリの名前と説明を入力します。

4 「プロパティ」タブで、プロパティ・カテゴリに割り当てるプロパティを「使用可能」リストから選択し、矢印を使用して「選択済」リストに移動します。

注： 「[Ctrl]」または「[Shift]」を押しながらクリックすると、複数のプロパティを選択できます。選択または選択解除するには、プロパティをダブルクリックします。

5 矢印を使用して、選択したプロパティの順序を変更するか、をクリックしてアルファベット順に並べます。

6 「ユーザー」タブで、プロパティ・カテゴリに割り当てるユーザーを「使用可能」リストから選択し、矢印を使用して「選択済」リストに移動します。

7 選択したリストでユーザーの行を選択し、「アクション」列でをクリックします。

8 「アクセス」列で「読取り」または「編集」を選択し、プロパティ・カテゴリへのアクセス権のレベルをユーザーに割り当てます。

9 「アクション」列で、変更を保存する場合は、変更を破棄する場合はをクリックします。

10 をクリックします。

プロパティ・カテゴリの編集

➤ プロパティ・カテゴリを編集するには:

1 「ホーム」ページで、「管理」を選択します。

2 プロパティ・カテゴリを選択してをクリックします。

3 「プロパティ」タブで、プロパティ・カテゴリに割り当てるプロパティを「使用可能」リストから選択し、矢印を使用して「選択済」リストに移動します。

注： 「[Ctrl]」または「[Shift]」を押しながらクリックすると、複数のプロパティを選択できます。選択または選択解除するには、プロパティをダブルクリックします。

4 矢印を使用して、選択したプロパティの順序を変更するか、  をクリックしてアルファベット順に並べます。

5 「ユーザー」タブで、プロパティ・カテゴリに割り当てるユーザーを「使用可能」リストから選択し、矢印を使用して「選択済」リストに移動します。

6 選択したリストでユーザーの行を選択し、「アクション」列で  をクリックします。

7 「アクセス」列で「読取り」または「編集」を選択し、プロパティ・カテゴリへのアクセス権のレベルをユーザーに割り当てます。

8 「アクション」列で、変更を保存する場合は  、変更を破棄する場合は  をクリックします。

9  をクリックします。

プロパティ・カテゴリの削除

➤ プロパティ・カテゴリを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「プロパティ・カテゴリ」を展開します。

3 プロパティ・カテゴリを選択して  をクリックします。

4 「このアイテムを削除」を選択し、削除を確認します。

注： プロパティ・カテゴリを削除しても、カテゴリに関連付けられているプロパティは削除されません。これらのプロパティは、引き続きアプリケーションで使用できます。

7 プロパティ定義の管理

この章の内容

プロパティ定義は、Data Relationship Management でバージョン、階層、ノードの属性を管理するために使用されます。プロパティには、テキスト、数値、日付、他のデータ・オブジェクトへの参照など様々なデータ型を格納できます。プロパティに明示的な値を格納し、継承を使用して自動的に値を子孫ノードに割り当てる、または式や参照テーブルに基づいて計算させることができます。プロパティ・カテゴリを使用してプロパティをグループ化し、関連するセットに編成すれば、その使用もユーザー・アクセス権の制御も簡単になります。

標準の製品機能では、デフォルトで使用できるシステム定義のプロパティが使用されます。アプリケーション管理者がユーザー定義のプロパティ定義を作成すると、ビジネスまたはシステム統合の要件をサポートする上で必要な追加の属性を管理できます。

Data Relationship Management におけるプロパティ定義のソースは様々です。たとえば、次のようなプロパティがあります。

- Data Relationship Management のシステム定義
- アプリケーション管理者が作成するユーザー定義のプロパティ
- 他の Oracle 製品で使用されるアプリケーション・テンプレートからロード
- 移行ユーティリティを使用する他の Data Relationship Management アプリケーションまたは環境からロード

ネームスペース

プロパティ定義では、ソースの異なるプロパティが類似の名前を持ち、データ整合性のために区別が必要な競合状態を回避するために、ネームスペースを使用します。プロパティ名は、ネームスペースを接頭辞にする規則で区別されます。

表 5　ネームスペースを使用したプロパティ定義の例

フィールド	例
完全修飾名	Custom.AccountType
ネームスペース	Custom

フィールド	例
名前	AccountType
ラベル	AccountType

Data Relationship Management には、競合を防ぐためにネームスペースに適用される特別なルールがあります。

- システム定義のプロパティは「Core」ネームスペースを使用します。
- ユーザー定義のプロパティは「Custom」ネームスペースを使用します。
- その他のネームスペースは、他の Oracle 製品用の Data Relationship Management アプリケーション・テンプレートで使用するために予約されています。

データ型

プロパティのデータ型を、次の表で説明します。

表 6　プロパティのデータ型

プロパティのデータ型	説明
関連グループ	関連ノード・グループ。複数のノードをポイントします。そのノードが「関連グループ」のノードを逆にポイント、またノード相互間でポイントします。アナロジー: 兄弟。 **注：** このデータ型は、グローバル・ノード・レベルのプロパティでのみ使用してください。 **注意** インポートによってロードされる関連ノードのプロパティは、ノードのインポート順に基づいてバージョンにまだ存在しないため、他のノードをすべて正しくポイントしないことがあります。
関連ノード	関連ノード。他の 1 つのノードをポイントします。そのノードが、「関連ノード」のノードを逆にポイントします。アナロジー: 婚姻。 **注：** このデータ型は、グローバル・ノード・レベルのプロパティでのみ使用してください。 **注意** インポートによってロードされる関連ノードのプロパティは、ノードのインポート順に基づいてバージョンにまだ存在しないため、他のノードをすべて正しくポイントしないことがあります。
関連ノード(複数)	関連ノードのリスト。複数のノードをポイントします。そのノードが「関連ノード(複数)」のノードを逆にポイント、またノード相互間でポイントします。アナロジー: 友人。 **注：** このデータ型は、グローバル・ノード・レベルのプロパティでのみ使用してください。 **注意** インポートによってロードされる関連ノードのプロパティは、ノードのインポート順に基づいてバージョンにまだ存在しないため、他のノードをすべて正しくポイントしないことがあります。
ブール	TRUE または FALSE
日付	日付 **注：** ユーザーのセッションに関連付けられた地域の設定に基づいてフォーマットされます。
日付/時間	日付と時刻。
浮動小数点	浮動小数点値 **注：** ユーザーのセッションに関連付けられた地域の設定に基づいてフォーマットされます。

プロパティのデータ型	説明
フォーマット済メモ	フォーマット済メモ: テキストのすべてのフォーマット(スペース、タブ、改行など)を維持します。フォーマット済メモにはハイパーリンク・テキストを含めることもできます。ハイパーリンクの URL のフォーマットの詳細は、「ハイパーリンク」データ型を参照してください。 **注：** プロパティ値でテキストとハイパーリンクの両方を使用するとき、URL 以外のテキストは抑止されます。
グローバル・ノード	バージョンのノードをポイントします。値が割り当てられる場合、プロパティ・グリッドの値フィールドにのみノード名が表示されます。
グループ	カンマで区切られたアイテム・リスト
階層	階層をポイントします。
階層グループ	階層グループをポイントします。 階層グループのプロパティを使用すると、表示するコンテキストに基づいて複数の方法で階層をグループ化できます。使用方法に基づいて、階層は同じバージョン内でも異なる形でグループ化できます。
ハイパーリンク	URL テキストのハイパーリンク機能を有効にします。URL 入力が複数の場合は、スペースを入れずキャリッジ・リターン+改行(CRLF、0x0D0A)で区切ります。入力された URL は、ナビゲート可能なハイパーリンクとして表示されます。このフィールドに URL 以外のテキストがあると抑止されます。解析済の区切られた URL、またはフォーマット済 URL のみが表示されます。URL は、次のフォーマットです。 [url=http_URL]URL_Title[/url] ここで、http_URL にハイパーリンク・テキストを指定し、URL_Title にはユーザーに表示されるテキストを指定します。 たとえば、次のようなマークアップになります。[url=http://support.oracle.com]Oracle Support[/url] は、プロパティ・グリッドを Oracle サポートとして表示します。
整数	整数値
リーフ・ノード	階層のリーフ・ノードをポイントします。値が割り当てられる場合、プロパティ・グリッドの値フィールドで階層名とノード名が表示されます。
リム・ノード	階層のリム・ノードをポイントします。値が割り当てられる場合、プロパティ・グリッドの値フィールドで階層名とノード名が表示されます。
グループのリスト	アイテムのチェック・リスト。複数のアイテムをリストから選択できます。
メモ	「メモ」フィールド: フォーマットは保存されず、データは 1 行のテキストにマージされます。メモのハイパーリンクも同様です。ハイパーリンクの URL のフォーマットの詳細は、「ハイパーリンク」データ型を参照してください。 **注：** プロパティ値でテキストとハイパーリンクの両方を使用するとき、URL 以外のテキストは抑止されます。
複数ノード	複数のノードをポイントします。
ノード	階層のノードをポイントします。値が割り当てられる場合、プロパティ・グリッドの値フィールドに階層名が表示されます。
ノード・プロパティ	ノードのプロパティをポイントします。

プロパティのデータ型	説明
プロパティ	プロパティをポイントします。
範囲リスト	値の範囲を定義します。使用できるのは整数値のみです。
ソート	ソートに使用される整数値
ソート・プロパティ	ソート・プロパティをポイントします。
標準クエリー	標準クエリーをポイントします。
文字列	文字列値
時間	時間 **注：** ユーザーのセッションに関連付けられた地域の設定に基づいてフォーマットされます。
バージョン	バージョンをポイントします。

プロパティの作成

詳細は、「階層制約の使用」を参照してください。

➤ プロパティ定義を作成するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「プロパティの定義」を選択します。

3 プロパティの名前を入力します。

注： プロパティは、Custom ネームスペースに割り当てられます。「完全修飾名」と「ラベル」のフィールドは、名前を入力すると自動的に入力されます。プロパティの「完全修飾名」は一意である必要があります。プロパティ・ラベルは、ユーザーにわかりやすい記述子です。アプリケーション管理を除くすべての機能のプロパティ定義で表示されます。ネームスペースが同じでなければ、複数のプロパティが同じラベルを使用できます。プロパティの「説明」はオプションで、プロパティ・エディタの下部に表示される長い記述子です。

4 プロパティのパラメータを次のように定義します。

注： パラメータのすべては示していません。表示されるパラメータは、選択したデータ型によって異なります。

- 「データ型」--プロパティのデータ型を参照してください

  次のいずれかのデータ型を選択して、ユーザーに表示するノードのリストを制限できます:「関連グループ」、「関連ノード」、「関連ノード(複数)」、「グローバル・ノード」、「リーフ・ノード」、「リム・ノード」、「複数ノード」または「ノード」。いずれかのデータ型を選択すると、「制約条件」タブが表示されます。

- 「プロパティ・レベル」--プロパティ定義のレベル:
  - **ローカル・ノード**--特定の階層にあるノードのプロパティ値が管理され、そのレベルでのみアクセスできます。
  - **グローバル・ノード**--バージョンのノードのプロパティ値が管理されますが、ローカル・ノードのレベルでもアクセスできます。
  - **階層**--階層のプロパティ値が管理されますが、ローカル・ノードのレベルでもアクセスできます。
  - **バージョン**--バージョンのプロパティ値が管理されますが、グローバル・ノードまたはローカル・ノードのレベルでもアクセスできます。

  注： グローバル・ノード継承のプロパティを定義する場合は、そのグローバル・プロパティの制御階層を定義する必要があります。これには、「ホーム」ページの「階層」タブで、制御されるプロパティを階層に割り当てます。
- 「プロパティ・タイプ」
  - 定義済--値がユーザーによって定義されて格納されます。
  - 検索--他のプロパティと参照テーブルに基づいて参照されます。
  - 導出--派生クラスを使用して計算します。

    注： スクリプト派生クラスを使用する派生プロパティは、バージョン、階層およびノードのプロパティに使用されます。式派生クラスは、グローバルまたはローカルのノード・プロパティにのみ使用されます。
- 「デフォルト値」--プロパティのデフォルト値
- 「ドメイン」--データ型が「ノード」、「リム・ノード」、「リーフ・ノード」、複数ノード、「関連ノード」、「関連ノード(複数)」、「関連グループ」(値として格納されているノードまたは複数のノードのすべてを表す)のプロパティの場合は、「ドメイン」ドロップダウン・リストを使用できます。ドロップダウンには、システムに定義されているすべてのドメインが含まれ、必要に応じて既存のドメインのいずれかを選択できます。
- 「列の幅」--プロパティ・タイプが「定義済」の場合の固定幅列の幅。
- 「最小値/最小長」--データ型に基づくプロパティの値または長さ。
- 「最大値/最大長」--データ型に基づくプロパティの値または長さ。

5 **次のオプションのいずれかを選択します。**

- 「リスト」--プロパティ値を、事前定義済の値リストのみから選択できます。

  注： リスト・プロパティに格納されるプロパティ値を、EnforceListProps システム・プリファレンスを使用するリストの値のみに制限できます。

注： 値のリストは、定義済プロパティまたは上書き可能な派生プロパティに対して使用できます。

- 「継承」--プロパティを継承として定義します

  注： このオプションを選択しても、「派生」プロパティ・タイプには影響しません。ただし、AncestorProp や DualAncestorProp のようなプロパティ派生が使用され、かつプロパティがグローバルである特別な場合は例外です。このような場合、プロパティは文字どおりに値を継承しませんが、「継承」オプションを使用して制御階層を指定できます。

- 「上書き可能」--プロパティ・グリッドでプロパティを上書きできます。

  注： このオプションが有効なのは、「派生」プロパティ・タイプの場合のみです。

- 「非表示」--プロパティ・グリッドでプロパティを非表示にします。

6 **次のいずれかの操作を行います。**

- カテゴリにプロパティを割り当てる場合は、「使用可能」リストからカテゴリを選択し、「選択済」リストに移動します。
- 「リスト」オプションとともに「定義済」プロパティ・タイプを選択した場合は、「リストの値」タブで次の手順を実行します。
  1. 「追加」をクリックして、リストに値を入力します。
  2. 行の「アクション」列で「保存」をクリックします。

  注： 各行で「移動」または「削除」を使用して、リスト値を順序変更または削除します。「編集」を使用するか行をダブルクリックして編集し、編集を取り消す場合には「取消し」を選択します。
- 「検索」プロパティ・タイプを選択した場合は、「参照テーブル」タブを選択して次の手順を実行します。
  1. 「追加」をクリックして、新しいキー/値ペアをリストに入力します。
  2. 行の「アクション」列で「保存」をクリックします。

  注： 各行で「移動」または「削除」を使用して、リスト値を順序変更または削除します。「編集」を使用するか行をダブルクリックして編集し、編集を取り消す場合には「取消し」を選択します。
- 階層制約を許可するデータ型を選択した場合、「制約条件」タブを選択し、次を実行します:
  1. **階層グループ・プロパティ**からプロパティを選択し、階層グループを選択します。

     ノード・セレクタでは、ユーザーは、選択した階層グループに属する階層からのノードのみを参照できます。
  2. **オプション:** Web クライアント、インポート、アクション・スクリプトまたは Web サービス API を介してプロパティが更新された場合、**「サー**

バー・プロパティの更新で制約を適用」を選択して、この制約を検証します。

- 派生プロパティ・タイプを選択した場合は、「パラメータ」タブを選択し、派生プロパティの式またはスクリプトを定義してください。

  式の詳細は、式の作成を参照してください。スクリプトの詳細は、動的スクリプトの作成を参照してください。

7  をクリックします。

階層制約の使用

階層制約を使用すると、ノード・データ型のプロパティ値を更新する際に、表示および選択の対象とする階層とノードを制限できます。階層制約は、ノード・データ型を使用するプロパティ定義のオプションの構成です。階層制約機能では、階層制約を割り当てる前に構成する必要がある階層グループおよび階層グループ・プロパティを使用します。

次のデータ型で、階層制約を使用できます:

- 関連グループ
- 関連ノード
- 関連ノード(複数)
- グローバル・ノード
- リーフ・ノード
- リム・ノード
- 複数ノード
- ノード

注： 関連グループ、関連ノードおよび関連ノード(複数)のノード・データ型の場合、関連ノードが相互参照を作成するため、階層制約の設定時にさらに考慮が必要なことがあります。階層制約を定義した場合、階層グループには、相互に関連付けられるすべての階層が含まれていることに注意する必要があります。例では、従業員と費用センターの階層内のノード間の相互参照を示しています。階層制約で使用するための階層グループ・プロパティおよび階層グループを別に作成する必要がある場合があります。

プロパティ定義の編集

➤ プロパティ定義を編集するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「メタデータ」の下で、「プロパティの定義」を展開します。**

3 **プロパティ定義のタイプに応じて、「コア」または「カスタム」を展開します。**

4 **プロパティをダブルクリックします。**

5 **編集可能なパラメータを変更します。**

注意 「プロパティ・タイプ」を定義済の値(「RWDerived」または「定義済」)から、格納されないタイプ(「派生」または「検索」)に変更すると、定義されているプロパティ値は削除され、このデータは失われます。このタイプの変更を行う前に、データが失われてもよいかどうか確認してください。

詳細は、52 ページの「プロパティの作成」を参照してください。

6 をクリックします。

プロパティの削除

➤ プロパティを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「プロパティの定義」を展開します。

3 プロパティを選択してをクリックします。

4 「プロパティ定義の削除」を選択し、削除を確定します。

注意 プロパティ定義を削除すると、そのプロパティに格納されているすべての値も削除され、またそれが使用されているすべてのメタデータ・オブジェクトのプロパティも削除されます。

8 検証の管理

この章の内容

検証を利用すると、バージョン、階層、ノードおよびプロパティに対してビジネス・ルールを適用できます。検証は、リアルタイムかバッチ、または両方のモードで実行できます。リアルタイム検証は変更の際に実行され、適用されているルールにアクションが違反する場合に変更を保存しないようにします。バッチ検証は編集を行う前または後に明示的に実行され、無効なため処理が必要なデータの条件を特定します。

検証クラス

検証クラスを使用すると、異なるタイプのビジネス・ルールを適用できます。汎用的に使用できる検証クラスと、特定の目的に使用する検証クラスがあります。既存の検証クラスのセットから、検証を作成できます。ノードに対するビジネス・ルールの多くは、そのロジックへのクエリーを使用する検証クラスで適用できます。そのため、分析の目的で作成されているクエリーを検証で利用し、データ整合性を管理することも可能です。バージョンおよび階層、またはノードの特殊ケースに関するルールは、他の検証クラスを使用して実行できます。検証クラスの一部は、製品テストの目的にのみ使用され、本番環境では使用されません。

表 7　検証クラス

検証クラス	レベル	説明	パラメータ
BoolNodeInHier	ノード	指定された階層で、指定されたブール・プロパティの値が True であることを検証します。	「プロパティ」、「階層」
ContainAllProp	グローバル・ノード	指定された階層に、指定されたプロパティが True であるノードがすべて含まれることを検証します。	「階層」、「プロパティ」
ContainAllWith	グローバル・ノード	指定された階層に、指定されたプロパティが指定された値であるノードがすべて含まれることを検証します。	「階層」、「プロパティ」、「値」

検証クラス	レベル	説明	パラメータ
CustPropQuery	ノード	事前定義済クエリーと予測される結果を使用して検証します。 ローカル・プロパティ・クエリーのみを使用できます。	「プロパティ」クエリー名、「失敗」値
DateRangeCheck	ノード	「開始日」が「終了日」より前であることを検証します。	「開始日プロパティ」、「終了日プロパティ」
式	ノード	式で表されるビジネス・ロジックを使用してノードを検証します。式の結果が False の場合は検証が失敗します。	式
GlobalPropQuery	グローバル・ノード	事前定義済クエリーと予測される結果を使用して検証します。	「プロパティ」クエリー名、「失敗」値
HierContainsRef	ノード	ブール・プロパティが True の場合、またはノードがリーフで 3 番目のブール・プロパティが True の場合は、階層にノードへの参照が含まれます。	「階層名」、すべてのノードのブール・プロパティ、リーフ・ノードのブール・プロパティ
HierFail	階層	テストのために、階層レベルで自動的に失敗します。	なし
InvalidNameLength	ノード	ノード名が指定された長さに等しくないことを検証します。	Length
MaxChildren	バージョン	ノード当たりの子の数が指定の制限を超えていないことを検証します。	子の最大数
MaxHierNodes	階層	階層内のノード数が指定の制限を超えていないことを検証します。	ノードの最大数
MaxVersionNodes	バージョン	バージョン内のノード数が指定の制限を超えていないことを検証します。	ノードの最大数
MergeEquiv	マージ	影響されるノードとマージ・ノードで、指定されたプロパティの値が同じであることを検証します。	「グローバル・ノード」プロパティ
MergePropSet	マージ	影響されるノードのプロパティ値が設定(上書き)されている場合に、指定されたプロパティに対してマージ・ノードのプロパティ値が指定されていることを検証します(プロパティ値が同じである必要はありません)。	プロパティ
MixedKids	ノード	リムおよびリーフの両方の子があるノードをチェックします。	なし
NoBoolBranch	ノード	指定されたブール・プロパティが、指定された分岐で少なくとも 1 回は True に設定されていることを検証します。	プロパティ
NodeFail	グローバル・ノード	テストのために、バージョン・レベルのノードが自動的に失敗します。	なし
NodeFailRandom	ノード	テストのために、指定された割合のノードで自動的に失敗します。	失敗の割合
NoDefaults	ノード	指定されたプロパティでデフォルト値が使用されないことを検証します。	プロパティ

検証クラス	レベル	説明	パラメータ
NoPropBranch	ノード	指定されたプロパティが、指定された分岐で少なくとも 1 回指定されていることを検証します。	プロパティ
PropEquivBool	ノード	3 番目のブール・プロパティ値が True の場合のプロパティ同等性。	評価するブール・プロパティ、1 番目のプロパティ、2 番目のプロパティ
PropLength	ノード	指定されたプロパティが最小の長さ以上、かつ最大の長さ以下であることを検証します。	「プロパティ」、「最小長」、最大長
PropRemove	除去	(prop1、prop2、prop3 のパラメータで)指定されたプロパティが指定された値(値 1、値 2、値 3 のパラメータ)に等しい場合にノードが削除されないようにします。	プロパティ 1、プロパティ 2、プロパティ 3、値 1、値 2、値 3
RequiredField	ノード	指定されたプロパティが指定された値を持つすべてのノードについて、必須リストの各プロパティが値を持つことを検証します。 ● 「デフォルトのレコードを拒否」フラグが True の場合は、必須リストの各プロパティがデフォルト以外の値を持つ必要があります。 ● 「デフォルトのレコードを拒否」フラグが False の場合は、デフォルト値が受け入れられます。	「プロパティ」、「値」、「デフォルトのレコードを拒否」、「必須プロパティ」
Script	ノード、階層、バージョン、マージ、グローバル・ノード、移動、削除、マージ	動的スクリプトを使用してデータを検証します。戻り値が True の場合は検証が成功します。戻り値が False の場合は検証が失敗します。	スクリプト
SingleBoolBranch	ノード	指定されたブール・プロパティが、分岐ごとに 1 回のみ True に設定されていることを検証します。	プロパティ
SinglePropBranch	ノード	指定されたプロパティが、分岐ごとに 1 回のみ設定されていることを検証します。	プロパティ
StrandedParent	ノード	すべてのリム・ノードに子があることを検証します。	なし
StrPropEqual	ノード	指定されたプロパティが指定された値に等しいノードのすべてで失敗します。	「プロパティ」、「値」
UniqueProp	ノード	指定されたプロパティで、階層内の値が重複していないことを検証します(「デフォルトを含む」フラグが False の場合、デフォルト値を持つノードは含まれません)。	「プロパティ」、「デフォルトを含む」
UniquePropBranch	ノード	指定されたプロパティの値が、分岐内で一意であることを検証します。	プロパティ
VersionFail	バージョン	テストのために、バージョン・レベルで自動的に失敗します。	なし

