



Documentation Produit

EBX5 Version 5.6.1 Fix A

EBX5 includes:

- software developed by the Apache Software Foundation - <http://www.apache.org/>
- software developed by the JDOM Project - <http://www.jdom.org/>
- software developed by the JSON Project - <http://www.json.org/>
- the Gagawa HTML Generator - <http://code.google.com/p/gagawa/>
- the Jericho HTML Parser - <http://jericho.htmlparser.net/docs/index.html>
- the H2 Database Engine - <http://www.h2database.com/>
- Font Awesome by Dave Gandy - <http://fontawesome.io/>

Table des matières

Guide utilisateur

Introduction

1. Notions clés.....	11
2. Interface utilisateur EBX5.....	15
3. Glossaire.....	19

Modèles de données

4. Introduction aux modèles de données.....	30
---------------------------------------------	----

Implémentation des modèles de données

5. Création du modèle de données.....	35
6. Configuration du modèle de données.....	37
7. Modélisation de la structure des données.....	41
8. Propriétés des éléments du modèle de données.....	47
9. Contrôles sur les éléments du modèle de données.....	61
10. Actions sur les modèles de données existants.....	67

Publication et gestion de versions des modèles de données

11. Publication du modèle de données.....	71
12. Gestion des versions de modèles de données embarqués.....	73

Espaces de données

13. Introduction aux espaces de données.....	78
14. Création d'un espace de données.....	81
15. Actions sur les espaces de données existants.....	83
16. Images.....	93

Jeux de données

17. Introduction aux jeux de données.....	100
18. Création du jeu de données.....	103
19. Visualisation des données.....	105
20. Edition des données.....	113
21. Actions sur les jeux de données existants.....	117
22. Héritage entre jeux de données.....	121

Modèles de workflow

23. Introduction aux modèles de workflow.....	126
24. Modélisation du workflow.....	131
25. Configuration du modèle de workflow.....	141
26. Publication d'un modèle de workflow.....	149

Workflows de données

27. Introduction aux workflows de données.....	152
28. Utilisation de l'interface utilisateur de la section Workflow de données.....	155
29. Bons de travail.....	159

Gestion de workflows de données

30. Lancement et monitoring de workflows de données.....	163
31. Administration de workflows de données.....	165

Services de données

32. Introduction aux services de données.....	170
33. Génération de WSDL pour services de données.....	173

Manuel de référence (en anglais)

Intégration

34. Overview of integration and extension.....	179
35. Using EBX5 as a web component.....	183
36. User interface services (UI services).....	189
37. Built-in UI services.....	199

Services de données

38. Introduction.....	213
39. WSDL generation.....	219
40. Operations.....	227

Services d'import et d'export

41. XML import and export.....	255
42. CSV import and export.....	261
43. Supported XPath syntax.....	267

Localisation

44. Labeling and localization.....	274
45. Extending EBX5 internationalization.....	277

Persistence

46. Overview of persistence.....	280
47. Relational mode.....	283
48. History.....	289
49. Replication.....	297
50. Data model evolutions.....	303

Divers

51. Inheritance and value resolution.....	310
52. Permissions.....	317
53. Criteria editor.....	329
54. Performance guidelines.....	331

Guide d'administration (en anglais)

Installation & configuration

55. Supported environments.....	343
56. Java EE deployment.....	347
57. EBX5 main configuration file.....	361
58. Installing a repository using the configuration assistant.....	375
59. Deploying and registering EBX5 add-ons.....	377

Technical administration

60. Repository administration.....	380
61. Front end administration.....	391
62. Users and roles directory.....	407
63. Audit trail.....	411
64. Data model administration.....	415
65. Data workflow administration.....	417
66. Task scheduler.....	421

Distributed Data Delivery (D3)

67. Introduction to D3.....	428
68. D3 broadcasts and delivery data spaces.....	433
69. D3 administration.....	437

Guide du développeur (en anglais)

70. Notes to developers.....	446
------------------------------	-----

Model design

71. Introduction.....	450
72. Packaging EBX5 modules.....	453
73. Data types.....	457
74. Tables and relationships.....	467
75. Constraints, triggers and functions.....	485
76. Labels and messages.....	503
77. Additional properties.....	509

Java reference

78. Mapping to Java.....	518
79. Tools for Java developers.....	523

Guide utilisateur

Introduction

CHAPITRE 1

Notions clés

Ce chapitre contient les sections suivantes :

1. [Concepts et outils associés](#)
2. [Architecture](#)

1.1 Concepts et outils associés

Le Master Data Management (MDM) est un moyen de modéliser, gérer et gouverner les données partagées. Quand des données sont partagées par plusieurs systèmes informatiques, ainsi que des équipes professionnelles différentes, l'existence d'une seule version gouvernée des données de référence est essentielle.

Avec EBX5, les utilisateurs métier et les informaticiens peuvent collaborer sur une solution unifiée, afin de concevoir des modèles de données et gérer le contenu des données de référence.

EBX5 est un logiciel de gestion des données de référence qui permet de modéliser tout type de données de référence et d'y appliquer une gouvernance grâce à des outils avancés comme le workflow collaboratif, le contrôle de l'édition de données, la gestion hiérarchique des données, le contrôle de version, et la sécurité.

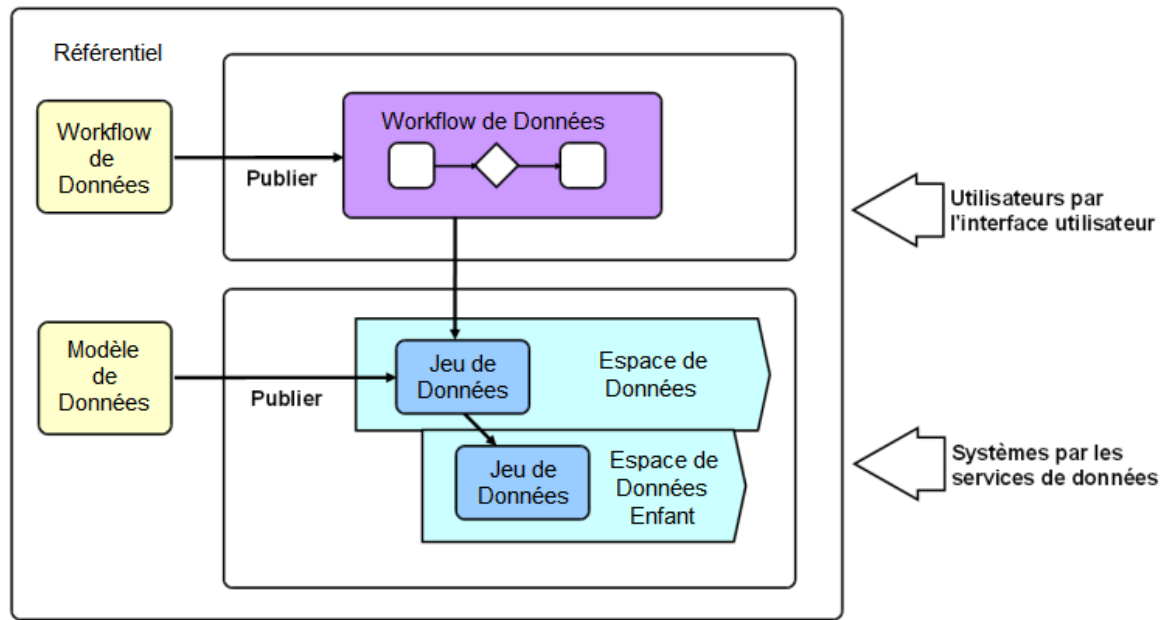
Un projet MDM qui utilise EBX5 commence par la création d'un *modèle de données*. Celui-ci définit les tables, les champs, les liens et les règles métier permettant de décrire les données de référence. De bons exemples sont les catalogues de produits, les hiérarchies financières, les listes de fournisseurs ou simplement les tables de référence.

Ce modèle de données peut ensuite être publié en tant que *jeu de données*, stockant le contenu des données de référence. Les jeux de données sont organisés dans des *espaces de données*. Un espace de données est un conteneur qui permet d'isoler toutes les mises à jour qui sont effectuées. Cela permet de travailler sur plusieurs versions parallèles des données.

Les *workflows* sont indispensables pour effectuer les processus de modification et d'approbation sur les données. Un workflow permet de modéliser un processus étape par étape, comprenant la participation de plusieurs utilisateurs humains et automatisés.

Les *modèles de workflow* définissent les tâches à effectuer, ainsi que les participants associés à chaque tâche. Dès qu'un modèle de workflow est publié, il peut être exécuté en tant que *workflow de données*. Les workflows de données permettent d'envoyer aux utilisateurs des notifications concernant les événements pertinents et les tâches à accomplir, le tout dans un contexte collaboratif.

Les *services de données* aident à intégrer EBX5 à des systèmes tiers ("middleware"), en leur permettant d'accéder aux données, ou de gérer des espaces de données et des workflows.



Voir aussi

[Modèle de données](#) [p 20]

[Jeu de données](#) [p 22]

[Espace de données](#) [p 23]

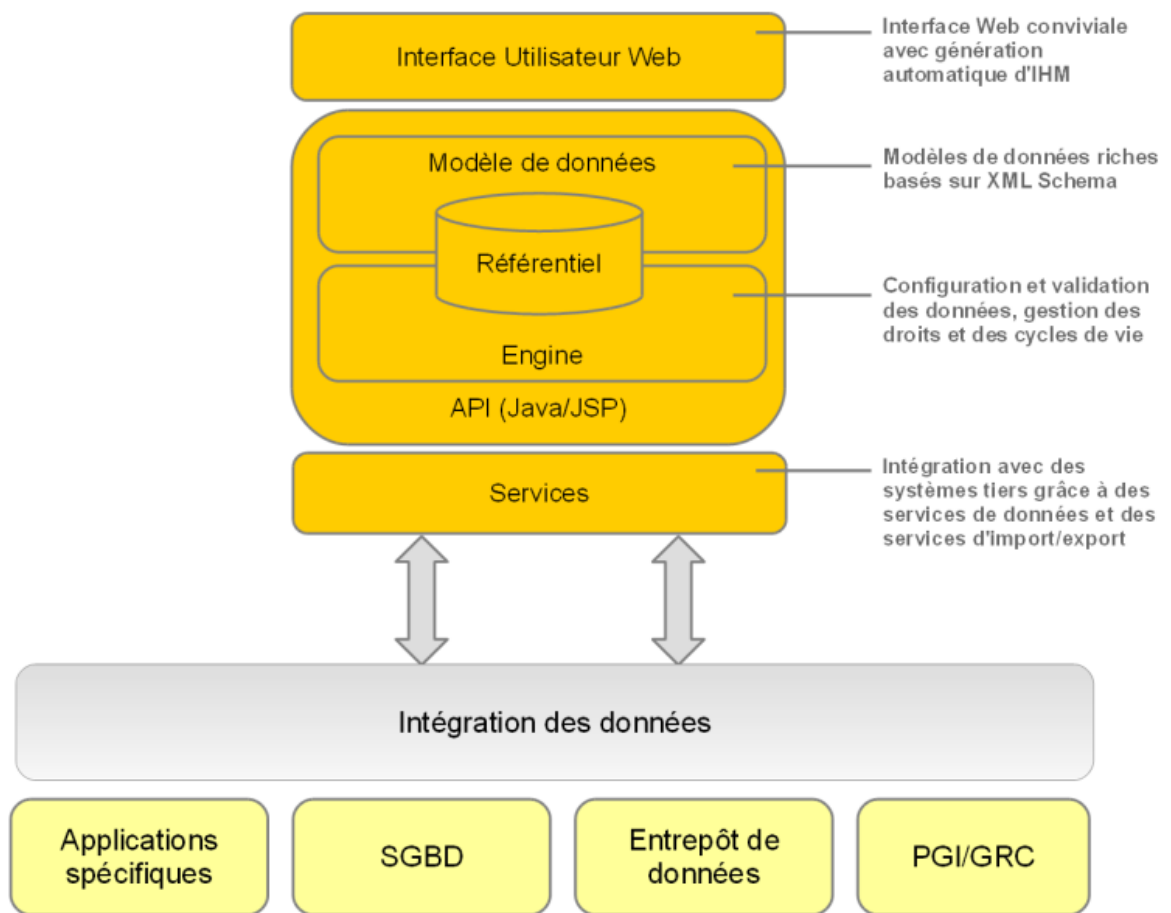
[Modèle de workflow](#) [p 25]

[Workflow de données](#) [p 26]

[Service de données](#) [p 27]

1.2 Architecture

Le schéma suivant présente l'architecture de EBX5.



CHAPITRE 2

Interface utilisateur EBX5

Ce chapitre contient les sections suivantes :

1. [Présentation](#)
2. [Perspective avancée](#)
3. [Perspectives](#)
4. [Fonctionnalités de l'interface utilisateur](#)
5. [Où trouver de l'aide sur EBX5](#)

2.1 Présentation

La mise en page générale des espaces de travail sur EBX5 est entièrement personnalisable par un administrateur. Si des perspectives personnalisées existent, elles peuvent être sélectionnées via le menu déroulant 'Perspective' qui permet de passer d'une vue à l'autre.

La perspective avancée est accessible par défaut.

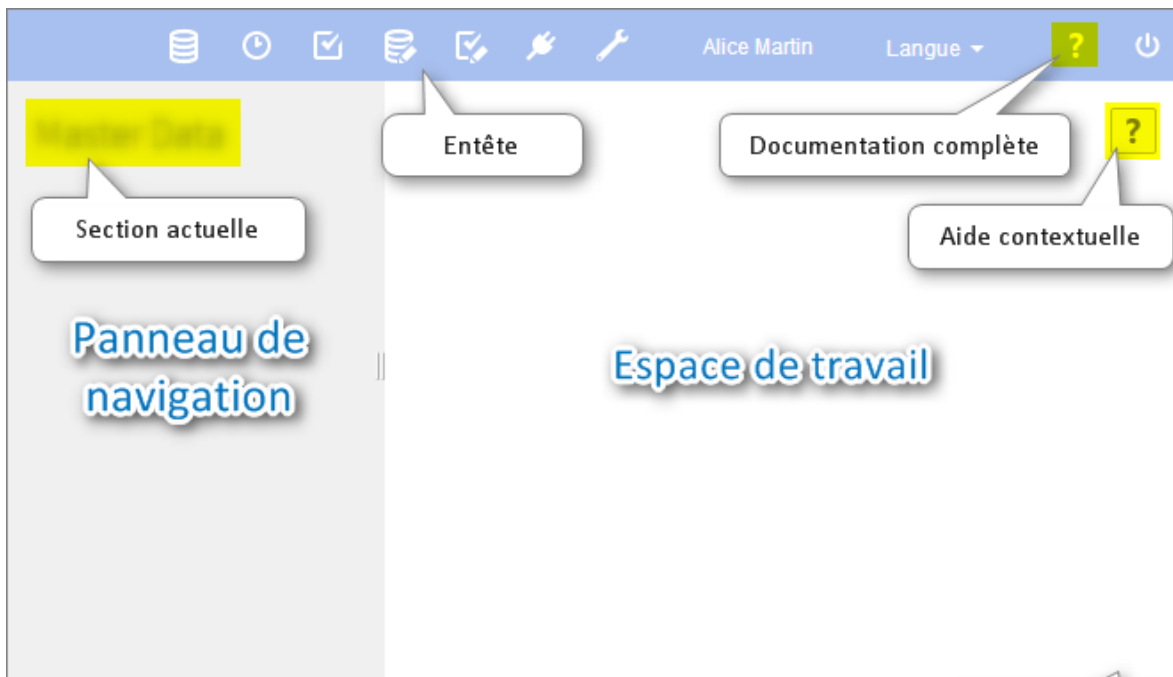
Voir aussi [Front end administration](#) [p 391]

2.2 Perspective avancée

Par défaut, la perspective avancée d'EBX5 est accessible à tous les utilisateurs. Cependant, son accès peut être restreint à certains profils uniquement. Cette vue est divisée en plusieurs zones principales, référencées dans la documentation sous les termes suivants:

- **En-tête** : le nom de l'utilisateur actuel s'affiche dans cette zone, ainsi que le menu de sélection de la langue (lorsque plusieurs sont disponibles), le menu de sélection des perspectives (lorsque plusieurs sont disponibles), un lien vers la documentation et un bouton pour fermer la session en cours.
- **Barre de menu** : cette zone comprend toutes les fonctionnalités accessibles à l'utilisateur actuel et lui permet de naviguer entre elles.
- **Panneau de navigation** : cette zone résume visuellement les diverses possibilités de navigation. Par exemple : sélectionner une table dans un jeu de données, ou un bon de travail dans un workflow.
- **Espace de travail** : zone de travail principale dépendant du contexte. Par exemple, la table sélectionnée dans le panneau de navigation s'affiche dans l'espace de travail, ou bien un bon de travail en cours s'y exécute.

Les sections fonctionnelles suivantes sont affichées dans l'interface selon les permissions de l'utilisateur actuel : *Données*, *Espace de données*, *Modélisation*, *Workflow de données*, *Services de données*, et *Administration*.



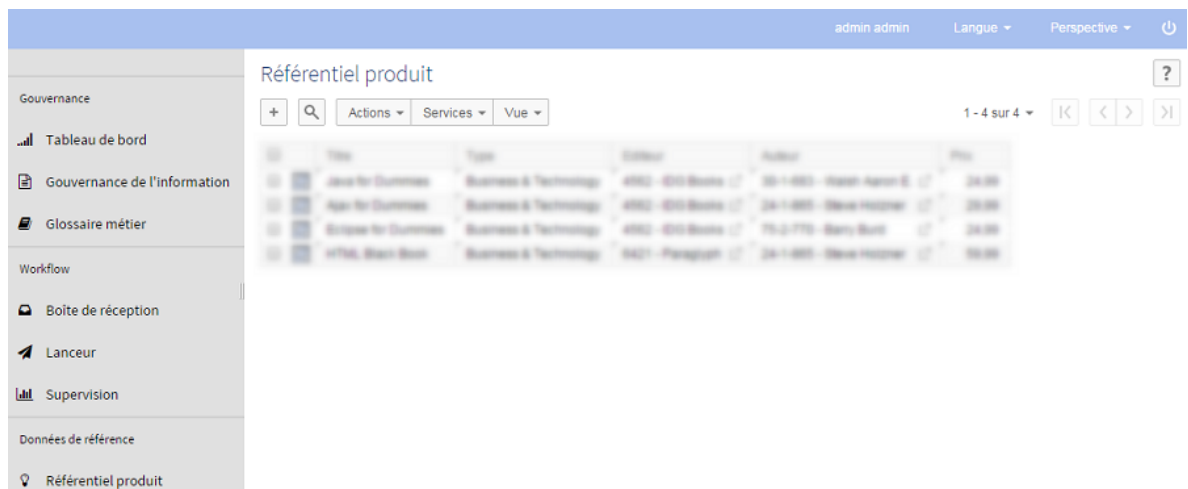
2.3 Perspectives

Les perspectives dans EBX5 sont des vues configurables avec une audience définie. Les perspectives permettent aux utilisateurs métier de bénéficier d'une interface simplifiée. Une perspective peut être affectée à un ou plusieurs profils. Cette vue est divisée en plusieurs zones principales, référencées dans la documentation sous les termes suivants:

- **En-tête** : le nom de l'utilisateur actuel s'affiche dans cette zone, ainsi que le menu de sélection de la langue (lorsque plusieurs sont disponibles), le menu de sélection des perspectives (lorsque plusieurs sont disponibles) et un bouton pour fermer la session en cours.
- **Panneau de navigation** : cette zone affiche le menu hiérarchique tel qu'il a été configuré par l'administrateur de perspectives. Ce panneau peut être développé ou réduit et permet d'accéder aux entités et services correspondant à l'activité de l'utilisateur.
- **Espace de travail** : zone de travail principale dépendant du contexte.

Une perspective est configurée par un utilisateur ayant les autorisations nécessaires. Pour plus d'informations sur la configuration d'une perspective, voir [perspective administration \(en anglais\)](#) [p 393].

Exemple de présentation d'un menu hiérarchique :



2.4 Fonctionnalités de l'interface utilisateur

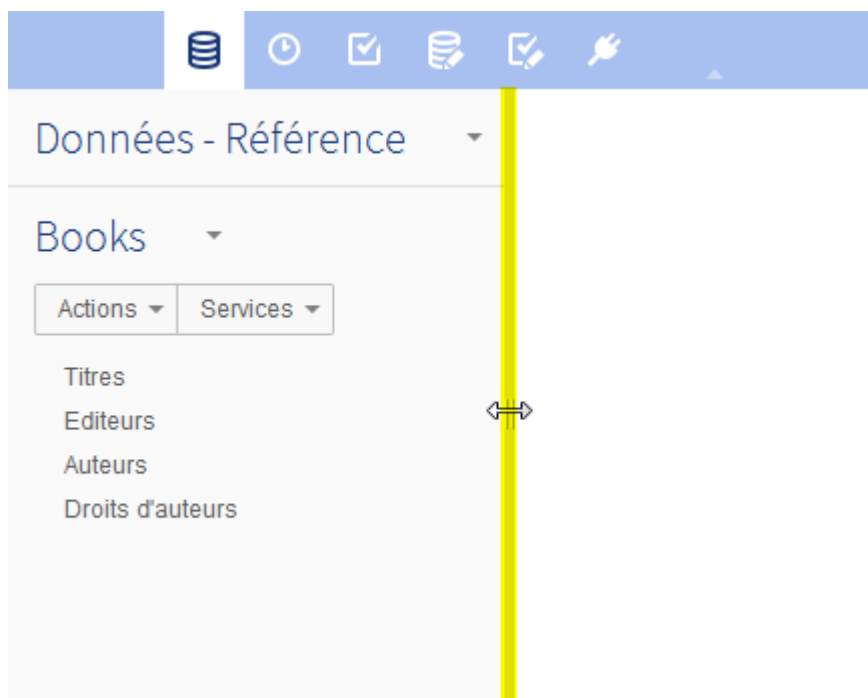
Cacher l'en-tête

L'en-tête de l'interface utilisateur peut être caché en le survolant, puis en cliquant sur le bouton en forme de flèche.



Réinitialiser la largeur du panneau de navigation

Si la largeur du panneau de navigation a été modifiée, elle peut être réinitialisée en double-cliquant sur la bordure.



2.5 Où trouver de l'aide sur EBX5

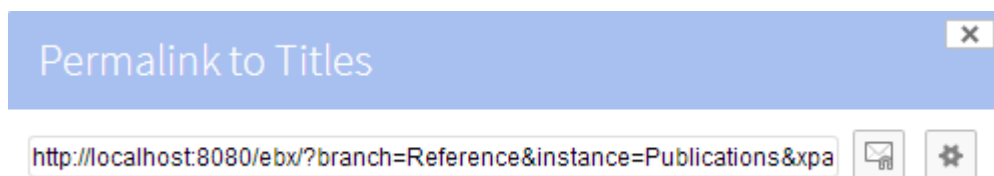
En plus de la documentation complète du produit, l'aide est accessible de plusieurs façons dans l'interface :

Aide contextuelle

Lorsque la souris survole un élément pour lequel une aide spécifique existe, un point d'interrogation apparaît. En cliquant sur le point d'interrogation, un panneau de documentation affiche les informations associées.



Un bouton dans le coin supérieur gauche du panneau permet de récupérer un permalien vers l'élément. Ce bouton n'est pas disponible pour tous les types d'élément.



CHAPITRE 3

Glossaire

Ce chapitre contient les sections suivantes :

1. [Gouvernance](#)
2. [Modélisation de données](#)
3. [Gestion des données](#)
4. [Cycle de vie des données](#)
5. [Historique](#)
6. [Modélisation de workflow](#)
7. [Workflows de données](#)
8. [Services de données](#)
9. [Transverse](#)

3.1 Gouvernance

référentiel

Entité de stockage côté serveur contenant toutes les données gérées par EBX5. Le référentiel est organisé en espaces de données.

Voir [espace de données](#) [p 23].

profil

Terme générique désignant soit un utilisateur, soit un rôle. Les profils sont utilisés pour définir des règles de permission et des workflows de données.

Voir [utilisateur](#) [p 19], [rôle](#) [p 20].

API Java : Profile^{API}.

utilisateur

Entité créée dans le référentiel afin de permettre à des personnes physiques ou des systèmes externes de s'authentifier et d'accéder à EBX5. Un utilisateur peut avoir un ou plusieurs rôles et posséder diverses informations de compte telles que nom, prénom, login, e-mail, etc.

Voir [annuaire des utilisateurs et des rôles](#) [p 20], [profil](#) [p 19].

Concept apparenté : [User and roles directory](#) [p 407].

API Java : `UserReferenceAPI`.

rôle

Classification d'utilisateur utilisée pour les règles de permission et les workflows de données. Chaque utilisateur peut appartenir à plusieurs rôles.

Dès qu'un profil de type rôle est configuré dans EBX5, le comportement résultant de cette configuration s'applique à tous les utilisateurs membres de ce rôle. Par exemple, dans un modèle de workflow, un rôle peut être configuré pour le(s) destinataire(s) d'un bon de travail.

Voir [annuaire des utilisateurs et des rôles](#) [p 20], [profil](#) [p 19].

Concept apparenté : [User and roles directory](#) [p 407].

API Java : `RoleAPI`.

administrateur

Rôle prédéfini permettant d'accéder à l'administration technique et à la configuration de EBX5.

annuaire des utilisateurs et des rôles

Annuaire définissant les méthodes disponibles pour l'authentification d'accès au référentiel, ainsi que les rôles disponibles et les utilisateurs autorisés à accéder au référentiel, avec leurs rôles respectifs.

Voir [utilisateur](#) [p 19], [rôle](#) [p 20].

Concept apparenté : [User and roles directory](#) [p 407].

API Java : `DirectoryAPI`, `DirectoryHandlerAPI`.

session utilisateur

Contexte d'accès au référentiel associé à un utilisateur (utilisateur ayant été authentifié par rapport à l'annuaire des utilisateurs et des rôles).

Concept apparenté : [User and roles directory](#) [p 407].

API Java : `SessionAPI`.

3.2 Modélisation de données

Section de la documentation [Modèles de données](#) [p 30]

modèle de données


Définition structurée des données à gérer dans le référentiel EBX5. Un modèle de données décrit la structure des données en termes d'organisation, de type et de relations sémantiques. Le but d'un modèle de données est de définir la structure et les caractéristiques d'un jeu de données, qui est une instance d'un modèle de données contenant les données gérées par le référentiel.

Voir [jeu de données](#) [p 22].

Concept apparenté : [Modèles de données](#) [p 30].

champ

Élément de base du modèle de données défini par un nom et un type de données simple. Un champ peut être directement défini à la racine du modèle de données, ou en tant que colonne d'une table. Il est possible d'assigner des contraintes de base sur la valeur du champ, par exemple sur sa longueur, ainsi que des règles de validation plus complexes impliquant des calculs. La valeur du champ peut être automatiquement calculée à l'aide du mécanisme d'héritage de données, ou de règles de calcul. Un champ peut être défini comme étant une liste agrégée en spécifiant une cardinalité maximale supérieure à 1. Chaque élément de la liste ainsi définie sera du même type que le champ initial. Les champs peuvent être regroupés pour faciliter l'organisation du modèle de données.

Par défaut, les champs sont représentés par l'icône .

Voir [enregistrement](#) [p 22], [groupe](#) [p 21], [table \(modèle de données\)](#) [p 21], [règle de validation](#) [p 22], [héritage](#) [p 23].

Concept apparenté : [Propriétés des éléments de structure](#) [p 47], [Contrôles sur les champs de données](#) [p 61].

API Java : `SchemaNode`^{API}.

clé primaire

Champ ou composition de plusieurs champs identifiant de manière unique un enregistrement dans une table.

Les clés primaires sont représentées par l'icône .

Concept apparenté : [Tables](#) [p 467].

clé étrangère

Champ ou composition de plusieurs champs référençant un enregistrement d'une autre table, via sa clé primaire.

Les clés étrangères sont représentées par l'icône .

Voir [clé primaire](#) [p 21].

Concept apparenté : [Clé étrangère](#) [p 471].

table (modèle de données)

Élément du modèle de données composé de champs et/ou de groupes. Une table doit au moins être composée d'un champ défini comme étant une clé primaire. Une table peut être utilisée pour la création d'un type réutilisable, afin de créer d'autres éléments basés sur la structure de cette table.

Les tables sont représentées par l'icône .

Voir [enregistrement](#) [p 22], [clé primaire](#) [p 21], [type réutilisable](#) [p 22].

groupe

Entité de classification utilisée pour organiser les données du modèle. Un groupe peut contenir des champs, d'autres groupes, et des tables. Si un groupe contient des tables, alors celui-ci ne pourra pas être inclus dans une autre table. Un groupe peut être utilisé pour la création d'un type réutilisable afin de créer d'autres éléments basés sur la structure de ce groupe.

Les groupes sont représentés par l'icône .

Voir [type réutilisable](#) [p 22].

API Java : `SchemaNodeAPI`.

type réutilisable

Définition d'un type simple ou complexe qui peut être partagée entre différents éléments d'un modèle.

règle de validation

Association d'une ou plusieurs règles de contrôle définies sur un champ ou une table. Toute donnée saisie ne respectant pas ces contrôles sera déclarée invalide, selon la sévérité associée à la règle de validation.

assistant de modélisation de données (DMA)

L'interface utilisateur inclut un outil d'aide à la modélisation des données. Cet outil permet de définir la structure d'un modèle, de créer et éditer ses éléments, puis de configurer et publier le modèle.


Voir [Modèles de données](#) [p 30].

3.3 Gestion des données

Section de la documentation [Jeux de données](#) [p 100]

enregistrement

Ensemble de données identifié de manière unique par une clé primaire. Un enregistrement correspond à une ligne dans une table. Chaque enregistrement respecte la structure de données définie dans le modèle de données associé. C'est ce modèle de données qui indique les types et les cardinalités des champs qui composent l'enregistrement.

Les enregistrements sont représentés par l'icône .

Voir [table \(jeu de données\)](#) [p 22], [clé primaire](#) [p 21].

table (jeu de données)

Ensemble d'enregistrements (lignes) de même structure contenant des données. Chaque enregistrement est identifié de manière unique par sa clé primaire.

Les tables sont représentées par l'icône .

Voir [enregistrement](#) [p 22], [clé primaire](#) [p 21].

jeu de données

Instance d'un modèle de données qui contient les données. La structure et le comportement d'un jeu de données sont basés sur les définitions fournies par le modèle de données qu'il implémente. En fonction de son modèle de données, un jeu de données peut contenir des données sous la forme de tables, groupes et champs.

Voir [table \(jeu de données\)](#) [p 22], [champ](#) [p 21], [groupe](#) [p 21], [vue personnalisée](#) [p 23].


Les jeux de données sont représentés par l'icône .

Concept apparenté : [Jeux de données](#) [p 100].


héritage

Mécanisme par lequel une donnée d'une entité peut être valorisée par défaut à partir d'une autre entité. Dans EBX5, deux types d'héritage sont possibles : le premier entre deux jeux de données, le deuxième entre deux champs.

Lorsqu'il est activé, l'héritage entre jeux de données permet à un jeu de données enfant d'obtenir comme valeur par défaut les données du jeu de données parent. Il est possible de surcharger dans les jeux de données enfants les valeurs héritées du parent. Par défaut, l'héritage est désactivé. Il peut être activé lors de la définition du modèle de données.

L'héritage depuis le jeu de données parent est représenté par l'icône .

L'héritage de champ fonctionne de manière similaire, mais s'applique entre champs d'un même jeu de données. Le champ hérité prend pour valeur par défaut la valeur du champ source.

Les champs hérités sont représentés par l'icône .

Concept apparenté : [Inheritance and value resolution](#) [p 310].

vue personnalisée

Configuration d'affichage applicable sur une table. Une vue peut être créée pour un utilisateur ou un rôle. La vue personnalisée permet de sélectionner sous quel mode seront affichés les enregistrements : mode hiérarchique ou tabulaire ; ainsi que de définir des critères de filtrage et de tri.

Voir [vue hiérarchique](#) [p 23].

Concept apparenté : [Vues](#) [p 107].

vue hiérarchique

Vue personnalisée qui affiche les données d'une table sous forme d'arbre. Une vue hiérarchique peut être utile pour montrer les relations entre les données du modèle. Lors de la création d'une vue hiérarchique, une dimension doit être sélectionnée pour déterminer les relations à exploiter. Dans une vue hiérarchique, il est possible de naviguer à travers des relations récursives, ainsi qu'entre plusieurs tables en utilisant des clés étrangères.

Voir [vue personnalisée](#) [p 23].


Concept apparenté : [Hiérarchies](#) [p 108].

3.4 Cycle de vie des données

Section de la documentation [Espaces de données](#) [p 78]

espace de données

Contient les jeux de données. L'espace de données est utilisé pour isoler différentes versions de jeux de données ou pour les organiser. Des espaces de données enfants peuvent être créés à partir d'un espace de données. Un espace de données enfant est initialisé dans le même état que son parent au moment de sa création. Ultérieurement, l'enfant pourra être fusionné avec son parent. A tout moment, une comparaison avec d'autres espaces de données est possible.

Les espaces de données sont représentés par l'icône .

Voir [héritage](#) [p 23], [référentiel](#) [p 19], [fusion](#) [p 24].

Concept apparenté : [Data spaces](#) [p 78].

espace de données de référence

Ancêtre commun des espaces de données du référentiel. N'ayant pas de parent, cet espace de données ne peut pas être fusionné.

Voir [espace de données](#) [p 23], [fusion](#) [p 24], [référentiel](#) [p 19].


fusion

Intégration, dans l'espace de données parent, des changements réalisés dans un espace de données enfant depuis sa création. L'espace de données enfant est fermé après une fusion réalisée avec succès. Pour effectuer cette fusion, un passage en revue des différences entre les deux espaces de données est requis afin de résoudre les éventuels conflits. En effet, des conflits peuvent survenir en cas de modification des mêmes données tant sur l'enfant que le parent. Une décision doit être prise pour chacun de ces conflits, afin de déterminer quelle modification doit prendre le pas sur l'autre.

Concept apparenté : [Fusion](#) [p 87].

image

Copie statique d'un espace de données qui capture son état et tout son contenu à un moment donné, afin d'être utilisée comme référence. Une image peut être consultée, exportée, et comparée à d'autres espaces de données, mais jamais modifiée directement.

Les images sont représentées par l'icône .

Concept apparenté : [Image](#) [p 93]

3.5 Historique

Section de la documentation [History](#) [p 289]

historisation

Mécanisme qui peut être activé au niveau d'une table afin de suivre les modifications dans le référentiel. Deux vues d'historique sont disponibles quand l'historisation est activée : la vue historique de table et la vue historique des transactions. Dans toutes les vues d'historique, les fonctionnalités classiques des tables telles que l'export, la comparaison et les filtres restent disponibles.

L'activation de l'historique nécessite la configuration d'un profil d'historisation. L'historisation des tables n'est pas activée par défaut.

Voir [vue historique de table](#) [p 25], [vue historique des transactions](#) [p 25], [profil d'historisation](#) [p 24].

profil d'historisation

Ensemble de préférences spécifiant d'une part les espaces de données dont les modifications doivent être enregistrées dans l'historique de la table et, d'autre part, si les transactions doivent échouer quand l'historisation n'est pas disponible.

Voir [profil d'historisation](#) [p 24].

vue historique de table

Vue contenant la trace de toutes les modifications effectuées sur une table donnée, notamment les créations, mises à jour et suppressions. Chaque entrée présente les informations transactionnelles telles que : la date et l'heure, l'utilisateur ayant effectué l'action, ainsi que l'état des données à l'issue de la transaction. Ces informations peuvent aussi être consultées au niveau d'un enregistrement ou d'un jeu de données.

Référence technique apparentée [History](#) [p 289].

vue historique des transactions

Vue présentant les données techniques et d'authentification des transactions au niveau du référentiel ou d'un espace de données. Etant donné qu'une transaction peut effectuer de multiples opérations/actions et peut affecter plusieurs tables dans un ou plusieurs jeux de données, cette vue montre toutes les opérations qui ont été effectuées dans le périmètre en question pour chaque transaction.

Référence technique apparentée [History](#) [p 289].

3.6 Modélisation de workflow

Section de la documentation [Modèles de workflow](#) [p 126]

modèle de workflow

Définition de la succession d'opérations à effectuer sur les données.


Un modèle de workflow de données décrit la totalité du parcours que doivent suivre les données pour être traitées, que ce soit en termes d'états ou d'actions associées à effectuer par des utilisateurs et des tâches automatiques.

Concept apparenté : [Modèles de workflow](#) [p 126].

tâche automatique

Tâche de workflow de données effectuée par une procédure automatique, sans intervention humaine.

Les tâches automatiques les plus communes sont la création d'espace de données, la fusion d'espace de données et la création d'image.


Les tâches automatiques sont représentées par l'icône .

Voir [modèle de workflow](#) [p 25].

tâche utilisateur

Tâche de workflow composée d'un ou plusieurs bons de travail réalisés en parallèle par des utilisateurs (intervention humaine).

Les bons de travail sont proposés ou assignés aux utilisateurs, en fonction du modèle de workflow de données. L'avancement du workflow de données positionné sur une tâche utilisateur dépend de la satisfaction du critère de fin de tâche défini dans le modèle de workflow de données.

Les tâches utilisateur sont représentées par l'icône .

Voir [modèle de workflow](#) [p 25].

condition de workflow

Etape de décision dans le workflow de données.

Une condition de workflow de données décrit le critère utilisé pour déterminer quelle sera la prochaine étape à exécuter.

Les conditions de workflow sont représentées par l'icône .

appel à des sous-workflows

Etape qui met le workflow de données courant en attente et qui lance un ou plusieurs autres workflows de données. Si une telle étape lance plusieurs sous-workflows, les sous-workflows sont exécutés en parallèle.

tâche d'attente

Etape d'un workflow de données qui met en pause le workflow en cours en attendant un événement donné. Lorsque l'événement est reçu, le workflow est réveillé et va automatiquement à l'étape suivante.

contexte des données

Ensemble de données qui peuvent être partagées entre les étapes pendant toute la durée de vie d'un workflow afin de garantir la communication entre les étapes.

3.7 Workflows de données

Section de la documentation [Workflows de données](#) [p 152]

publication de workflow

Version particulière d'un modèle de workflow de données qui est mise à disposition des utilisateurs ayant les permissions nécessaires pour exécuter des workflows.

workflow de données

Instance particulière d'un modèle de workflow qui exécute les étapes définies dans le modèle de workflow de données (les tâches utilisateur, les tâches automatiques et les conditions).

Voir [modèle de workflow](#) [p 25].

Concept apparenté : [Workflows de données](#) [p 152].

corbeille

Liste des workflows de données publiés affichés en fonction des permissions de l'utilisateur. Les utilisateurs qui ont la permission de lancer des workflows peuvent le faire à partir de leur corbeille. Tous les bons de travail nécessitant une action de l'utilisateur sont affichés sous la publication de workflow associée dans la corbeille. De plus, si l'utilisateur est administrateur de workflows de données, il a la possibilité de voir leur état dans sa corbeille, et ainsi d'intervenir, si nécessaire, sur les workflows qu'il supervise.

Voir [workflow de données](#) [p 26]

bon de travail

Action unitaire d'une tâche utilisateur qui doit être réalisée par un utilisateur.

Les bons de travail alloués sont représentés par l'icône .

Voir [tâche utilisateur](#) [p 25].

jeton

Repère de position d'un workflow qui indique quelle étape courante est actuellement exécutée par un workflow de données. Les jetons sont utilisés durant l'avancement d'un workflow de données, et sont uniquement visibles par les administrateurs du référentiel.

3.8 Services de données

Section de la documentation [Services de données](#) [p 170]

service de données

EBX5 partage les données de référence conformément à l' [Architecture Orientée Service](#) en utilisant la technologie XML web service. Tous les services de données sont générés directement à partir des modèles ou services built-in. Ils peuvent être utilisés pour accéder à une partie des fonctionnalités disponibles via l'interface utilisateur.

Les services de données d'EBX5 proposent :

- Un générateur WSDL pour les modèles de données ainsi que pour les services built-in. Le fichier WSDL peut-être produit indifféremment au travers de l'interface utilisateur ou du connecteur HTTP(S) pour un logiciel d'intégration ou une application cliente. Les opérations de services sont invoquées par des messages XML communiqués au point d'entrée d'EBX5.
- Un connecteur SOAP ou point d'entrée pour les messages SOAP permet aux systèmes externes d'interagir avec le contenu du référentiel. Ce connecteur répond aux demandes issues des WSDL produits par EBX5. Après authentification, il accepte les messages XML et les exécute selon les permissions de l'utilisateur authentifié.

lignage

Mécanisme de mise en place de profils de droits d'accès pour des services de données. Les profils de droits d'accès ainsi définis sont utilisés pour accéder aux données via des interfaces WSDL.

Concept apparenté : [Générer un WSDL pour un lignage](#) [p 175].