検証クラス	レベル	説明	パラメータ
VersionUnique2Prop	グローバル・ノード	指定されたプロパティで、バージョン内の値が重複していないことを検証します(「デフォルトを含む」フラグが False の場合、デフォルト値を持つノードは含まれません)。	第 1 プロパティ、第 2 プロパティ、「デフォルトを含む」
VersionUniqueProp	グローバル・ノード	指定されたプロパティで、バージョン内の値が重複していないことを検証します(「デフォルトを含む」フラグが False の場合、デフォルト値を持つノードは含まれません)。	「プロパティ」、「デフォルトを含む」

検証レベル

検証レベルは、ビジネス・ルールのスコープを定義します。ノード検証の場合、レベルには検証を実行するために実行する必要のあるアクションのタイプも含まれます。次の表に、各検証レベルの定義と次のことを示します。

- 検証をバッチ・モード、リアルタイム・モードまたはその両方のモードのいずれで実行できるか。
- 検証が割り当てられる場所。
- 検証の実行対象となるオブジェクト。

表 8　検証レベル

検証レベル	バッチまたはリアルタイムでの実行	割当て場所	実行対象
ノード--ノードの関係とプロパティを見直し、条件が満たされていることを確認します。 ノード・レベルの文字列プロパティ値の長さが有効かどうかを判定するために使用します。	リアルタイムまたはバッチ	バージョン、階層またはノード	ローカル・ノード
階層--階層のプロパティを見直し、条件が満たされていることを確認します。階層またはバージョンのレベルで割当てと実行が可能です。 階層でノード数が 10,000 を超えていないことを確認します。	バッチ	バージョンまたは階層	階層
バージョン--バージョンのプロパティを確認します。 バージョンでノード数が 100,000 を超えていないことを確認します。	バッチ	バージョン	バージョン
グローバル・ノード--バージョン・レベルで割り当てられます。階層にかかわらず、孤立ノードも含めてバージョンのすべてのノードを検証します。グローバルとして定義されているプロパティのみが確認されます。 バージョン内のすべてのノードのプロパティ値が一意であることを確認します。	バッチ	バージョン	グローバル・ノード
マージ--マージの必要な操作(削除または非アクティブ化など)を実行するときに実行されます。バージョン・レベルで割り当てられます。 リーフ・ノードが他のリーフ・ノードにのみマージされるようにします。	リアルタイム	バージョン	グローバル・ノード
移動--ノードの移動を試みたときに検証がトリガーされます。階層レベルで割り当てられます。 階層内でコスト・センターが移動されないようにします。	リアルタイム	階層	ローカル・ノード

検証レベル	バッチまたはリアルタイムでの実行	割当て場所	実行対象
削除--移動レベルと類似しています。ノードを階層さら移動または削除しようとしたときに実行されます。これを使用して、指定したタイプのノードを削除されないようにできます。 階層内でコスト・センター・ノードが削除されないようにします。	リアルタイム	バージョンまたは階層	グローバル・ノード

検証の作成

➤ 検証を作成するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「検証」を選択します。

3 検証の名前を入力します。

注： 検証は、Custom ネームスペースに割り当てられます。検証の「完全修飾名」は一意である必要があります。「ラベル」フィールドは、名前を入力すると自動的に入力されます。検証ラベルは、ユーザーにわかりやすい記述子です。アプリケーション管理を除くすべての機能の検証で表示されます。ネームスペースが同じでなければ、複数の検証が同じラベルを使用できます。

4 検証に失敗した場合にユーザーに表示されるメッセージを入力します。

5 検証クラスを選択します。「検証クラス」を参照してください。

注： 有効なレベルが、選択したクラスに応じて移入されます。

6 ノード・レベルでリアルタイムに実行できるクラスの場合は、アクションのタイプを含むレベルを選択します。

7 検証のオプションを次の中から選択します。

- リアルタイム--変更時に実行されます
- バッチ--明示的に要求された場合に実行されます
- 継承--選択したノードとその子孫に対して実行されます

注： 選択する検証クラスによっては、オプションの一部を使用できない場合や、値の編集が必要なパラメータが表示される場合があります。

8 選択した検証クラスにパラメータを定義します。

各検証クラスのパラメータは、検証クラスを参照してください。式の作成の詳細は、式の作成を参照してください。スクリプト作成の詳細は、動的スクリプトの作成を参照してください。

9 をクリックします。

検証の割当て

検証を作成すると、バージョン、階層、ドメイン、ノードに割り当てることができます。複数の検証を同時に割り当てられます。

注： ドメイン・レベルで割り当てる場合、検証はそのドメインのメンバーであるすべてのノードによって継承されます。バージョン・レベルで割り当てる場合、検証はそのバージョン内のすべての階層とノードによって継承されます。階層レベルで割り当てる場合、検証はその階層内のすべてのノードによって継承されます。

検証をドメインに割り当てる方法の詳細は、第 5 章「ドメインの管理」を参照してください。検証をバージョン、階層およびノードに割り当てる方法の詳細は、Oracle Data Relationship Management User's Guide を参照してください。

検証の編集

➤ 検証を編集するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「検証」を展開します。

3 検証を選択して  をクリックします。

4 検証を変更します。

注： 操作の「クラス」、「レベル」、「モード」の各パラメータは、検証を保存した後では変更できません。

5 「保存」をクリックします。

検証の削除

検証を削除すると、バージョン、階層、ノードに対する検証の割当てもすべて削除されます。

➤ 検証を削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「検証」を展開します。

3 検証を選択して  をクリックします。

4 「このアイテムを削除」を選択し、削除を確認します。

9 式の管理

この章の内容

式を使用すると、Data Relationship Management ネイティブの式言語を使用して、派生プロパティと検証の複雑なロジックを定義できます。式は関数と文字列リテラルで構成され、特定の構文ルールに従う必要があります。

詳細は、次を参照してください:

- プロパティの作成
- 検証の管理

関数の操作

関数名は大文字小文字が区別されず、パラメータが必要かどうかにかかわらず直後にはカッコが必要です。

詳細は、次を参照してください:

- 64 ページの「特殊文字」
- 64 ページの「リテラル」
- 72 ページの「関数の定義」
- 64 ページの「フォーマット文字列のパラメータ」
- 66 ページの「日時フォーマット文字列」

関数のパラメータは、所定のタイプと数である必要があります。パラメータには、ネストした関数や文字列リテラルを使用できます。パラメータのタイプが正しくない場合は、エラーが報告されます。パラメータが少なすぎる場合は、リスト・インデックスが範囲外というエラーが報告されます。パラメータが多すぎる場合は、余分なパラメータが無視されます。

特殊文字

パラメータ値に特殊文字(カンマ、スペース、タブなど)を含む一部の関数では、大カッコ([])を使用します。たとえば `FlipList(PropValue(Custom.NodeList),[comma])`は、関数呼出し `PropValue(Custom.NodeList)`から返されるカンマ区切りのリストに対して FlipList リスト関数を実行します。

`ArrayCount`、`ArrayIndex`、`ArrayItem`、`FlipList`、`Intersection`、`ListContains`、`PadList`、`RangeListContains`、`IsRangeListSubset`、`MinList`、`MaxList`、`AvgList`、`SumList`、`SortList`、`ListDistinct`、`ListNodePropValues`、`ListNodesWith` の各関数では、区切りパラメータとして `comma`、`space`、`tab` を大カッコ([])に指定できます。

`ReplaceStr` 関数では、置換前と置換後のパターンを表すパラメータが必要ですが、通常のテキスト文字列に加えて、`comma`、`space`、`tab`、`crlf`、`cr`、`lf`、`openparen`、`closeparen` を大カッコ([])に指定できます。

注： パラメータ値にカンマをそのまま(,)使用すると、「パラメータの数が無効です。」という構文エラーになります。関数呼出しの結果として渡されるカンマ区切りのリストは有効であり、想定どおりに処理されます。例:

無効な構文: `FlipList(a,b,c,[comma])`

有効な構文: `FlipList(PropValue(Custom.NodeList),[comma])` where `Custom.NodeList` value = a,b,c

リテラル

有効な関数名ではない値の後にカッコを続けると、リテラルとみなされます。リテラルとみなされるのは、文字列、整数、浮動小数点数またはブール・リテラルです。文字列リテラルの場合、スペースは文字として扱われます。したがって、適切な結果を導出するために必要な場合を除いて、余分なスペースは使用しないでください。「スペースの除去」オプションを使用すると、保存する前に式からスペースを削除できます。

フォーマット文字列のパラメータ

文字列のフォーマット・ルーチンに渡されるフォーマット文字列には、リテラル文字列とフォーマット指定子の 2 種類のオブジェクトが含まれます。リテラル文字列は、結果の文字列にすべてそのままコピーされます。フォーマット指定子は、指定されたプロパティからプロパティ値を取得し、フォーマットをそれに適用します。フォーマット文字列に含めることができる指定子は 1 つのみです。

フォーマット指定子には、次のフォームを使用します。

```
"%" [ "-" ][width][ "." prec]type
```

文字	説明
%	フォーマット指定子の開始を表します。

文字	説明
["-"]	左揃えのインジケータ(オプション) 値の後に空白を追加して、結果を左揃えします。デフォルトでは、値の前に空白を追加して結果を右揃えします。
[width]	幅の指定子(オプション) 変換の最小フィールド幅を設定します。結果の文字列が最小フィールド幅より短い場合は、空白を埋め込んでフィールド幅が広げられます。
["."prec]	精度指定子(オプション)
type	変換タイプ文字 変換文字は、大文字または小文字で指定できます。浮動小数点形式の場合、小数点と桁区切りに使用される実際の文字は、DecimalSeparator と ThousandSeparator のグローバル変数、またはそれに相当する TFormatSettings から取得されます。type の有効な値は、次の表のとおりです。

タイプ値	説明
d	小数 プロパティ値は整数で指定する必要があります。値は 10 進法の文字列に変換されます。フォーマット文字列に精度指定子が含まれている場合は、指定された桁数以上の数字が結果の文字列に含まれる必要があります。値の数字がそれより少ない場合は、結果の文字列の左側にゼロで追加されます。
u	符号なし 10 進数 d とほぼ同じですが符号が出力されません。
e	指数 プロパティ値は浮動小数点値で指定する必要があります。値は「-d.ddd...E+ddd」という形式の文字列に変換されます。負の数の場合は、結果の文字列がマイナス符号で始まります。小数点の前に常に 1 桁があります。結果の文字列における合計の桁数(小数点の前の 1 桁も含む)は、フォーマット文字列の精度指定子で指定します。精度指定子がない場合は、デフォルトの精度 15 が使用されます。結果の文字列における指数文字「E」の後には、必ずプラスまたはマイナス符号と 3 桁以上が続きます。
f	固定 プロパティ値は浮動小数点値で指定する必要があります。値は「-ddd.ddd...」という文字列に変換されます。負の数の場合は、結果の文字列がマイナス符号で始まります。小数点以下の桁数は、フォーマット文字列の精度指定子で指定します。精度指定子がない場合は、小数点以下はデフォルトで 2 桁となります。
g	全般 プロパティ値は浮動小数点値で指定する必要があります。値は、固定または指数形式を使用してできるかぎり短い小数文字列に変換されます。結果の文字列における有効桁数は、フォーマット文字列の精度指定子で指定します。精度指定子がない場合は、デフォルトの精度 15 が使用されます。後続のゼロは結果文字列から削除され、小数点は必要な場合にのみ表示されます。値の小数点の左にある桁数が、指定した精度以下の場合と、値が 0.00001 以上の場合、結果の文字列には固定小数点形式が使用されます。それ以外の場合、結果の文字列には指数形式が使用されます。
n	数値 プロパティ値は浮動小数点値で指定する必要があります。値は「-d,ddd,ddd.ddd...」という文字列に変換されます。n 形式は f 形式に相当しますが、結果の文字列に桁区切りが含まれる点のみが例外です。

タイプ値	説明
m	通貨 プロパティ値は浮動小数点値で指定する必要があります。値は通貨額を表す文字列に変換されます。変換は CurrencyString、CurrencyFormat、NegCurrFormat、ThousandSeparator、DecimalSeparator、CurrencyDecimals のグローバル変数、またはそれに相当する TFormatSettings データ構造で制御されます。フォーマット文字列に精度指定子が含まれている場合は、CurrencyDecimals グローバル変数またはそれに相当する TFormatSettings によって指定される値より優先されます。
s	String プロパティ値は文字、文字列、または PChar 値で指定する必要があります。文字列または文字は、フォーマット指定子のかわりに挿入されます。フォーマット文字列に精度指定子が存在する場合は、結果の文字列の最大長が指定されます。プロパティ値の文字列がこの最大長を超える場合、文字列は切り捨てられます。
x	16 進数 プロパティ値は整数値で指定する必要があります。値は 16 進法の文字列に変換されます。フォーマット文字列に精度指定子が含まれている場合は、指定された桁数以上の数字が結果の文字列に含まれる必要があります。値の数字がそれより少ない場合は、結果の文字列の左側にゼロで追加されます。

日時フォーマット文字列

日時フォーマット文字列には、日時の値(TDateTime)が文字列に変換されるときのフォーマットを指定します。日時フォーマット文字列は、フォーマット文字列に挿入される値を表す指定子で構成されます。一部の指定子(d など)は、数字または文字列をフォーマットします。その他の指定子(/など)は、グローバル変数からのロケール固有の文字列を表します。指定子の大文字小文字はフォーマットで無視されますが、am/pm と a/p の指定子は例外です。

指定子	表示
c	日付とそれに続く時刻 **注：** 日時の値が深夜ちょうどを示す場合、時刻は表示されません。
d	先行するゼロを伴わない数字で表される日付(1-31)
dd	先行するゼロを伴う数字で表される日付(01-31)
ddd	省略表記の曜日(Sun-Sat)
dddd	完全表記の曜日(Sunday-Saturday)
ddddd	短い形式の日付
dddddd	長い形式の日付
e	先行するゼロを伴わない数字で表される現在の元号(ロケールが日本、韓国、台湾の場合のみ)
ee	先行するゼロを伴う数字で表される現在の元号(ロケールが日本、韓国、台湾の場合のみ)
g	省略表記の元号(ロケールが日本と台湾の場合のみ)

指定子	表示
gg	完全表記の元号(ロケールが日本と台湾の場合のみ)
m	先行するゼロを伴わない数字で表される月(1-12) **注意** m 指定子を h または hh 指定子の直後に指定した場合は、月ではなく分が表示されます。
mm	先行するゼロを伴う数字で表される月(01-12) **注意** mm 指定子を h または hh 指定子の直後に指定した場合は、月ではなく分が表示されます。
mmm	省略表記の月の名前(Jan-Dec)
mmmm	完全表記の月の名前(January-December)
yy	2 桁で表される年(00-99)
yyyy	4 桁で表される年(0000-9999)
h	先行するゼロを伴わない時(0-23)
hh	先行するゼロを伴う時(00-23)
n	先行するゼロを伴わない分(0-59)
nn	先行するゼロを伴う分(00-59)
s	先行するゼロを伴わない秒(0-59)
ss	先行するゼロを伴う秒(00-59)
z	先行するゼロを伴わないミリ秒(0-999)
zzz	先行するゼロを伴うミリ秒(000-999)
t	ShortTimeFormat グローバル変数で指定された形式を使用する時刻
tt	LongTimeFormat グローバル変数で指定された形式を使用する時刻
am/pm	先行する h または hh 指定子について 12 時間制を使用し、正午前の時刻には「am」、正午より後の時刻には「pm」を表示します。am/pm 指定子には小文字、大文字、大小文字の混在を使用でき、結果は指定したとおりに表示されます。
a/p	先行する h または hh 指定子について 12 時間制を使用し、正午前の時刻には「a」、正午より後の時刻には「p」を表示します。a/p 指定子には小文字、大文字、大小文字の混在を使用でき、結果は指定したとおりに表示されます。
ampm	先行する h または hh 指定子について 12 時間制を使用します。
/	地域の設定で指定される日付の区切り文字
:	地域の設定で指定される時刻の区切り文字
'xx'/"xx"	一重または二重の引用符で囲んだ文字はそのまま表示され、フォーマットに影響しません。

式の評価

プロパティ定義や検証を作成または変更するとき、式をテストできます。式は、指定されたプロパティ値を使用して評価され、式の結果が計算されます。このプロセスで、単純な構文検証では見つからないロジック上または実装上のエラーが式に見つかる場合があります。式の結果が表示され、式のエラーやステータス・メッセージがある場合には表示されます。

式は左から右に評価され、関数と文字列リテラルの評価は出現順に実行されます。この方法に従い、ネストした関数が評価されてから、その右に表示される追加のパラメータが評価されます。関数は、式で明示的にネストすることもできる他に、他の式のプロパティから値を取得することによって暗黙的にネストされる場合もあります。循環参照(明示的または暗黙的にプロパティ自身を参照するプロパティ式)は、原則として避けるようにしてください。問題の原因になる循環参照は Data Relationship Management によって検出され、防止されますが、必要があり十分に理解されている場合を除いて避ける必要があります。

詳細は、次を参照してください:

- 式の構文チェック
- 構文チェックでのプロパティ名

式の構文チェック

保存する前に、次の点について式の構文が検証されます。

- 関数名が正しいです。
- プロパティ名が正しいです。
- 開きカッコと閉じカッコの数が同じです。
- 関数ごとに、実際のパラメータ数が所定のパラメータ数以上あります。

`Concat` などの関数の場合、パラメータ数に指定はありません。パラメータ数の検証では、実際のパラメータ数が所定のパラメータ数と同じかそれ以上であることを確認します。したがって、パラメータ数が多い場合にはエラーにならず、少ない場合にのみエラーになります。

構文の検証で式は評価されないため、無効な定数を入力した場合にはエラーが発生する可能性があります。たとえば、`IntToStr(ABC,3)` は構文の検証にパスしますが、Data Relationship Management アプリケーションでエラーが生成されます。このタイプのエラーを回避するには、保存する前にそれぞれの式を評価する必要があります。

構文チェックでのプロパティ名

プロパティ名に関する構文の検証を正確に実行するために、プロパティ名がリテラルでなく関数の結果であるまれな場合には、プロパティ名を必要とする関数が部分的に評価されます。

次の例を確認してください。

- 式 `PropValue(Concat(Core.Abbrev))`は有効ですが、プロパティ名を検証するには `Concat` 関数の評価が必要です(構文の検証のみではなく)。
- 式 `PropValue(If(NodeIsLeaf(),Core.Abbrev,Custom.Label))`は有効ですが、プロパティ名を検証するには If 関数の評価が必要です。

問題のプロパティ名が式の一部でしかない場合、プロパティ名の決定に必要な部分のみが評価されます。たとえば、式 `Add(PropValue(Concat(Core.,I,D)),If(NodeIsLeaf(),0,1)`の場合、構文検証のために評価される式の部分は `Concat` 関数とそのパラメータのみです。

このような式の一部が評価されるという事実は、`PropValue(PropValue(NodeType))`などの場合に重要です。この式では、Custom.NodeType プロパティに値を指定しないかぎり構文検証は失敗します。

式の使用に関する考慮事項

次の項では、式を使用する際にさらに考慮が必要な点について説明します:

- データ型の変換
- プロパティ・レベルの制限
- 他のノードからのプロパティの参照
- グローバル・ノード・プロパティからローカル・ノード・プロパティを参照
- 関数のネスト
- プロパティを他のプロパティの変数として使用
- 再帰を使用した階層関係の走査

データ型の変換

一部の関数では、特定のデータ型のデータ値を適切に評価する必要があります。たとえば、数学的な計算を実行する関数では入力引数が整数値または浮動小数点値の必要があり、文字列を操作する関数では文字列値を指定する必要があります。場合によっては、データ値の型を変換して正しく導出する必要があります。Data Relationship Management には、式でデータ型の変換を処理する一連の関数が用意されています。

プロパティ・レベルの制限

通常、粒度の低いレベルでデータを管理するために作成されるプロパティ定義は、それより粒度の高いレベルでデータを管理する他のプロパティを参照できます。

- ローカル・ノード--他のローカル・ノード、グローバル・ノード、階層またはバージョン・プロパティを参照できます
- グローバル・ノード--他のグローバル・ノードまたはバージョン・プロパティを参照できます
- 階層--他の階層またはバージョン・プロパティを参照できます(検索のみ)

- バージョン--他のバージョン・プロパティを参照できます(検索のみ)

他のノードからのプロパティの参照

派生プロパティまたは検証が、式を計算中である現在のノードとは異なるノードからのプロパティ値を評価または取得することは普通です。Data Relationship Management には、同じバージョン内のノードのプロパティ値にアクセスできる機能があります。

- NodePropValue
- ParentPropValue
- HierNodePropValue
- AncestorProp
- DualAncestorProp
- AscNodeProp
- ReplacePropValue
- ListPropValues
- ListNodePropValues

グローバル・ノード・プロパティからローカル・ノード・プロパティを参照

グローバル・ノード・プロパティは、値を返すときに階層コンテキストを必要としませんが、ローカル・ノード・プロパティには階層を指定する必要があります。グローバル・ノードに対して計算される派生プロパティまたは検証が、標準の PropValue または NodePropValue 関数を使用してローカル・ノード・プロパティ値を参照することはできません。グローバル・ノード・プロパティはローカル・ノード・プロパティを参照できますが、そのとき使用する HierNodePropValue 関数で特定の階層を指定し、その階層におけるローカル・ノードのプロパティ値を取得する必要があります。

関数のネスト

関数を同じ式に組み合せることを、関数のネストと呼びます。同じ式の中で、1 つの関数の出力が別の関数の入力引数として使用されます。ネストした関数を評価するとき、Data Relationship Management は最も内側の関数を最初に実行し、実行を順に外側へと移していきます。関数は、同じ式の中で明示的にネストする場合も、異なる式を使用するプロパティを参照する 1 つの式によって暗黙的にネストされる場合もあります。

プロパティを他のプロパティの変数として使用

Data Relationship Management では、ネストした関数の組合せを使用して他のプロパティまたはノードとリテラル値を参照できるため、結果として長く複雑な式に

なることがあります。別のプロパティ定義を使用して式のロジックをモジュール化すれば、同じ結果を得るために必要な式の構文を単純化できます。この方法を利用すれば、式の管理は大幅に簡単になります。

また、同じプロパティ定義内で、あるいは特定のノードに対する複数のプロパティ定義にまたがって、式が同じデータを評価したり、何度も同じ計算を実行することが可能です。このロジックをはるかに大きい式に埋め込む、またはプロパティ定義内で実装すると、このような評価と計算が複数回実行されるため、プロパティの計算を必要とする操作のパフォーマンスに影響することがあります。別々のプロパティ定義の中で重複する式ロジックを切り分け、冗長な処理を最小限に抑えることができます。

再帰を使用した階層関係の走査

低い階層レベルにおけるノードのビジネス・ルールが、それより上位にある祖先ノードのプロパティ値の評価を必要とする場合があります。このように低いレベルのノードからプロパティ値を参照する方法の 1 つとして、参照すべき値を管理する継承をプロパティ定義で有効にすることが考えられます。ただし、多くの場合、プロパティ定義に継承を使用する方法は適切ではありません。

現在のプロパティ定義を自己参照する特定の階層式関数を使用して、階層の分岐まで上位方向に再帰すれば、祖先ノードのプロパティ値を取得または評価できます。

ParentPropValue--現在の階層で祖先の分岐まで上位方向に再帰するには、この関数を使用します。例: `If(Equals(Integer,PropValue(Core.Level),1),Label Only,ParentPropValue(Essbase.DataStorage))`