3.9 Transverse

noeud

Un noeud est un élément d'une arborescence ou d'un graphe. Dans EBX5, 'Noeud' peut faire référence à différents concepts selon le contexte d'utilisation :

- Dans le contexte du [modèle de workflow](#) [p 25], un noeud est une étape du workflow ou une condition.
- Dans le contexte du [modèle de données](#) [p 20], un noeud est un groupe, une table ou un champ.
- Dans le contexte des [hiérarchies](#) [p 23], un noeud représente la valeur d'une dimension.
- Dans un arbre d'[héritage de jeux de données](#) [p 23], un noeud est un jeu de données.
- Dans un [jeu de données](#) [p 22], un noeud est le noeud du modèle de données évalué dans le contexte du jeu de données ou de l'enregistrement.

Modèles de données

CHAPITRE 4

Introduction aux modèles de données

Ce chapitre contient les sections suivantes :

1. [Présentation](#)
2. [Utilisation de l'interface utilisateur de la section Modèles de données](#)

4.1 Présentation

La fonction d'un modèle de données

La première étape de toute gouvernance de données dans EBX5 est le développement d'un modèle de données. Le but d'un modèle de données est de définir la structure des données gérées dans le référentiel en termes d'organisation, de types de données, et de relations sémantiques. Une fois que le modèle de données a été défini et publié, il devient possible de créer des jeux de données à partir de celui-ci.

Afin de définir un modèle de données dans le référentiel, créez d'abord un nouveau modèle de données, puis définissez sa structure et les propriétés de ses éléments (tables, champs et groupes). Le modèle de données ainsi défini doit être publié pour devenir disponible. Les utilisateurs pourront créer des jeux de données à partir de cette publication qui contiendront les données gérées par le référentiel EBX5.

Concepts de base utilisés dans la modélisation des données

Une compréhension des termes suivants est nécessaire pour commencer la création de modèles de données :

- [champ](#) [p 21]
- [clé primaire](#) [p 21]
- [clé étrangère](#) [p 21]
- [table](#) [p 21]
- [groupe](#) [p 21]
- [type réutilisable](#) [p 22]
- [règle de validation](#) [p 22]

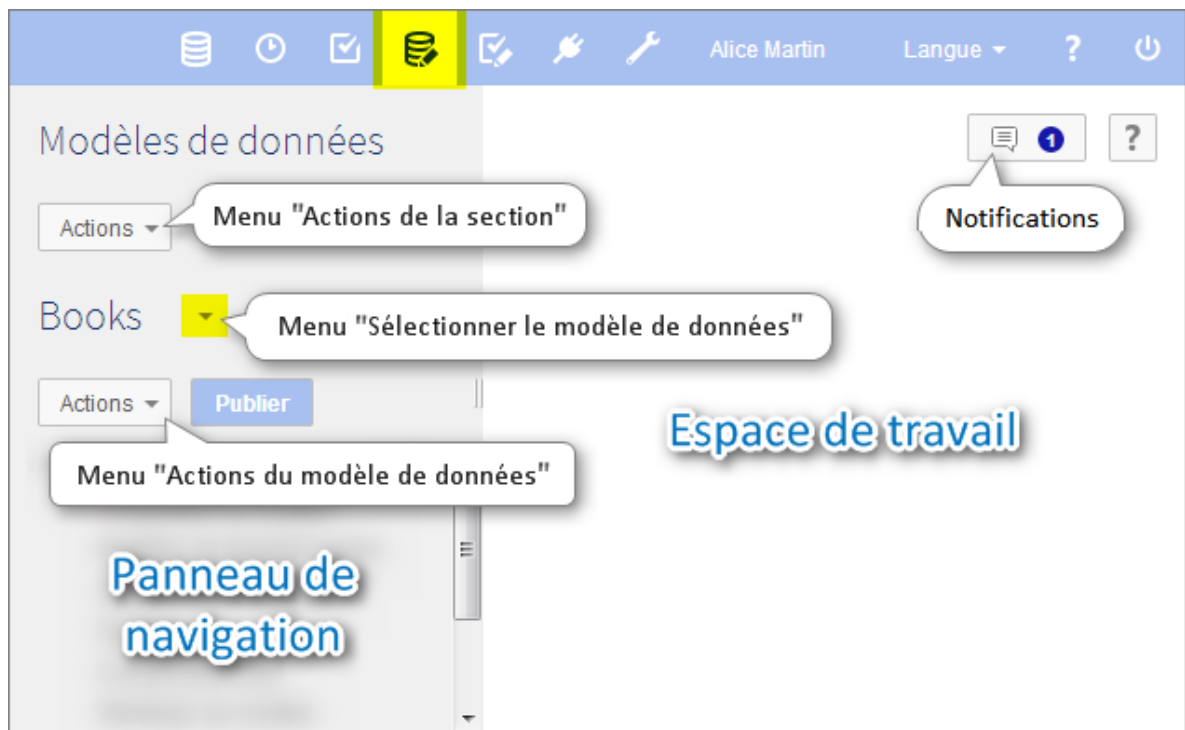
4.2 Utilisation de l'interface utilisateur de la section Modèles de données

Navigation dans le Data Model Assistant

Les modèles de données peuvent être importés, édités, et publiés dans la section **Modèles de données**. Le Data Model Assistant de EBX5 facilite le développement des modèles de données.

Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée'.



Le panneau de navigation est organisé selon les sections suivantes :

Configuration	La configuration technique du modèle de données.
Propriétés du modèle	Les propriétés techniques du modèle de données.
Modèle de données inclus	Les modèles de données inclus dans le modèle courant. Les types de données définis dans les modèles inclus peuvent être réutilisés dans le modèle de données courant.
Librairie de composants	Les composants Java utilisables dans le modèle de données.
Services	Les services disponibles dans le modèle de données.
Composants Ajax	Les composants Ajax disponibles dans le modèle de données.
'Bindings' du modèle	Les propriétés des types Java générés depuis le modèle de données.
Répliquations	Les unités de réplication disponibles dans le modèle de données.
Structure de données	Structure du modèle de données. Définit les relations entre les éléments du modèle de données et permet d'accéder à la définition de chaque élément.
Types de données simples	Types simples réutilisables définis dans le modèle de données courant.
Types de données complexes	Types complexes réutilisables définis dans le modèle de données courant.
Types de données simples inclus	Types simples réutilisables définis dans un modèle de données inclus dans le modèle courant.
Types de données complexes inclus	Types complexes réutilisables définis dans un modèle de données inclus dans le modèle courant.

Voir aussi

[*Modélisation de la structure des données*](#) [p 41]

[*Configuration du modèle de données*](#) [p 37]

[*Types réutilisables*](#) [p 43]

Icônes des éléments du modèle de données

 [champ](#) [p 21]

 [clé primaire](#) [p 21]

 [clé étrangère](#) [p 21]

 [table](#) [p 21]

 [groupe](#) [p 21]

Concepts apparentés

[Espaces de données](#) [p 78]

[Jeux de données](#) [p 100]

CHAPITRE 5

Création du modèle de données

Ce chapitre contient les sections suivantes :

1. [Création d'un modèle de données](#)
2. [Sélection d'un type de modèle de données](#)

5.1 Création d'un modèle de données

Pour créer un modèle de données, cliquez sur le bouton **Créer** dans le sélecteur, puis suivez les instructions de l'assistant de création de modèle de données.

5.2 Sélection d'un type de modèle de données

Si l'utilisateur a le rôle "Administrateur", il pourra sélectionner le type de modèle de données. Un utilisateur avec le rôle "Administrateur" peut créer un modèle de données *sémantique* ou *relationnel*.

Modèles sémantiques

Les modèles sémantiques permettent l'utilisation de toutes les fonctionnalités de gestion des données fournies par EBX5, comme la gestion du cycle de vie, en utilisant des espaces de données. Par défaut, les modèles de données dans EBX5 sont sémantiques.

Modèles relationnels

Les modèles relationnels sont utilisés lorsque les tables du modèle doivent être accessibles par le biais d'un système de gestion de base de données (SGBD). Le principal avantage des modèles relationnels est la possibilité d'interroger les tables par des requêtes SQL externes. Cependant, les modèles relationnels perdent certaines fonctionnalités de gestion des données, par exemple l'héritage, les champs multi-valués, et la gestion de cycle de vie par l'utilisation des espaces de données.

Note

Un modèle de données relationnel peut être utilisé par un seul jeu de données. L'espace de données contenant le jeu de données doit également être déclaré comme relationnel.

Voir aussi

[Relational mode](#) [p 283]

[Espaces de données](#) [p 78]

CHAPITRE 6

Configuration du modèle de données

Ce chapitre contient les sections suivantes :

1. [Informations associées au modèle de données](#)
2. [Permissions](#)
3. [Propriétés du modèle de données](#)
4. [Modèles de données inclus](#)
5. [Réplication des données vers tables relationnelles](#)

6.1 Informations associées au modèle de données

Pour visualiser et éditer les informations concernant le propriétaire et la documentation du modèle de données, sélectionnez "Informations" dans le menu ["Actions" du modèle de données](#) [p 31] dans le panneau de navigation.

Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée'.

Nom unique	Le nom unique du modèle de données. Ce nom ne peut pas être modifié après la création du modèle.
Propriétaire	Spécifie le propriétaire du modèle de données, qui a le droit de modifier les informations du modèle et ses permissions.
Documentation localisée	Libellés et descriptions localisés pour le modèle de données.

6.2 Permissions

Pour définir les permissions d'accès au modèle de données, sélectionnez "Permissions" dans le menu ["Actions" du modèle de données](#) [p 31] dans le panneau de navigation.

La définition des permissions d'un modèle de données s'effectue de la même manière que pour les jeux de données. Les détails se trouvent dans la section [Permissions](#) [p 118].

6.3 Propriétés du modèle de données

Dans le panneau de navigation, sous Configuration > Propriétés du modèle, vous pouvez accéder aux propriétés techniques suivantes :

Nom du module	Définit le module contenant les ressources qui seront utilisées par ce modèle de données. Ce nom de module désigne également le module cible lors de la publication du modèle de données s'il est publié dans un module.
Chemin du module	Localisation physique du module sur le système de fichiers du serveur.
Chemin des sources	Emplacement des codes sources utilisés pour configurer les composants Java dans la "Librairie de composants". Si le chemin est relatif, il sera résolu à partir du "Chemin du module".
Mode de publication	<p>Les deux modes possibles sont : la publication du modèle dans un document XML Schema au sein d'un module ou en tant que modèle de données embarqué dans le référentiel EBX5. Les modèles de données embarqués permettent d'avoir des fonctionnalités additionnelles, comme la gestion des versions et la restauration des versions précédentes du modèle.</p> <p>Voir Modes de publication [p 71] pour plus d'informations.</p> <p>Chemin du modèle dans le module: Définit l'emplacement du document XML Schema pour la publication du modèle de données dans un module. Le chemin doit commencer par "/".</p>
Héritage des données	<p>Spécifie si l'héritage des données, entre jeux de données, est activé pour ce modèle de données. L'héritage des données est désactivé par défaut.</p> <p>Voir héritage [p 121] pour plus d'informations.</p>
Documentation	<p>Documentation du modèle de données définie à l'aide d'une classe Java. Cette classe Java définit les libellés et descriptions des éléments du modèle de données. Les libellés et descriptions définis par cette classe Java sont affichés, dans les jeux de données associés, en priorité par rapport aux libellés et descriptions définis localement par les éléments du modèle de données.</p> <p>Voir Dynamic labels and descriptions [p 504] pour plus d'informations.</p>

Extensions spéciales	Permissions définies à l'aide de règles programmatiques.
Désactiver les contrôles de l'auto-incrément	Indique si le contrôle de la valeur d'un champ auto-incrémenté par rapport à la valeur maximale trouvée dans la table en cours de mise à jour doit être désactivé. Voir Valeurs auto-incrémentées [p 500] pour plus d'informations.

6.4 Modèles de données inclus

Les types de données définis dans un autre modèle de données peuvent être utilisés dans le modèle de données courant en ajoutant une entrée pour l'autre modèle de données dans la table Configuration > Modèles de données inclus.

En accédant à l'enregistrement du modèle inclus depuis cette table, des informations techniques liées à ce modèle sont consultables sous l'onglet **Information**. Cet onglet contient aussi le rapport de validation du modèle de données inclus.

Seuls les modèles de données sans erreur de validation, qui ont été définis et publiés comme modèle "embarqué" ou dans un module, peuvent être inclus.

Les types de données doivent être uniques à la fois dans le modèle courant et dans tous les modèles inclus. Il est impossible d'inclure un modèle contenant des types de données déjà existant dans le modèle courant ou dans les autres modèles de données inclus.

Voir aussi [Including external data models](#) [p 464]

6.5 Réplication des données vers tables relationnelles

Dans n'importe quel type de modèle de données, il est possible de définir des *unités de réplication* afin que les données du référentiel soient copiées dans des tables relationnelles dédiées. Ainsi, ces tables relationnelles permettent d'accéder directement aux données en utilisant des requêtes et des vues SQL.

Pour définir une unité de réplication en utilisant l'interface utilisateur, créez un nouvel enregistrement dans la table "Répliquations" située dans la section Configuration du modèle de données dans le panneau de navigation. Chaque unité de réplication concerne un jeu de données spécifique dans un espace de

données particulier. Une unité de réplication peut inclure plusieurs tables, tant qu'elles sont dans le même jeu de données. Une unité de réplication définit les informations suivantes :

Nom	Nom de l'unité de réplication. Ce nom identifie l'unité de réplication dans le modèle de données. Ce nom doit être unique.
Espace de données	Indique l'espace de données concerné par la réplication. Cet espace de données ne peut ni être une version ni être relationnel.
Jeu de données	Indique le jeu de données concerné par la réplication.
Mode de rafraichissement	<p>Définit le mode de synchronisation. Les différents modes de synchronisation sont les suivants :</p> <ul style="list-style-type: none"> • Sur "commit" : les données répliquées contenues dans la base de données sont toujours à jour par rapport à la table source. Chaque transaction de mise à jour de la table source produit les mises à jour correspondantes dans la table contenant les données répliquées dans la base de données. • Sur demande : les données répliquées contenues dans la base de données sont mises à jour uniquement lorsqu'une opération manuelle de rafraichissement est effectuée.
Tables	<p>Indique les tables du modèle de données qui doivent être répliquées dans la base de données.</p> <p>Chemin de la table: Indique le chemin de la table dans le modèle de données qui doit être répliquée dans la base de données.</p> <p>Nom dans la base de données: Indique le nom de la table dans la base de données qui contiendra les données répliquées. Ce nom doit être unique par rapport à toutes les unités de réplication.</p>

Voir aussi [Replication](#) [p 297]

CHAPITRE 7

Modélisation de la structure des données

Pour définir la structure du modèle de données, sélectionnez le modèle de données avec lequel vous voulez travailler dans le panneau de navigation.

La structure du modèle de données est accessible depuis le panneau de navigation dans la section "Structure de données". Cette section permet de visualiser et de définir la structure des champs, groupes, et tables du modèle de données.

Ce chapitre contient les sections suivantes :


1. [Actions et propriétés communes](#)
2. [Types réutilisables](#)
3. [Détails de la création des éléments du modèle de données](#)
4. [Modification des éléments existants](#)

7.1 Actions et propriétés communes

Créer des éléments

Les éléments suivants peuvent être ajoutés à un modèle de données :

- champs
- groupes
- tables
- clés primaires
- clés étrangères
- associations

Ajoutez un de ces éléments sous un élément existant en cliquant sur la flèche  située à la droite de l'élément existant, puis en sélectionnant une option de création parmi les options présentées dans le menu. Suivez ensuite l'assistant de création pour créer un élément.

Note


L'élément root est ajouté par défaut lors de la création d'un modèle de données. Cet élément représente la racine de la structure du modèle de données. S'il faut renommer cet élément, il peut être supprimé et recréé avec un nom différent.

Noms, libellés, descriptions, et informations

Le nom de l'élément à créer est obligatoire. Ce nom doit être unique au sein d'un même niveau dans la structure de données. En effet, sous un même groupe, deux éléments ne peuvent avoir le même nom. Une fois l'élément créé, son nom ne peut plus être modifié.


Il est possible de définir des libellés localisés qui seront affichés dans l'interface utilisateur au lieu du nom unique de l'élément. Il est aussi possible de définir une description localisée de l'élément. Contrairement au nom de l'élément, les libellés et descriptions sont modifiables après la création de l'élément. Selon la préférence de langue de chaque utilisateur, EBX5 affichera le libellé et la description localisés de l'élément.

Supprimer des éléments

Tous les éléments du modèle de données peuvent être supprimés de la structure de données en utilisant la flèche  située à la droite de l'élément à supprimer.

Si un groupe ou une table n'utilisant pas un type réutilisable est supprimé, la suppression est effectuée récursivement sur tous les éléments situés sous le groupe ou la table.

Dupliquer des éléments existants


Pour dupliquer un élément, cliquez sur la flèche  située à la droite de l'élément à dupliquer. Spécifiez le nom de l'élément dupliqué. Ce nom doit être unique au sein d'un même niveau dans la structure de données. Toutes les propriétés de l'élément source sont dupliquées.

L'élément dupliqué est rajouté dans le modèle de données au même niveau que l'élément d'origine, en dernière position. Lorsqu'un élément contenant d'autres éléments est dupliqué, tous les sous-éléments sont dupliqués avec leurs noms originaux.

Note

En cas de duplication d'un champ appartenant à une clé primaire, les propriétés du champ sont dupliquées mais le nouveau champ n'est pas ajouté à la clé primaire de la table parente.

Déplacer des éléments

Pour déplacer un élément, cliquez sur la flèche  située à la droite de l'élément à déplacer et sélectionnez "Déplacer". Sélectionnez ensuite la flèche qui correspond à l'élément *avant lequel* positionner l'élément actuel.

Note

Le déplacement d'un élément est uniquement possible au sein d'un même niveau dans la structure de données du modèle.



7.2 Types réutilisables


Les types réutilisables sont des éléments partagés qui, après leur création, peuvent être réutilisés par différents éléments du modèle de données.

Note

En modifiant la définition d'un type réutilisable, la structure de tous les éléments basés sur ce type réutilisable est aussi modifiée. L'arborescence "Structure de données" affiche, en lecture seule, la structure d'un groupe ou d'une table qui utilise un type réutilisable. Pour éditer la structure du type réutilisable associé, accédez au type dans la section "Types de données simples" ou "Types de données complexes".

Définition d'un type réutilisable

En utilisant le menu avec la flèche  des sections "Types de données simples" ou "Types de données complexes" dans le panneau de navigation, il est possible de créer des types simples et des types complexes réutilisables qui seront disponibles pour créer d'autres éléments avec la même structure et les mêmes propriétés. Il est également possible de convertir les tables et groupes existants en types réutilisables en utilisant le menu avec la flèche  situé à côté de l'élément à convertir.

Il est possible de visualiser les éléments du modèle utilisant un type réutilisable, en éditant ce type et en cliquant sur le lien "Références vers ce type". Ce lien affiche une table listant tous les éléments utilisant ce type. Si le type n'est utilisé par aucun élément, il peut être supprimé en sélectionnant "Supprimer type" en utilisant le menu avec la flèche  situé à droite du type à supprimer.

Utilisation d'un type réutilisable

Les structures et les propriétés de nouveaux éléments peuvent être définies par des types réutilisables en sélectionnant un type réutilisable à la création d'un élément. L'élément créé utilisera la structure et les propriétés du type réutilisable.

Inclusion des types de données définis dans d'autres modèles de données

Les types réutilisables peuvent aussi être partagés entre plusieurs modèles de données. En configurant l'inclusion d'un modèle de données externe, il est possible d'utiliser les types de données inclus pour

créer des éléments dans la structure de données, de la même manière que pour les types réutilisables définis en local.

Note

Les types de données devant être uniques pour tous les types définis en local et inclus, il n'est pas possible de créer un type réutilisable portant le même nom qu'un type de données dans un modèle de données inclus. De la même manière, il n'est pas possible d'inclure un modèle de données externe qui définit un type de données portant le même nom qu'un type réutilisable défini en local ou dans un autre modèle de données inclus.

Les types de données inclus apparaissent dans les sections "Types de données simples inclus" et "Types de données complexes inclus" dans le panneau de navigation. Les détails de ces types réutilisables sont consultables, mais ils ne sont éditables que dans leurs modèles de données d'origine.

Voir [Modèles de données inclus](#) [p 39] pour plus d'informations.

7.3 Détails de la création des éléments du modèle de données

Création de champs

A la création d'un champ, un type de données doit être sélectionné. Il définira le type de données associé aux valeurs saisies dans un jeu de données basé sur ce modèle. Le type de données du champ ne peut pas être modifié après la création du champ.

Durant la création d'un champ, il est également possible de le désigner comme clé étrangère, champ obligatoire, ou comme clé primaire si le champ est créé sous une table.

Création de tables

Lors de la création d'une nouvelle table, un type réutilisable existant peut être utilisé pour définir la structure et les propriétés de cette nouvelle table. Voir [Types réutilisables](#) [p 43] pour plus d'informations.

Chaque table nécessite la désignation d'au moins un champ clé primaire, qui peut être créé comme un élément enfant de la table dans la section "Structure de données" du panneau de navigation.

Création de groupes

Lors de la création d'un groupe, il est possible d'utiliser un type réutilisable existant pour définir la structure et les propriétés du nouveau groupe. Voir [Types réutilisables](#) [p 43] pour plus d'informations.

Création de clés primaires

Pour chaque table, il est nécessaire de définir une clé primaire. Pour cela, ajoutez un nouvel élément enfant à partir du menu d'actions disponible sur la table dans la section "Structure de données" du panneau de navigation.

Il est aussi possible d'ajouter un champ existant à la définition de la clé primaire, sur l'onglet "Clé primaire" dans les "Propriétés avancées" de la table.

Création ou définition de clés étrangères

Les champs associés à une clé étrangère sont de type "Chaîne de caractères". Pour créer une clé étrangère sur une table, ajoutez un nouvel élément enfant à partir du menu d'actions disponible sur la table dans la section "Structure de données" du panneau de navigation. Il est également possible de définir directement les propriétés d'une clé étrangère en éditant un champ de type "Chaîne de caractères". Pour convertir un champ existant de type "Chaîne de caractères" en clé étrangère, activez la propriété "Contrainte de clé étrangère" dans les "Contrôles avancés" du champ et définissez les propriétés associées.

Il faut toujours spécifier la table référencée par une clé étrangère.

Création d'associations

Une association permet de définir un lien sémantique entre deux tables. Une association peut être définie en créant un élément dans une table de la section "Structure de données" du panneau de navigation en sélectionnant la propriété "association" dans le formulaire de création. Une association peut uniquement être créée dans une table et il n'est pas possible de convertir un champ existant en association.

Lors de la création d'une association, spécifiez son type. Pour cela, deux options sont disponibles :

- Relation inverse d'une *clé étrangère*. Dans ce cas, l'association est définie dans une *table source* et fait référence à une *table cible*. Ce type d'association est l'inverse d'une clé étrangère qui est définie dans la table cible et qui référence la table source. Définissez la clé étrangère qui référence la table contenant l'association. Pour cela, des assistants de saisie sont disponibles lors de la création de l'association.
- Utilisation d'une *table de liens*. Dans ce cas, l'association est définie dans une *table source* et fait référence à une *table cible* qui est inférée à partir d'une *table de liens*. Cette table de liens doit définir deux clés étrangères : une clé étrangère référençant la table source et une autre référençant la table cible. La clé primaire de la table de liens doit aussi être composée de champs auto-incrémentés et/ou de clés étrangères vers la table source ou cible de l'association. Définissez la table de liens et ces deux clés étrangères. Pour cela, des assistants de saisie sont disponibles à la création de l'association.

Pour ces deux types d'association, nous appelons *enregistrements associés* les enregistrements de la table cible sémantiquement liés aux enregistrements de la table source.

Une fois l'association créée, il est possible de définir d'autres propriétés :

- filtrer les enregistrements associés en définissant un filtre XPath. Il est uniquement possible d'utiliser les champs de la table source et de la table cible lors de la définition du filtre XPath. Il n'est donc pas possible d'utiliser les champs d'une table de liens dans un filtre XPath. Un assistant de saisie est disponible pour définir les champs à utiliser dans un filtre XPath.
- configurer une vue tabulaire pour définir les champs de la table cible devant être affichés dans les formulaires des jeux de données. Il n'est pas possible de configurer ou de modifier une vue tabulaire si la table cible n'est pas définie ou n'existe pas. Par défaut, tous les champs de la table cible qu'un utilisateur a le droit de voir seront affichés dans les jeux de données si la vue tabulaire n'est pas définie.
- définir comment doivent être présentés les enregistrements associés dans les formulaires des jeux de données. Indiquez si les enregistrements associés doivent être inclus dans le formulaire ou dans un onglet spécifique. Par défaut, les enregistrements associés seront inclus dans le formulaire au niveau de l'association dans le modèle de données.
- inclure / exclure les enregistrements associés dans les opérations de sélection de Data Service. Par défaut, les enregistrements ne sont pas inclus dans les opérations de sélection des Data Service.
- spécifier les nombres minimum et maximum d'enregistrements associés requis. Dans les jeux de données associés, un message de validation est ajouté lorsque les nombres minimum ou maximum d'enregistrements associés ne correspondent pas à ces critères. Par défaut, les nombres minimum et maximum d'enregistrements associés requis ne sont pas restreints.
- définir une contrainte de validation en utilisant un prédicat XPath pour restreindre les enregistrements associés. Il est uniquement possible d'utiliser les champs de la table source et de la table cible lors de la définition du prédicat XPath. Il n'est donc pas possible d'utiliser les champs d'une table de liens dans un prédicat XPath. Un assistant de saisie permet de sélectionner les champs à utiliser dans un prédicat XPath. Dans les jeux de données associés, un message de validation sera ajouté pour tout enregistrement associé ne vérifiant pas cette contrainte.

7.4 Modification des éléments existants

Suppression d'un champ de la clé primaire

Tout champ appartenant à la clé primaire peut être supprimé de la clé primaire d'une table sur l'onglet "Clé primaire" dans les "Propriétés avancées" de la table.

Voir [clé primaire](#) [p 21] dans le glossaire.

CHAPITRE 8

Propriétés des éléments du modèle de données

Après avoir créé un élément, il est possible de définir des propriétés supplémentaires pour compléter sa définition.

Voir aussi [Contrôles sur les éléments du modèle de données](#) [p 61]

Ce chapitre contient les sections suivantes :

1. [Propriétés basiques des éléments](#)
2. [Propriétés avancées des éléments](#)

8.1 Propriétés basiques des éléments

Propriétés basiques communes

Les propriétés basiques suivantes sont disponibles pour plusieurs types d'éléments :

Information	Informations supplémentaires non internationalisées associées à l'élément.
Nombre minimum de valeurs	<p>Nombre minimum de valeurs pour cet élément.</p> <p>Les clés primaires ne pouvant être multi-valuées, cette propriété doit être égale à "1" ou être "non définie".</p> <p>Pour les éléments de type "Noeud de sélection", le nombre minimum est automatiquement défini à "0".</p>
Nombre maximum de valeurs	<p>Nombre maximum de valeurs pour cet élément. Si cette propriété est supérieure à "1", l'élément est considéré comme multi-valué.</p> <p>Les clés primaires ne pouvant être multi-valuées, cette propriété doit être égale à "1" ou être "non définie".</p> <p>Pour une table, le nombre maximum d'éléments est défini à "unbounded" par défaut lors de sa création.</p> <p>Pour les éléments de type "Noeud de sélection", le nombre maximum d'éléments est défini à "0" par défaut lors de la définition des propriétés du noeud de sélection.</p>
Règles de validation	<p>Cette propriété est disponible pour les champs situés dans une table, sauf pour les champs de type Mot de passe, les types réutilisables, les champs des types complexes réutilisables et les noeuds de sélection. Cette propriété est utilisée pour définir des règles de validation riches et complexes à l'aide d'un éditeur de prédicats XPath 1.0.</p> <p>Voir éditeur de prédicats [p 329].</p> <p>Une règle de validation peut être utile lorsque la validation de la valeur dépend de critères complexes ou des valeurs d'autres champs.</p> <p>En utilisant l'assistant associé, il est possible de définir des libellés localisés pour la règle de validation, ainsi qu'un message localisé avec la sévérité qui s'affichera si le critère n'est pas satisfait.</p> <p>Si une règle de validation est définie sur un champ de type table, alors cette règle sera considérée comme une "contrainte sur table" et sera exécutée sur chaque</p>

enregistrement de la table. Voir [Contraintes sur table](#) [p 493] pour plus d'informations.

Propriétés basiques des champs

Les propriétés basiques suivantes sont spécifiques aux champs simples :

Valeur par défaut	<p>Définit une valeur par défaut pour ce champ.</p> <p>Cette valeur sera insérée automatiquement dans le champ de saisie du formulaire de création des nouveaux enregistrements. Le type de la valeur par défaut doit être compatible avec le type du champ courant.</p> <p>Voir Valeur par défaut [p 509] pour plus d'informations.</p>
Message d'erreur de conversion	<p>Messages d'erreur internationalisés affichés aux utilisateurs lors de la saisie d'une valeur qui n'est pas compatible avec le type du champ courant.</p>
Règle de calcul	<p>Cette propriété est disponible pour les champs des tables, excepté pour les types réutilisables. Définit une règle de calcul pour la valeur du champ avec l'aide d'un éditeur de prédicats XPath 1.0.</p> <p>Voir Editeur de prédicats [p 329] pour plus d'informations.</p> <p>Une règle de calcul peut être utile lorsqu'une valeur dépend d'autres valeurs dans le même enregistrement, mais qu'un calcul programmatique n'est pas nécessaire.</p> <p>Les limitations suivantes existent pour les règles de calcul :</p> <ul style="list-style-type: none">• Les règles de calcul s'utilisent uniquement avec les champs simples d'une table.• Les règles de calcul ne peuvent pas être définies sur des champs du type OResource ou Password.• Les règles de calcul ne peuvent pas être définies sur les noeuds de sélection ni les champs clés primaires.• Les règles de calcul ne peuvent pas être définies en accédant à l'élément depuis le rapport de validation.

8.2 Propriétés avancées des éléments

Propriétés avancées communes

Les propriétés avancées suivantes sont disponibles pour plusieurs types d'éléments :

UI bean	<p>Cette propriété est disponible pour tous les types d'éléments sauf les tables.</p> <p>Spécifie une classe Java permettant de personnaliser l'interface utilisateur associée à cet élément dans un jeu de données. Un UI bean peut afficher l'élément différemment et/ou modifier sa valeur en étendant la classe <code>UIBeanEditor^{API}</code> dans l'API Java.</p>
Transformation à l'export	<p>Cette propriété est disponible pour les champs et pour les groupes qui sont des noeuds terminaux.</p> <p>Spécifie une classe Java définissant des opérations de transformation à effectuer lors de la création d'une archive à partir d'un jeu de données associé. La transformation à effectuer porte sur la valeur de l'élément courant.</p> <p>Voir <code>NodeDataTransformer^{API}</code> pour plus d'informations.</p>
Propriétés d'accès	<p>Définit le mode d'accès pour l'élément courant, à savoir : s'il peut être écrit et/ou lu.</p> <ul style="list-style-type: none"> • "Lecture & Ecriture" correspond au mode RW dans le XSD du modèle de données. • "Lecture seule" correspond au mode R- dans le XSD du modèle de données. • "Elément hors d'un jeu de données" correspond au mode CC dans le XSD du modèle de données. • "Noeud non terminal" correspond au mode -- dans le XSD du modèle de données. <p>Voir Access properties [p 509] pour plus d'informations.</p>
Affichage par défaut	<p>Cette propriété permet d'indiquer :</p> <ul style="list-style-type: none"> • Si cet élément doit être affiché dans la vue par défaut d'un jeu de données. Lorsque la propriété "Caché" est sélectionnée, cet élément ne sera pas affiché dans la vue par défaut d'un jeu de données. Si cette propriété n'est pas renseignée, alors l'élément sera affiché par défaut. • Si cet élément doit être affiché dans les outils de recherche d'un jeu de données. Lorsque la propriété "Caché dans toutes les recherches" est sélectionnée, cet élément ne sera pas affiché dans les outils de

recherche textuelle et typée d'un jeu de données. Lorsque la propriété "Caché uniquement dans une recherche textuelle" est sélectionnée, cet élément sera masqué uniquement dans l'outil de recherche textuelle d'un jeu de données. Si cette propriété n'est pas renseignée, alors l'élément sera affiché par défaut dans l'outil de recherche textuelle d'un jeu de données.

- Si cet élément doit être inclus lors de l'opération select d'un data service. Lorsque la propriété "Exclu des Data Services" est sélectionnée, cet élément ne sera pas inclus lors de l'opération select d'un data service. Si cette propriété n'est pas renseignée, alors cet élément sera inclus par défaut lors de l'opération select d'un data service.

Voir [Default view](#) [p 511] pour plus d'informations.

Affichage par défaut > Sélecteur combo box

Définit le nom de la vue publiée utilisée dans le menu de sélection de la clé étrangère.

Un bouton de sélection sera affiché en bas à droite de la liste déroulante du menu de sélection. Lors de la définition d'une clé étrangère, cette fonctionnalité permet d'accéder à une vue de sélection avancée en cliquant sur le bouton 'Sélecteur' qui ouvre la vue de sélection avancée, permettant ainsi d'utiliser les options de tri et de recherche.

Si aucune vue publiée n'a été définie, la vue de sélection avancée sera désactivée.

Si le chemin de la table référencée est absolu, alors seules les vues publiées correspondant à cette table seront affichées. Si le chemin de la table référencée est relatif, alors toutes les vues associées au modèle de données contenant la table cible seront affichées.

Voir [Defining a view for the combo box selector of a foreign key](#) [p 512] pour plus d'informations.

Catégorie du noeud

Définit les catégories permettant de restreindre la visualisation des données. Un noeud de catégorie "Hidden" est caché par défaut dans un jeu de données.

Restriction : la définition de catégorie ne s'applique pas aux noeuds d'enregistrement de tables, à l'exception de la catégorie "Hidden".

Voir [Catégories](#) [p 515] pour plus d'informations.

Mode de comparaison

Définit un mode de comparaison pour l'élément. Ce mode permet de restreindre la comparaison des données associées à l'élément.

- "Défaut" implique que l'élément est pris en compte lors de la comparaison de données.
- "Ignoré" implique qu'aucune modification ne sera détectée lors de la comparaison de deux entités modifiées (jeux de données ou enregistrements).

Lors de la fusion de données, les valeurs des éléments ignorés dans des jeux de données ou enregistrements modifiés ne sont pas fusionnées. Les valeurs contenues dans de nouveaux jeux de données ou enregistrements sont fusionnées.

Lors de l'import d'une archive, les valeurs des éléments ignorés dans des jeux de données ou enregistrements modifiés ne sont pas importées. Les valeurs contenues dans de nouveaux jeux de données ou enregistrements sont importées.

Voir [Comparison mode](#) [p 514].

Règle d'application des dernières modifications

Indique si cet élément doit être inclus ou non dans le service permettant d'appliquer à d'autres enregistrements de la même table les dernières modifications effectuées.

- "Défaut" indique que la dernière modification effectuée sur cet élément peut être appliquée à d'autres enregistrements.
- "Ignoré" indique que la dernière modification effectuée sur cet élément ne peut pas être appliquée à d'autres enregistrements. Cet élément ne sera pas visible dans le service permettant d'appliquer les dernières modifications.

Voir [Apply last modifications policy](#) [p 514] pour plus d'informations.

Propriétés avancées des champs

Les propriétés avancées suivantes sont spécifiques aux champs.

Contrôle de saisie sur valeur nulle

Implémente la propriété `osd:checkNullInput`. Cette propriété est utilisée pour activer et vérifier une contrainte sur valeur `null` lors de la saisie utilisateur.

Par défaut, pour permettre une saisie temporairement incomplète, la validation des éléments obligatoires est effectuée non pas lors de la saisie utilisateur, mais pendant la validation du jeu de

données. Si un élément obligatoire doit être vérifié immédiatement lors de la saisie utilisateur, cette propriété doit avoir la valeur "oui".

Note

Une valeur est considérée comme obligatoire si le "Nombre minimum de valeurs" est égal ou supérieur à "1". Pour les éléments terminaux, les valeurs obligatoires sont seulement vérifiées dans les jeux de données activés. Pour les éléments non terminaux, les valeurs sont vérifiées indépendamment de l'état d'activation du jeu de données.

Voir [Constraints, triggers and functions](#) [p 496] pour plus d'informations.

UI bean

Voir [Propriétés avancées communes](#) [p 50].

Fonction (valeur calculée)

Cette propriété est disponible pour les champs qui ne sont pas des clés primaires. Elle spécifie une classe Java permettant de calculer la valeur de ce champ programmatiquement. Peut être utile si la valeur de ce champ dépend d'autres valeurs dans le référentiel, ou si le calcul de la valeur doit récupérer des données depuis un système externe.

Une fonction peut être créée en implémentant l'interface `ValueFunctionAPI`.

Transformation à l'export

Voir [Propriétés avancées communes](#) [p 50].

Propriétés d'accès

Voir [Propriétés avancées communes](#) [p 50].

Noeud de sélection

Cette propriété est disponible uniquement pour les champs de type "chaîne de caractères". Elle définit une sélection d'enregistrements en utilisant une expression XPath sur une table dans un jeu de données. Peut être utile pour associer des enregistrements liés au champ courant. Les enregistrements qui répondent au critère sont affichés quand l'utilisateur clique sur le lien généré. Lorsque cette propriété est renseignée, les propriétés "Nombre minimum de valeurs" et "Nombre maximum de valeurs" sont automatiquement définies à "0".

Voir [Noeuds de sélection](#) [p 481] pour plus d'informations.

Auto-incrément

Cette propriété est disponible uniquement pour les champs de type "entier" contenus dans une table. Si elle est activée, la valeur du champ est calculée automatiquement lors de la création d'un nouvel

enregistrement. Peut être utile pour les clés primaires, car l'auto-incrément génère un identifiant unique pour chaque enregistrement. Deux attributs peuvent être spécifiés :

Valeur de démarrage	Valeur initiale de l'auto-incrément. Si aucune valeur n'est définie, la valeur par défaut est "1".
Pas de l'incrément	La valeur ajoutée à la valeur précédente de l'auto-incrément. Si aucune valeur n'est définie, la valeur par défaut est "1".
Désactiver les contrôles de l'auto-incrément	Indique s'il faut désactiver le contrôle de la valeur d'un champ auto-incrémenté par rapport à la valeur maximale trouvée dans la table en cours de mise à jour.

Les valeurs auto-incrémentées ont le comportement suivant :

- Le calcul et l'allocation de la valeur de ce champ sont effectués dès qu'un nouvel enregistrement est inséré et que la valeur du champ est indéfinie.
- Aucune allocation ne peut être réalisée si l'insertion programmatique spécifie déjà une valeur différente de null. Par conséquent, cette allocation n'est pas effectuée à l'insertion d'un enregistrement en mode occultation ou réécriture.
- Si une archive importée spécifie cette valeur, alors elle est préservée.
- Une valeur nouvellement allouée est, autant que possible, unique au sein du référentiel.

Plus précisément, le caractère unique de l'allocation s'étend à tous les jeux de données basés sur le modèle de données, et également à tous les espaces de données. Ce dernier cas de figure permet de fusionner un espace de données à son parent avec une garantie raisonnable d'absence de conflit lorsque la valeur auto-incrémentée fait partie des enregistrements d'une clé primaire.

Ce principe a une limitation très spécifique : quand une opération majeure de mise à jour spécifiant des valeurs est réalisée en simultané d'une opération allouant une valeur au même champ, il est possible que cette dernière opération alloue une valeur qui sera fixée par la première opération (il n'y a pas de verrouillage entre plusieurs espaces de données).

Voir [Valeurs auto-incrémentées](#) [p 500] pour plus d'informations.

Affichage par défaut

Voir [Propriétés avancées communes](#) [p 50].

Catégorie du noeud

Voir [Propriétés avancées communes](#) [p 51].

Champ hérité

Définit une relation entre le champ courant et un champ dans une autre table afin de chercher sa valeur automatiquement.

Enregistrement source	Une clé étrangère ou une séquence de clés étrangères, séparées par des espaces, permettant de trouver l'enregistrement dont hérite le champ courant. Si cette propriété n'est pas renseignée, alors l'élément courant sera utilisé comme source d'héritage de champ.
Élément source	Chemin XPath définissant l'élément à hériter. L'élément source doit être terminal, se trouver dans "Enregistrement source", et son type doit être le même que le type de ce champ. Cette propriété est obligatoire pour l'héritage de champ.

Voir [héritage](#) [p 23] dans le glossaire.

Pour plus d'informations, voir aussi [Inherited fields](#) [p 313].

Propriétés avancées des tables

Les propriétés avancées suivantes sont spécifiques aux tables.