HierNodePropValue--別の階層で祖先の分岐まで上位方向に再帰するには、この関数を使用します。例:

```
If(Equals(Boolean,PropValue(Custom.PlanPoint),True),Abbrev(),HierNodePropValue(Geography,HierNodePropValue(Geography,Abbrev(),Core.Parent),Custom.PlanMember))
```

式の作成

式は、派生プロパティの定義と検証の作成または編集を行うための「パラメータ」タブからアクセス可能な式エディタで作成します。

➤ 式を作成するには:

1 **テキストの式を入力するか、「パラメータ」タブで、次のようにして関数およびプロパティを挿入します:**

- 関数を挿入するには、式にカーソルを置いて「関数の挿入」をクリックします。関数のリストが表示されます。関数を展開して入力パラメータを表示します。パラメータ値を入力し、「OK」をクリックします。
- プロパティを挿入するには、式にカーソルを置いて「プロパティの挿入」をクリックします。プロパティのリストが表示されます。プロパティを選択して「OK」をクリックします。

2 **次のオプションから選択します:**

- 「スペースの除去」--デフォルトで選択されています。選択されている場合、式が評価されるときとプロパティが保存されるときに、式のスペースがすべて削除されます。式でリテラル値として評価するために空白を保持するには、このオプションを無効化します。
- 式を評価するには、次のオプションを選択します。
  - 「選択したノードで評価」--[…]をクリックし、ノードを選択します。式ではノードの現在のプロパティ値が使用されます。「評価」をクリックします。結果は、式デザイナの下部に表示されます。
  - 「スクラッチ・パッドで評価」--プロパティ値を手動で入力します。ノードから値をコピーし、評価のために変更することもできます。「ノードからコピー」で[…]をクリックし、プロパティ値を表示するノードをグリッドから選択します。列見出しの下にあるフィルタ行を使用して、プロパティのリストをフィルタします。「アクション」列の「編集」ボタンを使用し、式の評価に必要なプロパティ値を変更します。「評価」をクリックします。評価の結果は、式デザイナの下部に表示されます。

3 式をテストするには、「評価」をクリックします。

関数の定義

派生式プロパティの定義で使用できる関数のリストを、アルファベット順に示します。

Abbrev

説明

現在のノードの名前(略称)を戻します。

構文

```
Abbrev(): String
```

例

```
Abbrev()
```

戻り値はノードの名前です。

追加

説明

指定された 2 つの整数値を加算し、結果を戻します。

構文

```
Add(Int1,Int2:Integer):Integer
```

例

```
Add(1,4)
```

戻り値は 5 です。

AddedBy

説明

「追加者」の変更追跡プロパティの値を戻します。

構文

```
AddedBy():String
```

例

```
AddedBy()
```

現在のノードをバージョンに追加したユーザーの名前を戻します。

AddedOn

説明

「追加日」の変更追跡プロパティの値を日付/時刻として戻します。

構文

```
AddedOn():Date/Time
```

例

```
AddedOn()
```

現在のノードがバージョンに追加された日時を戻します。

AddFloat

説明

指定された 2 つの浮動小数点値を加算し、結果を戻します。

構文

```
AddFloat(Float1,Float2:Float):Float
```

例

```
AddFloat(2.14,3.75)
```

戻り値は 5.89 です。

AncestorProp

説明

プロパティが指定された値に等しい最初の祖先のプロパティ値を戻します。

注： 現在のノードが基準に対して有効である場合に戻されます。

構文

```
AncestorProp(Operator:String,Property:String,Value:String,FromTop:Boolean,ReturnProp:String)
```

「Operator」は、値を含むプロパティを比較する際に使用する演算子です。有効な値: =、<、>、>=および<=。

「Property」は、使用するプロパティの名前です。

「Value」は、比較する値です。

「FromTop」は、階層の最上位ノードから検索するかどうかを指定します。False の場合、検索は現在のノードから実行されます。

「ReturnProp」は、戻すプロパティの名前です。

And

説明

指定されたすべてのブール式が True と評価された場合に True を戻します。

構文

```
And(Expression1,Expression2,...ExpressionN:Boolean):Boolean
```

例

```
And(1,T,True)
```

戻り値は True です。

ArrayCount

説明

指定されたリスト(配列)のアイテム数を戻します。

構文

```
ArrayCount(List:String,Delimiter:String):Integer
```

「List」は、その内容を検索する文字列のリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
ArrayCount(Diet Cola;Root Beer;Cola,[comma])
```

戻り値は 3 です。

ArrayIndex

説明

リスト(配列)内で、指定されたアイテムが出現する最初の位置を戻します。アイテムが見つからない場合はゼロ(0)を戻します。

構文

```
ArrayIndex(Item:String,List:String,Delimiter:String):Integer
```

「Item」は、テストする文字列値を指定します。

「List」は、その内容を検索する文字列のリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
ArrayIndex(Cola,Diet Cola;Root Beer;Cola,[comma])
```

戻り値は 3 です。

ArrayItem

説明

リスト(配列)内の指定されたインデックス位置にあるアイテムを戻します。

構文

```
ArrayItem(List:String,Delimiter:String,Index:Integer):String
```

「List」は、その内容を検索する文字列のリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「Index」は、リスト内での文字列の位置です。負の値はリスト内の最後のアイテムを示します。

例

```
ArrayItem(Diet Cola;Root Beer;Cola,[comma],3)
```

戻り値は Cola です。

AscNodeProp

説明

指定されたプロパティで参照される関連ノードのプロパティ値を戻します。

構文

```
AscNodeProp(LookUpProperty,ReturnProperty)
```

「LookupProperty」は、ノードをポイントするプロパティの名前です。プロパティのデータ型は Node または AscNode のどちらかである必要があります。

「ReturnProperty」は、戻す関連ノードのプロパティの名前です。プロパティはグローバルである必要があります。

AvgList

説明

空白アイテムは無視して、リスト内のアイテムの平均を戻します。指定されたタイプではないアイテムがリストに含まれている場合は、空白の文字列を戻します。

構文

```
AvgList(InputList:String,Delimiter:String,ItemType:String):String
```

「InputList」は、使用するリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「ItemType」は、リスト・メンバーに予期するアイテム・データ型を示します。有効な値: integer、float および datetime。デフォルト値は float です。

例

```
AvgList(1;2;3,[comma],Integer)
```

戻り値は 2 です。

BoolToStr

説明

True または False に変換されたブール値を戻します。入力がブール値を表さない場合は、False を戻します。

構文

```
BoolToStr(Expression:Boolean):String
```

例

```
BoolToStr(1)
```

戻り値は True です。

Changed

説明

「ノード変更」の変更追跡プロパティの値をブールとして戻します。

構文

```
Changed()
```

ChangedBy

説明

バージョンで現在のノードを最終更新したユーザーの名前を戻します。

構文

```
ChangedBy():String
```

例

```
ChangedBy()
```

ChangedOn

説明

「変更日」の変更追跡プロパティの値を戻します。

構文

```
ChangedOn():Date/Time
```

例

```
ChangedOn()
```

バージョンで現在のノードが最終更新された日時を戻します。

Concat

説明

指定された 2 つ以上の文字列を 1 つに連結し、結果を戻します。

構文

```
Concat(Item1,Item2,... ItemN:String):String
```

例

```
Concat(Abbrev,-,Descr())
```

現在のノード名が 100 で、現在のノードの説明が Colas の場合、戻り値は 100-Colas です。

ConcatWithDelimiter

説明

2 つ以上の文字列を 1 つの区切りリストに連結し、結果を戻します。

構文

```
ConcatWithDelimiter(Delimiter:String,SkipBlanks:Boolean,Items:String)
```

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「SkipBlanks」は、文字列のリストで空白値をスキップするかどうかを示します。有効な値: 1、0、T、F、t、f。

「Items」は、連結する文字列のリストを指定します。

例

```
ConcatWithDelimiter([comma],1,Item1,Item2,Item3,Item4)
```

戻り値は Item1; Item2; Item3; Item4 です。

Decode

説明

[openparen]、[closeparen]、[comma]、[tab]、[space]、[crlf]、[cr]および[lf]のすべてのインスタンスを適切な文字に置き換えて、入力文字列を戻します。

注： これは、特殊文字を使用するプロパティ定義名をアップグレードするための関数です。これらの特殊文字は、派生プロパティ式で解析上の問題の原因になる場合があります。この関数は主として、非推奨となった派生クラスを使用する既存のプロパティを Formula 派生クラスに変換するために使用されます。

構文

```
Decode(CodedString:String):String
```

「CodedString」は、関数を実行する対象の文字列値です。

DefaultProp

説明

プロパティのデフォルト値を戻します。

構文

```
DefaultProp(Property:String)
```

「Property」は、使用するプロパティの名前です。

Descr

説明

現在のノードの説明を戻します。

構文

```
Descr():String
```

例

現在のノードの説明が Colas の場合、戻り値は Colas です。

Divide

説明

指定された 2 つの整数値を除算し、結果を戻します。

構文

```
Divide(Int1,Int2:Integer):Integer
```

例

```
Divide(200,10)
```

戻り値は 20 です。

DivideFloat

説明

2 つの浮動小数点数(小数)を除算し、結果を戻します。

構文

```
Divide(Float1,Float2:Float):Float
```

例

```
DivideFloat(2.535,1.5)
```

戻り値は 1.69 です。

DualAncestorProp

説明

2 つのプロパティが指定された値と等しい最初の祖先のプロパティ値を戻します。

構文

```
DualAncestorProp(Operator1:String,Property1:String,Value1:String,Operator2:String,
Property2:String,Value2:String,FromTop:Boolean,ReturnProp:String):String
```

「Operator1」は、最初のプロパティと値を比較する際に使用する演算子です。有効な値: =、<、>、>=および<=。

「Property1」は、チェックする最初のプロパティの名前です。

「Value1」は、比較する最初の値です。

「Operator2」は、2 番目のプロパティと値を比較する際に使用する演算子です。有効な値: =、<、>、>=および<=。

「Property2」は、チェックする 2 番目のプロパティの名前です。

「Value2」は、比較する 2 番目の値です。

「FromTop」は、階層の最上位ノードから検索するかどうかを指定します。False の場合、検索は現在のノードから実行されます。

「ReturnProp」は、戻す祖先のプロパティの名前です。

Equals

説明

指定された 2 つの値が等しい場合に True を戻します。

構文

```
Equals(ParamType:String,Param1:String,Param2:String):Boolean
```

「ParamType」は、値の比較に使用するデータ型です。有効な値: string、integer、float、date。デフォルト値は integer です。

「Param1」は、比較する最初の値です。

「Param2」は、比較する 2 番目の値です。

例

```
Equals(integer,01,1)
```

戻り値は True です。

FlipList

説明

指定されたリストの逆を表す文字列を戻します。

構文

```
FlipList(List,Delimiter:String):String
```

「List」は、反転する文字列のリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
FlipList(DietCola;Orange Soda;Root Beer;Lemonade,[comma])
```

戻り値は、Lemonade、Root Beer、Orange Soda、Diet Cola です。

FloatToStr

説明

浮動小数値を文字列に変換して戻します。入力値が浮動小数を表さない場合は、ゼロ(0)を戻します。

構文

```
FloatToStr(Float1:Float):String
```

例

```
FloatToStr(1.001)
```

戻り値は 1.001 です。

Format

説明

指定されたフォーマット文字列のパラメータ・タイプ識別子と、指定されたタイプのパラメータ値を使用して値をフォーマットします。この関数は値が 1 つのパラメータに限定されます。

構文

```
Format(Format:String,ParamType:String, ValueToFormat:String):String
```

「Format」は、適用するフォーマットです。

「ParamType」は、値の比較に使用するデータ型です。有効な値: string、integer、float、date。デフォルト値は integer です。

「ValueToFormat」は、関数を実行する対象の値です。

例

```
Format('%8.2f',Float,123.456)
```

戻り値は 123.46 です。

FormattedDate

説明

指定されたフォーマット文字列を使用してフォーマットされた日付プロパティの値を戻します。

構文

```
FormattedDate(PropertyName:String,FormatString:String): String
```

「PropertyName」は、使用するプロパティの名前です。

「FormatString」は、適用する日付フォーマットを指定します。

GreaterThan

説明

2 つの値を比較して、最初の値が 2 番目の値より大きい場合に True を戻します。

構文

```
GreaterThan(Value1:Integer,Value2:Integer,ParamType:String):Boolean
```

「Value1」は、比較する最初の値です。

「Value2」は、比較する 2 番目の値です。

「ParamType」は、値の比較に使用するデータ型です。有効な値: string、integer、float、date。デフォルト値は integer です。

例

```
GreaterThan(1,2)
```

戻り値は False です。

GreaterThanOrEqual

説明

2 つの値を比較して、最初の値が 2 番目の値以上の場合に True を戻します。

構文

```
GreaterThanOrEqual(Value1:Integer,Value2:Integer,ParamType:String):Boolean
```

「Value1」は、比較する最初の値です。

「Value2」は、比較する 2 番目の値です。

「ParamType」は、値の比較に使用するデータ型です。有効な値: string、integer、float、date。デフォルト値は integer です。

例

```
GreaterThanOrEqual(2,2)
```

戻り値は True です。

HasCharacters

説明

指定された Input に、Character クラスの文字、特殊文字、または CharList でリストされた文字が含まれている場合に True を戻します。

構文

```
HasCharacters(Input:String,CharList:String):Boolean
```

「Input」は、テストする文字列値です。

「CharList」は、オプションの特殊な値を含む、テスト対象の文字のリストです。特殊文字の値は大カッコで囲まれてカンマで区切られます。有効な値: [alpha]、[numeric]、[whitespace]、[punctuation]、[uppercase]、[lowercase]、[comma]、[space]、[tab]、[crlf]、[cr]、[lf]、[openparen]および[closeparen]。

HasChildWith

説明

指定された式が現在のノードのどの子についても True の場合に True を戻します。

構文

```
HasChildWith(Expression:Boolean):Boolean
```

例

```
HasChildWith(GreaterThan(ID(),200))
```

現在のノードに、ID が 200 より大きい子がある場合、戻り値は True です。

HasParentNode

説明

現在のローカル・ノードに親ノードがある場合に True を戻します。

構文

```
HasParentNode():Boolean
```

例

```
HasParentNode()
```

ノードが階層の最上位ノードまたはいずれかの子孫ノードの子である場合、戻り値は True です。

HasSiblingWith

説明

指定された式が現在のノードのどの兄弟についても True の場合に True を戻します。

構文

```
HasSiblingWith(Expression:Boolean):Boolean
```

例

```
HasSiblingWith(PropValue(Leaf))
```

子のいずれかがリーフの場合、戻り値は True です。

HierNodePropValue

説明

指定された階層にある指定されたノードの指定されたプロパティの値を戻します。

構文

```
HierNodePropValue(HierAbbrev:String,NodeAbbrev:String,PropAbbrev:String):String
```

「HierAbbrev」は、使用する階層の名前です。

「NodeAbbrev」は、使用するノードの名前です。

「PropAbbrev」は、使用するプロパティの名前です。

例

```
HierNodePropValue(Assets,1000,Description)
```

Assets 階層のノード 1000 の説明が Banking の場合、戻り値は Banking です。

ID

説明

現在のノードの ID を戻します。

構文

```
ID():Integer
```

例

```
ID()
```

現在のノード ID が 2000 の場合、戻り値は 2000 です。

If

説明

指定された式が True と評価される場合に、TrueResult パラメータの値を戻します。
それ以外の場合には、FalseResult パラメータの値を戻します。

構文

```
If(Expression:Boolean, TrueResult:String,FalseResult:String):String
```

「Expression」は、評価するブール式です。

「TrueResult」は、条件が True である場合に戻す文字列値です。

「FalseResult」は、条件が False である場合に戻す文字列値です。

例

```
If(Equals(String,Descr(),),Abbrev(),Concat(Abbrev,-,Descr()))
```

ノード名が Colas で現在のノードの説明が空白の場合、戻り値は Colas です。

ノード名が 100 で現在のノードの説明が Colas の場合、戻り値は 100-Colas です。

InheritedPropOrigin

説明

継承されたプロパティ値の元になるノードの名前を戻します。指定されたプロパティがグローバルの場合は、元の階層も戻されます。指定されたプロパティが継承していない場合や、ノードまたはプロパティが見つからない場合は、False を戻します。

構文

```
InheritedPropOrigin(PropAbbrev:String,Node:String):String
```

例

```
InheritedPropOrigin(Custom.AccountType,Abbrev())
```

「PropAbbrev」は、使用するプロパティの名前です。

「Node」は、使用するノードの名前です。

InRange

説明

指定された値が、指定された値の範囲内にある場合に True を戻します。入力パラメータが文字列の場合は、Min および Max パラメータにはチェック対象の文字列の長さの範囲を指定します。その他のタイプの場合は、Min および Max にはチェック対象の数値範囲または日付範囲を指定します。

注： MinExclusive または MaxExclusive が True の場合は、Min または Max に等しい値が範囲に含まれ、それ以外の場合は範囲から除外されます。

構文

```
InRange(DataType:String,Input:String,Min:String,Max:String,MinExclusive:String,MaxExclusive:String):Boolean
```

「DataType」は、使用するデータ型です。有効な値: string、integer、float および datetime。

「Input」は、テストする文字列値です。

「Min」は、長さまたは範囲チェックの最小値です。

「Max」は、長さまたは範囲チェックの最大値です。

「MinExclusive」は、Min の値をチェック対象の範囲から除外するかどうかを指定します。

「MaxExclusive」は、Max の値をチェック対象の範囲から除外するかどうかを指定します。

例

```
InRange(Integer,5,1,10,False,False)
```

戻り値は True です。

InternalPrefix

説明

現在のノードの名前から数値以外の接頭辞を戻します。

構文

```
InternalPrefix()
```

Intersection

説明

指定された値のリスト両方に共通するアイテムのセットを戻します。結果の順序は、指定された最初のリストでのアイテムの位置に基づきます。

構文

```
Intersection(List1:String,List2:String,Delimiter:String):String
```

「List1」は、その内容を検索する文字列のリストを指定します。

「List2」は、その内容を検索する文字列のリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
Intersection(A;B;C;D;E,C;E;F;A,[comma])
```

戻り値は A、C、E です。

IntToStr

説明

指定された整数値を、文字列データ型に変換して戻します。入力値が整数を表さない場合は、ゼロ(0)を戻します。

構文

```
IntToStr(Int1:Integer):String
```

例

```
IntToStr(12345)
```

戻り値は 12345 です。

InvertedLevel

説明

現在のノードより下位の子孫の最大深さを戻します。

構文

```
InvertedLevel()
```

IsAlpha

説明

指定された文字列にアルファベット文字(大文字小文字は区別しない)のみが含まれている場合に True を戻します。

構文

```
IsAlpha(String:String):Boolean
```

例

```
IsAlpha(A23D)
```

戻り値は False です。

IsAlphaNumeric

説明

指定された文字列にアルファベット文字(大文字と小文字は区別しない)または数字のみが含まれている場合に True を戻します。

構文

```
IsAlphaNumeric(String:String,AllowBlanks:Boolean):Boolean
```

「String」は、テストする文字列値です。

「AllowBlanks」は、空白文字列を数値とみなすかどうかを指定します。デフォルトは False です。

例

```
IsAlphaNumeric(ABC123,True)
```

True を戻します。

IsBlank

説明

指定された入力値が空の文字列(ゼロ長)である場合に True を戻します。

構文

```
IsBlank(Input:String):Boolean
```

例

```
IsBlank(Descr())
```

ノードの説明が空白の場合は、True を戻します。

IsBottomNode

説明

指定されたノードに子ノードがない場合に True を戻します。ノードが見つからない場合は、False を戻します。

構文

```
IsBottomNode(Node:String):Boolean
```

「Node」は、使用するノードの名前です。

例

```
IsBottomNode(Abbrev)
```

ノードに子ノードがない場合は、True を戻します。

IsDataType

説明

入力値が指定されたデータ型に一致する場合に True を戻します。

構文

```
IsDataType(DataType:String,Input:String):Boolean
```

「DataType」は、使用するデータ型です。有効な値: boolean、string、integer、float および datetime。

「Input」は、テストする文字列値です。

例

```
IsDataType(123,Integer)
```

True を戻します。

IsDefinedPropVal

説明

指定されたノードの指定されたプロパティに定義(上書き)されている値がある場合に True を戻します。ノードまたはプロパティが見つからない場合は、False を戻します。

構文

```
IsDefinedPropVal(PropAbbrev:String,Node:String):Boolean
```

「PropAbbrev」は、使用するプロパティの名前です。

「Node」は、使用するノードの名前です。

例

```
IsDefinedPropVal(Custom.AccountType,Abbrev())
```

AccountType プロパティに定義(上書き)されている値がある場合は、True を戻します。

IsNodeAbove

説明

最初のノードが現在の階層の 2 番目のノードの祖先である場合に True を戻します。Node1 または Node2 が見つからない場合は、False を戻します。

構文

```
IsNodeAbove(Node1:String,Node2:String):Boolean
```

「Node1」は、使用する最初のノードの名前です。

「Node2」は、使用する 2 番目のノードの名前です。

例

```
IsNodeAbove(Parent,Child)
```

ノード Parent が Child ノードの祖先の場合は、True を戻します。

IsNodeBelow

説明

最初のノードが現在の階層の 2 番目のノードの子孫である場合に True を戻します。Node1 または Node2 が見つからない場合は、False を戻します。

構文

```
IsNodeBelow(Node1:String,Node2:String):Boolean
```

「Node1」は、使用する最初のノードの名前です。

「Node2」は、使用する 2 番目のノードの名前です。

例

```
IsNodeBelow(Child,Parent)
```

ノード Child が Parent ノードの子孫の場合は、True を戻します。

IsNumeric

説明

指定された値に数字(0-9)のみが含まれている場合に True を戻します。

構文

```
IsNumeric(String: String,AllowBlanksAsNumeric:Boolean):Boolean
```

「String」は、テストする文字列値です。

「AllowBlanksAsNumeric」は、空白値を文字列とみなすかどうかを指定します。デフォルト値は False です。

例

```
IsNumeric(12345)
```

戻り値は True です。

IsRangeListSubset

説明

指定された値が指定された範囲リストのサブセットである場合に True を戻します。

構文

```
IsRangeListSubset(RangeList:Range List,SubsetRangeList:Range
List,Delimiter:String):Boolean
```

「RangeList」は、指定された区切り文字で区切られた、検索する整数範囲のリストです。

「SubsetRangeList」は、指定された区切り文字で区切られた、検索する整数範囲のサブセット・リストです。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

Length

説明

指定された文字列値の文字数を戻します。

構文

```
Length(String:String):Integer
```

例

```
Length(Desc())
```

現在のノードの説明が Colas の場合、戻り値は 5 です。

LessThan

説明

2 つの値を比較して、最初の値が 2 番目の値より小さい場合に True を戻します。

構文

```
LessThan(Value1:Integer,Value2:Integer,ParamType:String):Boolean
```

「Value1」は、比較する最初の値です。

「Value2」は、比較する 2 番目の値です。

「ParamType」は、値の比較に使用するデータ型です。有効な値: string、integer、float、date。デフォルト値は integer です。

例

```
LessThan(1,2)
```

戻り値は True です。

LessThanOrEqual

説明

2 つの値を比較して、最初の値が 2 番目の値以下の場合に True を戻します。

構文

```
LessThanOrEqual(Value1:Integer,Value2:Integer,ParamType:String):Boolean
```