Table

Clé primaire	<p>Liste des champs composant la clé primaire de la table. Il est possible d'ajouter ou de supprimer des champs de la clé primaire.</p> <p>Chaque champ de la clé est identifié par un chemin XPath absolu qui débute sous la table.</p> <p>Si la clé est composite, la liste des champs doit être séparée par des espaces. Par exemple, "/name /startDate".</p>
Présentation	<p>Spécifie la manière dont les enregistrements de cette table sont affichés dans l'interface utilisateur dans les jeux de données associés.</p>
Labellisation des enregistrements	<p>Définit les champs composant le libellé par défaut et les libellés localisés pour les enregistrements de la table.</p> <p>Peut aussi définir une classe Java pour affecter le libellé programmatiquement, ou définir le libellé dans une hiérarchie. Cette classe Java doit implémenter l'interface <code>UILabelRenderer^{API}</code> ou <code>UILabelRendererForHierarchy^{API}</code> de l'API Java.</p> <p>Attention : Les droits d'accès définis dans les jeux de données associés ne sont pas appliqués lors de l'affichage des libellés. Les champs habituellement cachés à cause de droits d'accès seront affichés dans les libellés des enregistrements.</p>
Présentation par défaut des groupes dans un formulaire	<p>Spécifie la politique d'affichage par défaut des groupes contenus dans cette table. Si cette propriété n'est pas définie, alors la politique par défaut sera utilisée pour afficher les groupes.</p> <p>Voir Ergonomie & comportement [p 400] pour plus d'informations.</p> <p>Présentation activée des groupes : spécifie un mode de présentation autorisé en complément de "Développé" et "Réduit", qui sont toujours disponibles. Les liens doivent être activés sur la table afin de définir spécifiquement le mode de présentation des groupes en tant que liens.</p> <p>Présentation par défaut des groupes : spécifie la présentation par défaut des groupes contenus dans cette table. Si un groupe ne spécifie aucune vue par défaut, alors le mode de présentation par défaut défini par cette table sera utilisé. En fonction des performances du réseau et du navigateur, ajustez la manière d'afficher chaque groupe du formulaire. En ce qui concerne le poids de</p>

la page téléchargée, le mode "Lien" est léger, les modes "Développé" et "Réduit" sont plus lourds.

Note : quand les onglets sont activés dans une table, tous les groupes qui utiliseraient les liens dans les autres cas sont convertis automatiquement en mode "Réduit". Cela permet d'éviter les complexités d'affichage qui se posent quand les liens et les onglets sont dans la même interface utilisateur.

Présentation spécifique d'un formulaire	Définit une présentation spécifique pour personnaliser le formulaire d'un enregistrement dans les jeux de données associés.
Historisation	<p>Spécifie si l'historisation est activée, et le niveau de garantie demandé. Les profils d'historisation peuvent être modifiés dans la section Administration > Historique et journaux.</p> <p>Voir Configuration de l'historique dans le référentiel [p 289] pour plus d'informations.</p>
Index	<p>Propriété permettant d'indexer certains champs de la table. L'indexation améliore le temps d'accès des requêtes portant sur ces champs. La combinaison des champs dans chaque index doit être unique.</p> <p>Nom de l'index: Nom unique pour cet index.</p> <p>Champs à indexer: Les champs à indexer, où chaque champ d'index est identifié par un chemin XPath absolu qui débute sous la table.</p>
Filtres spécifiques	Définit des filtres pour afficher uniquement les enregistrements correspondant à certains critères.
Actions	Indique les actions autorisées ou interdites dans le contexte d'un jeu de données. Toutes les actions sont autorisées par défaut, cependant des droits d'accès peuvent être définis dans les jeux de données associés pour restreindre ces actions.

Contraintes d'unicité

Indique les champs dont les valeurs doivent être uniques dans la table.

Triggers à la mise à jour

Spécifie des classes Java définissant des méthodes exécutées automatiquement lorsque des opérations sont effectuées : création, mise à jour, suppression, etc.

Un trigger natif est inclus par défaut pour lancer les workflows de données.

Voir [Triggers](#) [p 499] pour plus d'informations.

Propriétés d'accès

Voir [Propriétés avancées communes](#) [p 50].

Affichage par défaut

Voir [Propriétés avancées communes](#) [p 50].

Catégorie du noeud

Voir [Propriétés avancées communes](#) [p 51].

Propriétés avancées des groupes

Les propriétés avancées suivantes sont spécifiques aux groupes.

Classe du conteneur de valeurs (JavaBean)

Définit une classe Java pour contenir les valeurs des éléments situés sous le groupe. La classe Java spécifiée doit être conforme à la spécification JavaBean. Cela signifie que chaque élément enfant de ce groupe doit correspondre à une propriété de la classe Java. Toutes les propriétés de la classe Java doivent avoir des méthodes d'accès (getters et setters).

UI bean

Voir [Propriétés avancées communes](#) [p 50].

Transformation à l'export

Voir [Propriétés avancées communes](#) [p 50].

Propriétés d'accès

Voir [Propriétés avancées communes](#) [p 50].

Affichage par défaut

Visibilité	Voir Propriétés avancées communes [p 50].
-------------------	-----------------------------------------------------------

Présentation	<p>Définit la manière dont ce groupe sera présenté dans un jeu de données. Si cette propriété n'est pas définie, alors la vue par défaut définie par la table parente sera utilisée. Les options "Onglet" et "Lien" sont disponibles uniquement si la table parente a activé ces options de présentation.</p> <p>Propriétés de l'onglet</p> <p>Position: Cet attribut spécifie la position de l'onglet par rapport à tous les onglets définis dans le modèle. Cette position est utilisée pour ordonnancer la liste des onglets au moment de l'affichage d'un formulaire. Si cette propriété n'est pas définie, alors l'onglet sera positionné en fonction de la position du groupe dans le modèle de données.</p>
---------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Catégorie du noeud

Voir [Propriétés avancées communes](#) [p 51].

Concepts apparentés [Contrôles sur les éléments du modèle de données](#) [p 61]

CHAPITRE 9

Contrôles sur les éléments du modèle de données

Après la création d'un élément, complétez sa définition avec des contrôles supplémentaires.

Voir aussi [Propriétés des éléments du modèle de données](#) [p 47]

Ce chapitre contient les sections suivantes :

1. [Contrôles simples](#)
2. [Contrôles avancés](#)

9.1 Contrôles simples

Il est possible d'établir des contrôles simples sur le contenu d'un champ en définissant des contrôles statiques. Les contrôles disponibles dépendent du type du champ.

Longueur fixe	Nombre exact de caractères requis pour ce champ.
Longueur minimale	Nombre minimum de caractères requis pour ce champ.
Longueur maximale	Nombre maximum de caractères autorisés pour ce champ.
Pattern	Expression régulière à laquelle la valeur du champ doit être conforme. Il est interdit de définir un pattern à la fois sur un champ et sur son type de donnée.
Partie décimale	Nombre maximum de chiffres autorisés dans la partie décimale de la valeur d'un champ de type décimal.
Nombre maximum de chiffres	<p>Nombre maximum de chiffres composant la valeur d'un champ de type entier ou décimal.</p> <p>Pour les champs de type "Décimal", la longueur maximale par défaut est fixée à 15 chiffres. Cette valeur par défaut est définie pour éviter des problèmes de précision ou d'arrondi lors de la saisie de nombres dépassant les 15 chiffres. En effet, si une valeur dépassant les 15 chiffres était saisie dans un formulaire, la mauvaise valeur remplacerait l'originale lors de la soumission de ce formulaire.</p>
Énumération	Définit une liste de valeurs possibles pour ce champ. Si une énumération est définie à la fois sur ce champ et sur son type de données, alors une énumération représentant l'intersection entre ces deux énumérations sera utilisée dans les jeux de données associés au modèle de données.
Supérieur à [constante]	Définit la valeur minimale autorisée pour ce champ.
Inférieur à [constante]	Définit la valeur maximale autorisée pour ce champ.

Voir [Facettes XML schema supportées](#) [p 485].

9.2 Contrôles avancés

Il est possible d'établir des contrôles avancés sur le contenu d'un élément en définissant des contrôles dynamiques contextuels. Les contrôles disponibles pour un élément dépendent de sa nature (table, groupe, etc.) et de son type de données.

Voir aussi [Dynamic constraints](#) [p 489]

Contrainte de clé étrangère	
Table	Définit la table ciblée par la clé étrangère. Une clé étrangère référence une table dans le même jeu de données par défaut. Elle peut aussi référencer une table d'un autre jeu de données du même espace de données, ou un jeu de données d'un autre espace de données.
Mode	Emplacement de la table ciblée par la clé étrangère. "Défaut" : modèle courant. "Autre jeu de données" : jeu de données qui se trouve dans le même espace de données. "Autre espace de données" : jeu de données appartenant à un espace de données différent.
Table référencée	Expression XPath indiquant l'emplacement de la table. Par exemple, /racine/MaTable.
Jeu de données référencé	Obligatoire si la table est dans un autre jeu de données. Le nom unique du jeu de données contenant la table référencée.
Espace de données référencé	Obligatoire si la table est dans un autre espace de données. Nom unique de l'espace de données contenant la table référencée.
Libellé	Définit les champs composant un libellé par défaut et des libellés localisés pour présenter les enregistrements de la table cible. Peut aussi spécifier une classe Java qui définit le libellé programmatiquement quand "Expression XPath" a la valeur "Non". Cette classe Java doit implémenter l'interface <code>TableRefDisplay^{API}</code> de l'API Java. Attention : Les droits d'accès définis dans les jeux de données associés ne sont pas appliqués lors de l'affichage des libellés. Les champs habituellement cachés à cause de droits d'accès seront affichés dans les libellés des enregistrements.
Filtre	Définit un filtre de clé étrangère en utilisant une expression XPath. Peut aussi spécifier une classe Java qui implémente l'interface <code>TableRefFilter^{API}</code> de l'API Java.

Supérieur à [variable]	Définit un champ qui spécifie la valeur minimale autorisée pour ce champ.
Inférieur à [variable]	Définit un champ qui spécifie la valeur maximale autorisée pour ce champ.
Longueur fixe [dynamique]	Définit un champ qui spécifie le nombre exact de caractères requis pour ce champ.
Longueur minimale [dynamique]	Définit un champ qui spécifie le nombre minimum de caractères requis.
Longueur maximale [dynamique]	Définit un champ qui spécifie le nombre maximum de caractères autorisés.
Valeurs exclues	Définit une liste de valeurs non autorisées pour ce champ.
Plage exclue de valeurs	<p>Définit une plage de valeurs non autorisées pour ce champ.</p> <p>Valeur maximale exclue : La valeur maximale non autorisée pour ce champ.</p> <p>Valeur minimale exclue : La valeur minimale non autorisée pour ce champ.</p>
Contrainte spécifique (composant)	Spécifie une ou plusieurs classes Java qui implémentent l'interface <code>Constraint^{API}</code> de l'API Java. Voir Programmatic constraints [p 492] pour plus d'informations.
Énumération spécifique (composant)	Spécifie une classe Java pour définir une énumération. La classe doit définir une liste ordonnée de valeurs, en implémentant l'interface <code>ConstraintEnumeration^{API}</code> de l'API Java.
Énumération alimentée par un autre noeud	Spécifie un champ qui définit les valeurs autorisées pour cette énumération. Le champ spécifié doit être une liste ou doit définir une énumération.

Propriétés de validation

Chaque contrainte, excepté celles utilisant une classe Java, peut définir des messages de validation. Il est possible d'associer une sévérité à ces messages de validation et ils peuvent être localisés en utilisant les propriétés suivantes :

Validation	Spécifie le message de validation et la sévérité associée à la contrainte.
Sévérité	Spécifie la sévérité de la contrainte. Les valeurs possibles sont : "Erreur", "Avertissement", et "Information".
Gestion des erreurs	Spécifie le comportement de la contrainte en cas d'erreur de validation. Il est possible d'indiquer que la contrainte doit toujours être valide pour toute opération (mise à jour d'un jeu de données, création, mise à jour et suppression d'un enregistrement), ou lorsqu'un utilisateur soumet un formulaire. Dans ce cas, toutes les saisies ou opérations qui rendraient la contrainte invalide seront rejetées et les données seront non modifiées. Si cette propriété n'est pas spécifiée, alors la contrainte bloquera uniquement les erreurs lors de la soumission d'un formulaire, sauf dans le cas de modèles relationnels où les contraintes de clés étrangères bloquent par défaut toutes les erreurs. Cette propriété est uniquement disponible sur les contrôles statiques, valeurs exclues, plage exclue de valeurs et contraintes de clés étrangères. Le caractère bloquant pour toutes les opérations ne s'applique pas sur les filtres de clé étrangère. Une contrainte de clé étrangère n'est donc pas bloquante si un enregistrement référencé existe mais ne répond pas aux critères de filtrage. Il n'est pas possible de définir cette propriété sur les contraintes structurelles définies dans les modèles relationnels. Si cette propriété est définie sur les contraintes "Longueur fixe", "Longueur maximum", "Nombre total de chiffres" et "Partie décimale" dans des modèles relationnels, une erreur sera levée lors de la compilation du modèle. Le caractère bloquant d'une contrainte est ignoré lors de l'import d'archives. Toutes les contraintes bloquantes, sauf les contraintes dites structurelles, sont désactivées lors de l'import d'archives.
Message	Message à afficher lorsque la valeur de ce champ dans un jeu de données ne respecte pas cette contrainte. Ce message peut être localisé.

Concepts apparentés [Propriétés des éléments du modèle de données](#) [p 47]

CHAPITRE 10

Actions sur les modèles de données existants

Lorsque le modèle de données est créé, des actions sont disponibles dans le menu **"Actions" du modèle de données** [p 31] dans le panneau de navigation.

Ce chapitre contient les sections suivantes :

1. [Validation du modèle de données](#)
2. [Import et export de fichiers XML Schema Document \(XSD\)](#)
3. [Duplication d'un modèle de données](#)
4. [Suppression d'un modèle de données](#)

10.1 Validation du modèle de données

Il est possible de valider un modèle de données en sélectionnant **"Actions" du modèle de données** > **Valider** dans le panneau de navigation. Les éventuels messages issus de la validation du modèle de données sont présentés dans un rapport. Depuis le rapport de validation, cliquez sur le bouton **Revalider** pour mettre à jour ce rapport, ou cliquez sur le bouton **Réinitialiser le rapport de validation** pour supprimer tous les messages de validation actuellement associés au modèle de données afin de pouvoir relancer une validation complète.

Voir [Validation](#) [p 336] pour obtenir plus d'informations concernant la validation incrémentale de données.

10.2 Import et export de fichiers XML Schema Document (XSD)

EBX5 fournit des services intégrés pour importer et exporter des fichiers XML Schema Document (XSD). Les services d'import et d'export sont accessibles depuis le menu **"Actions" du modèle de données** [p 31] dans le panneau de navigation. Un import ou export est toujours effectué sur l'intégralité du modèle de données. Lors d'un import, la structure du modèle de données courant est entièrement remplacée par le contenu du document XML Schema importé. Lors d'un export, le modèle de données complet est exporté dans le document XML Schema cible.

Pour importer un fichier XSD, le fichier doit être valide et conforme aux règles de validation du référentiel EBX5. Si le document déclare des ressources situées dans un module, le module doit être déclaré aussi dans la configuration du modèle de données. Si le module n'a pas été déclaré, le fichier

XSD ne pourra pas être importé. Voir [Propriétés du modèle de données](#) [p 38] pour plus d'informations sur la déclaration des modules.

Pour importer un modèle, sélectionnez *Importer XSD* dans le menu **Actions** situé dans le panneau de navigation du modèle de données.

Un document XML Schema (XSD) peut être importé à partir du système de fichier local. Pour cela, sélectionnez *Importer à partir d'un document local* :

- **Nom du document** : chemin du document XSD à importer dans le système de fichiers local.

Il est possible d'importer un document XML Schema (XSD) contenu dans le référentiel. Pour cela, sélectionnez *Importer à partir du référentiel de modèles* :

- **Modèle** : nom du modèle de données à importer.

Un modèle de données XSD peut aussi être importé dans un module. L'import d'un modèle de données XSD depuis un module utilise les propriétés suivantes :

Module	Module qui contient le modèle de données.
Chemin du module	Localisation physique du module sur le système de fichier du serveur.
Chemin des sources	Codes sources utilisés pour configurer les "Règles et objets métier". Si le chemin est relatif, il sera résolu à partir du "Chemin du module". Cette propriété est obligatoire si le modèle définit des éléments programmatiques.
Modèle	Modèle de données à importer.

Note

Les fichiers XSD à importer doivent être encodés en "UTF-8". Les fichiers XSD exportés sont toujours encodés en "UTF-8".

10.3 Duplication d'un modèle de données

Pour dupliquer un modèle de données, sélectionnez "Dupliquer" dans le menu **Actions** du modèle. Nommez le nouveau modèle de données. Ce nom doit être unique dans le référentiel.

10.4 Suppression d'un modèle de données

Pour supprimer un modèle de données, sélectionnez "Supprimer" dans le menu [Actions du modèle de données](#) [p 31]. Quand un modèle de données est supprimé, toutes les publications associées au modèle restent disponibles. Si un nouveau modèle de données est recréé avec le même nom que celui qui a été supprimé, le nouveau modèle de données sera réassocié avec toutes les publications existant

dans le référentiel. Au moment de la publication, il sera possible de confirmer le remplacement d'une publication existante.

Note

Seul un administrateur peut supprimer les publications d'un modèle de données dans la section "Administration".

Voir [Publication des modèles de données](#) [p 71] pour plus d'informations sur le processus de publication.

CHAPITRE 11

Publication du modèle de données

Ce chapitre contient les sections suivantes :

1. [A propos des publications](#)
2. [Modes de publication](#)
3. [Mode de publication "Embarqué"](#)

11.1 A propos des publications

Chaque jeu de données dans le référentiel EBX5 basé sur un **modèle de données embarqué** est associé à une publication d'un modèle de données, et non directement au modèle de données défini dans le **Data Model Assistant**. Une publication est créée la première fois qu'un modèle de données est publié en utilisant le bouton **Publier** du panneau de navigation. Il est possible, à partir d'une publication, de créer des jeux de données dont la structure sera basée sur la structure du modèle de données publié.

Note

Le bouton **Publier** est uniquement affiché pour les utilisateurs qui ont le droit de publier le modèle de données. Voir [Permissions du modèle de données](#) [p 37] pour plus d'informations.

Les jeux de données étant basés sur des publications, les modifications faites sur le modèle de données n'impacteront pas les jeux de données existants. Les jeux de données seront impactés uniquement lors de la mise à jour d'une publication existante.

11.2 Modes de publication

Un modèle de données peut être publié en mode "Embarqué" ou "Dans un module". Le mode de publication "Embarqué" génère une publication gérée et persistée dans le référentiel EBX5 et possède des fonctionnalités spécifiques dédiées à la gestion de version. Le mode de publication "Dans un module" crée un fichier XML Schema Document (XSD) à l'intérieur d'un module qui n'est pas géré par le référentiel EBX5.

Selon la configuration du modèle de données, EBX5 détermine automatiquement le processus de publication à utiliser lorsque l'on clique sur le bouton **Publier** dans le panneau de navigation. Quand un modèle de données spécifie le mode de publication "Dans un module" ainsi qu'un fichier XSD à cibler, le processus de publication génère le fichier XSD dans le module défini dans la configuration.

11.3 Mode de publication "Embarqué"

La première fois qu'un modèle de données est publié en mode embarqué, une nouvelle publication avec le même nom que le modèle est automatiquement créée dans le référentiel. Si différentes publications ont été créées à partir du modèle, sélectionnez celle à mettre à jour.

Voir [Consultation et création des publications](#) [p 72] pour plus d'informations sur l'utilisation de différentes publications.

Pendant le processus de publication, si le modèle de données a déjà été publié, il est possible de visualiser dans une interface de comparaison les différences structurelles introduites par la nouvelle publication.

Le processus de publication permet aussi de créer une image en lecture seule de l'état actuel du modèle de données. Cette image sera utile si un précédent état du modèle doit être restauré après plusieurs modifications et publications du modèle de données.

Note

Les images sont des archives statiques du modèle de données et ne doivent pas être confondues avec les *versions* du modèle de données, qui sont des éditions du modèle évoluant en parallèle. Voir [Gestion des versions du modèle de données embarqué](#) [p 73] pour plus d'informations sur les versions des modèles de données.

Consultation et création des publications

Pour accéder aux publications existantes du modèle de données courant, sélectionnez "Gérer les publications" dans son menu ["Actions" du modèle de données](#) [p 31] dans le panneau de navigation. Vous pourrez consulter les détails des publications et en créer de nouvelles.

Dans certains cas, il faudra utiliser plusieurs publications du même modèle de données afin de permettre à différents jeux de données d'être basés sur des états différents du modèle. L'utilisation de plusieurs publications doit être réalisée avec précaution. En effet, les utilisateurs devront sélectionner la publication à mettre à jour lors de la publication du modèle de données et doivent donc avoir connaissance de l'utilisation de ces différentes publications. La création d'une nouvelle publication est disponible uniquement pour les utilisateurs ayant le rôle "Administrateur".

Pour créer une nouvelle publication, sélectionnez "Gérer les publications" dans le menu ["Actions" du modèle de données](#) [p 31] dans le panneau de navigation, puis cliquez sur le bouton **Créer une publication**. Le nom donné à la publication doit être unique dans le référentiel. Après sa création, la nouvelle publication est vide et ne représente pas un modèle de données. Pour pouvoir être utilisée par des jeux de données, la publication doit être mise à jour en publiant le modèle de données.