「Value1」は、比較する最初の値です。

「Value2」は、比較する 2 番目の値です。

「ParamType」は、値の比較に使用するデータ型です。有効な値: string、integer、float、date。デフォルト値は integer です。

例

```
LessThanOrEqual(3,3)
```

戻り値は True です。

ListAncestors

説明

最上位ノードから始まる、現在のノードの祖先の名前のカンマ区切りリストを戻します。現在のノードがローカル・ノードでない場合は、空白の文字列を戻します。

構文

```
ListAncestors(SortOrder:String):String
```

「ソート順」では、ノードの戻りリストのソート順を指定します。サポートされるソート順序値:

- [hier]--ローカル・コンテキストのデフォルト値。現在の階層について標準の階層ソート順序でノードのリストが戻されます。
- [alpha]--ノード名でソートして、ノードのリストが戻されます。
- [nodeid]--従来との互換性のために限定的に使用されます。戻りリストの各ノードの ID を基準に数字でソートして、ノードのリストが戻されます。

注： SortOrder パラメータを大カッコで囲む必要があります。

例

```
ListAncestors([alpha])
```

A、B、C、D が Z の子、Z が Y の子、現在のノードが D である場合、戻り値は Z、Y です。

ListChildren

説明

現在のノードの子のカンマ区切りリストを戻します。

構文

```
ListChildren(SortOrder:String):String
```

「ソート順」では、ノードの戻りリストのソート順を指定します。サポートされるソート順序値:

- [hier]--ローカル・コンテキストのデフォルト値。現在の階層について標準の階層ソート順序でノードのリストが戻されます。
- [alpha]--ノード名でソートして、ノードのリストが戻されます。
- [nodeid]--従来との互換性のために限定的に使用されます。戻りリストの各ノードの ID を基準に数字でソートして、ノードのリストが戻されます。

注： SortOrder パラメータを大カッコで囲む必要があります。

例

```
ListChildren([alpha])
```

A、B、C、D が Z の子で、現在のノードが Z である場合、戻り値は A、B、C、D です。

ListContains

説明

指定されたリストに指定された値が含まれている場合に True を戻します。

構文

```
ListContains(List:String,Item:String,Delimiter: String):Boolean
```

「List」は、その内容を検索する文字列のリストを指定します。

「Item」は、関数を実行する対象の文字列値を指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
ListContains(PropValue(NodeList),Colas,[comma])
```

戻り値は True です。

ListDescendants

説明

現在のノードの子孫のカンマ区切りリストを戻します。

構文

```
ListDescendants(SortOrder:String):String
```

「ソート順」では、ノードの戻りリストのソート順を指定します。サポートされるソート順序値:

- [hier]--ローカル・コンテキストのデフォルト値。現在の階層について標準の階層ソート順序でノードのリストが戻されます。
- [alpha]--ノード名でソートして、ノードのリストが戻されます。
- [nodeid]--従来との互換性のために限定的に使用されます。戻りリストの各ノードの ID を基準に数字でソートして、ノードのリストが戻されます。

注： SortOrder パラメータを大カッコで囲む必要があります。

例

```
ListDescendants([hier])
```

A、B、C、D が Z の子、Z が Y の子、現在のノードが Y である場合、戻り値は Z、A、B、C、D です。

ListDistinct

説明

指定されたリストから重複を削除して、アイテムの個別リストを戻します。

構文

```
ListDistinct(InputList:String,Delimiter:String):String
```

「InputList」は、使用するリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
ListDistinct(A;B:C;A;D,[comma])
```

戻り値は A、B、C、D です。

ListNodePropValues

説明

指定されたノード・リストに対して指定されたプロパティのプロパティ値のリストを戻します。見つからないノードについては、リスト内で空白の文字列を戻します。

構文

```
ListNodePropValues(NodeList:String,Delimiter:String,PropAbbrev:String):String
```

「NodeList」は、ノード名のカンマ区切りリストです。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「PropAbbrev」は、使用するプロパティの名前です。

例

```
ListNodePropValues(100;200;300,[comma],Core.Leaf)
```

ノード 100、200、300 がリーフ・ノードの場合には、True、True、True を戻します。

ListNodesWith

説明

指定されたノード・リストから、指定された式が True と評価されるノードのリストを戻します。

構文

```
ListNodesWith(NodeList:String,Delimiter:String,Expression:String):String
```

「NodeList」は、ノード名のカンマ区切りリストです。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「Expression」は、評価するブール式です。

例

```
ListNodesWith(100;200;300,[comma],NodeIsLeaf())
```

ノード 100、200、300 がリーフ・ノードの場合には、True、True、True を戻します。

ListRelatedNodesWith

説明

現在のノードとの関係を持ち、指定された式が True と評価されるノードのリストを戻します。

構文

```
ListRelatedNodesWith(Relation:String,Expression:String,SortOrder:String,Max:Intege
```

```
r):String
```

「関係」は次のとおりです:

- 祖先--指定した式でローカル・プロパティを参照できます
- 兄弟--指定した式でローカル・プロパティを参照できます
- 子--指定した式でローカル・プロパティとグローバル・プロパティを参照できます
- 子孫--指定した式でローカル・プロパティとグローバル・プロパティを参照できます

「Expression」は、評価するブール式です。

「ソート順」では、ノードの戻りリストのソート順を指定します。サポートされるソート順序値:

- [hier]--ローカル・コンテキストのデフォルト値。現在の階層について標準の階層ソート順序でノードのリストが戻されます。
- [alpha]--ノード名でソートして、ノードのリストが戻されます。
- [nodeid]--従来との互換性のために限定的に使用されます。戻りリストの各ノードの ID を基準に数字でソートして、ノードのリストが戻されます。

注： SortOrder パラメータを大カッコで囲む必要があります。

「Max」は、戻すノードの最大数を示す整数値です。ゼロを指定するか値を指定しない場合、制限がなく、すべてのノードが戻されます。

例

```
ListRelatedNodesWith(children,True,[alpha],1000)
```

ノードが現在のノードの子である場合、100、200、300 を戻します。

ListSiblings

説明

現在のノードの兄弟(ピア)のカンマ区切りリストを戻します。

構文

```
ListSiblings(SortOrder:String):String
```

「ソート順」では、ノードの戻りリストのソート順を指定します。サポートされているソート順の値:

- [hier]--ローカル・コンテキストのデフォルト値。現在の階層について標準の階層ソート順序でノードのリストが戻されます。
- [alpha]--ノード名でソートして、ノードのリストが戻されます。
- [nodeid]--従来との互換性のために限定的に使用されます。戻りリストの各ノードの ID を基準に数字でソートして、ノードのリストが戻されます。

例

```
ListSiblings([alpha])
```

A、B、C、D が Z の子で、現在のノードが B である場合、戻り値は A、C、D です。

LowerCase

説明

指定された文字列値を、小文字に変換して戻します。

構文

```
LowerCase(String:String):String
```

例

```
LowerCase(HOBBES)
```

戻り値は hobbes です。

LTrim

説明

指定された値を、文字列の先頭からすべてのスペースを切り取って戻します。

構文

```
LTrim(String: String): String
```

例

```
LTrim(“  101203 “)
```

戻り値は 101203 です。

MaxList

説明

空白アイテムは無視して、指定されたリストから最大アイテムを戻します。指定されたタイプではないアイテムがリストに含まれている場合は、空白の文字列を戻します。

構文

```
MaxList(InputList: String,Delimiter: String,ItemType: String)
```

「InputList」は、使用するリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「ItemType」は、リスト・メンバーに予期するアイテム・データ型を示します。有効な値: integer、float および datetime。デフォルト値は float です。

例

```
MaxList(1;2;3,[comma],Integer)
```

戻り値は 3 です。

MinList

説明

空白アイテムは無視して、指定されたリストから最小アイテムを戻します。指定されたタイプではないアイテムがリストに含まれている場合は、空白の文字列を戻します。

構文

```
MinList(InputList:String,Delimiter:String,ItemType:String)
```

「InputList」は、使用するリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「ItemType」は、リスト・メンバーに予期するアイテム・データ型を示します。有効な値: integer、float および datetime。デフォルト値は float です。

例

```
MinList(1;2;3,[comma],Integer)
```

戻り値は 1 です。

Modulus

説明

指定された 2 つの整数の除算の法(余り)を戻します。

構文

```
Modulus(Dividend: Integer, Divisor: Integer): Integer
```

「Dividend」は、除算中の分数の分子です。

「Divisor」は、除算中の分数の分母です。

例

```
Modulus(5,2)
```

戻り値は 1 です。

Multiply

説明

指定された 2 つの整数を乗算し、結果を戻します。

構文

```
Multiply(Int1: Integer, Int2: Integer): Integer
```

例

```
Multiply(2,5)
```

戻り値は 10 です。

MultiplyFloat

説明

指定された 2 つの浮動小数点数(小数)を乗算し、結果を戻します。

構文

```
Multiply(Float1: Float, Float2: Float): Float
```

例

```
MultiplyFloat(4.76,2.3)
```

戻り値は 10.948 です。

NextSibling

説明

現在の階層に使用されたソート順序に基づいて、現在のノードの次の兄弟を戻します。

構文

```
NextSibling(): String
```

例

```
NextSibling()
```

A、B、C、D が Z の子で、現在のノードが B である場合、戻り値は C です。

NodeAccessGroups

説明

現在のノードの現在のユーザーに対するノード・アクセス・グループのカンマ区切りリストを戻します。

構文

```
NodeAccessGroups(): String
```

例

```
NodeAccessGroups()
```

戻り値は Accounts、Finance です。

NodeExists

説明

指定されたノードが存在する場合に True を戻します。

構文

```
NodeExists(NodeAbbrev: string): Boolean
```

「NodeAbbrev」は、使用するノードの名前です。

例

```
NodeExists(2000)
```

ノード 2000 が存在する場合、戻り値は True です。

NodeInHier

説明

指定された階層に指定されたノードが存在する場合に True を戻します。

構文

```
NodeInHier(NodeAbbrev, HierAbbrev: string): Boolean
```

「NodeAbbrev」は、使用するノードの名前です。

「HierAbbrev」は、使用する階層の名前です。

例

```
NodeInHier(2000,Assets)
```

Assets 階層にノード 2000 が存在する場合、戻り値は True です。

NodeIsLeaf

説明

現在のノードがリーフ・ノードの場合に True を戻します。

構文

```
NodeIsLeaf(): Boolean
```

例

```
NodeIsLeaf()
```

現在のノードがリーフの場合、戻り値は True です。

NodeIsValidForPropertyHiers

説明

指定したノードが指定したプロパティの階層制約を満たす場合、True を戻します。プロパティがノード値を保管していない場合、またはプロパティの制約が定義されていない場合も、True を戻します。

構文

```
NodeIsValidForPropertyHiers(NodeAbbrev: String, PropAbbrev: String): Boolean
```

「NodeAbbrev」は、使用するノードの名前です。

「PropAbbrev」は、使用するプロパティの名前です。

NodePropValue

説明

ローカル・ノードの現在の階層で、またはグローバル・ノードの現在のバージョンで指定されたノードの指定されたプロパティの値を戻します。

構文

```
NodePropValue(NodeAbbrev: String, PropAbbrev: String): String
```

「NodeAbbrev」は、使用するノードの名前です。

「PropAbbrev」は、使用するプロパティの名前です。

例

```
NodePropValue(2000,Abbrev())
```

戻り値は 2000 です。

Not

説明

指定されたブール式の反対のブールを戻します。

構文

```
Not(Expression: Boolean): Boolean
```

例

```
Not(NodeIsLeaf())
```

現在のノードがリムの場合、戻り値は True です。

Now

説明

現在のシステムの日付または時刻(あるいはその両方)を戻します。

構文

```
Now([DateTimeType: String]): DateTime
```

「DataTimeType」は、日付の戻す部分を指定するオプションです。有効な値: Date、Time、Datetime。デフォルト値は Date です。

例

```
Now()
```

現在の日時、たとえば 3/25/2010 9:20:44 AM を返します。

```
Now(Time)
```

現在の時刻のみ、たとえば 9:20:44 AM を返します。

```
Now(Date)
```

現在の日付のみ、たとえば 3/25/2010 を返します。

NumChildWith

説明

現在のノードで、指定された式が True と評価される子の数を戻します。

構文

```
NumChildWith(Expression: Boolean): Integer
```

例

```
NumChildWith(NodeIsLeaf())
```

ノードに 2 つのリーフの子がある場合、戻り値は 2 です。

NumDescendantsWith

説明

現在のノードで、指定された式が True と評価される子孫の数を戻します。

構文

```
NumDescendantsWith(Expression: Boolean): Integer
```

例

```
NumDescendantsWith(NodeIsLeaf())
```

ノードに 2 つの子があり、それぞれの子に 10 のリーフの子がある場合、戻り値は 20 です。

Or

説明

指定されたいずれかのブール式が True と評価される場合に True を戻します。

構文

```
Or(Expression1, Expression2,... ExpressionN: Boolean): Boolean
```

例

```
Or(NodeIsLeaf(),Equals(Integer,PropValue(Level),3))
```

ノードがリーフである、または階層の第 3 レベルである場合、戻り値は True です。

OrigPropValue

説明

HasSiblingWith 関数または NumDescendantsWith 関数の使用時に、開始ノードの指定されたプロパティの値を戻します。

構文

```
OrigPropValue(PropAbbrev: String): String
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
HasSiblingWith(GreaterThan(OrigPropValue(ID),ID())
```

現在のノードの ID が 200 で、ノード ID が 200 より大きい兄弟を持つ場合、戻り値は True です。

PadChar

説明

指定された文字列を、指定されたパディング文字で延長して戻します。パディングは元の文字列の左側または右側に追加できます。結果の文字列は、少なくとも指定された桁数と同じ長さになります。元の文字列が指定された桁数より長い場合、結果は元の文字列になります。

構文

```
PadChar(String: String, PadChar: String; PadLeft: Boolean; NewLength: Integer):
String
```

「String」は、関数を実行する対象の文字列値です。

「PadChar」は、文字列のパディングに使用する文字です。

「PadLeft」は、文字列の左側にパディングするかどうかを指定します。有効な値: 1、0、T、F、t または f。

「NewLength」は、結果の長さを指定する整数です。

例

```
PadChar(102,0,1,6)
```

戻り値は 000102 です。

PadList

説明

指定されたリストを、指定されたパディング文字で延長して戻します。パディングは元のリストの左側または右側に追加できます。結果のリストは、少なくとも指定された桁数と同じ長さになります。元のリストが指定された桁数より長い場合、結果は元のリストになります。

構文

```
PadList(String, DelimChar, PadChr:String, PadLeft: Boolean, NewLength:Integer):
String
```

「StringList」は、指定された区切り文字で区切られた、パディングを適用する文字列のリストです。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「PadChar」は、文字列のパディングに使用する文字です。

「PadLeft」は、文字列の左側にパディングするかどうかを指定します。有効な値: 1、0、T、F、t または f。

「NewLength」は、結果の長さを指定する整数です。

例

```
PadList(1;2;3;4,;,T,3)
```

戻り値は 001;002;003,004 です。

ParentPropValue

説明

現在のノードの親ノードの指定されたプロパティの値を戻します。ノードに親がない場合、または現在のノードがローカル・ノードでない場合は、空白の文字列を戻します。

構文

```
ParentPropValue(PropAbbrev: String): String
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
ParentPropValue(Abbrev)
```

親ノードの名前が Colas の場合、戻り値は Colas です。

Pos

説明

大文字小文字を区別する検索を使用して、指定された文字列内の指定された部分文字列の最初の文字の位置(インデックス)を戻します。部分文字列が文字列値内で見つからない場合、ゼロ値が戻されます。

構文

```
Pos(SubString: String, String: String): Integer
```

「Substring」は、検索対象の文字列値です。

「String」は、関数を実行する対象の文字列値です。

例

```
Pos(D,ABCDEFG)
```

戻り値は 4 です。

PreviousSibling

説明

現在の階層に使用されたソート順序に基づいて、現在のノードの前の兄弟を戻します。

構文

```
PreviousSibling(): String
```

例

```
PreviousSibling()
```

A、B、C、D が Z の子で、現在のノードが B である場合、戻り値は A です。

PropControllingHier

説明

現在のバージョンにある指定されたプロパティの制御階層の名前を戻します。

構文

```
PropControllingHier(PropAbbrev: String): String
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
PropControllingHier(TimeBalance)
```

戻り値は Accounts です。

PropDefaultValue

説明

指定されたプロパティ定義のデフォルト値を戻します。

構文

```
PropDefaultValue(PropAbbrev: String): String
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
PropDefaultValue(Currency)
```

戻り値は USD です。

PropertyCategories

説明

現在のユーザーのプロパティ・カテゴリのカンマ区切りリストを戻します。

構文

```
PropertyCategories(AccessType: String) :String
```

「AccessType」は、プロパティ・カテゴリのアクセス・レベルです。有効な値:
ReadOnly、ReadWrite または Both。

例

```
PropertyCategories(Both)
```

戻り値は System、All、Essbase、Enterprise、HFM、Planning です。

PropMaxValue

説明

指定されたプロパティ定義の最大値を戻します。

構文

```
PropMaxValue(PropAbbrev: String): Integer
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
PropMaxValue(Volume)
```

戻り値は 10 です。

PropMinValue

説明

指定されたプロパティ定義の最小値を戻します。

構文

```
PropMinValue(PropAbbrev: String): Integer
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
PropMinValue(Volume)
```

戻り値は 1 です。

PropValue

説明

現在のノードの指定されたプロパティの値を戻します。

構文

```
PropValue(PropAbbrev: String): String
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
PropValue(Volume)
```

戻り値は 2 です。

RangeListContains

説明

指定された範囲リストに指定された値が含まれている場合に True を戻します。

構文

```
RangeListContains(RangeList: String, Value: Integer, Delimiter: String): Boolean
```

「RangeList」は、指定された区切り文字で区切られた、検索する整数範囲のリストです。たとえば、1-100、201-300 などです。

「Value」は、範囲のリストで検索する整数値です。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

例

```
RangeListContains(PropValue(MyRangeList),1,[Comma])
```

プロパティ MyRangeList の値が 1-10, 101-10000 の場合は、指定された範囲に 1 が含まれるため、戻り値は True です。一方、RangeListContains(PropValue(MyRangeList),11,[Comma])はFalse を戻します。11 は指定された範囲に含まれないためです。

注： MyRangeList を"1-5,6-10,101-1000"に変えると、RangeList が検証されて範囲に隣接する境界が結合されるため、この値は"1-10,101-1000"に置き換えられます。

ReplacementAbbrev

説明

ノードが非アクティブでマージ・ノードが指定されている場合、現在のノードの置換(マージ)ノードの名前を戻します。

構文

```
ReplacementAbbrev(): String
```

例

```
ReplacementAbbrev()
```

ReplacePropValue

説明

ノードが非アクティブでマージ・ノードが指定されている場合、現在のノードの置換(マージ)ノードに対して指定されているプロパティ値を戻します。

構文

```
ReplacePropValue(PropAbbrev: String): String
```

「PropAbbrev」は、使用するプロパティの名前です。

例

```
ReplacePropValue(Description)
```

ReplaceStr

説明

出現する古いパターンを新しいパターンで置換して文字列を戻します。

構文

```
   ReplaceStr(String: String,OldPattern: String,NewPattern: String,ReplaceAll:
Boolean): String
```

「String」は、関数を実行する対象の文字列値です。

「NewPattern」は、検出された文字列を置換する文字列値です。

「OldPattern」は、検索する文字列値です。

「ReplaceAll」は、検索文字列すべてを置換文字列で置換するかどうかを指定します。有効な値: 1、0、T、F、t または f。

例

```
   ReplaceStr(A1;A2;A3,A,B,T)
```

戻り値は B1;B2;B3 です。

RTrim

説明

指定された値を、文字列の末尾からすべてのスペースを切り取って戻します。

構文

```
   RTrim(String: String): String
```

「String」は、関数を実行する対象の文字列値です。

例

```
   RTrim(“100   “))
```

戻り値は 100 です。

SortList

説明

指定されたリストをソートされた順序で戻します。

構文

```
SortList(InputList: String,Delimiter: String,IgnoreCase: Boolean,ItemType: String)
```

「InputList」は、使用するリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「IgnoreCase」は、ソート時に大文字と小文字を区別しないかどうかを指定します。デフォルト値は False です。

「ItemType」は、結果リスト・アイテムのターゲットのデータ型を示します。有効な値: string、integer、float date、time および datetime。デフォルト値は string です。どのアイテムも指定された型に変換できない場合、関数は空白の文字列を戻します。

StripPadChar

説明

指定された文字列の最初から指定されたパディング文字を削除して、変更後の値を戻します。元の文字列に含まれるパディング文字の数が StripCount に指定された数より少ない場合は、元の文字列値が戻されます。

構文

```
StripPadChar(String: String, PadChar: String, StripCount: Integer): String
```

「String」は、関数を実行する対象の文字列値です。

「PadChar」は、文字列のパディングに使用する文字です。

「StripCount」は、文字列から削除する文字数を指定する整数です。ゼロの場合、パディング文字すべてを削除します。

例

```
StripPadChar(0003333,0,6)
```

戻り値は 3333 です。

StrToBool

説明

指定された文字列に基づいてブール値を戻します。文字列が Y、T、または 1 で始まる場合は、大文字小文字または後続の文字に関係なく True 値が戻されます。文字列が N、F、または 0(ゼロ)で始まる場合は、大文字小文字または後続の文字に関係なく False 値が戻されます。

構文

```
StrToBool(String: String): Boolean
```

「String」は、関数を実行する対象の文字列値です。

例

```
StrToBool(0)
```

戻り値は False です。

StrToFloat

説明

指定された文字列の浮動小数値を戻します。スペースや空白の文字列に対してはゼロ(0)を戻します。

指定された文字列が浮動小数点数を表さない場合は、エラーが戻されます。

構文

```
StrToFloat(String: String): Float
```

「String」は、関数を実行する対象の文字列値です。

例

```
StrToFloat(11.101)
```

戻り値は 11.101 です。

StrToInt

説明

指定された文字列の整数値を戻します。スペースや空白の文字列に対してはゼロ(0)を戻します。

指定された文字列が整数を表さない場合は、エラーが戻されます。

構文

```
StrToInt(String: String): Integer
```

「String」は、関数を実行する対象の文字列値です。

例

```
StrToInt(101)
```

戻り値は 101 です。

Stuff

説明

指定された文字を指定された文字列で置換して、指定された値を戻します。

構文

```
Stuff(PropAbbrev: String, CharsToReplace: String, ReplacementChars: String):
String
```

「PropAbbrev」は、使用するプロパティの名前です。

「CharsToReplace」は、検索する文字列値です。

「ReplacementChars」は、検出された文字列を置換する文字列値です。

例

```
Stuff(Abbrev(),GEO,RIO)
```

Abbrev が GEO101 の場合、戻り値は RIO101 です。

SubString

説明

指定されたインデックスから始まる指定された文字数の、指定された文字列の一部を戻します。

構文

```
SubString(String: String, Index: Integer, Count: Integer): String
```

「SubString」は、関数を実行する対象の文字列値です。

「Index」は、部分文字列の検索を開始するインデックス位置を表す整数です。ゼロは、文字列内の最初の文字の位置を示します。

「Count」は、開始インデックスから開始する、検索する文字列の番号を表す数字です。

例

```
SubString(Colas,1,2)
```

戻り値は Co です。

Subtract

説明

最初の値から 2 番目の整数値を減算し、結果を戻します。

構文

```
Subtract(Minuend: Integer,Subtrahend: Integer): Integer
```

「Minuend」は整数値です。

「Subtrahend」は整数値です。

例

```
Subtract(10,2)
```

戻り値は 8 です。

SubtractFloat

説明

最初の値から 2 番目の浮動小数点値を減算し、結果を戻します。

構文

```
SubtractFloat(Minuend,Subtrahend: Float): Float
```

「Minuend」は浮動小数点値です。

「Subtrahend」は浮動小数点値です。

例

```
SubtractFloat(8.09,3.76)
```

戻り値は 4.33 です。

SumList

説明

空白アイテムは無視して、リスト内のアイテムの合計を戻します。指定されたタイプではないアイテムがリストに含まれている場合は、空白の文字列を戻します。

構文

```
SumList(InputList: String,Delimiter: String,ItemType: String):Integer
```

「InputList」は、使用するリストを指定します。

「Delimiter」は、文字列リストのアイテムを区切るために使用される文字です。サポートされる特殊文字:

- [comma]
- [space]
- [tab]

注： 区切り文字の名前(文字ではない)を使用し、名前を大カッコで囲む必要があります。

「ItemType」は、リスト・メンバーに予期するアイテム・データ型を示します。有効な値: integer および float。デフォルト値は float です。

例

```
SumList(1;2;3,;,Integer)
```

戻り値は 6 です。

Trim

説明

指定された値を、文字列の先頭および末尾からすべてのスペースを切り取って戻します。

構文

```
Trim(String: String): String
```

「String」は、関数を実行する対象の文字列値です。

例

```
Trim(“  101  “)
```

戻り値は 101 です。

UpperCase

説明

大文字に変換された文字列値を戻します。

構文

```
UpperCase(String: String): String
```

「String」は、関数を実行する対象の文字列値です。

例

```
UpperCase(smaller)
```

戻り値は SMALLER です。

UserName

説明

現在のユーザーのユーザー名を戻します。

構文

```
UserName(): String
```

例

```
UserName()
```

戻り値はユーザー名です。

XOr

説明

指定されたブール式の 1 つのみが True と評価される場合に True を戻します。

構文

```
XOr(Expression1:Boolean, Expression2: Boolean): Boolean
```

例

```
XOr(NodeIsLeaf(),Equals(Integer,PropValue(Level),3))
```

ノードがリーフであるか、階層の第 3 レベルである、そのいずれかの場合、戻り値は True です。

関数グループ

次の表では、関数を用途別にグループ化しています。

表 9　関数グループ

関数グループ	関数
集約	• AvgList • MaxList • MinList • SumList

関数グループ	関数
変更追跡	● AddedBy ● AddedOn ● Changed ● ChangedBy ● ChangedOn ● Now
比較	● Equals ● GreaterThan ● GreaterThanOrEqual ● InRange ● IsBlank ● IsRangeListSubset ● LessThan ● LessThanOrEqual ● RangeListContains
条件	● And ● If ● Not ● Or ● XOr
データ型	● BoolToStr ● FloatToStr ● IntToStr ● IsDataType ● IsNumeric ● StrToBool ● StrToFloat ● StrToInt
リスト	● ArrayCount ● ArrayIndex ● ArrayItem ● Intersection ● ListContains ● ListDistinct ● ListNodePropValues ● ListNodesWith ● SortList

関数グループ	関数
算術	● Add ● AddFloat ● Divide ● DivideFloat ● Modulus ● Multiply ● MultiplyFloat ● Subtract ● SubtractFloat
ノード	● Abbrev ● ID ● InternalPrefix ● NodeExists ● NodeInHier ● NodeIsLeaf
プロパティ	● AncestorProp ● AscNodeProp ● DefaultProp ● Descr ● DualAncestorProp ● HierNodePropValue ● InheritedPropOrigin ● IsDefinedPropVal ● NodePropValue ● OrigPropValue ● ParentPropValue ● PropControllingHier ● PropDefaultValue ● PropMaxValue ● PropMinValue ● PropValue ● ReplacePropValue

関数グループ	関数
関係	● Children ● HasChildWith ● HasParentNode ● HasSiblingWith ● InvertedLevel ● IsBottomNode ● IsNodeAbove ● IsNodeBelow ● ListAncestors ● ListChildren ● ListDescendants ● ListRelatedNodesWith ● ListSiblings ● NextSibling ● NumChildWith ● NumDescendantsWith ● PreviousSibling ● ReplacementAbbrev
文字列操作	● Concat ● ConcatWithDelimiter ● Decode ● FlipList ● Format ● FormattedDate ● HasCharacters ● IsAlpha ● IsAlphaNumeric ● Length ● LowerCase ● LTrim ● PadChar ● PadList ● Pos ● ReplaceStr ● RTrim ● StripPadChar ● Stuff ● SubString ● Trim ● UpperCase

関数グループ	関数
ユーザー	● NodeAccessGroups ● PropertyCategories ● UserName

10 動的スクリプトの管理

この章の内容

動的スクリプトは、JavaScript を使用した、派生プロパティと検証のビジネス・ロジックの開発を可能にします。動的スクリプトは、標準のスクリプト言語を使用しており、式よりも堅牢でパフォーマンスに優れた代替手段です。スクリプトでは、複数の文や変数、インライン・コメントを使用することで、編成がわかりやすくなり、ロジックの複雑さを軽減できます。また、動的スクリプトでは、ループや正規表現などの高度な概念もサポートされています。

実行コンテキスト

スクリプトの実行には、次の 2 つのコンテキストがあります: プロパティ・コンテキストおよび検証コンテキスト。各コンテキストには別々の初期パラメータが定義され、異なる結果タイプが戻されます。

詳細は、次を参照してください:

- スクリプトを使用した派生プロパティ
- スクリプトを使用した検証

スクリプトを使用した派生プロパティ

スクリプト派生クラスを使用すると、派生プロパティで動的スクリプトを使用できます。スクリプトを使用した派生プロパティは、バージョン、階層およびノードで使用できます。

表 10　プロパティ・レベルの説明

プロパティ・レベル	パラメータ	オブジェクト
バージョン	version	VersionObject
階層	hierarchy	HierarchyObject

プロパティ・レベル	パラメータ	オブジェクト
グローバル・ノード	node	NodeObject
ローカル・ノード	node	LocalNodeObject

詳細は、次を参照してください:

- ノード派生プロパティ
- バージョンおよび階層のプロパティ

ノード派生プロパティ

このコンテキストでは、node と呼ばれるパラメータが渡されます。グローバル・プロパティの場合、node は NodeObject です。ローカル・プロパティの場合、node は LocalNodeObject です。派生プロパティのスクリプトは値を戻す必要があり、その値は、評価または実行されるプロパティのデータ型に適切であることが必要です。スクリプトにより戻された値がプロパティのデータ型と一致しない場合は、強制されます: たとえば、ブール・プロパティに戻された null 値は、false とみなされます。

注： すべての Data Relationship Management のプロパティ・データ型に、JavaScript 表現が存在するわけではありません。データ型の変換を参照してください。

バージョンおよび階層のプロパティ

このコンテキストでは、VersionObject を参照するバージョン・パラメータ、または HierObject を参照する階層パラメータを使用します。スクリプトを定義する際、スクリプトを評価または実行している場合は、バージョンがロードされないことがあります。バージョンまたは階層の派生プロパティがアクセスするのが、別のバージョンおよび階層のレベル・プロパティのみの場合、バージョンのロード・ステータスに関係なく、プロパティが計算されます。バージョンまたは階層の派生プロパティがノード・レベルの情報にアクセスを試行する場合は、バージョンがロードされている必要があり、ロードされないと、プロパティの計算でエラー値が生成されます。たとえば、バージョン・レベル・プロパティが孤立のリストの取得を試行すると、バージョンがロードされていない場合には、そのプロパティによりエラー値が生成されます;バージョンがロードされると、同じプロパティにより正しい値が生成されます。

スクリプトを使用した検証

スクリプト検証クラスを使用すると、検証に動的スクリプトを使用できます。いくつかの異なる検証レベルがあり、その中にはパラメータが複数存在するものもあります。検証レベルとパラメータを次に示します:

レベル	パラメータ	説明
任意のレベル	validation	現在実行中の検証に関する情報を提供します

レベル	パラメータ	説明
階層	hierarchy	検証する階層の HierarchyObject
GlobalNode	node	検証するグローバル・ノードの NodeObject
ノード	node	検証するノードの LocalNodeObject
除去	node	検証するノードの NodeObject
移動	node	移動するノードの LocalNodeObject
	move	移動に関する情報を含むオブジェクト: OldParent--元の親の LocalNodeObject NewParent--宛先の親の LocalNodeObject IsPost/IsPre--移動の直前または移動の完了直後に、このスクリプトが実行されているかどうかを示します。スクリプトは通常、移動前に 1 回、移動後に 1 回の 2 回実行されます。 Values--移動前のフェーズ中に、単純なキーと値のペアがこのオブジェクトに格納されます(たとえば、Values["key"] = "value")。これらの値が存在する場合、移動後のフェーズ中に、移動前の状態に関する情報を格納し、移動後の状態と比較することができます。すべての値が String、Number または Date オブジェクトに変換されます。複雑なオブジェクトは現在サポートされていません。
マージ	node	削除または非アクティブ化するノード
	merge	マージに関する情報を含むオブジェクト: Target--マージのターゲットの NodeObject IsInactivate--非アクティブ化操作の場合は True IsDelete--削除操作の場合は True
バージョン	version	検証するバージョンの VersionObject

列挙定数

特定のプロパティは名前付き定数に対応する数字で、コードの把握と維持を簡単にします。例えば、次を使用するかわりに:

`if(nodeProp.PropOrigin == 2)` 次を使用できます: `if(nodeProp.PropOrigin == PropOrigin.Overridden)`

プロパティの列挙定数

- DataType--Boolean、LeafNode、Date、Time、Float、Integer、Sort、Group、Node、LimbNode、String、Hier、Version、ListGroup、MultiNode、AscNode、AscNodes、AscGroup、Memo、FormatMemo、SortProp、Property、Query、StdQuery、GlobalNode、NodeProps、RangeList、DateTime、Hyperlink、HierarchyGroup
- PropLevel--Node、Hier、Version

- PropOrigin--Default、Inherited、Overridden、InheritedHier、InheritedVer、Derived、InheritedDomain、Unknown
- PropType--Invalid、System、Defined、Lookup、Derived、Stats、Validation、Verification、LimbAccessGroup、LeafAccessGroup、UserSpecific、RWDerived、SharedInfo

検証の列挙定数

- ValidationLevel--Node、Hier、Version、GlobalNodes、Merge、Move、Remove
- ValidationType--None、RealTime、Batch、Both

サポートされている JavaScript データ型

標準の JavaScript データ型が使用可能で、Data Relationship Management では、可能な場合にはそれらが使用されます。たとえば、日付は Date オブジェクトを使用して表されます。関数はそれ自体がオブジェクトであり、new で呼び出された関数は、ECMA 準拠の JavaScript 環境と同じように、プロトタイプが関数のコンストラクタ・プロトタイプを指すオブジェクトを作成します。

注： JavaScript Document Object Model (DOM)オブジェクトは、Data Relationship Management スクリプトではサポートされていません。

どのメソッドが使用可能であるかも含め、JavaScript 構文および組込みオブジェクトに精通している必要があります。使用可能なデータ型の一部を示します:

- Array--length、pop、push、concat、join、reverse、slice、shift、sort などが含まれます
- Boolean--True および False を表します
- Date--parseDate、month、day、year などが含まれます
- Error--try/catch エラー処理を使用して、エラー・メッセージにアクセスします
- Function--標準の call および apply 機能がサポートされています
- Math--random、max、pow、round、sin、cos、floor、sqrt、log などが含まれます
- Number--JavaScript のすべての数値は、不動小数点型の数値です
- RegExp--正規表現の言語サポートを使用することも、明示的にアクセスすることも可能です
- String--concat、indexOf、lastIndexOf、substr、split、splice、search、replace、toUpperCase、toLowerCase などが含まれます

グローバルに使用可能な parseInt、parseFloat、isNaN、decodeURI、encodeURI などの関数も使用きます。

データ型の変換

すべての Data Relationship Management のプロパティ定義データ型に、対応する JavaScript の表現が存在するわけではありません。対応する表現がないものは、StringValue および Value が等価となり、その文字列値の解析方法を把握しておく必要があります。これらのいずれかのデータ型のプロパティ値を戻す場合は、そのデータ型の適切な文字列表現も戻す必要があります。格納されている値をプロパティの有効なデータ型に変換できない場合、値は未定義になります。

リスト・プロパティは、データ型に適切なタイプのオブジェクトを含む配列の各要素とともに Array を戻します。たとえば、List とマークされた Date プロパティでは、Date オブジェクトを含む Array が戻されます。

参照ターゲットが無効な場合、参照テーブルにキーがない場合または参照テーブルの値がデータ型に対して有効でない場合には、参照プロパティにより、予測されるデータ型が戻されない場合があります。たとえば、キーと値のペアの値が TEST であるのに、データ型が Date の場合、結果は未定義になります。

次に、Data Relationship Management のデータ型を、対応する JavaScript の表現とともに示します。

プロパティ定義データ型	JavaScript データ型
AscGroup	NodeObject 配列
AscNode	NodeObject
AscNodes	NodeObject 配列
Boolean	Boolean
Date	Date
DateTime	DateTime
Float	Number
FormatMemo	String
GlobalNode	NodeObject
Group	String 配列
Hier	HierObject
Hierarchy Group	String (階層グループ名)
Hyperlink	String (URL を表します)
Integer	Number
LeafNode	LocalNodeObject
LimbNode	LocalNodeObject
ListGroup	String 配列

プロパティ定義データ型	JavaScript データ型
Memo	String
MultiNode	LocalNodeObject 配列
Node	LocalNodeObject
NodeProps	PropDefObject 配列
Query	String (クエリー名)
Property	PropDefObject
Sort	Number
SortProp	PropDefObject
StdQuery	String (クエリー名)
String	String
Time	String
Version	String (バージョン名)

その他の JavaScript の派生プロパティ(または別のノードの派生プロパティ)を呼び出す場合、その派生によって戻される値がすぐには文字列表現に変換されないため、複合オブジェクトで toString()を呼び出すことによって結果が戻されるまで、(Array からなど、組込み変換に記載されている場合を除き)派生と遅延強制の間で複合オブジェクトを渡すことができます。

Print 関数

Print 関数を使用すると、スクリプトの作成中にデバッグ情報を出力できます。結果は、スクリプト・エディタの「警告」セクションに表示されます。Print 関数で生成されるのはテスト・コンテキストの出力のみですが、エンジンでは引数を構成する必要があります;そのため、本番用にスクリプトを保存する前に、print 文をすべてコメント・アウトしてください。

Format 関数

Format 関数は、標準の JavaScript よりもはるかに豊富な文字列のフォーマット・メカニズムを備えています。最初のパラメータは、中カッコで囲まれたフォーマット指定子を含む文字列です。カッコを二重にするとエスケープされます。たとえば、「{{」は「{」と出力されます。フォーマット指定子はゼロから始まり、増分的に増えていきます。シーケンスから指定子を除外すると、Format 関数と同等のパラメータが無視されます。たとえば、「{1}」では Format に対する最初の値パラメータが無視され、2 番目が使用されます。

簡単な方法が 1 つあります。Format を呼び出し、カッコなしでフォーマット指定子を渡して、引数を 1 つのみ渡すことができます。結果は、Format("{0:<specifier>", <argument>)と同一です

フォーマット指定子は、Java や C#など、その他の言語と同じように機能します。構文は{<paramnum>}または{<paramnum>;<format>}で、paramnum はゼロから始まる正の整数であり、連続して増加します。フォーマット・パラメータは、パラメータとして渡されるオブジェクトのタイプによって異なります。

フォーマット・パラメータは、一般的に、ユーザーのロケールに適した値を戻します; たとえば、米国では「{0:0.00}」は「1.23」を戻し、ヨーロッパでは「1,23」を戻します。または、エスケープのサポートを使用して、ロケールを明示的に上書きし、すべてのユーザーに同じ値を出力できます。たとえば、「#\,###\,##0」は、文化の設定に関係なく、すべての地域でカンマを 3 桁ごとの区切り文字として使用し、数値をフォーマットします。

数値のフォーマット

数値は、「G」のような 1 つのショートカット文字を使用するか、「##0,000.0」のような指定子の組合せを使用してのみフォーマットできます。フォーマット指定子に 1 文字を超えるショートカット文字を使用すると、変更されずに出力にコピーされます(リテラル文字として扱われます)。

適切な文化を選択した状態で本番のエクスポートを実行し、出力が正しくフォーマットされていることを確認します。

表 11　1 文字ショートカットの数値フォーマット

フォーマット	説明
D	整数(負数の負の符号はロケール対応)
D<precision>	少なくとも<precision>の桁までフォーマットされた整数で、必要な場合はゼロでパディングされます。たとえば、「{0:D5}」が指定された 123 は 12300 と出力されます。
E	指数表記「1.234E+10」
F	浮動小数点数「123.456」(負数の小数点と負の符号はロケール対応)
F<precision>	小数点以下が<precision>の有効桁数まで切り捨てられた浮動小数点数
G	一般的な数値フォーマット
N	一般化された数値フォーマット「123,456.789」(負数のグループ区切り文字/小数点と負の符号はロケール対応)
N<precision>	小数点以下が<precision>の桁まで切り捨てられた一般化された数値
P	パーセント(負数のグループ区切り文字/小数点と負の符号はロケール対応で、0.20146 は「20.14%」と出力されます)
P<precision>	<precision>の有効桁数まで切り捨てられたパーセント({0:P0 と指定された 0.205 は 21%と出力されます)
X	16 進数は 4D2 と出力されます

表 12 数値フォーマット指定子

フォーマット	説明
0	ゼロがプレースホルダで、数字が存在する場合はその数字を出力し、それ以外の場合はゼロが出力されます
#	数字のプレースホルダで、数字が存在する場合はその数字を出力し、それ以外の場合は出力は生成されません
.	ロケール固有の小数点
,	プレースホルダ間に配置された場合は、ロケール固有のグループ区切り文字を出力します({0:#,#}と指定された 123456789 は 123,456,789 と出力されます)。小数点(または暗黙的な小数点)のすぐ左に 1 つ以上配置されている場合、数値はカンマごとに 1000 で区切られます({0:#,##0,,}と指定された 123456789 は 1,235 と出力されます)。
%	数値に 100 を掛け、指定された場所にロケール固有のパーセンテージ記号を出力します
E<sign>0	指数の表記。少なくともゼロが 1 つ必要で、ゼロの数は指数の最小桁数を指定しています。<sign>はオプションで、次のいずれかになります: ● + (記号+/-が必要に応じて必ず出力されます) ● - (負数にのみ-記号が出力されます)
\<char>	エスケープ文字(<char>はリテラル出力として処理されます)
;	セクション区切り。存在する場合、整数、負数およびゼロの異なるフォーマットを定義できます。 ● 1 つのセクション「{0:#,#;}」--セクションなしと同じです ● 2 つのセクション「{0:#,#;-#,0}」--最初のセクションは整数およびゼロに、2 つ目のセクションは負数に適用されます ● 3 つのセクション「{0:#,#;-#,0;zero}」--最初のセクションは整数に、2 つ目のセクションは負数に(空の場合、1 つ目のセクションは負数にも使用)、3 つ目のセクションはゼロに適用されます
その他の文字	変更されずにそのまま出力にコピーされます

日付のフォーマット

日付は、1 つのショートカット文字: G などを使用するか、指定子の組合せ: HH:mm などを使用してフォーマットできます。通常の指定子として、ショートカットではなく、1 つの文字を使用する場合は、文字列に接頭辞%を付けます。例: %m とすると、月と日にちではなく、パディングなしの分が出力されます。

表 13 1 文字ショートカットの日付フォーマット

フォーマット	説明
t	短い形式の時間 "午後 4:05"
T	長い形式の時間 "午後 4:05:07"
d	短い形式の日付 "2013/3/9"

フォーマット	説明
D	長い形式の日付 "2013 年 3 月 9 日金曜日"
f	長い形式の日付と短い形式の時間 "2013 年 3 月 9 日金曜日午後 4:05"
F	長い形式の日付と長い形式の時間 "2013 年 3 月 9 日金曜日午後 4:05:07"
g	短い形式の日付と短い形式の時間 "2013/3/9 午後 4:05"
G	短い形式の日付と長い形式の時間 "2013/3/9 午後 4:05:07"(デフォルト)
m	月と日にち "3 月 9 日"
y	年と月 "2013 年 3 月"
r	RFC 1123 "Fri, 09 Mar 2013 16:05:07 GMT"
s	ソート可能な日付/時間 "2013-03-09T16:05:07"
u	ソート可能なユニバーサルの日付/時間 "2013-03-09 16:05:07Z"

表 14 日付フォーマット指定子(複数文字)

フォーマット	説明 例は 2013-04-05 04:07:09 PM CST を元にしています
yy	年 "13"
yyyy	年 "2013"
M	月 "4"
MM	月 "04"
MMM	月 "Apr"
MMMM	月 "April"
d	日にち "5"
dd	日にち "05"
ddd	曜日 "Sun"
dddd	曜日 "Sunday"
h	12 時間 "4"
hh	12 時間 "04"
H	24 時間 "16" (午前 4 時の場合は"4")
HH	24 時間 "16" (午前 4 時の場合は"04")
m	分 "7"
MM	分 "07"

フォーマット	説明 例は 2013-04-05 04:07:09 PM CST を元にしています
s	秒 "9"
ss	秒 "09"
f	1 秒未満の部分(1-4 桁まで表示して精度を高められます)
F	後ろにゼロが続かない、1 秒未満の部分(1-4 桁まで表示できます)
t	AM または PM の指定子 P (24 時間表示のみの文化圏ではブランク)
tt	AM または PM の指定子 PM (24 時間表示のみの文化圏ではブランク)
z	GMT オフセット "-6"
zz	GMT オフセット "-06"
zzz	GMT オフセット "-06:00"
:	時間の区切り文字(ロケール固有)
/	日付の区切り文字(ロケール固有)
\<char>	エスケープ文字(<char>はリテラル出力として処理されます): たとえば、{0:HH\h}は 16h を出力します
その他の文字	変更されずにそのまま出力にコピーされます

Data Relationship Management オブジェクト

次に、Data Relationship Management オブジェクト、およびそのメソッドとプロパティを説明します。

SysObject

Sys と呼ばれる 1 つの SysObject は、自動的に作成されます。このオブジェクトはすべてのコンテキストで使用可能で、汎用的な関数および Data Relationship Management アプリケーションの情報を提供します。このオブジェクトにプロパティはありません。

表 15　SysObject メソッド

名前	説明
FormattedDate (value, formatString)	式のシステム・ルールに従って日付をフォーマットします。古い式プロパティに正確に一致させる下位互換に便利です。 ● 値は Date オブジェクトか有効な日時文字列であることが必要です ● formatString は、有効なフォーマット文字列であることが必要です(FormattedDate 関数を参照してください)

名前	説明
GetPropDef(abbrev)	指定されたプロパティ名の PropDefObject を戻します。名前は、完全修飾名であることが必要です。
GetSysPrefValue(abbrev)	指定されたシステム・プリファレンス(HierNodeSeparator など)の値を戻します
InRange(dataType, input, min, max, minExclusive, maxExclusive)	式関数 InRange に対応します。必須パラメータは dataType、input および min です。
IsNodeAbove(ancestor, child)	階層内で祖先が子より上の場合に True を戻します。パラメータが LocalNodeObjects ではない場合、または同じ階層にない場合に False を戻します。
IsNodeBelow(descendant, parent)	階層内で子孫が親より下の場合に True を戻します。パラメータが LocalNodeObjects ではない場合、または同じ階層にない場合に False を戻します。
RunFormula(node, propDef, formulaString)	Data Relationship Management の式を実行し、文字列の結果を戻します ● ノードは NodeObject または LocalNodeObject です。NodeObject を渡す際に、式文字列でローカル・プロパティを参照しないようにする必要があり、参照するとエラーが発生します。LocalNodeObject を渡す際は、使用可能なすべてのグローバルおよびローカルのプロパティを参照できます。 ● propDef--正しく解析または実行するには、一部の式関数にプロパティ定義が必要です。これらの関数を使用する際は、プロパティ定義を指定する必要があります。一般的に、プロパティ定義文字(レベル、グローバルとローカル、およびタイプなど)は一致する必要がありますが、formulaString 用の実際のプロパティである必要はありません。関連がなくてもかまいません。ほとんどの式で、このパラメータに null を渡すことができます。 ● formulaString は従来の Data Relationship Management 式です; 空白は式のリテラルの一部とみなされるため、必要に応じて削除する必要があります。 **注：** これはベスト・プラクティスではないため、従来の動作と完全に一致させる必要がある場合にのみ使用します。このメソッドを使用すると、パフォーマンスが低下します。