CHAPITRE 12

Gestion des versions de modèles de données

Ce chapitre contient les sections suivantes :

1. [A propos des versions](#)
2. [Accès aux versions](#)
3. [Actions sur les versions](#)
4. [Limitations connues sur la gestion de versions du modèle de données](#)

12.1 A propos des versions


Il est possible de créer des *versions* pour les modèles de données. Cela permet d'avoir différents états du modèle qui évoluent en parallèle. Les versions ne doivent pas être confondues avec les images des modèles de données, qui sont prises au moment de publication et sont sauvegardées uniquement pour consultation d'historique en lecture seule.

12.2 Accès aux versions

Pour voir les versions existantes de votre modèle de données, sélectionnez "Gérer les versions" dans le menu **"Actions" du modèle de données** [p 31].

Les versions existantes sont représentées dans une arborescence selon leurs relations parent-enfant. Chaque modèle de données a une version racine par défaut.

12.3 Actions sur les versions

Dans l'espace de travail, en utilisant le menu avec la flèche vers le bas  à côté de chaque version, accédez aux actions suivantes :

Accéder à la version du modèle	Pour visualiser la version correspondante du modèle de données.
Créer une nouvelle version	Crée une nouvelle version basée sur le contenu de la version sélectionnée. La nouvelle version est créée comme enfant de la version sélectionnée, mais les contenus évoluent indépendamment.
Définir en tant que version par défaut	Définit la version par défaut sélectionnée lorsqu'un utilisateur accède au modèle de données.
Exporter une archive	Exporte le modèle de données sélectionné dans une archive contenant les données de la version, ses permissions et ses informations. L'archive est exportée vers le répertoire d'archives, accessible par les administrateurs du référentiel. Voir Dossier archives [p 383] pour plus d'informations sur le répertoire d'archives.
Importer une archive	Importe le contenu d'une archive dans la version sélectionnée. L'archive à importer doit contenir un modèle de données avec le même nom que le modèle associé à cette version.

Une version peut être supprimée en cliquant le bouton **X** situé à sa droite. Une version ne peut pas être supprimée si elle est liée à une publication ou si elle a des sous versions. La version racine du modèle de données ne peut pas être supprimée.

Deux versions du même modèle peuvent être comparées dans l'espace de travail en sélectionnant leur case à cocher, puis "**Actions**" **du modèle de données** > **Comparer les versions sélectionnées**. La vue de comparaison côte-à-côte affiche les différences structurelles entre les versions du modèle de données, avec la plus ancienne à gauche et la plus récente à droite.

12.4 Limitations connues sur la gestion de versions du modèle de données

- Il est impossible de fusionner deux versions d'un modèle de données.
- L'interface de comparaison n'affiche pas les mises à jour des champs, uniquement les ajouts et suppressions.
- Il n'est pas possible de gérer des versions de modèles de données publiés dans un module.
- Les ressources contenues dans un module et utilisées par un modèle de données embarqué ne sont pas versionnées lorsqu'une version est créée. En effet, seules les références des ressources sont sauvegardées, et les développeurs doivent s'assurer que les ressources référencées restent compatibles avec les différentes versions.

Espaces de données

CHAPITRE 13

Introduction aux espaces de données

Ce chapitre contient les sections suivantes :

1. [Présentation](#)
2. [Utilisation de l'interface utilisateur de la section Espace de données](#)

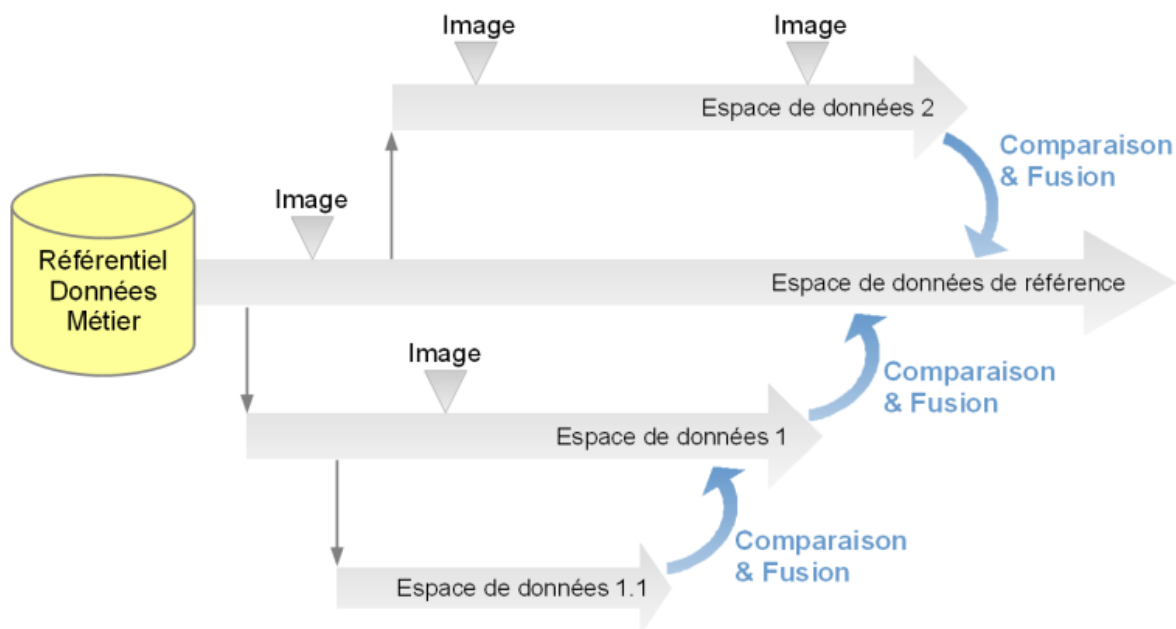
13.1 Présentation

Fonction d'un espace de données

Le cycle de vie des données est souvent complexe. Il est parfois nécessaire d'entretenir une version courante des données tout en travaillant sur des évolutions futures. De plus, il faut conserver une trace des états intermédiaires. EBX5 rend ces démarches possibles grâce aux espaces de données et aux images.

Un espace de données est un conteneur qui isole différentes versions de jeux de données et les organise. Des espaces de données enfants peuvent être créés à partir d'un espace de données. Un espace de données enfant est initialisé avec le même état que son parent au moment de sa création. Ainsi, des modifications peuvent être effectuées isolément dans l'espace de données enfant, sans impacter l'espace de données parent ni d'autres espaces de données. Lorsque les modifications dans l'espace de données enfant sont terminées, cet espace de données peut être fusionné avec son parent.

Une image est une copie statique d'un espace de données qui capture son état et tout son contenu à un moment donné. Les images peuvent être consultées, exportées, et comparées à d'autres espaces de données.



Concepts de base liés aux espaces de données

La compréhension des termes suivants est recommandée pour l'utilisation des espaces de données :

- [espace de données](#) [p 23]
- [image](#) [p 24]
- [jeu de données](#) [p 22]
- [fusion](#) [p 24]
- [espace de données de référence](#) [p 24]

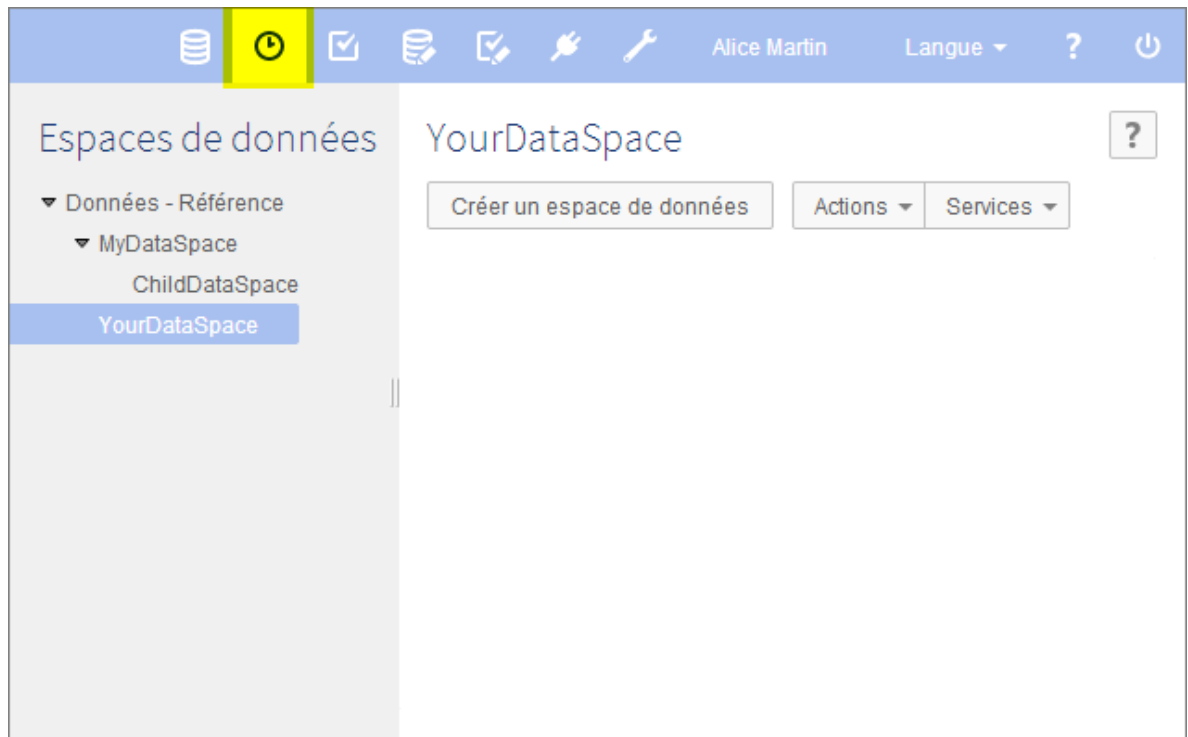
13.2 Utilisation de l'interface utilisateur de la section Espace de données

Les espaces de données sont créés, consultés et modifiés dans la section **Espaces de données**.

Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée'.

Le panneau de navigation affiche l'organisation hiérarchique des espaces de données existants, tandis que l'espace de travail affiche les informations concernant l'espace de données sélectionné et liste ses images.



Voir aussi

[*Création d'un espace de données*](#) [p 81]

[*Images*](#) [p 93]

Concepts apparentés [*Jeux de données*](#) [p 100]

CHAPITRE 14

Création d'un espace de données

Ce chapitre contient les sections suivantes :

1. [Création d'un espace de données](#)
2. [Propriétés](#)
3. [Mode relationnel](#)

14.1 Création d'un espace de données

Par défaut, les espaces de données dans EBX5 sont en *mode sémantique*. Ce mode supporte toutes les fonctionnalités permettant la gestion du cycle de vie des données.

Pour créer un nouvel espace de données en mode sémantique, sélectionnez un espace de données existant sur lequel il se basera, puis cliquez sur le bouton **Créer un espace de données** situé dans l'espace de travail.

Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée'.

L'espace de données nouvellement créé devient un espace de données enfant de celui qui était sélectionné. Son contenu est automatiquement initialisé avec le contenu de son parent au moment de la création. Une image initiale représentant cet état est créée.

Excepté l'espace de données de référence, qui est la racine de tous les espaces de données sémantiques du référentiel, les espaces de données sémantiques sont toujours l'enfant d'un autre espace de données.

Voir aussi [Mode relationnel](#) [p 82]

14.2 Propriétés

Les informations suivantes sont requises lors de la création d'un nouvel espace de données :

Identifiant	Identifiant unique de l'espace de données.
Propriétaire	Utilisateur possédant l'espace de données et qui est autorisé à en modifier les informations et les permissions. Le propriétaire d'un espace de données n'est pas obligatoirement son créateur.
Libellé	Libellé et description associés à l'espace de données en plusieurs langues.
Mode relationnel	Spécifie si l'espace de données est en mode relationnel. Cette option existe seulement lors de la création d'un espace de données à partir de l'espace de données de référence.

14.3 Mode relationnel

Les espaces de données en mode relationnel ne peuvent être créés qu'à partir de l'espace de données de référence. Ils sont limités en fonctionnalités par rapport aux espaces de données en mode sémantique. Par exemple, les espaces de données en mode relationnel n'ont pas d'images et ne peuvent pas avoir d'espaces de données enfants.

Le mode relationnel implique que les tables de données de cet espace soient stockées directement dans un SGBDR.

Voir aussi [Relational mode](#) [p 283]

CHAPITRE 15

Actions sur les espaces de données existants

Ce chapitre contient les sections suivantes :

1. [Informations de l'espace de données](#)
2. [Permissions sur un espace de données](#)
3. [Fusion d'un espace de données](#)
4. [Comparaison d'un espace de données](#)
5. [Validation d'un espace de données](#)
6. [Archives d'espaces de données](#)
7. [Fermeture d'un espace de données](#)

15.1 Informations de l'espace de données

Certaines propriétés associées à un espace de données peuvent être modifiées en sélectionnant **Actions** > **Information** dans l'espace de travail de la section Espaces de Données.

Documentation	Libellé et description localisés associés à l'espace de données.
Stratégie de chargement	<p><i>Seul l'administrateur a le droit de modifier cette option.</i></p> <ul style="list-style-type: none"> • Chargement et déchargement sur demande : le principal avantage de cette stratégie est de permettre de libérer de la mémoire si nécessaire. L'inconvénient est que cela implique un coût de chargement quand on accède à une ressource pour la première fois depuis le démarrage du serveur, ou si elle a été déchargée entre-temps. Ce mode est utilisé par défaut. • Chargement forcé : ce mode est particulièrement recommandé pour les espaces de données et images utilisés de façon intensive, ou exigeants en termes de temps de réponse. • Chargement forcé et pré-validation : ce mode est particulièrement recommandé pour les espaces de données et images utilisés de façon intensive ou exigeants en termes de temps de réponse, et pour lesquels le processus de validation peut être long. <p>Note : Toute modification de la stratégie de chargement requiert un redémarrage du serveur.</p>
Politique de fusion	<p>La politique de fusion des espaces de données enfants ne s'applique qu'aux fusions initiées par un utilisateur. Elle ne s'applique pas aux fusions programmatiques, telles que celles initiées par les scripts de workflow.</p> <p>Les politiques possibles sont :</p> <ul style="list-style-type: none"> • Autoriser des erreurs de validation dans le résultat : c'est la politique par défaut. Un espace de données enfant peut être fusionné quelle que soit la validité du résultat de cette fusion. • Fusion pré-validante : un espace de données enfant ne peut être fusionné que si le résultat de la fusion est valide.
Propriétaire	L'utilisateur possédant l'espace de données et qui est autorisé à en modifier les informations et les permissions.

Le propriétaire d'un espace de données n'est pas obligatoirement son créateur.

Tri des espaces enfants	Définit l'ordre d'affichage des espaces de données enfants dans l'arbre des espaces de données. Si la valeur n'est pas définie, l'ordre défini par le parent est pris en compte. La valeur par défaut est "par libellé".
Changement de propriétaire	Définit si le propriétaire de l'espace de données a le droit de modifier l'attribut "Propriétaire". Si la valeur est "Non habilité", seul l'administrateur a le droit d'effectuer cette modification.
Changement des permissions	Définit si le propriétaire de l'espace de données a le droit de modifier les permissions de l'espace de données. Si la valeur est "Non habilité", seul l'administrateur a le droit d'effectuer cette modification.

15.2 Permissions sur un espace de données

Permissions générales

Identifiant de l'espace de données	Indique l'espace de données auquel les permissions vont être appliquées.
Sélection du profil	Indique le profil concerné par la permission définie.
Restriction d'accès	Indique si la permission définie restreint les permissions affectées à un utilisateur donné par des politiques définies pour d'autres profils. Voir Restriction d'accès [p 323].
Permissions d'accès à l'espace de données	<p>Indique la permission globale d'accès à l'espace de données.</p> <p>Lecture seule</p> <ul style="list-style-type: none"> • Permet de visualiser l'espace de données et son image, et de voir les espaces de données enfants selon les droits s'appliquant à chacun d'eux. • Permet de visualiser le contenu de l'espace de données sans pouvoir le modifier, à condition que les droits s'appliquant au contenu en autorisent l'accès. <p>Ecriture</p> <ul style="list-style-type: none"> • Permet de visualiser l'espace de données et son image, et de voir les espaces de données enfants selon les droits s'appliquant à chacun d'eux. • Permet de modifier le contenu de l'espace de données, à condition que les droits s'appliquant au contenu en autorisent l'accès. <p>Non visible</p> <ul style="list-style-type: none"> • Ni l'espace de données, ni ses images ne peuvent être vus. • A partir d'un espace de données enfant, l'espace de données courant est visible sans toutefois pouvoir être sélectionné. • Le contenu de l'espace de données n'est pas accessible. • Aucune action ne peut être effectuée sur l'espace de données.

Actions possibles

Les utilisateurs peuvent être autorisés à effectuer les actions suivantes :

Créer un espace de données enfant	Indique si le profil peut créer un espace de données enfant.
Créer une image	Indique si le profil peut créer une image de l'espace de données.
Initier une fusion	Indique si le profil peut fusionner un espace de données avec son parent.
Exporter une archive	Indique si le profil peut exporter l'espace de données.
Importer une archive	Indique si le profil peut importer dans l'espace de données.
Fermer un espace de données	Indique si le profil peut fermer l'espace de données.
Fermer une image	Indique si le profil peut fermer les images de l'espace de données.
Droits sur les services	Spécifie les permissions d'accès aux services.
Permissions des espaces de données enfants à la création	Spécifie les permissions d'accès aux espaces de données enfants créés à partir de l'espace de données courant.

15.3 Fusion d'un espace de données

Quand le travail dans un espace de données est terminé, il est possible d'effectuer une fusion unidirectionnelle de l'espace de données vers son espace de données parent. Le processus de fusion est le suivant :

1. l'espace de données parent et l'espace de données enfant sont verrouillés pour tous les utilisateurs, excepté l'utilisateur qui a initié la fusion ainsi que les utilisateurs administrateurs. Les verrous sont maintenus pendant la durée de la fusion. Ainsi, les contenus des deux espaces de données peuvent être lus, mais ne peuvent pas être modifiés.

Note : en plus de s'appliquer aux modifications directes sur l'espace de données, cette restriction s'applique également aux autres fusions des autres espaces de données enfants. Ainsi, il est impossible de fusionner d'autres espaces de données enfants tant que la fusion en cours n'est pas terminée.

2. les changements qui ont été effectués dans l'espace de données depuis sa création sont intégrés dans l'espace de données parent ;
3. l'espace de données enfant est fermé ;
4. l'espace de données parent est déverrouillé.

Initiation d'une fusion

Suivez ces étapes pour initier la fusion d'un espace de données avec son espace de données parent :

1. sélectionnez l'espace de données à fusionner dans le panneau de navigation de la section Espaces de données ;
2. dans l'espace de travail, sélectionnez **Fusionner l'espace de données** dans le menu **Actions**.

Revue et acceptation des changements

Avant la fusion définitive, sélectionnez les changements survenus dans l'espace de données enfant (source) qui doivent être fusionnés dans l'espace de données parent (cible).

Note

Cette étape de revue n'existe pas pour une fusion faite à travers un service de données ou un service programmatique : pour les fusions automatisées, tous les changements dans l'espace de données enfant remplacent les données dans l'espace de données parent.

Pendant la fusion, un écran de comparaison récapitule tous les changements devant être revus. Deux colonnes de *listes de changements* sont affichées, prenant en compte les changements des comparaisons d'espaces de données suivantes :

- l'espace de données enfant par rapport à son image initiale ;
- l'espace de données parent par rapport à l'image initiale de l'espace de données enfant.

Par défaut, tous les changements détectés sont sélectionnés pour la fusion. Pour exclure un changement de la fusion, désélectionnez-le. Les changements relatifs aux différents éléments sont visibles en les sélectionnant dans le panneau de navigation.

Afin de détecter les conflits, la fusion implique l'espace de données courant, l'image initiale et l'espace de données parent. En effet, les données peuvent avoir été modifiées à la fois dans l'espace de données courant et dans son parent.

Le processus de fusion concerne aussi les droits d'accès aux tables. Il faut également passer en revue les modifications des permissions pour décider si elles seront incluses dans la fusion.

Lorsque vous avez choisi les changements à fusionner, vous devez cliquer sur le bouton **Marquer les différences comme revues** pour indiquer que vous avez vérifié les changements dans le périmètre actuel. Tous les changements doivent être revus pour effectuer la fusion.

Types de modifications

Le processus de fusion considère les opérations suivantes comme des modifications à évaluer :

- la création d'un enregistrement ou d'un jeu de données ;
- la modification de toute entité ;
- la suppression d'un enregistrement, d'un jeu de données, ou de la valeur d'un noeud ;
- la modification des permissions d'une table.