PropDefObject

このオブジェクトにメソッドはありません。

表 16　PropDefObject プロパティ

名前	説明
Abbrev	プロパティ定義名(完全修飾ネームスペースなど)
Cascade	プロパティ値を継承する場合は True
ColumnWidth	デフォルトのエクスポート列の幅
DataType	DataType.String などの DataType 列挙値(列挙定数を参照)
Descr	説明
DefaultValue	プロパティ定義のデフォルト値。タイプはプロパティ定義のデータ型によって異なります。
EditorLabel	ラベル

名前	説明
Global	プロパティがグローバル・ノード・プロパティの場合は True
Hidden	プロパティをプロパティ・グリッドから非表示にする場合は True
ID	ID
Level	PropLevel.Node などの PropLevel 列挙値(列挙定数を参照)
List	prop を値のリストから選択できる場合は True
ListValues	ユーザーが選択可能な値の配列
LookupValues	参照プロパティのキーと値のペアを検索します。この配列にあるオブジェクトのキーと値のプロパティを使用します。
MaxValue	最大値
MinValue	最小値
Namespace	プロパティ定義のネームスペース
PropType	PropType.Defined などの PropType 列挙値(「列挙定数」を参照)
PropClass	派生クラス(式またはスクリプト)
ReadOnly	プロパティが読取り専用(コア統計プロパティなど)の場合は True

VersionObject

表 17　VersionObject プロパティ

名前	説明
Abbrev	名前
Descr	説明
HierCount	階層の数
ID	ID
NodeCount	ノードの数

表 18　VersionObject メソッド

名前	説明
GetHierarchies()	現在のユーザーが使用可能なバージョンのすべての階層の配列を取得します
GetGlobalNodes()	バージョンのすべてのグローバル・ノード(NodeObjects)の配列を取得します
GetOrphans()	バージョンのすべての孤立(NodeObjects)の配列を取得します
HierByAbbrev(abbrev)	HierarchyObject を名前ごとに取得します

名前	説明
HierByID(id)	HierarchyObject を ID ごとに取得します
NodeByAbbrev(abbrev)	NodeObject を名前ごとに取得します
NodeByID(id)	NodeObject を ID ごとに取得します
NodeExists(abbrev)	指定された名前のグローバル・ノードが存在する場合は True を戻します
Prop(abbrev)	バージョンの指定されたプロパティの NodePropObject を取得します
PropValue(abbrev)	バージョンの指定されたプロパティの値を取得します。戻り値の型はプロパティ定義のデータ型によって異なります。

HierarchyObject

表 19　HierarchyObject プロパティ

名前	説明
Abbrev	名前
Descr	説明
HierarchyUrl	階層 URL
ID	ID
NodeCount	階層のノード数
SharedNodesEnabled	共有ノードを有効にする場合は True
TopNode	LocalNodeObject の最上位ノード
Version	VersionObject
VersionAbbrev	バージョンの名前
VersionID	バージョンの ID

表 20　HierarchyObject メソッド

名前	説明
NodeByAbbrev(abbrev)	NodeObject を名前ごとに取得します
NodeByID(id)	NodeObject を ID ごとに取得します
NodeExists(abbrev)	指定された名前のグローバル・ノードが存在する場合は True を戻します
Prop(abbrev)	バージョンの指定されたプロパティの NodePropObject を取得します
PropValue(abbrev)	バージョンの指定されたプロパティの値を取得します。戻り値の型はプロパティ定義のデータ型によって異なります。

共通のノード・プロパティおよびメソッド

NodeObject および LocalNodeObject の 2 つのオブジェクトではプロトタイプ・チェーンは共有されていませんが、一部のプロパティおよびメソッドは、この両方のオブジェクトに共通です。

コンテキストがグローバルであるかローカルであるかによって値が異なるあらゆるケースで、そのコンテキストに適切な値が戻されます。たとえば、NodeObject の GetChildren()を呼び出すと、結果の Array には NodeObject が含まれます。LocalNodeObject で同じ呼出しを行うと、結果の配列には LocalNodeObject が含まれます。

表 21　NodeObject および LocalNodeObject の共通プロパティ

名前	説明
Abbrev	Core.Abbrev
AddedBy	Core.AddedBy
AddedOn	Core.AddedOn
Changed	Core.Changed
ChangedBy	Core.ChangedBy
ChangedOn	Core.ChangedOn
ChildNodeCount	子ノードの数
Descr	Core.Descr
DomainAbbrev	Core.DomainAbbrev
DomainNodeAbbrev	Core.DomainNodeAbbrev
ID	Core.ID
Inactive	Core.Inactive
IsPrimary	ノードが共有ノードのプライマリである場合は True; ノードが共有されていない場合やプライマリでない場合は False
IsShared	ノードが共有ノードの場合は True
Leaf	Core.Leaf
NodeApproved	Core.NodeApproved
Version	ノードの所有者は VersionObject
VersionAbbrev	ノードのバージョン名
VersionID	ノードのバージョン ID

表 22　NodeObject および LocalNodeObject の共通メソッド

名前	説明
GetChildren(sorted)	このノードの直接の子の Array を取得し、オプションでソート順にできます。sorted のデフォルトは False です。
GetDescendants(inclusive, sorted)	このノードの子孫の Array を取得し、オプションでこのノードを含めること/ソート順にすることが可能です。inclusive のデフォルトは True です。sorted のデフォルトは False です。
NodeByAbbrev(abbrev)	NodeObject を名前ごとに取得します
NodeByID(id)	NodeObject を ID ごとに取得します
NodeExists(abbrev)	指定された名前のグローバル・ノードが存在する場合は True を戻します
Prop(abbrev)	バージョンの指定されたプロパティの NodePropObject を取得します
PropValue(abbrev)	バージョンの指定されたプロパティの値を取得します。戻り値の型はプロパティ定義のデータ型によって異なります。

LocalNodeObject

階層内の他のノードの検索には、様々な xxxWith 関数を使用することをお薦めします。たとえば、ChildrenWith は、GetChildren()を呼び出して結果を反復するよりはるかに高速で実行されます。同様に、GetReferenceInHier も、GetReferences()を呼び出して結果を反復するより、はるかに高速で簡単に使用できます。

表 23　LocalNodeObject プロパティ

名前	説明
GlobalNode	現在のノードのグローバル NodeObject
Hier	ノードが存在する階層の HierarchyObject
HierAbbrev	Core.HierAbbrev
HierID	Core.HierID
レベル	階層におけるノードのレベルを表す数値
MissingPrimary	プライマリ・ノードが見つからない場合は True
NodeUrl	ノード URL
Parent	このノードの親ノードの LocalNodeObject。階層の最上位ノードには Null が戻されます。
ParentNodeAbbrev	親ノードの名前
Primary	この共有ノードのプライマリ・ノード。プライマリがこの階層に存在しない場合は、それが存在する最初の階層のプライマリが戻されます。プライマリが存在する階層のリストが必要な場合は、戻されたプライマリ・ノードで GetReferences()を呼び出します。共有ノードまたはプライマリが見つからない場合は、null が戻されます。
PrimaryNotInHier	プライマリ・ノードは存在するが、この階層ではない場合は True

表 24　LocalNodeObject メソッド

名前	説明
AncestorsWith(func, maxResults, searchFromTop, inclusive)	指定された関数を満たすノードの祖先チェーンを検索します。これは、祖先を検索する最速の方法です。LocalNodeObject の結果の Array が戻されます。 ● func は単一のノード引数を使用する関数であることが必要で、結果にノードが含まれる必要がある場合は True が、テストに失敗した場合は False が戻されます。 ● maxResults はオプションで、デフォルトで 1 に設定されます。制限なし(条件を満たすすべてのノード)の場合は 0 を使用します。 ● searchFromTop はオプションで、デフォルトで False に設定されます。階層の最上位で開始する場合は True を使用します。 ● inclusive はオプションで、デフォルトで False に設定されます。潜在的な一致に現在のノードを含めるには True を使用します(テストは通過する必要があります)。
ChildrenWith(func, maxResults)	指定された関数を満たすノードの子リストを検索します。これは、子を検索する最速の方法です。LocalNodeObject の結果の Array が戻されます。 ● func は単一のノード引数を使用する関数であることが必要で、結果にノードが含まれる必要がある場合は True が、テストに失敗した場合は False が戻されます。 ● maxResults はオプションで、デフォルトで 1 に設定されます。制限なし(条件を満たすすべての子)の場合は 0 を使用します。
DescendantsWith(func, maxResults, inclusive, depthFirst)	指定された関数を満たすノードの子孫チェーンを検索します。これは、子孫を検索する最速の方法で、LocalNodeObject の結果の Array が戻されます。 ● func は単一のノード引数を使用する関数であることが必要で、結果にノードが含まれる必要がある場合は True が、テストに失敗した場合は False が戻されます。 ● maxResults はオプションで、デフォルトで 1 に設定されます。制限なし(条件を満たすすべてのノード)の場合は 0 を使用します。 ● inclusive はオプションで、デフォルトで False に設定されます。潜在的な一致に現在のノードを含めるには True を使用します(テストは通過する必要があります)。 ● depthFirst はオプションで、デフォルトで True に設定されます。True の場合、ツリーをバックアップして次の分岐に移動する前に、各分岐がヒントまですべて調査されます。False の場合は、ノードのすべての子が先に調査され、次に各子のノードが、その次にその子というように調査されます。ノードがツリー形式になっている場所がわかる場合は、ここで適切な値を選択すると、検索を大幅に高速化できます。
GetAncestorEnumerator()	祖先ノードを列挙する NodeEnumeratorObject を取得します
GetAncestors(inclusive)	LocalNodeObject の祖先の Array を取得します
GetChildEnumerator(sorted)	子ノードを列挙する NodeEnumeratorObject を取得します。sorted が True の場合、子はソートされた順序で表示されます。
GetDescendantEnumerator()	子孫ノードを列挙する NodeEnumeratorObject を取得します
GetImplicitly SharedDescendants(inclusive)	この共有ノードが関連するプライマリ・ノードの子ノードを取得します
GetInvertedLevel()	式 InvertedLevel 関数と等価です

名前	説明
GetReferences()	このノード(このノードが表示されるすべての階層)の参照先である LocalNodeObjects の Array を取得します
GetReferenceInHier(hierAbbrev)	指定された階層で、このノードへの参照を取得します。階層にアクセスできない場合や、階層にこのノードが存在しない場合、結果は null になります。
NextSibling()	ソート順でこのノードの次の兄弟を取得します
PreviousSibling()	ソート順でこのノードの前の兄弟を取得します
SiblingsWith(func, maxResults, inclusive)	指定された関数を満たすノードの兄弟を検索します。LocalNodeObject の結果の Array が戻されます。 ● func は単一のノード引数を使用する関数であることが必要で、結果にノードが含まれる必要がある場合は True が、テストに失敗した場合は False が戻されます。 ● maxResults はオプションで、デフォルトで 1 に設定されます。制限なし(条件を満たすすべての祖先)の場合は 0 を使用します。 ● inclusive はオプションで、デフォルトで False に設定されます。潜在的な一致に現在のノードを含めるには True を使用します(テストは通過する必要があります)。

NodePropObject

表 25　NodePropObject プロパティ

名前	説明
Abbrev	プロパティ定義の名前
ControllingHierarchy	このバージョンにおけるプロパティ定義の制御階層の HierarchyObject。プロパティがグローバル・ノード・プロパティではない場合、制御階層がない場合、または制御階層が見つからない場合、戻り値は null になります。
Locked	値がロックされている場合は True
Origin	PropOrigin.Overridden などの PropOrigin 列挙値(「列挙定数」を参照)
Owner	この値が関連付けられているオブジェクト(VersionObject、HierarchyObject、NodeObject または LocalNodeObject)
PropType	PropType.Defined などの PropType 列挙値(「列挙定数」を参照)
StringValue	このプロパティの raw 文字列値。Derived または RWDerived プロパティの場合は、プロパティ定義のデフォルト値または上書きされた値になります。
Value	このプロパティの解釈された値(たとえば、DataType.Float および DataType.Integer の場合、この値は数値オブジェクトになります)。すべての DataTypes に、必ずしも文字列以外の表現があるとはかぎりません。

表 26　NodePropObject メソッド

名前	説明
GetPropDef()	ノード・プロパティの PropDefObject を取得します

RangeListObject

RangeListObject は値の RangeList を表し、手動で文字列を解析することなく、RangeList プロパティの検査に使用できます。適切なデータ型の派生プロパティから戻されるよう、新しい RangeListObject を構成することも可能です。

コンストラクタの例

```
var x = new RangeListObject();


var y = new RangeListObject("1-10,20-25");


var z = new RangeListObject([{start:1, end:10},{start:20, end:25}]);
```

表 27　RangeListObject コンストラクタ・パラメータ

パラメータ	オプション	説明
ranges	True	初期化の範囲の値。このパラメータはオプションです。2 つのフォーマットを使用できます: ● Array--配列の各要素が、範囲を示す開始と終了のプロパティを持つオブジェクトである配列。これらのプロパティがない配列のオブジェクトは無視されます。 ● String--文字列エントリのカンマ区切りリスト。各エントリには、ダッシュ(-)または等号(=)記号で区切られた、開始値と終了値が含まれます。

表 28　RangeListObject プロパティ

名前	説明
Ranges	オブジェクトの配列。各オブジェクトには 2 つのプロパティがあります: ● start--範囲エントリの開始 ● end--範囲エントリの終了 このプロパティは読取り専用です。範囲を変更するには、次のメソッドを使用します。

表 29　RangeListObject メソッド

名前	説明
AddRange(start, end)	範囲リストに新しい範囲を追加します。これにより、既存の範囲エントリを拡張するか、新たに作成します。リストに 1 つの数字を追加するには、開始と終了の両方のパラメータにその数字使用します。必要な場合は、両方のパラメータが整数に強制されます。
Contains(value)	値が範囲リスト内にある場合は True、それ以外の場合は False が戻されます。 必要な場合は、値が整数に強制されます。
IsSupersetOf(range)	現在の RangeListObject が、指定された RangeListObject のスーパーセットである場合は True が戻されます。別のタイプのオブジェクトを渡すとエラーが発生します。
RemoveRange(start, end)	リストから範囲を削除します。この削除により、既存の範囲エントリが 2 つに分割されるか、エントリ全体が削除されます。リストから 1 つの数字を削除するには、開始と終了の両方のパラメータにその数字を使用します。必要な場合は、両方のパラメータが整数に強制されます。

NodeEnumeratorObject

NodeEnumeratorObject を使用すると、ノードのリストをより効率的に操作できます。一度にリスト全体をすべて作成するかわりに、列挙子が必要に応じて一度に 1 つのノードを取得します。リストの中ほどで探していたものが見つかった場合は、列挙子を中断できます。ノード・オブジェクトの Array を戻すプロパティおよびメソッドは、配列の最後のアイテムにアクセスするかどうかに関係なく、配列全体をすぐに作成する必要があります。

列挙子は、現在の値を null として開始されます。列挙子をリストの最初のノードに進めるには、MoveNext()を呼び出す必要があります。最後に達すると、GetCurrent()メソッドにより再度 null が戻されます。

注： 検索するノードが、可能性のあるすべての一致の中のいくつかのみで、リストを反復するのが一度のみの場合は、AncestorsWith または SiblingsWith メソッドなど、With メソッドを使用することをお薦めします。祖先ノードのリストを何度も繰り返す必要がある場合や、祖先の大部分またはすべてが必要であることがわかっている場合は、列挙子の方が高速です。

表 30　NodeEnumeratorObject メソッド

名前	説明
GetCurrent()	現在のノード(コンテキストにより、NodeObject または LocalNodeObject)。現在のノードがない場合、またはリストの最後に到達した場合は、null が戻されます。
MoveNext()	列挙子を次のノードに進めます。列挙するノードが存在しない場合は False が戻されます。

ValidationObject

- ブール値(True または False)を戻すことができます。True を戻す場合、検証は成功です; それ以外の場合は失敗です。値を戻さない場合は、False が戻されたのと同じであるとみなされます。
- 数値や文字列など、ブール以外のオブジェクトを戻すと、ブールに変換されてから戻されます。標準の JavaScript 変換が適用されます。ゼロに等しい数値、空の文字列、null または未定義のオブジェクトは false と解釈されます。その他のすべての値は true です。
- 「success」という名前のプロパティを含む複合オブジェクトを戻す場合、その success プロパティはブールに変換され、検証の戻り値として使用されます。「parameters」という名前のプロパティに、オプションで、値の配列を戻すことができます。パラメータは、文字列置換を使用して、検証のエラー・メッセージに置き換えられます。エラー・メッセージのプレースホルダに対応する、適切な数の値を戻す必要があります。余分なパラメータを戻した場合は無視されます。十分な数のパラメータを戻さないと、不足しているパラメータは空の文字列とみなされます。エラー・メッセージのパラメータでは、標準のフォーマット指定子を使用でき、たとえば{0:00}は、数値パラメータに 2 つの十進数がフォーマットされます。

表 31　ValidationObject メソッド

名前	説明
Abbrev	検証の名前(完全修飾ネームスペースを含む)
Descr	説明
EditorLabel	ラベル
Cascade	検証の割当てを継承する場合は True
ValidationClass	検証クラスの名前
ValidationLevel	ValidationLevel.Node などの ValidationLevel 列挙値(列挙定数を参照)。

実行環境

Data Relationship Management エンジンはマルチスレッド化された、複数マシンの環境で、スクリプトは複数のスレッドでマシン全体にわたって同時に実行されます。値を作成してグローバル・スコープに格納できますが、別のスレッドでスクリプトが実行される場合にはそのグローバル値が存在しないため、この動作に依存することはお薦めできません。同様に、Data Relationship Management エンジンのインスタンスやマシン全体でグローバル値が更新されるわけではありません。さらに、Data Relationship Management では、複数のライブ・バージョンがサポートされているため、ノードの値の計算やその値をグローバル・スコープに格納することに依存していると、別のスクリプトが他のノードのプロパティにアクセスした場合には、誤った値が生成されます。

注： 変数をグローバル・スコープに格納しないことと同じ理由で、すべてのエンジン・インスタンスおよびスレッド全体に変更が行われたことを確認できないため、組込みの Data Relationship Management オブジェクト・プロトタイプも変更しないことをお薦めします。

スクリプト・タイムアウトの設定

エンジンが過度にロックされないよう、値を戻さずに長時間実行されるスクリプトは、タイムアウト設定に基づいて終了されます。スクリプト・タイムアウトは、プロパティ定義および検証ごとに設定できます。

タイムアウトは実行コンテキストごとにであるため、エクスポートで 100 ノードのスクリプト・プロパティをエクスポートしていて、そのプロパティのタイムアウトが 30 秒に設定されている場合、各ノードでのプロパティの検証に 30 秒かかるため、そのエクスポートには最大 50 分かかります。ただし、スクリプト・プロパティで別のスクリプト・プロパティが呼び出される場合には、タイムアウトは増加しません。たとえば、PropA のタイムアウトが 10 秒、PropB のタイムアウトが 20 秒で、長時間の計算を開始する PropB を PropA が呼び出した場合、10 秒が経過すると、元のタイムアウトが過ぎているため PropA の検証は終了します。

無限ループの防止

無限ループ(スタック・オーバーフローとも呼ばれる)に陥るスクリプトは、サーバー・プロセスが予期せず終了する原因となる重大なエラーです。Data Relationship Management はそのようなスクリプトが実行されないよう試みますが、自己参照を行うスクリプトや再帰的なスクリプトを記述する際には注意が必要です。本番環境にデプロイする前に、開発環境で必ず新しいスクリプトをテストしてください。

無限にループするスクリプトの簡単な例を次に示します。スクリプトにそのスクリプト自体の呼出しが含まれていますが、実行が終了されないため、関数を実行しているエンジンが、最終的にリソース不足で停止します。最後に、スクリプトでは Data Relationship Management エンジンが呼び出されないため、オーバーフローを捕捉してスクリプトを停止する機会がありません。

```
function badFunc(a) { badFunc(a); }


badFunc("oops");
```

パフォーマンスに関する考慮事項

最適なパフォーマンスを実現するため、式から派生したプロパティとスクリプトの間で相互に参照することは回避してください。一般的にスクリプトは、64 ビット・プロセッサを含むネイティブ・ハードウェアの Just-In-Time (JIT)コンパイルなどの考慮事項のため、式と比較して、パフォーマンス・チューニングの最適化の非常によい機会です。また、スクリプトは、実際の実行特性に合せて JIT コン

パイラによりチューニングされるため、時間の経過とともにより短時間で実行されるようになります。

動的スクリプトの作成

動的スクリプトは、派生プロパティの定義と検証用の「パラメータ」タブからアクセス可能なスクリプト・エディタで作成します。

➤ 動的スクリプトを作成するには:

1 **テキスト領域にスクリプトを入力します。**

注： プロパティを挿入するには、スクリプトにカーソルを置いて「プロパティの挿入」をクリックします。プロパティのリストが表示されます。プロパティを選択して「OK」をクリックします。

2 **次のオプションから選択します:**

- 「スクリプト・タイムアウト」--スクリプトがタイムアウトするまでの秒数。
- 選択したノードでスクリプトを評価するには、[...]をクリックし、ノードを選択します。スクリプトではノードの現在のプロパティ値が使用されます。「評価」をクリックします。結果は、スクリプト・デザイナの下部に表示されます。

3 **スクリプトをテストするには、「評価」をクリックします。**

11 ノード・タイプの管理

この章の内容

ノード・タイプを利用すると、その関係と属性に基づいて、階層ノードを別々に表示し管理することができます。特定のノード・タイプのノードは、次の同じ特性を共有します。

- プロパティ
- 検証
- グリフ

階層はノード・タイプの異なるノードを持つことができ、同じノードが異なる階層の異なるノード・タイプでもかまいません。ノード・タイプの使用例としては、GL 勘定科目、費用センター、連結エンティティ、製品グループ、予測ポイントなどがあります。

ノードをノード・タイプ別に分類するには:

1. 階層内でノードの分類に必要なノード・タイプを決定します。
2. 各ノード・タイプに関連する(または関連しない)プロパティを特定します。
3. 各ノード・タイプに関連する(または関連しない)検証を特定します。
4. 必要に応じて、各ノード・タイプにグリフを割り当てます。

ノード・タイプの定義

➤ ノード・タイプを定義するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「ノード・タイプ」を選択します。

3 ノード・タイプの名前と説明を入力します。

4 「オプション」: ノード・タイプに使用するグリフを選択します。

5 「プロパティ」タブで、ノード・タイプに関連付けるプロパティを「使用可能」リストから選択します。矢印を使用して、プロパティを「選択済」リストに移動します。

6 「検証」タブで、ノード・タイプに関連付ける検証を「使用可能」リストから選択します。矢印を使用して、検証を「選択済」リストに移動します。

7 「保存」をクリックします。

ノード・タイプの編集

➤ ノード・タイプを編集するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「ノード・タイプ」を展開します。

3 ノード・タイプを選択して  をクリックします。

4 次のいずれかの操作を行います。

- 説明を編集します。
- ノード・タイプに使用するグリフを変更
- プロパティを追加または削除
- 検証を追加または削除

5 「保存」をクリックします。

ノード・タイプの削除

➤ ノード・タイプを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「ノード・タイプ」を展開します。

3 ノード・タイプを選択して  をクリックします。

4 「このアイテムを削除」をクリックし、削除を確定します。

ノード・グリフの使用

グリフは、ノード・タイプに関連付けられ、Data Relationship Management のユーザー・インタフェースでノードのアイコンとして表示される画像です。新しいグリフを作成することも、既存のグリフを修正することもできます。不要になったグリフの削除も可能です。グリフは、PNG フォーマットで指定する必要があります。

➤ ノード・グリフを追加するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「グリフ」を選択します。

3 グリフの名前を入力し、説明を追加します。

4 「参照」をクリックして PNG ファイルを選択します。

5 「アップロード」をクリックします。

6 「保存」をクリックします。

➤ ノード・グリフを変更するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「グリフ」を展開します。

3 グリフを選択して  をクリックします。

4 「参照」をクリックして別の PNG ファイルを選択します。

5 「アップロード」をクリックします。

6 「保存」をクリックします。

➤ グリフを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「グリフ」を展開します。

3 グリフを選択して  をクリックします。

4 「このアイテムを削除」をクリックし、削除を確定します。

12 システム・プリファレンスの操作

この章の内容

「システム・プリファレンス」で、管理ユーザーは Data Relationship Management の動作を制御する設定を編集できます。

システム・プリファレンス

次の表に、Data Relationship Management のシステム・プリファレンスをまとめます。

表 32 システム・プリファレンス

システム・プリファレンス	タイプ	説明
AllowAsOf	ブール	True の場合、コア・アクションの取得を強制してベースライン・バージョンを作成し、「時点」バージョンの作成を許可します。このプリファレンスを False に設定すると、「時点」バージョンは作成できません。 デフォルト値は True です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
AllowNextIDGeneration	ブール	True の場合、次の ID を自動生成できます。 デフォルト値は False です。
AllowNextIDKeyCreation	役割	NextID 機能で新しいキーを作成できる役割のリスト。 デフォルト値は「インタラクティブ・ユーザー」、「データ作成者」、「データ・マネージャ」です。
AllowPru	ブール	True の場合、子があるノードを非管理ユーザーが削除できる削除オプションが有効です。False の場合、子があるノードを非管理ユーザーは削除できません。 デフォルト値は True です。
AllowRelaxedMove	ブール	True の場合、ノードを移動するとき、他の階層にあるノードの競合する親関係よりも新しい親が優先されるようにします。 デフォルト値は False です。

システム・プリファレンス	タイプ	説明
AllwSpac	ブール	True の場合、ノード名に空白を使用できます。 デフォルトは True です。
ApprovalGroups	文字列	承認グループのカンマ区切りのリスト。
ApprovalGroupTrackProperties	文字列	グループによって追跡される承認プロパティの区切りリスト。
ApprovalPropertyByApprovalGroup	文字列	承認グループ別のグローバル・ブール承認プロパティ。
AuthMethod	文字列	ユーザー認証の方法: ● 内部--ユーザーは Data Relationship Management 内でのみ認証されます。 ● CSS (外部)--ユーザーは外部でのみ認証されます。共有サービスへのアクセスが必要です。 ● 混合--ユーザーは、ユーザーごとの設定に基づいて内部と外部で認証されます。 デフォルト値は「内部」です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
CopyLcl	ブール	True の場合、ノードをコピーする際にローカル値がコピーされます。 デフォルト値は True です。
DefaultCurrentVersion	バージョン	デフォルトの現在のバージョン。このプリファレンスは、バージョンの「デフォルトに設定」を使用して設定できます。
DefaultPreviousVersion	バージョン	デフォルトの前のバージョン。このプリファレンスは、バージョンの「デフォルトに設定」を使用して設定できます。
DefaultPropCopyMode	文字列	デフォルトのプロパティ・コピー・モード。 有効な値は「上書き済」、「選択済」、「すべて強制」です。 デフォルト値は「上書き済」です。
EnablePropCopyOptions	役割	プロパティのコピー・オプションにアクセスできる役割のリスト。 デフォルト値は「インタラクティブ・ユーザー」、「データ作成者」、「データ・マネージャ」です。
EnforceListProps	ブール	True の場合、事前定義済リストのみからの値でリスト・プロパティを更新できます。 デフォルト値は True です。
FiltrChr	文字列	エクスポートの「出力オプション」画面における「置換」機能用の文字のセット。

システム・プリファレンス	タイプ	説明
FindByProperties	プロパティ	階層を参照するとき検索に使用できるプロパティのリスト。 **注：** 表示されるのは、ユーザーにアクセス権があるプロパティです。また、表示されるプロパティはすべての階層に適用されるとはかぎりません。
FindWildCardAppend	ブール	True の場合、「完全一致」を選択していないときに「検索」基準の後ろにアスタリスク(*)が追加されます。 デフォルト値は False です。
FindWildCardPrepend	ブール	True の場合、「完全一致」を選択していないときに「検索」基準の前にアスタリスク(*)が追加されます。 デフォルト値は False です。
GlobalPropLocalOverride	プロパティ	グローバル・プロパティに対するローカル・チェックから除外するプロパティのリスト。GlobalPropLocalSecurity が有効なときに使用されます。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
GlobalPropLocalSecurity	ブール	True の場合、グローバル・プロパティに対してローカル・セキュリティが適用されます。グローバル・プロパティに対する変更は、ノードが存在するすべての階層で、ユーザーのローカル・セキュリティ(ノード・アクセス・レベル)に対してチェックされます。 デフォルト値は False です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
HierSep	文字列	階層とノードの区切り文字。 デフォルト値はチルダ(~)です。
IdleTime	整数	アプリケーション・サーバーにおけるセッション・タイムアウトまでの分数。 デフォルト値は 60 です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
非アクティブ化	役割	ノードの非アクティブ化を許可される役割のリスト。 デフォルト値はすべての役割です。
InactiveChanges	役割	非アクティブなノードの変更を許可される役割のリスト。 デフォルト値は「データ・マネージャ」、「アプリケーション管理者」、「アクセス・マネージャ」です。
InvDescr	文字列	ノード説明プロパティで無効な文字のリスト。
InvName	文字列	ノード名で無効な文字のリスト。
JobResultsRetentionAge	整数	アーカイブされたジョブ結果詳細を履歴で保持する日数。値ゼロはジョブ結果が履歴からは削除されないことを示します。

システム・プリファレンス	タイプ	説明
LeafEdit	役割	リーフ・プロパティの変更を許可される役割のリスト。 デフォルト値は「データ・マネージャ」、「データ作成者」、「アプリケーション管理者」、「アクセス・マネージャ」です。
LockoutInactivity	整数	非アクティブなユーザーがロックアウトされるまでの最大日数。 最大値は 30 です。ゼロの場合は最大数を設定しません。
LockoutInvalidLogins	整数	ユーザーがロックアウトされるまでの無効なログインの最大回数。 最大値は 6 です。ゼロの場合は最大数を設定しません。
LossLevel	文字列	取り込む損失レベル。 有効な値は次のとおりです。 ● 定義済 ● すべて デフォルト値は「定義済」です。「すべて」を選択すると、多くのプロパティ値を持つノードの移動または削除で、システム・パフォーマンスに大きく影響することがあります。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
MaxDescr	整数	ノードの説明の最大文字数。有効な値は 12 から 255 です。 デフォルト値は 80 です。
MaxLeaf	整数	リーフ名の最大文字数。有効な値は 3 から 20 です。 デフォルト値は 255 です。
MaxLimb	整数	リム名の最大文字数。有効な値は 3 から 20 です。 デフォルト値は 255 です。
PasswordDuration	整数	ユーザーのパスワードが有効な日数。有効な値は 1 から 9999 です。 デフォルト値は 30 です。
PasswordMaxLength	整数	ユーザー・パスワードの最大文字数。有効な値は 0 から 255 です。ゼロの場合は最小数を設定しません。 デフォルト値はゼロです。
PasswordMinLength	整数	ユーザー・パスワードの最小文字数。有効な値は 0 から 9999 です。ゼロの場合は最小数を設定しません。 デフォルト値は 6 です。
PasswordPolicyEnabled	ブール	True の場合、次の要素のうち 3 つをパスワードに含める必要があります。 ● 大文字 ● 小文字 ● 数字 ● 特殊文字 デフォルト値は True です。

システム・プリファレンス	タイプ	説明
PasswordWarningPeriod	整数	正または負の数。実際にログインできなくなるパスワード有効期限日の何日前(-)または何日後(+)に、パスワードを変更するようユーザーに警告するかを指定します。有効な値は-30 から 30 です。 デフォルト値は 1 です。
RenameLeaf	役割	リーフ・ノードの名前を変更できる役割のリスト。 デフォルト値は「データ・マネージャ」、「アプリケーション管理者」、「アクセス・マネージャ」です。
RenameLimb	役割	リム・ノードの名前を変更できる役割のリスト。 デフォルト値はすべての役割です。
ReqMerge	ブール	True の場合、UseMerge が有効なときに非アクティブ化または削除にマージが必要です。 デフォルト値は False です。
SharedNodeDelimiter	文字列	ノード名とノード接尾辞の間の区切る文字を指定します。 デフォルト値はコロン(:)です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
SharedNodeIdentifier	文字列	共有ノードの区切り文字の後に使用する識別子を指定します。 デフォルト値は「共有」です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
SharedNodeMaintenanceEnabled	ブール	True の場合、共有ノードを有効にします。 デフォルト値は False です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
SharedNodeNamingType	文字列	共有ノードの代替名を指定します。有効な値は、「接尾辞」または「接頭辞」です。 デフォルトは「接尾辞」です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
SharedNodeSequenceLength	整数	数値シーケンス・タイプを使用する場合の一意性キーの長さを指定します。 デフォルト値は 3 です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
SharedNodeSequenceSeparator	文字列	共有ノード識別子の後に配置される区切り文字を指定します。 デフォルト値はダッシュ(-)です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。

システム・プリファレンス	タイプ	説明
SharedNodeSequenceType	文字列	一意性キーのタイプを指定します。有効な値は、「数値」または「祖先」です。 デフォルトは「数値」です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
SortLimbsFirst	ブール	True の場合、最初にリム・ノードのソートを制御し、その後リーフ・ノードのソートを制御します。False の場合、リム・ノードとリーフ・ノードを同時にソートできます。このプリファレンスは、階層のエクスポート、表示、ノード・リストに影響します。 デフォルト値は True です。
TopNodeParentString	文字列	「インポート」と「エクスポート」で、最上位ノードの親値を示すために使用されます。 デフォルト値は「なし」です。
TransactionLevels	文字列	取り込むトランザクション・レベルのリスト。「時点」をオンに設定するか、結果アクションまたは損失アクションを指定すると、コア・アクションが取得されます。 有効な値は次のとおりです。 ● ログに記録されたアクション ● コア・アクション ● 結果アクション ● 損失アクション デフォルト値は「ログに記録されたアクション」、「コア・アクション」、「結果アクション」、「損失アクション」です。 **注：** このプリファレンスを変更するには、Data Relationship Management アプリケーションの再起動が必要です。
UpName	ブール	True の場合、ノード名に常に大文字を使用します。 デフォルト値は False です。
UseChangeApproval	ブール	True の場合、変更承認が有効です。 デフォルト値は False です。
UseMerge	ブール	True の場合、非アクティブなノードと削除されたノードに対して「マージ」手法を使用できます。 **注：** ReqMerge が True の場合、マージ・ノードを指定する必要があります。ReqMerge が False の場合、ノードの承認済プロパティが True でないかぎり、マージ・ノードはオプションです。バージョンがファイナライズされるとき、または適切なアクセス権を持つユーザーによって明示的に True に設定されるとき、ノードの承認済プロパティは True に設定されます。 デフォルト値は False です。
ValSec	ブール	True の場合、ノード・アクセス・グループのセキュリティをチェックして、ユーザーがノードのバッチ検証を実行できるかどうかを判断します。 デフォルト値は False です。

システム・プリファレンス	タイプ	説明
WarnHL	整数	「子孫」、「子」、「クエリー結果」などのリストに表示されるノードの最大数。最小値は 1000 です。1000 より少ない値に設定すると、1000 ノードが表示されます。 デフォルト値は 5000 です。

詳細は、次を参照してください:

- トランザクション履歴ロギング・レベルの設定
- 変更承認の設定
- 「グローバル・プロパティ」でのローカル・セキュリティ

トランザクション履歴ロギング・レベルの設定

Data Relationship Management トランザクション履歴ロギング・レベルを設定するには、アプリケーション管理者の権限が必要です。トランザクション履歴に取り込むアクション・タイプを指定するには、「TransactionLevels」システム・プリファレンスを設定します。

➤ トランザクション履歴ロギング・レベルを設定するには:

1 **Data Relationship Management Web クライアントで、「管理」を選択します。**

2 **「メタデータ」の下で、「システム・プリファレンス」を展開して「TransactionLevels」プリファレンスを編集します。**

3 **「TransactionLevels」で、次のトランザクション・レベル・タイプを選択します:**

- **「ロギング・アクション」**を指定すると、基本的なロギング情報(ログイン、ログアウトなど)が記録されます。
- **「コア・アクション」**を指定すると、バージョン、階層、またはノードの情報を変更するアクション(ノードの追加、プロパティの変更、ノードの移動など)が記録されます。
- **「結果アクション」**を指定すると、コア・アクションから生じたアクションが記録されます。たとえば、下のすべてを消去するコア・アクションが実行されると、個々のノードからプロパティが消去されます。個々のノードからプロパティを消去するアクションが、結果アクションです。
- **「損失アクション」**を指定すると、コア・アクションによるデータの損失が記録されます。たとえば、ノードが削除されると、そのノードの定義済プロパティが削除されます。このアクションが、損失アクションです。損失アクションは、「LossLevel」システム・プリファレンスで制御されます。

  注: 「損失アクション」を指定したり、AllowAsOf システム・プリファレンスを有効にすると、TransactionLevels システム・プリファレンスで設定されていない場合でもコア・アクションが追跡されます。

4 **LossLevel プリファレンスの設定:**

- **定義済**--特にノードで設定されている値のみがノードの削除時に追跡されます。
- **すべてのアイテム**--派生した値、デフォルト値および継承値が LossAction で追跡されます。

5 **アプリケーションを停止して再起動するか、Data Relationship Management サービスを再起動します。**

変更承認の設定

Data Relationship Management には変更承認のシステムがあり、承認グループを定義して、一連のプロパティまたは指定されたアクションによってトリガーされる承認フラグにそのグループを結び付けることができます。それにより通常のユーザーが変更を行うことができ、承認者はクエリーを実行して必要に応じて承認フラグを設定できます。

Data Relationship Management で、変更承認の動作を決定するシステム・プリファレンスは次のとおりです。

- UseChangeApproval--変更承認の使用をオンにする場合は、True に設定します。
- ApprovalGroups--システムで使用される承認グループの名前のカンマ区切りリスト。
- ApprovalGroupTrackProperties--UseChangeApproval が True の場合、このグループに対する承認フラグを False に変更するトリガーとして追跡されるプロパティを定義します。フォーマットは xxx[a,b,c],yyy[d,e,f]...です。このとき、xxx と yyy は ApprovalGroups プリファレンスで定義される販売グループ、a、b、c、d、e、f はプロパティ名です。たとえば、Sales[Custom.SalesGroup,{NodeMove}],Treasury[Custom.AccountDescription,{NodeAdd}]とします。

  プロパティ・リストに含めることができる特別なアクションは、次のとおりです。

  - {NodeAdd}--追加されるノードで承認必須のメカニズムをトリガーします。
  - {NodeInactivate}--非アクティブ化されるノードで承認必須のメカニズムをトリガーします。
  - {NodeReactivate}--再アクティブ化されるノードで承認必須のメカニズムをトリガーします。
  - {NodeInsert}--挿入されるノードで承認必須のメカニズムをトリガーします。
  - {NodeRemove}--削除されるノードで承認必須のメカニズムをトリガーします。
  - {NodeMove}--移動されるノードで承認必須のメカニズムをトリガーします。
- ApprovalPropertyByApprovalGroup--UseChangeApproval が True の場合、トリガー・プロパティのいずれかが変更されたとき、または特別アクションが使用されたときに False に設定される、グローバル・ブール・プロパティを定義します。フォーマットは xxx:bbbb,yyy:cccc…です。このとき、xxx と yyy は ApprovalGroups プリファレンスで定義される販売グループ、bbbb と cccc はグ

ループの承認フラグを格納するためのグローバル・ブール・プロパティの名前です。たとえば、Sales:Custom.SalesApprovedFlag,Treasury:Custom.TreasuryApprovedFlag とします。

「グローバル・プロパティ」でのローカル・セキュリティ

グローバル・プロパティでローカル・セキュリティを制御するには、GlobalPropLocalSecurity と GlobalPropLocalOverride の 2 つのシステム・プリファレンスを使用します。

システム・プリファレンスの構成

➤ システム・プリファレンスを構成するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「システム・プリファレンス」を展開します。

3 システム・プリファレンスを選択して  をクリックします。

4 値を変更して「保存」をクリックします。

13 外部接続の使用

この章の内容

アプリケーション管理者は、外部のファイル・システムとデータベースに対してよく使用する接続を定義し、構成することができます。インポート、エクスポート、ブックが接続を共有すれば、接続情報のメンテナンスも最小限で済みます。外部接続を使用すると、アプリケーション・サーバーからネットワーク・リソースに対して直接のアクセス、読取り、書込みが可能です。

注： 外部接続を定義する前に、外部リソースを設定する必要があります。

外部接続の定義

➤ 外部接続を定義するには:

1 「ホーム」ページで、「管理」を選択します。

2 「新規」から「外部接続」を選択します。

3 名前と説明を入力します。

4 「オブジェクトのアクセス権」から、「標準」、「システム」またはカスタム・グループを選択します。

5 接続タイプを「サーバー・ファイル」、「FTP」、「データベース・テーブル」の中から選択します。

6 次を実行します:

- 「サーバー・ファイル」を選択した場合は、サーバーへの UNC パスを入力します。

注： 「サーバー・ファイル」接続には、Data Relationship Management アプリケーション・サーバーで使用する Windows ユーザー・アカウントが自動的に使用されます。Windows のサービス**「Oracle DRM サーバー・プロセス」**に使用されるデフォルトの Windows ユーザー・アカウントは、**ローカル・システム・アカウント**です。「サーバー・ファイル」に適切に接続するには、サービスに使用されるアカウントが UNC パスにアクセスできる必要があります。また、サービス・アカウントがファイルに対して読取りと書込むを行うには、UNC パスに適切な権限が必要です。

- 「FTP」を選択した場合は、次の情報を入力します。
  - ホスト・サーバー
  - ユーザー ID
  - ユーザー・パスワード
- 「データベース・テーブル」を選択した場合は、次を行います:
  - 「データ・アクセス・プロバイダ」を選択: 「Oracle」、「SqlServer」、または「OleDb」
  - 接続文字列の入力
  - ユーザー ID の入力
  - ユーザー・パスワードの入力
  -  をクリックし、「使用可能」リストからテーブルを選択します。矢印を使用して、テーブルを「選択済」リストに移動します。

7 **「接続のテスト」をクリックします。**

注： 「データベース」接続の場合は、「リフレッシュ」をクリックしてテーブルのリストを取得します。

外部接続の編集

➤ 外部接続を編集するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「メタデータ」の下で、「外部接続」を展開します。**

3 **外部接続を選択して**  **をクリックします。**

4 **必要な変更を行います。**

5 **「保存」をクリックします。**

外部接続の削除

外部接続を削除すると、その接続を使用しているインポート・プロファイルとエクスポート・プロファイルがすべて影響されます。

➤ 外部接続を削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「メタデータ」の下で、「外部接続」を展開します。

3 外部接続を選択して  をクリックします。

4 「このアイテムを削除」を選択し、削除を確認します。

14 ガバナンス・ワークフローの構成

この章の内容

ガバナンス・ワークフローは、ノード、関係およびプロパティ値への変更の入力、承認、検証およびコミットメントの制御に使用される形式化されたプロセスです。

アプリケーション管理者は、ワークフロー・タスクおよびワークフロー・モデルを定義して、ビジネス・ユーザーが送信した変更要求とデータ・スチュワードが送信した改善要求を管理します。

ワークフロー・タスクの管理

ワークフロー・タスクは、要求のコンテキスト内のローカル・ノードに対してユーザーが実行する、単一の変更セットです。要求の要求アイテムは、ワークフロー・タスクによって制御されます。

ワークフロー・タスクは、アクション・タイプ、表示または編集するプロパティ、および検証で構成されます。ワークフロー・タスクのアクション・タイプは、ノードの追加、移動または更新など、実行するアクションの基本的なタイプを指定します。各アクション・タイプは、ノードおよび親の選択、プロパティ更新のアプリケーション、要求の検証およびコミット時に実行するアクションに関するルールを定義します。

タスク・プロパティは、プロパティ値を表示専用にするか編集可能にするかを制御するために構成します。編集可能なプロパティは、必要に応じて構成できます。アクション・タイプのデフォルト・プロパティは、タスクから削除できません。タスク検証は、特定のワークフロー・ステージに対して要求を承認する前に、要求アイテム用に正常に実行する必要がある、オプションのノード・レベルのバッチ検証です。関連付けられたプロパティ値にエラーを関連付けるため、検証ごとに、タスク・プロパティが割り当てられます。検証は、要求の要求アイテムに関連付けられたローカル・ノードに対してのみ実行されます。

詳細は、次を参照してください:

- ワークフロー・タスクの作成
- ワークフロー・タスクの編集
- ワークフロー・タスクの削除

ワークフロー・タスクの作成

➤ ワークフロー・タスクを作成するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「新規」から、「ワークフロー・タスク」を選択します。**

3 **ワークフロー・タスクの名前を入力します。**

4 **「アクション・タイプ」から、次に示すタスクのアクションのタイプを選択します:**

- 「リーフの追加」--グローバル・プロパティとローカル・プロパティを含むリーフ・ノードを追加します
- 「リムの追加」--グローバル・プロパティとローカル・プロパティを含むリム・ノードを追加します
- 「削除」--ノードのグローバル/ローカル・プロパティを更新し、ノードを削除します
- 「非アクティブ化」--ノードのグローバル・プロパティおよびローカル・プロパティを更新し、ノードを非アクティブ化します
- 「挿入」--ノードを階層に挿入し、グローバル/ローカル・プロパティを更新します
- 「移動」--ノードを異なる親に移動し、グローバル/ローカル・プロパティを更新します
- 「除去」--ノードのグローバル/ローカル・プロパティを更新し、ノードを除去します
- 「更新」--ノードのグローバル・プロパティおよびローカル・プロパティを更新します

5 **「オプション」:「指示」フィールドに、ユーザー用のテキストを入力します。**

6 **「プロパティ」タブで、「使用可能」リストからプロパティを選択して、タスクに割り当てます。矢印を使用して、プロパティを「選択済」リストに移動します。上下の矢印を使用して、プロパティを順序付けします。**

7 **をクリックして、プロパティの「編集可能」オプションまたは「必須」オプションを変更します。**

をクリックして変更を保存するか、をクリックして変更を取り消します。

8 **「検証」タブで、「使用可能」リストから検証を選択して、タスクに割り当てます。矢印を使用して、検証を「選択済」リストに移動します。上下の矢印を使用して、プロパティを順序付けします。**