Types de conflits

Cette interface de revue affiche également les conflits détectés. Des conflits peuvent survenir quand une entité contient des modifications dans l'espace de données source et l'espace de données cible.

Les conflits sont catégorisés comme suit :

- conflit de création d'un enregistrement ou d'un jeu de données,
- conflit de modification de toute entité,
- conflit de suppression d'un enregistrement ou d'un jeu de données,
- tout autre conflit.

Finalisation d'une fusion

Lorsque tous les changements ont été vérifiés et que ceux à inclure dans le résultat de la fusion ont été choisis, cliquez sur le bouton **Fusionner** >> dans le panneau de navigation.

En fonction de la politique de fusion de l'espace de données parent, le processus de finalisation peut différer. Par défaut, une fusion peut être effectuée même si le résultat de la fusion contient des erreurs de validation. En revanche, l'administrateur de l'espace de données parent peut configurer la politique de fusion pour que les fusions de ses espaces de données enfants soient finalisées uniquement si le résultat ne contient pas d'erreur de validation.

Si la politique de fusion pré-validante est utilisée, un espace de données dédié est d'abord créé pour accueillir le contenu de la fusion. S'il est valide, cet espace de données est automatiquement fusionné avec l'espace de données parent.

Dans le cas contraire, si des erreurs de validation sont détectées dans l'espace de données dédié, seuls seront accessibles l'espace de données d'origine et l'espace de données dédié contenant le résultat de la fusion, nommé "[fusion] <nom de l'espace de données enfant>". Les options suivantes sont alors proposées dans le menu de l'espace de travail **Actions** > **Fusion en cours** :

- **Abandonner la fusion** : cette action abandonne la fusion en cours et restitue l'espace de données enfant dans son état précédant la fusion.
- **Poursuivre la fusion** : cette action permet de tenter de nouveau la fusion après avoir effectué les corrections nécessaires dans l'espace de données de fusion dédié.

Configuration de la politique de fusion d'un espace de données

En tant qu'administrateur d'un espace de données, il est possible, à travers l'interface utilisateur, d'interrompre la finalisation des fusions de ses espaces de données enfants si le résultat contient des erreurs de validation. Pour ce faire, cliquez sur **Actions** > **Informations** dans l'espace de travail de l'espace de données parent. Sur la page des informations de cet espace de données, positionnez la **Politique de fusion des enfants** à "Fusion pré-validante". Cette politique de fusion sera appliquée pour toutes les fusions des espaces de données enfants avec l'espace de données parent ainsi paramétré.

Note

Si la fusion est effectuée à travers un composant web, le comportement pour la politique de fusion est le même ; la politique définie par l'espace de données parent sera automatiquement utilisée lors de la fusion des modifications de l'espace de données enfant. Cependant, cette politique n'est pas appliquée pour les fusions programmatiques. Elle est donc ignorée pour les tâches automatiques des workflows de données.

Voir aussi [Politique de fusion](#) [p 89]

Abandon d'une fusion

Une fusion est effectuée dans le contexte d'une session utilisateur et doit être achevée en une seule opération. Si vous décidez de ne pas poursuivre la fusion après l'avoir initiée, cliquez sur le bouton **Annuler** afin d'abandonner l'opération.

Si vous naviguez vers une autre page pendant une fusion, celle-ci sera abandonnée mais les verrous sur l'espace de données parent et l'espace de données enfant seront maintenus. Il faudra les déverrouiller dans la section **Espaces de Données**.

Pour déverrouiller un espace de données, sélectionnez l'espace de données dans le panneau de navigation, et cliquez sur le bouton **Déverrouiller** dans l'espace de travail. Si vous effectuez le déverrouillage depuis l'espace de données enfant, les deux espaces de données seront déverrouillés. Si vous effectuez le déverrouillage depuis l'espace de données parent, lui seul sera déverrouillé, vous devrez donc aussi effectuer un déverrouillage de l'espace de données enfant.

15.4 Comparaison d'un espace de données

Il est possible de comparer le contenu d'un espace de données à celui d'un autre espace de données ou à une image dans le référentiel. Pour effectuer une comparaison, ouvrez l'espace de données dans le panneau de navigation, puis sélectionnez **Actions > Comparer** dans l'espace de travail. L'assistant permet de choisir un autre espace de données ou une image à comparer avec l'espace de données courant.

Pour une comparaison plus rapide, qui ignore les champs avec une valeur héritée ou calculée, sélectionnez le filtre "Valeurs persistantes seulement".

Voir aussi [Compare contents](#) [p 202]

15.5 Validation d'un espace de données

Le contenu d'un espace de données peut être validé globalement en utilisant le service de validation au niveau de l'espace de données. Ce service est accessible en sélectionnant **Actions > Validation** dans l'espace de travail.

Note

Ce service est proposé uniquement si l'utilisateur a la permission de valider tous les jeux de données contenus dans l'espace de données.

15.6 Archives d'espaces de données

Export d'une archive

Le contenu d'un espace de données peut être exporté dans une archive en sélectionnant **Actions > Export** dans le panneau de navigation. L'archive est sauvegardée sur le système de fichiers du serveur, où seul un administrateur peut la récupérer.

Note

Voir [Archives directory](#) [p 383] dans le Guide d'administration.

Pour réaliser un export, les informations suivantes sont requises:

Nom du fichier archive	Nom de l'archive exportée.
Type d'export	<p>Obligatoire.</p> <p>Le type d'export par défaut est "Contenu complet de l'espace de données". Il permet d'exporter l'ensemble des données sélectionnées dans l'archive.</p> <p>Il peut être utile d'inclure uniquement les différences entre l'espace de données et son image initiale dans l'archive. Il existe deux types d'export qui incluent un delta : "Mises à jour avec leur contenu complet" et "Mises à jour seules". Le premier exporte l'état actuel de l'espace de données ainsi que le delta qui contient les différences entre l'état actuel de l'espace de données et l'image initiale. Le second exporte uniquement le delta qui contient les différences entre l'état actuel de l'espace de données et l'image initiale. Les deux options affichent une page de comparaison, sur laquelle il est possible de sélectionner les différences à inclure dans le delta. Les différences sont détectées au niveau table.</p>
Jeu de données à exporter	<p>Jeux de données de cet espace de données à exporter. Pour chaque jeu de données, il est possible de spécifier si les données, les permissions et les informations doivent être exportées.</p>

Importer une archive

Le contenu d'une archive peut être importé dans un espace de données en sélectionnant **Actions > Import**.

Si l'archive sélectionnée ne contient pas de delta, l'état actuel de l'espace de données sera remplacé par le contenu de l'archive.

Si l'archive sélectionnée se compose d'un contenu complet et d'un delta, vous pouvez choisir d'appliquer le delta afin de fusionner les différences incluses. Un écran de comparaison sera affiché, depuis lequel les différences à fusionner peuvent être sélectionnées.

Si l'archive sélectionnée contient uniquement un delta, il est possible de sélectionner les différences à fusionner dans un écran de comparaison.

15.7 Fermeture d'un espace de données

Si un espace de données n'est plus utilisé, il peut être fermé. Dès qu'il est fermé, l'espace de données n'apparaîtra plus dans la section **Espaces de Données** de l'interface utilisateur, et ne sera plus accessible.

Un administrateur peut rouvrir un espace de données fermé, tant que celui-ci n'a pas encore été purgé du référentiel.

Pour fermer un espace de données, sélectionnez **Actions > Fermer l'espace de données**.

Voir aussi [*Closing unused data spaces and snapshots*](#) [p 386]

CHAPITRE 16

Images

Ce chapitre contient les sections suivantes :

1. [Présentation des images](#)
2. [Création d'une image](#)
3. [Visualisation de contenu d'une image](#)
4. [Informations de l'image](#)
5. [Comparaison d'une image](#)
6. [Validation d'une image](#)
7. [Export d'une image](#)
8. [Fermeture d'une image](#)

16.1 Présentation des images

Une image est une copie en lecture seule d'un espace de données. Une image sert de référence de l'état et du contenu d'un espace de données à un instant donné.

Voir aussi [image](#) [p 24]

16.2 Création d'une image

Une image est créée en sélectionnant un espace de données dans le panneau de navigation, puis en cliquant sur le bouton 'Créer une image' sous le menu 'Actions'.

Les informations suivantes sont requises :

Identifiant	Identifiant unique de l'image. Le modèle suivant doit être respecté: [a-zA-Z0-9_:\.-\]*.
Libellé	Libellé et description localisés associés à l'image.

16.3 Visualisation de contenu d'une image

Pour visualiser le contenu d'une image, ouvrir l'image, puis sélectionner *Actions > Voir ou éditer les jeux de données* dans l'espace de travail.

16.4 Informations de l'image

Certaines propriétés associées à une image peuvent être modifiées en sélectionnant l'image, puis *Actions > Informations* dans l'espace de travail de la section *Espaces de données*.

Documentation	Libellé et description localisés associés à l'image.
Mode relationnel	Indique que les tables des jeux de données présents dans cet espace de données sont contenues dans une base de données relationnelle. Il ne sera pas possible de créer des images et des espaces de données enfants.
Stratégie de chargement	<p>Toute modification de ce champ requiert de redémarrer le serveur. Pour un espace de données en mode sémantique, la stratégie par défaut, 'Charger et décharger à la demande', permet de libérer la mémoire si nécessaire. En contrepartie, cela implique un coût de chargement quand la ressource accédée l'est pour la première fois depuis le redémarrage du serveur ou quand elle a été déchargée depuis. La stratégie 'Chargement forcé' est recommandée pour les espaces de données et images à vie longue et qui sont intensivement utilisées. Pour un espace de données en mode relationnel, les stratégies sont 'Aucune' (défaut) ou 'Prévalidation'.</p> <p>Seul l'administrateur a le droit de modifier cette option. Voir Stratégie de chargement [p 84].</p>
Politique de fusion des enfants	Un espace de données peut définir une politique qui définit la manière de fusionner un espace de données enfant. La politique par défaut est 'Autoriser des erreurs de validation dans le résultat'. Une autre politique, la 'Fusion pré-validante', permet de s'assurer que le résultat de la fusion est valide avant d'effectuer définitivement la fusion. La politique de fusion des enfants est uniquement appliquée en utilisant les interfaces utilisateur. Ce paramétrage est ignoré pour une fusion programmatique (incluant les tâches automatiques des workflows de données).
Propriétaire	L'utilisateur possédant l'image, et qui est autorisé à en modifier les informations et les permissions. Le propriétaire d'une image n'est pas obligatoirement son créateur.
Tri des espaces enfants	Définit l'ordre d'affichage des espaces de données enfants dans l'arbre des espaces de données. Si non défini, l'ordre défini par le parent est pris en compte. La valeur par défaut est 'par libellé'.

Changer le propriétaire	Définit si le propriétaire de l'image a le droit de modifier l'attribut "Propriétaire". Si la valeur est "Non habilité", seul l'administrateur a le droit d'effectuer cette modification.
Changer les permissions	Spécifie si un utilisateur qui est propriétaire de l'espace de données a le droit de modifier les permissions de cet espace de données. Si ce n'est pas le cas, seul un administrateur ou un 'super propriétaire' de l'espace de données a le droit de modifier les permissions.

16.5 Comparaison d'une image

Il est possible de comparer le contenu d'une image à celui d'une autre image ou d'un espace de données dans le référentiel. Pour effectuer une comparaison, ouvrir l'image, puis sélectionner *Actions* > *Comparer* dans l'espace de travail. L'assistant permet de choisir une autre image ou un espace de données à comparer avec l'image courante.

Pour une comparaison plus rapide, qui ignore les champs avec une valeur héritée ou calculée, sélectionner le filtre 'Valeurs persistantes seulement'.

Voir aussi [Compare contents](#) [p 202]

16.6 Validation d'une image

Le contenu d'une image peut être validé globalement en utilisant le service de validation au niveau de l'image. Ce service est accessible en sélectionnant *Actions* > *Valider* dans l'espace de travail.

Note

Pour utiliser ce service, l'utilisateur doit avoir la permission de valider tous les jeux de données contenus dans l'image.

16.7 Export d'une image

Le contenu d'une image peut être exporté dans une archive en sélectionnant *Actions* > *Exporter* dans l'espace de travail. L'archive est sauvegardée sur le système de fichiers du serveur, où seul un administrateur peut la récupérer.

Note

Voir [Archives directory](#) [p 383] dans le Guide d'administration.

Pour réaliser un export, les informations suivantes sont requises :

Nom du fichier archive à créer	Le nom de l'archive exportée.
Type d'export	Le contenu complet de l'espace de données, Mises à jour avec leur contenu complet (*), Mises à jour seules (*). (*) : <i>les mises à jour à exporter sont sélectionnées dans les pages suivantes.</i>
Jeu de données (ou arbre de jeux de données) à exporter	Les jeux de données de cette image à exporter. Pour chaque jeu de données, il est possible de choisir si les données, les permissions et les informations doivent être exportées.

16.8 Fermeture d'une image

Si une image n'est plus utilisée, elle peut être fermée. Dès qu'elle est fermée, l'image n'apparaît plus dans la section 'Espaces de données' de l'interface utilisateur et n'est plus accessible.

Un administrateur peut rouvrir une image fermée, tant qu'elle n'a pas été purgée du référentiel.

Pour fermer une image, sélectionner *Actions > Fermer l'image*.

Voir aussi [Closing unused data spaces and snapshots](#) [p 386]

Jeux de données

CHAPITRE 17

Introduction aux jeux de données

Ce chapitre contient les sections suivantes :

1. [Présentation](#)
2. [Utilisation de l'interface utilisateur de la section Données](#)

17.1 Présentation

Fonction d'un jeu de données

Un jeu de données est un conteneur de données qui se base sur les définitions de structure fournies par le modèle de données qu'il implémente. Lors de la publication d'un modèle de données, il est possible de créer des jeux de données basés sur sa définition. Par la suite, si ce modèle est modifié et republié, tous ses jeux de données associés sont mis à jour automatiquement.

Dans un jeu de données, les valeurs de données sont consultables et modifiables. A l'aide des vues, il est possible d'afficher les tables d'une manière adaptée à la nature des données et du mode d'accès. Les recherches et les filtres peuvent aussi être utilisés pour restreindre l'affichage ou rechercher des données.

Des permissions peuvent aussi être affectées à différents rôles pour contrôler l'accès au niveau du jeu de données. Ainsi, en appliquant des permissions spécifiques, on peut permettre à certains utilisateurs d'afficher ou de modifier des données tout en les cachant à d'autres.

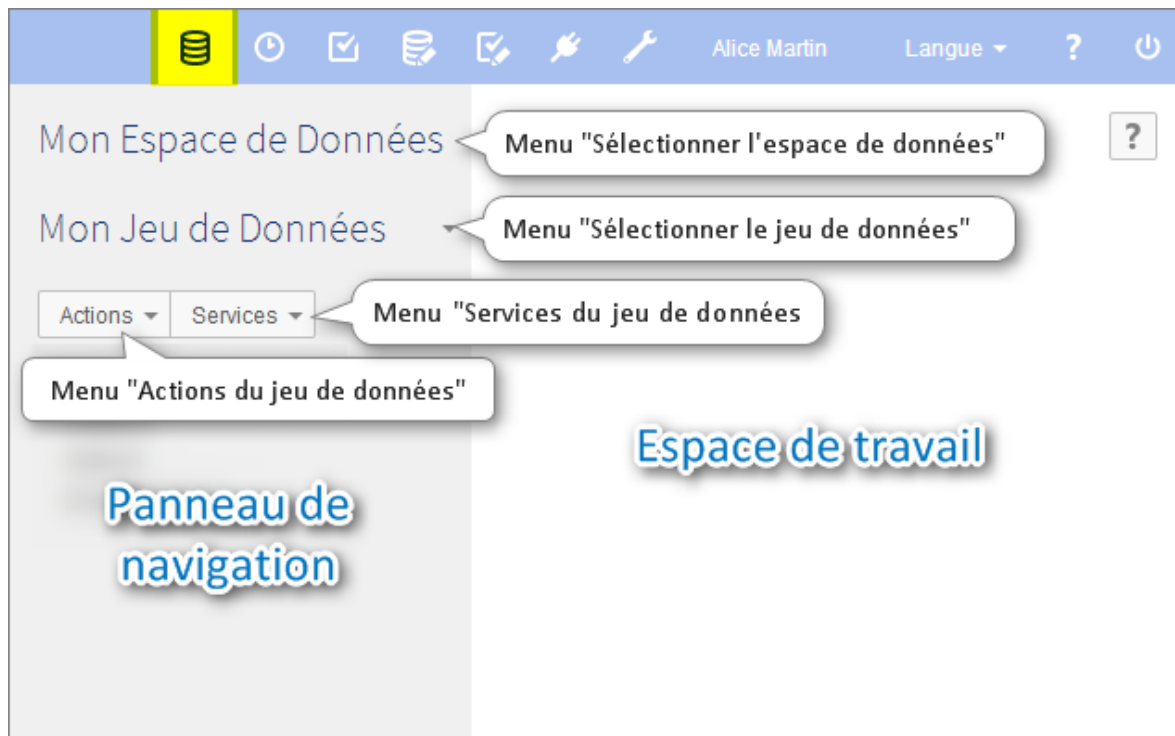
Concepts de base liés aux jeux de données

La compréhension des termes suivants est recommandée pour utiliser les jeux de données :


- [espace de données](#) [p 23]
- [jeu de données](#) [p 22]
- [enregistrement](#) [p 22]
- [champ](#) [p 21]
- [clé primaire](#) [p 21]
- [clé étrangère](#) [p 21]
- [table \(jeu de données\)](#) [p 22]
- [groupe](#) [p 21]

17.2 Utilisation de l'interface utilisateur de la section Données

Dans le cadre de l'utilisation de la [Perspective avancée](#) [p 15], ou d'une perspective spécifique, les jeux de données sont créés, consultés et modifiés dans la section 'Données'. Seuls les utilisateurs autorisés peuvent accéder à ces interfaces spécifiques.



Pour sélectionner ou créer un jeu de données, cliquer sur 'Sélectionner le jeu de données' dans le panneau de navigation. La structure de données du jeu de données s'affichera dans le panneau de navigation et les formulaires d'enregistrement ainsi que les vues de table s'afficheront dans l'espace de travail.

Lors de la visualisation d'une table du jeu de données dans l'espace de travail, le bouton  permet d'afficher les recherches et filtres disponibles pour restreindre l'affichage des enregistrements.

Les actions applicables au jeu de données sont disponibles dans les menus 'Actions' et 'Services' du panneau de navigation.

Voir aussi

[Création du jeu de données](#) [p 103]

[Recherche et filtrage des données](#) [p 106]

[Actions sur les enregistrements dans l'interface utilisateur](#) [p 113]

[Héritage](#) [p 23]

Concepts apparentés

[Modèle de données](#) [p 30]

[Espace de données](#) [p 78]

CHAPITRE 18

Création du jeu de données

Ce chapitre contient les sections suivantes :

1. [Création d'un jeu de données racine](#)
2. [Création d'un jeu de données enfant utilisant l'héritage](#)

18.1 Création d'un jeu de données racine

Afin de créer un jeu de données racine, qui n'hérite pas d'un jeu de données parent, il faut sélectionner le menu '[Sélectionner le jeu de données](#) [p 101]' dans le panneau de navigation, cliquer sur le bouton 'Créer un jeu de données' dans la fenêtre contextuelle, et suivre l'assistant.

Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée' ou via une perspective spécifique.

L'assistant vous permet de sélectionner un des trois types de modèles de données, sur lequel le nouveau jeu de données sera basé : packagé, embarqué ou externe.

- Un *modèle de données packagé* est un modèle de données défini à l'intérieur d'un module (application web).
- Un *modèle de données embarqué* est un modèle de données créé et publié en utilisant l'assistant de modèles de données. Il est entièrement géré à l'intérieur du référentiel EBX5. Un modèle de données doit avoir été préalablement publié pour que cette fonctionnalité soit disponible.
- Un *modèle de données externe* est un modèle de données référencé au moyen d'une URI.


Après avoir choisi le modèle de données sur lequel le nouveau jeu de données sera basé, vous devez fournir un nom unique, sans espaces ni caractères spéciaux. Facultativement, vous pouvez donner des libellés localisés pour le jeu de données, qui seront affichés aux utilisateurs dans l'interface en fonction de leurs préférences de langue.

Attention

Le contenu des tables n'est pas copié lors de la duplication d'un jeu de données.

18.2 Création d'un jeu de données enfant utilisant l'héritage

Le mécanisme d'héritage permet d'utiliser les relations parent-enfant, grâce auxquelles les jeux de données enfants héritent par défaut des valeurs de leurs jeux de données ancêtres. Pour pouvoir créer des jeux de données enfants depuis un jeu de données parent, il faut activer l'héritage entre jeux de données dans le modèle de données associé.