9 **をクリックして、検証のプロパティを変更します。**

をクリックして変更を保存するか、をクリックして変更を取り消します。

10 **をクリックしてワークフロー・タスクを保存します。**

ワークフロー・タスクの編集

➤ ワークフロー・タスクを編集するには:

1 「ホーム」ページで、「管理」を選択します。

2 「ワークフロー」の下で、「ワークフロー・タスク」を展開します。

3 タスクを選択し、をクリックします。

4 「プロパティ」タブおよび「検証」タブで、プロパティおよび検証の選択を変更します。

5 をクリックします。

ワークフロー・タスクの削除

➤ ワークフロー・タスクを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「ワークフロー」の下で、「ワークフロー・タスク」を展開します。

3 ワークフロー・タスクを選択し、をクリックします。

4 「ワークフロー・タスクの削除」をクリックして、削除を確認します。

ワークフロー・モデルの管理

ワークフロー・モデルは、モデルに基づき、1 つの要求に一緒に含めることが可能な定義済タイプの一連の変更管理タスクを定義します。モデルは、変更をバージョンにコミットする前に必要な一連の承認およびエンリッチメントのステップを定義します。ワークフロー・ステージは、ワークフロー・モデルごとに定義され、ワークフロー・モデル間で共有できません。

ワークフロー・ステージ

ステージがワークフロー・モデルに割り当てられると、ステージ・タイプ属性は、ワークフローのそのステージにおけるユーザーに対して、参加のタイプを定義します。

表 33 ワークフロー・ステージ

ワークフロー・ステージ	説明	アクション・タイプ
送信	送信ページは、要求に含める最初の要求アイテムの定義に使用されます。このステージ・タイプには、複数のワークフロー・タスクを関連付けられます。送信ステージ中に、少なくとも 1 つの要求アイテムを要求に追加する必要があります。 **注：** 各要求の送信ステージは 1 つのみです。	● リムの追加 ● リーフの追加 ● 更新 ● 非アクティブ化 ● 挿入 ● 移動 ● 除去 ● 削除
エンリッチ	エンリッチ・ステージは、送信ステージで追加された要求アイテムの更新、または要求アイテムの追加に使用されます。 エンリッチ・ステージには、関連付けられた単一のワークフロー・タスクがあります。一般的なエンリッチ・ステージでは、既存の要求アイテムの更新アクションを含むワークフロー・タスクを使用します。ただし、エンリッチ・ステージでは、たとえば次のような追加のライン・アイテムを作成する必要がある場合があります: ● 複数の階層への単一ノードの挿入 ● 複数の階層での単一ノードのローカル・プロパティの更新 このステージは、送信ステージとコミット・ステージの間で発生します。 **注：** ワークフロー・モデルには、任意の数のエンリッチ・ステージを定義できます。	● 更新(既存の要求アイテム) ● 挿入(新規アイテムの追加) ● 移動(既存の要求アイテム)
承認	承認ステージは、送信ステージで追加されたか、エンリッチ・ステージで追加または更新されたすべての要求アイテムを表示および承認するために使用されます。ユーザーは、承認ステージ中に要求アイテムを追加または編集できません。 承認ステージには、関連付けられた更新タスクに基づく単一のワークフロー・タスクがあります。 このステージは、送信ステージとコミット・ステージの間で発生します。 **注：** ワークフロー・モデルには、任意の数の承認ステージを定義できます。	更新(既存の要求アイテム)
コミット	コミット・ステージは、ターゲット・バージョンへの要求で要求アイテムのコミットをトリガーする際に要求の最終承認を行うために使用されます。コミット・ユーザーは、要求でのすべての要求アイテムを承認する必要があります。 コミット・ステージには、関連付けられたワークフロー・タスクはありません。かわりに、コミット・ステージでは、プロパティのスーパーセットを表示し、前の送信ステージおよびエンリッチ・ステージの要求アイテムで使用できる検証のスーパーセットを実行します。コミット・ステージにおけるユーザーは、最終調整を許可するために要求アイテムに対して表示される、編集可能なプロパティへの更新を行うことができます。 これは、最後のワークフロー・ステージです。 **注：** 各要求のコミット・ステージは 1 つのみです。	N/A

ワークフロー・モデルの作成

➤ ワークフロー・モデルを作成するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 **「新規」から、「ワークフロー・モデル」を選択します。**

3 **ワークフロー・モデルの名前を入力します。説明はオプションです。**

4 **オプション: 次のオプションに日数を入力します:**

- 「要求期間」--要求が期限切れとみなされるタイミングを示します。要求期間が要求の経過時間より少ない場合、その要求は「遅滞」とマークされます。
- 「請求期間」--請求された要求が自動的に未請求になり、他のユーザーが請求に使用可能になるタイミングを示します。ステージ期間が請求期間を超える場合、要求は自動的に未請求となります。

注: いずれかのオプションの値が 0 の場合、期限切れおよび自動未請求の機能はワークフロー・モデルに対して無効になることを示します。

5 **「ワークフロー・ステージ」タブで、ワークフロー・ステージごとに、をクリックし、次のオプションを構成します:**

- 「ワークフロー・タスク」--ステージに割り当てるワークフロー・タスクを選択します。各ステージに割り当てることができるアクションの詳細は、ワークフロー・ステージを参照してください。
- 「ノード・アクセス」--ワークフロー・ステージに関連付けるワークフロー・ノード・アクセス・グループを選択します。ワークフロー・タイプのノード・アクセス・グループのみをステージに割り当てることができます。
- 「ワークフロー・メソッド」--次のいずれかのオプションを選択して、要求のステージを承認する必要があるノード・アクセス・グループを指定します:
  - 「任意のグループ」--割り当てられたノード・アクセス・グループの任意のユーザーは、次のワークフロー・ステージに進めるために要求を承認できます。ノード・アクセス・グループを、現在のステージ・タイプまたはそれを超えるステージ・タイプへのアクセス権がある階層に割り当てる必要があります。ステージに割り当てられたいずれのアクセス・グループにも、要求内の要求アイテムへの適切なデータ・アクセス権がない場合、必要な値が指定され、すべての要求アイテムに対する検証が成功しているかぎり、ステージはスキップされます。
  - 「すべてのグループ」--割り当てられたすべてのノード・アクセス・グループのうち少なくとも 1 つのユーザーが、次のステージに進む前に要求を承認する必要があります。ステージに割り当てられたいずれのアクセス・グループにも、要求内の要求アイテムへの適切なデータ・アクセス権がない場合、その要求はデータ・マネージャにエスカレートされて解決されます。

- 「再承認」--要求を前のステージに戻し、戻す間に要求アイテムが変更される場合、その要求への変更は、元の要求に対する最初の承認を行ったユーザーによって再承認される必要があります。このオプションによって、戻す間に各ステージで行われた変更が他のユーザーによって再承認される必要があるかどうかが決まります。次のいずれかのオプションを選択してください:
  - 「現在」--このステージでの要求への変更は、現在のステージでのみ再承認される必要があります。承認後、要求は、前に要求を戻したユーザーに割り当てられます。
  - 「すべて」--このステージでの要求への変更は、後続のステージで再承認される必要があります。
- 「通知」--アラートおよび通知を、ワークフロー・ステージのワークフロー・ユーザーに送信するかどうかと、そのタイミングを決定します。要求がワークフロー・パス内で異なるステップ(ワークフロー・ステージおよびユーザー・グループ)に移動するたびに、アラートおよび通知を送信できます。次のいずれかのオプションを選択します:
  - 「なし」--このワークフロー・ステージに対して実行されたアクションについて、ユーザーは通知されません。
  - 「担当者」--要求を割り当てられたユーザーが通知されます。
  - 「参加者」--要求に参加したユーザーが通知されます。
  - 「担当者および参加者」--担当者と参加者が通知されます。

6 **「オプション」: エンリッチメント・ステージと承認ステージをワークフロー・モデルに追加するには、「追加」をクリックして、追加された新しいステージごとにステップ 5 に従います。**

7 **「オプション II」: ユーザーが表示および選択できるバージョン、階層およびノードのタイプを特定の要求のタイプに制限するには、「フィルタ」タブで選択を行います:**

- 「バージョン変数」--特定のワークフロー・モデルの要求の要求アイテムに対して、バージョンの選択を制限します。
- 「階層グループ・プロパティ」--特定のワークフロー・モデルの要求の要求アイテムに対して、ノードを選択できる階層を制限します。
- 「階層グループ」--「階層グループ・プロパティ」を指定した場合は必要です。
- 「ノード・タイプ」--特定のワークフロー・モデルの要求への要求アイテムとして追加できるノードを制限します。

8 **  をクリックしてワークフロー・モデルを保存します。**

ワークフロー・モデルの編集

➤ ワークフロー・モデルを編集するには:

1 **「ホーム」ページで、「管理」を選択します。**

2 「ワークフロー」の下で、「ワークフロー・モデル」を展開します。

3 モデルを選択し、をクリックします。

4 ワークフロー・モデルを変更し、をクリックします。

ワークフロー・モデルの削除

➤ ワークフロー・モデルを削除するには:

1 「ホーム」ページで、「管理」を選択します。

2 「ワークフロー」の下で、「ワークフロー・モデル」を展開します。

3 モデルを選択し、をクリックします。

4 「ワークフロー・タスクの削除」をクリックして、削除を確認します。

15 外部ワークフロー・アプリケーションの統合

外部ワークフロー・アプリケーションは、外部ソースから Data Relationship Management に対して提案された変更の処理に使用できます。Web サービス API は、複数の変更をグループ化して検証し、外部ワークフローの処理中に単一の作業単位としてコミットできる外部要求インタフェースを提供します。API ユーザーが外部要求を行うには、ワークフロー・ユーザー役割が必要です。この要求インタフェースは汎用であり、ワークフロー・モデル、ワークフロー・タスクまたは Web クライアントのワークリスト・ページを使用することはできません。これらの汎用的な外部要求が記録され、アクセスは「要求履歴」からのみになります。

外部要求に関する API サポートの詳細は、Oracle Data Relationship Management API リファレンスを参照してください。

次のために外部要求を作成できます:

- 階層の追加
- ノードの追加
- ノードの挿入および移動
- ノードのアクティブ化、非アクティブ化および除去
- プロパティの更新
- プロパティ値の削除

外部要求は、承認のためにドラフト状態で格納し、変更をバージョンに即時コミットせずに Data Relationship Management バージョンに対して検証できます。この保留状態の外部要求は、複数のユーザーが別々のタイミングで更新し、必要に応じて再検証できます。リクエスト内のトランザクションは、リクエストの承認時に Data Relationship Management バージョンに対してコミットされます。

注： 外部要求が承認された後、要求は変更できず、関連するバージョンが削除されるまでは要求は削除できません。

外部要求は、次の要素で構成されています:

- ターゲット Data Relationship Management バージョン。
- リクエストの所有者: 有効な Data Relationship Management ユーザー ID。
- カスタム・ワークフロー ID: ワークフロー・アプリケーション内のリクエストの識別子。
- カスタム・ワークフロー・ラベル: ワークフロー・アプリケーション内のリクエストの簡単な説明。

- カスタム・ワークフロー・ステータス: ワークフロー・アプリケーション内のリクエストのステータスを管理します。
- カスタム・ワークフロー情報: ワークフロー・アプリケーションに必要な追加情報を格納します。
- リクエスト・コメント: リクエストの注釈。
- 作成者: 初期リクエストを作成したユーザー。
- 作成日: リクエストが作成された日付。
- 更新者: リクエストを最後に更新したユーザー。
- 更新日: リクエストが最後に更新された日付。
- 承認者: リクエストを承認したユーザー。
- 承認日: リクエストが承認された日付。
- 検証済フラグ: リクエストが最後に更新されてから検証されたかどうかを示します。
- 承認済フラグ: リクエストが承認されたかどうかを示します。
- 検証または承認操作時にリクエスト内のアクションに対してのみ適用する必要がある追加バッチ検証
- 現在のリクエストの階層およびノードに影響するアクション・アイテムのリスト

16 Data Relationship Managementメタデータの移行

この章の内容

アプリケーション管理者は、Data Relationship Management の移行ユーティリティを使用して、Data Relationship Management アプリケーション間でメタデータ・オブジェクト・タイプを移動できます。

移行ユーティリティでは、次が可能です:

- Data Relationship Management アプリケーションから XML ファイルにメタデータ・オブジェクト・タイプを抽出し、その結果から HTML レポートを生成
- XML ファイルから Data Relationship Management アプリケーションにメタデータをロード
- 2 つのソース間でメタデータの差分を比較し、差分を使用して XML ファイルを作成し、その結果から HTML レポートを生成
- XML ファイルでメタデータを表示し、ファイルから HTML レポートを生成

次のタイプのメタデータを抽出、ロード、比較および表示できます。

- プロパティ定義
- プロパティ・カテゴリ
- 検証
- ノード・タイプ
- グリフ
- ノード・アクセス・グループ
- 階層グループ
- クエリー(標準、システムおよびカスタム)
- 比較(標準、システムおよびカスタム)

- ドメイン
- バージョン変数(標準、システムおよびカスタム)
- エクスポート(標準、システムおよびカスタム)
- エクスポート・ブック(標準、システムおよびカスタム)
- インポート(標準、システムおよびカスタム)
- ブレンダ(標準、システムおよびカスタム)
- システム・プリファレンス
- 外部接続(標準、システムおよびカスタム)

  注： 外部接続で表示されるのは接続名のみです; オブジェクト・アクセス・グループ名の接頭辞は追加されません。
- オブジェクト・アクセス・グループ
- ワークフロー・タスク
- ワークフロー・モデル

移行ユーティリティを開く

デフォルトでは、移行ユーティリティは次の場所にインストールされます。

```
MIDDLEWARE_HOME\EPMSystem11R1\products\DataRelationshipManagement
\client
```

➤ 移行ユーティリティを開くには、「Data Relationship Management 移行ユーティリティ」をクリックします。

メタデータの抽出

Data Relationship Management アプリケーションから抽出するメタデータのタイプを選択できます。情報は XML ファイルに抽出され、それを表示する、他の Data Relationship Management アプリケーションにロードする、他の XML と比較する、あるいは他の Data Relationship Management アプリケーションと比較することができます。このファイルは、バックアップ、格納、監査の目的にも使用できます。

作成される XML ファイルの情報からレポートを生成できます。

➤ Data Relationship Management アプリケーションからメタデータを抽出するには:

1 **「メイン・メニュー」で「抽出」をクリックします。**

2 **Data Relationship Management の接続情報を入力し、「ログイン」をクリックします。**

3 **オブジェクト・タイプまたは抽出するオブジェクトを選択し、「次」をクリックします。**

注： 階層ツリーのプラス記号をクリックすると、オブジェクトが表示されます。オブジェクト・タイプのチェック・ボックスを選択してオブジェクト・タイプとそのオブジェクトをすべて選択するか、抽出するオブジェクトのチェック・ボックスを選択します。オブジェクト名をクリックすると、オブジェクト・タイプ定義が新しいウィンドウで表示されます。

4 **オプション: メタデータ・オブジェクト・タイプまたはオブジェクトを検索する場合は、「検索」をクリックします。**

注： 入力したテキストを含むオブジェクト・タイプが戻されます。結果の中で特定のオブジェクトに移動するには、「ジャンプ先」リンクをクリックします。

5 **要約情報を確認します。**

注： 移行ユーティリティは、依存性を持つオブジェクト・タイプに対して別途チェックを実行します。たとえば、エクスポートがプロパティ定義に依存している、またはプロパティ定義が他のプロパティ定義を参照している場合があります。要約に依存性が欠落している場合は、含める特定の依存性を選択できます。除外されている依存性をすべて含めることも、すべての依存性を除外することもできます。

注： ページ・サイズを大きくすると、1 ページに表示するオブジェクト・タイプの数を定義できます。

6 **オプション: この抽出のメタデータ詳細を入力します。**

入力できる情報は次のとおりです。

- 「タイトル」--最大 255 文字
- 「目的」--フォーマット済メモ
- 「使用状況」--フォーマット済メモ
- 「アプリケーション・バージョン」--最大 20 文字
- 「ファイル・バージョン」--最大 20 文字

7 **「抽出の実行」をクリックします。**

8 **次のいずれかの操作を行います。**

- XML ファイルを開くか、保存する場合は、**「メタデータ・ファイルのダウンロード」**をクリックします。
- XML ファイルの詳細を表示する場合は、**「メタデータ・ファイルの表示」**をクリックします。
- XML ファイルを Data Relationship Management アプリケーションにロードする場合は、**「メタデータ・ファイルのロード」**をクリックします。詳細は186 ページの「メタデータのロード」を参照してください。
- XML ファイルからレポートを生成する場合は、**「メタデータ・ファイルのレポートの生成」**をクリックします。詳細は189 ページの「レポートの生成」を参照してください。

メタデータのロード

Data Relationship Management アプリケーションにロードできるのは、Data Relationship Management XML 形式のファイルのみです。ロードの実行後にはログ・ファイルが作成され、データの重要度として監査、情報、警告、エラー・メッセージのいずれかが表示されます。

注： メタデータ・ファイルをロードする前に、以前の構成に戻す場合のために、既存メタデータの抽出を実行することをお薦めします。メタデータをロードする前に、特に移行ファイルを本番環境にロードする場合には、データベースのバックアップを実行するのが理想的です。

➤ XML ファイルから Data Relationship Management アプリケーションにメタデータをロードするには:

1 「メイン・メニュー」で**「ロード」**をクリックします。

2 **「参照」**をクリックし、ロードする XML ファイルを選択して**「アップロード」**をクリックします。

3 アップロードされるファイルの情報を確認し、「次」をクリックします。

4 Data Relationship Management の接続情報を入力し、**「ログイン」**をクリックします。

5 オブジェクト・タイプまたはロードするオブジェクトを選択し、**「次」**をクリックします。

注： 階層ツリーのプラス記号をクリックすると、オブジェクトが表示されます。オブジェクト・タイプのチェック・ボックスを選択してオブジェクト・タイプとそのオブジェクトをすべて選択するか、ロードするオブジェクトのチェック・ボックスを選択します。オブジェクト名をクリックすると、オブジェクト・タイプ定義が新しいウィンドウで表示されます。

6 要約情報を確認し、「次」をクリックします。

注： ページ・サイズで、1 ページに表示するオブジェクト・タイプの数を定義できます。

7 **オプション:** エラーが発生した場合でもロードを続行するには、**「エラー後もロードを続行」**を選択します。

8 **「ロードの実行」**をクリックします。

9 ロード結果を確認します。

表示する詳細の重要度を監査、情報、警告およびエラーの中から選択すると、ログ・ファイルの表示を変更できます。ログ・ファイルを保存するには、**「ダウンロード」**をクリックします。

注： 列見出しのリンクを使用すると、任意の列を基準にしてログ・アイテムをソートできます。

メタデータの比較

2 つのメタデータ・ソースを比較できます。メタデータの差異を比較できるのは、2 つの Data Relationship Management アプリケーション間、2 つの XML ファイル間、または Data Relationship Management アプリケーションと XML ファイルの間です。2 つのメタデータ・ソース間の差異を含む XML ファイルを生成できます。その結果を使用すると、データを復元する、無許可の変更を元に戻す、または誤ったオブジェクト・タイプ構成を検査することができます。

作成される XML ファイルの情報からレポートを生成できます。

➤ メタデータを比較するには:

1 **「メイン・メニュー」で「差分」をクリックします。**

2 **「ソース#1」ドロップダウン・リストから、ソースのタイプとして「サーバー接続」または「XML ファイル」を選択します。**

3 **次のいずれかの操作を行います:**

- **「サーバー接続」**を選択した場合は、Data Relationship Management の接続情報を入力して**「ログイン」**をクリックします。
- **「XML ファイル」**を選択した場合は、**「参照」**をクリックし、比較に使用する XML ファイルを選択して**「アップロード」**をクリックします。

4 **ファイルをアップロードした場合は、アップロードしたファイルの情報を確認して「次」をクリックします。それ以外の場合は、次の手順にスキップします。**

5 **「ソース#2」についても、2 から 4 の手順を繰り返します。**

6 **「次」.をクリックします。**

7 **次の操作で、差分ファイルに含めるオブジェクト・タイプを選択します。**

- フィルタを選択します。
- 「>」をクリックして、「ソース#1」からオブジェクト・タイプを選択します。
- 「<」をクリックして、「ソース#2」からオブジェクト・タイプを選択します。
- オブジェクト・タイプの選択を解除するには、「X」をクリックします。
- 左の列見出しをクリックすると、選択したフィルタに基づいて「ソース#1」からすべてのオブジェクトが選択されます。
- 右の列見出しをクリックすると、選択したフィルタに基づいて「ソース#2」からすべてのオブジェクトが選択されます。
- 中央の列見出しをクリックすると、選択したフィルタに基づいてすべてのオブジェクトの選択が解除されます。
- 比較結果の最上部にあるページ・リンクをクリックすると、別のページに切り替わります。

注 : ページ・サイズで、1 ページに表示するオブジェクト・タイプの数を定義できます。

8 「差分ファイルの作成」をクリックします。

9 次のいずれかの操作を行います。

- XML ファイルを開くか、保存する場合は、**「メタデータ差分ファイルのダウンロード」**をクリックします。
- XML ファイルの詳細を表示する場合は、**「メタデータ差分ファイルの表示」**をクリックします。
- ファイルを Oracle Data Relationship Management アプリケーションにロードする場合は、**「メタデータ差分ファイルのロード」**をクリックします。詳細は 186 ページの「メタデータのロード」を参照してください。
- XML ファイルからレポートを生成する場合は、**「メタデータ・ファイルのレポートの生成」**をクリックします。詳細は189 ページの「レポートの生成」を参照してください。

メタデータの表示

メタデータ・ファイルを表示し、その情報からレポートを生成できます。

➤ XML ファイルをメタデータの表示するには:

1 「メイン・メニュー」で**「ファイルの表示」**をクリックします。

2 **「参照」**をクリックし、表示する XML ファイルを選択して**「アップロード」**をクリックします。

3 アップロードされるファイルの情報を確認し、「次」をクリックします。

4 階層ツリーのプラス記号をクリックすると、メタデータ・オブジェクトが表示されます。

5 **オプション:** ファイル内でアイテムを検索する場合は、**「検索」**をクリックします。

注: テキストを含むオブジェクト・タイプが戻されます。結果の中で特定のオブジェクトに移動するには、「ジャンプ先」リンクをクリックします。

6 **オプション:** ファイルから HTML レポートを生成するには、「レポート」タブをクリックします。

メタデータ・ファイルの制限

移行ユーティリティでアップロードされるファイルのデフォルトの制限は 4MB です。移行ユーティリティを使用して大きなメタデータ・ファイルをロードまたは表示する際、ファイル・サイズが構成された制限を超えると、次のエラーが発生する可能性があります。

「予期しないエラー: 要求の処理中に予期しないエラーが発生しました: 要求の最大長を超えています。」

大きなファイル・サイズの構成の詳細は、Oracle Data Relationship Management Installation Guide の移行ユーティリティの構成に関する項を参照してください。

レポートの生成

抽出後に生成された XML ファイル、差分レポート、および表示中のメタデータ・ファイルから HTML レポートを生成できます。

➤ HTML レポートを生成するには:

1 **次のいずれかの操作を行います:**

- メタデータの抽出後、または差分レポートの作成後に、「メタデータ・ファイルのレポートの生成」をクリックします。
- メタデータ・ファイルの表示後に、「レポート」をクリックします。

2 **次のいずれかの操作を行います:**

- レポートを表示するには「レポートの表示」をクリックします。
- レポートを保存するには「レポートのダウンロード」をクリックします。

索引

A - Z

あ行

か行

さ行

た行

な行

は行

ま行

や行

わ行