Pour créer un jeu de données enfant, cliquer sur [Créer ou sélectionner un jeu de données](#) [p 101] dans le panneau de navigation, puis sur le bouton  correspondant au jeu de données parent.

Le jeu de données enfant sera basé sur le même modèle de données que celui de son parent, et donc les seules informations à spécifier sont son nom unique, et éventuellement les libellés localisés.

Voir aussi [Héritage entre jeux de données](#) [p 121]

CHAPITRE 19

Visualisation des données

Ce chapitre contient les sections suivantes :

1. [Menu 'Vue'](#)
2. [Tri des données](#)
3. [Recherche et filtrage des données](#)
4. [Vues](#)
5. [Gestion des vues](#)
6. [Historique](#)

19.1 Menu 'Vue'

Le menu déroulant 'Vue' permet d'accéder facilement à toutes les fonctionnalités majeures de visualisation.

Les vues sont gérées dans le sous-menu dédié '[Gestion de vues](#)' [p 111].

Les vues peuvent également être regroupées. L'administrateur doit avoir défini des groupes au préalable dans la table 'Groupes de vues' de la section 'Configuration des vues'. L'utilisateur peut ainsi définir une vue comme appartenant à un groupe dans le champ 'Groupe de la vue' lors de la création ou de la modification d'une vue. Voir [Description d'une vue](#) [p 108] pour plus d'informations.

19.2 Tri des données

Les critères de tri contrôlent l'ordre dans lequel les enregistrements sont présentés.

Par défaut, les enregistrements sont ordonnés par clé primaire ascendante.


Afin de définir des critères de tri spécifiques, sélectionnez *Vue > Critères de tri* dans l'espace de travail.

Chaque critère de tri est défini par un nom de colonne et un ordre de tri (ascendant ou descendant). Utiliser les flèches 'Déplacer à gauche/droite' pour ajouter ou enlever un critère dans la table 'Trié'. Une fois un critère sélectionné, il est possible de choisir son ordre de tri en cliquant sur le bouton 'ASC' ou 'DESC' à droite.

Pour changer la priorité d'un critère de tri, sélectionnez-le dans la liste, puis cliquez sur les flèches vers le haut ou vers le bas pour le déplacer.

Pour rétablir la vue sans critère de tri spécifique, sélectionner *Vue > Rétablir*.

19.3 Recherche et filtrage des données

Des outils spécifiques permettent la recherche d'enregistrements dans une table. On peut y accéder par l'icône , qui affiche les panneaux des filtres.

Chaque panneau de filtre (ou de recherche) propose une case à cocher sur sa barre de titre : cocher cette case applique le filtre ; la décocher le désapplique.

Note

Appliquer une vue réinitialisera et retirera tous les filtres actuellement appliqués.

Recherche

En mode simple, la 'Recherche' permet d'ajouter des critères contextualisés sur un ou plusieurs champs. Lors de l'ajout d'un critère, les opérateurs proposés seront en adéquation avec le type du champ correspondant.

En activant le mode avancé, il est possible de créer des sous-blocs contenant des critères, afin de créer des opérations logiques plus complexes dans l'élaboration du filtre.

Note

En mode avancé, les critères dont l'opérateur est "correspond à" ou "correspond à (respecter la casse)" suivent la syntaxe standard des expressions régulières de Java.

Voir aussi [Regex pattern](#)

Recherche textuelle

La recherche textuelle est utilisée pour une recherche de texte brut sur un ou plusieurs champs. Cette recherche ne prend pas en compte le type du champ.

- Si le texte entré contient un ou plusieurs mots sans caractère de remplacement (* ou ?), les champs trouvés seront ceux contenant tous ces mots. Les mots entre guillemets ("aa bb" par exemple) sont considérés comme un seul mot.
- Les caractères usuels de remplacement sont proposés : l'étoile * (tout texte) ou le point d'interrogation ? (tout caractère). Pour des raisons de performance, leur utilisation est restreinte à un seul par recherche.
- Les caractères de remplacement peuvent être considérés comme de simple caractères en les échappant à l'aide du caractère '\', par exemple, '*'.

Exemples :

- aa bb : le champ contient "aa" et "bb".
- aa "bb cc" : le champ contient "aa" et "bb cc".
- aa* : le champ commence par "aa".
- *bb : le champ se termine par "bb".
- aa*bb : le champ commence par "aa" et se termine par "bb".
- aa? : le champ commence par "aa" et a une taille de 3 caractères.
- ?bb : le champ se termine par "bb" et a une taille de 3 caractères.
- aa?bb : le champ commence par "aa" et se termine par "bb" et a une taille de 5 caractères.
- aa*bb : le champ contient "aa*bb" tel quel.

Sur les tables de grande taille, il est recommandé de ne sélectionner qu'un seul champ de la table ; si le champ n'est pas du type "chaîne de caractères", il est conseillé d'entrer un texte au bon format, par exemple :

- booléen : Oui, Non
- date : 01/01/2000
- nombre : 100000 ou 100 000
- champ énuméré : Rouge, Bleu...

L'option *Sensible à la casse* oblige la recherche à tenir compte de la casse, en distinguant les majuscules des minuscules.

Recherche sur validation

La recherche par validation permet de voir les enregistrements en fonction de leur statut dans la dernière validation effectuée. Il est possible de voir les enregistrements avec les niveaux 'Erreur', 'Avertissements' et 'Informations'.

Note

Cette recherche s'applique seulement aux enregistrements de tables qui ont déjà été validés ; pour ce faire, sélectionner *Actions > Valider* au niveau de la table dans l'espace de travail, ou au niveau du jeu de données dans le panneau de navigation.

Recherches spécifiques sur tables

Pour chaque table, le modèle peut spécifier des filtres additionnels pour la recherche.

Voir aussi [Specifying UI filters on a table](#) [p 181]

19.4 Vues

Une vue est créée en sélectionnant *Vue > Créer une nouvelle vue* dans l'espace de travail. Pour appliquer une vue, la sélectionner dans *Vue > nom de la vue*.

Deux modes avancés de visualisation sont disponibles à la création d'une nouvelle vue :

- 'Vue tabulaire simple' : une vue tabulaire qui permet de trier et filtrer les enregistrements affichés ;
- 'Vue hiérarchique' : une arborescence qui lie les données de différentes tables en utilisant leurs relations.

Description d'une vue

Lors de la création ou de la modification d'une vue, la première page permet de définir les informations générales concernant la vue.

Documentation	Libellé et description localisés associés à la vue.
Propriétaire	Nom du propriétaire de la vue, qui peut à ce titre la gérer et la modifier.
Autres profils autorisés	Détail des profils ayant le droit d'utiliser la vue.
Mode de vue	Vue tabulaire simple ou vue hiérarchique.
Groupe de la vue	Groupe d'appartenance de cette vue (le cas échéant).

Vue tabulaire simple

Les vues tabulaires simples offrent la possibilité de définir des critères pour filtrer les enregistrements et de sélectionner les colonnes à afficher.

Colonnes affichées	Spécifie, à l'aide de flèches, les colonnes de la table à afficher dans la vue.
Colonnes triées	Spécifie l'ordre d'affichage des colonnes et indique si les enregistrements de chaque colonne sont triés par ordre croissant ou décroissant. Voir Tri des données [p 105].
Filtre	Définit les critères utilisés pour filtrer les enregistrements. Voir Editeur de critères [p 329].

Vues hiérarchiques

Une hiérarchie est une arborescence permettant de présenter les relations existant entre les tables. Elle peut être structurée sur plusieurs niveaux, appelés niveaux de dimension. En outre, il est possible de définir des filtres sur des niveaux afin de filtrer les enregistrements.

Dimension d'une hiérarchie

Une dimension définit une dépendance dans la hiérarchie. Par exemple, une dimension pourrait être précisée pour afficher des produits par catégorie. Plusieurs dimensions peuvent être définies pour une vue hiérarchique.

Options de configuration d'une vue hiérarchique

Ce formulaire permet de configurer les options d'une vue hiérarchique.

Afficher les enregistrements dans une nouvelle fenêtre	Si "oui", une nouvelle fenêtre sera ouverte pour afficher l'enregistrement. Sinon, il sera affiché dans une nouvelle page de la fenêtre courante.
Hiérarchie élaguée	Si "oui", les noeuds de la hiérarchie qui n'ont pas d'enfants et qui n'appartiennent pas à la table cible ne seront pas affichés.
Afficher les orphelins	Si "oui", les noeuds de la hiérarchie qui n'ont pas de parent seront affichés.
Afficher le noeud racine	Si 'Non', le noeud racine de la hiérarchie sera caché de la vue.
Libellé du noeud racine	Libellé localisé du noeud racine de la hiérarchie.

Libellés

Pour chaque niveau de dimension faisant référence à une autre table, il est possible de définir les libellés localisés pour les noeuds correspondants dans la hiérarchie. Utiliser l'assistant pour sélectionner les champs utilisés dans les définitions des libellés.

Filtre

L'éditeur de critères permet de définir un filtre d'enregistrement pour la vue.

Voir aussi [Editeur de critères](#) [p 329]

Champ d'ordonnancement

Afin de pouvoir déplacer les noeuds dans la vue hiérarchique, il faut désigner un champ d'ordonnancement éligible. Celui-ci est défini dans la table sur laquelle est appliquée la vue hiérarchique. Un champ d'ordonnancement doit avoir le type de données 'Entier' et doit avoir la vue par défaut 'Caché' dans les propriétés avancées du modèle de données.


Des actions de positionnement sur chaque noeud sont alors possibles, à moins que le champ d'ordonnancement ne soit en lecture seule ou qu'un filtre ne soit défini sur la hiérarchie.

En l'absence d'un noeud d'ordonnancement, les noeuds enfants sont triés selon l'ordre alphabétique des libellés des noeuds.

Attention

Le champ d'ordonnancement doit être un champ technique dédié, et non un champ contenant des données métier. En effet, ces données seront modifiées automatiquement par les actions de réordonnancement.

Actions sur un noeud de hiérarchie

Chaque noeud d'une vue hiérarchique possède un menu correspondant  qui offre des actions contextuelles.

Les noeuds terminaux peuvent être détachés de leur parent en sélectionnant l'option 'Détacher du parent'. L'enregistrement devient ainsi un noeud orphelin dans l'arborescence, rangé sous un conteneur portant le nom 'non défini'.

Les noeuds terminaux peuvent aussi changer de noeud parent, grâce à l'option 'Attacher à un autre parent'. Si, selon le modèle de données, un noeud peut avoir plusieurs parents, le noeud sera à la fois sous le parent d'origine et rajouté également sous le nouveau parent. Sinon, le noeud terminal sera déplacé sous le nouveau noeud parent.

Publication de vue

Une vue peut être publiée afin de la rendre accessible à tous les autres utilisateurs via les composants web, les tâches utilisateurs du workflow, et les services de données. Pour publier une vue, sélectionner 'Publier' dans le menu *Vue > Gérer les vues > nom de la vue*. Dès que la vue sera publiée, les autres utilisateurs pourront appliquer cette vue à leur interface en la sélectionnant dans *Vue > nom de la vue* dans leur espace de travail.

Cette publication de vue sera accessible aux utilisateurs qui appartiennent aux profils autorisés dans la définition de la vue.

19.5 Gestion des vues

Gérer les vues recommandées

Quand un utilisateur se connecte sans spécifier de vue, la vue recommandée, si elle existe, s'applique. Sinon, la vue par défaut s'affiche. L'action 'Gérer les vues recommandées' permet de définir les règles d'attribution des vues recommandées par utilisateur et par rôle.

Les actions disponibles sur les vues recommandées sont les suivantes : changer l'ordre d'attribution des règles, ajouter une règle, éditer une règle, supprimer une règle existante.

Pour un utilisateur donné, la règle appliquée sera la première de la liste qui correspond au profil de l'utilisateur.

Note

La fonctionnalité 'Gérer les vues recommandées' est uniquement accessible au propriétaire du jeu de données.

Gérer les vues

Le sous-menu 'Gérer les vues' permet d'accéder aux actions suivantes :

Définir cette vue comme ma favorite	Uniquement disponible lorsque la vue en cours d'affichage n'est PAS la vue recommandée. La vue favorite sera automatiquement appliquée lors de l'accès à la table.
Définir la vue recommandée comme ma favorite	Uniquement disponible lorsqu'une vue favorite est définie. En cliquant sur cette action, la vue courante ne sera plus la vue favorite de l'utilisateur. Une vue recommandée, à l'instar de la vue favorite, est appliquée automatiquement lors de l'accès à la table. Cette action n'est pas affichée si aucune vue favorite n'a été définie.

19.6 Historique

La fonctionnalité d'historique permet le suivi des modifications de données.

La visualisation de l'historique de données nécessite que la fonctionnalité ait été préalablement activée au niveau des tables dans le modèle de données. Voir [Propriétés avancées des tables](#) [p 55] pour plus d'informations.

Pour visualiser l'historique d'un jeu de données, sélectionner *Actions > Historique* dans le panneau de navigation.

Pour visualiser l'historique d'une table ou d'une sélection d'enregistrements, sélectionner *Actions > Voir l'historique* dans l'espace de travail.

Il existe différents modes d'historique qui permettent de visualiser l'historique des données selon différentes perspectives:

Historique dans l'espace de données en cours	La vue d'historique de table affiche les opérations sur la branche en cours. C'est le mode par défaut.
Historique dans l'espace de données courant et ses ancêtres	La vue d'historique de table affiche les opérations sur la branche en cours ainsi que tous ses ancêtres.
Historique dans l'espace de données courant et ses enfants fusionnés	La vue d'historique de table affiche les opérations sur la branche en cours ainsi que toutes ses branches filles fusionnées.
Historique dans tous les espaces de données	La vue d'historique de table affiche les opérations sur toute la hiérarchie de la branche en cours.

Dans la vue d'historique, utiliser le menu *View* pour passer à un autre mode de l'historique.

Voir aussi [Historique](#) [p 289]

CHAPITRE 20

Edition des données

Ce chapitre contient les sections suivantes :

1. [Actions sur les enregistrements dans l'interface utilisateur](#)
2. [Import et export de données](#)
3. [Restauration depuis l'historique](#)


20.1 Actions sur les enregistrements dans l'interface utilisateur

L'édition des enregistrements s'effectue dans l'espace de travail de l'interface utilisateur.

Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée' ou via une perspective spécifique.

Création d'un enregistrement

Dans la vue tabulaire, un nouvel enregistrement peut être créé à l'aide du bouton  situé en haut à gauche de la table.

Dans une vue hiérarchique, sélectionnez "Créer un enregistrement" dans le menu du noeud parent du nouvel enregistrement.

Dans les deux cas, un formulaire s'affiche, permettant d'entrer des données. Les données obligatoires sont repérées par une astérisque rouge.

Modification d'un enregistrement

Un enregistrement peut être édité par double clic. Le formulaire qui s'affiche permet d'éditer l'enregistrement, tandis que le bouton *Rétablir* permet de recharger le formulaire sans soumettre aucun des changements effectués.

Dupliquer un enregistrement

Pour dupliquer un enregistrement, sélectionnez-le, puis sélectionnez *Actions > Dupliquer*.

Un formulaire apparaît, pré-rempli à partir des valeurs de l'enregistrement copié. La clé primaire doit ensuite être modifiée pour pouvoir créer ce nouvel enregistrement, à moins qu'elle ne soit générée automatiquement (à l'exemple d'une valeur auto-incrémentée).

Supprimer

Pour supprimer un ou plusieurs enregistrements sélectionnés, sélectionnez *Actions > Supprimer*.

Comparer

Deux enregistrements sélectionnés peuvent être comparés en sélectionnant *Actions > Comparer*.

Note

La comparaison ne prend pas en compte le contenu des noeuds terminaux complexes, comme les listes agrégées ou les attributs utilisateurs. Toutes les différences portant sur de tels noeuds seront ignorées.

20.2 Import et export de données

Dans une table, les enregistrements peuvent être importés ou exportés, depuis ou vers les formats CSV ou XML.

Vous pouvez soit exporter l'ensemble de la table, soit sélectionner manuellement certains enregistrements à exporter, grâce aux cases à cocher.

Voir aussi

[Services CSV](#) [p 261]

[Services XML](#) [p 255]

20.3 Restauration depuis l'historique

Dans une table où l'historique est activé, il est possible de restaurer un enregistrement à un état antérieur en se basant sur son historique. Dans le cas où l'enregistrement (identifié par sa clé primaire) existe encore dans la table, il sera mis à jour par les valeurs historisées à restaurer. Dans le cas contraire, il sera créé.

Pour restituer un enregistrement à un état antérieur, sélectionner un enregistrement dans la vue d'historique, puis sélectionner *Actions > Restaurer depuis l'historique* dans l'espace de travail. Un écran de résumé s'affiche avec les détails de la mise à jour ou de la création à effectuer.

La fonctionnalité de restauration est uniquement applicable sur un enregistrement à la fois.

Si un trigger de table doit avoir un comportement spécifique pour la restauration, c'est à dire différent de celui des opérations classiques de création et de mise à jour, le développeur peut utiliser la méthode `TableTriggerExecutionContext.isHistoryRestore`^{API}.

Note

Il existe des limitations liées à la fonctionnalité de l'historique :

- La restauration depuis l'historique n'est pas disponible pour les tables contenant des listes non supportées par l'historique. Voir [Limitations du modèle de données](#) [p 294].
- Les types de champ suivants ne sont pas historisés : les valeurs calculées, les champs cryptés, ainsi que les champs dont l'historique est désactivé. Ces champs seront donc ignorés lors de la restauration d'un enregistrement depuis l'historique.

Voir aussi [Historique](#) [p 289]

CHAPITRE 21

Actions sur les jeux de données existants

Ce chapitre contient les sections suivantes :

1. [Validation du jeu de données](#)
2. [Duplication d'un jeu de données](#)
3. [Désactivation d'un jeu de données](#)
4. [Gestion des permissions de jeux de données](#)


21.1 Validation du jeu de données

Il est possible de valider un jeu de données en sélectionnant *Actions > Valider* depuis le panneau de navigation. Les éventuels messages issus de la validation du jeu de données sont présentés dans un rapport. Depuis le rapport de validation, cliquer sur le bouton *Revalider* pour mettre à jour ce rapport. Pour supprimer tous les messages de validation actuellement associés au jeu de données, et pouvoir relancer une validation complète, cliquer sur le bouton *Réinitialiser le rapport de validation*.

Dans la section 'Données', il est également possible de valider une table, en la sélectionnant dans le panneau de navigation et en utilisant l'action *Actions > Valider* dans l'espace de travail.

Voir [Validation](#) [p 336] pour obtenir plus d'informations concernant la validation incrémentale de données.

21.2 Duplication d'un jeu de données

Pour dupliquer un jeu de données existant, sélectionnez-le dans le menu "[Sélectionner le jeu de données](#) [p 101]"  dans le panneau de navigation, puis sélectionnez *Actions > Dupliquer*.

21.3 Désactivation d'un jeu de données

Si un jeu de données est activé, il sera sujet à la validation. Tous les éléments obligatoires doivent être définis pour que le jeu de données soit valide. Si un jeu de données est activé et valide, on considère qu'il peut être exporté de façon sécurisée vers des systèmes externes (ou utilisé par d'autres applications Java).

Il est possible de désactiver un jeu de données dont les éléments obligatoires ne sont pas définis, en spécifiant "Non" pour la propriété "Activé" dans *Actions > Informations*.

21.4 Gestion des permissions de jeux de données

Les permissions de jeux de données sont accessibles en sélectionnant *Actions > Permissions* dans le panneau de navigation.

Les permissions sont définies en créant des *profils*. Pour créer un nouveau profil de permissions, créez un nouvel enregistrement dans la table "Droits d'accès par profil".

Voir aussi [Profil](#) [p 19]

Profil	Indique le profil concerné par la permission définie.
Restriction d'accès	Indique si la permission définie restreint celles affectées à un utilisateur donné par des politiques définies pour d'autres profils. Voir Restriction d'accès [p 323].
Actions sur les jeux de données	Cette section spécifie les permissions des actions sur les jeux de données.
Créer un le jeu de données enfant	Indique si le profil peut créer un jeu de données enfants. L'héritage doit aussi être activé dans le modèle de données.
Dupliquer le jeu de données	Indique si le profil peut dupliquer le jeu de données.
Supprimer le jeu de données	Indique si le profil peut supprimer le jeu de données.
Activer/désactiver le jeu de données	Indique si le profil peut modifier la propriété "Activé" dans les informations du jeu de données. Voir Désactivation d'un jeu de données [p 117].
Créer une vue	Indique si le profil peut créer des vues et des hiérarchies.
Droits sur tables	Spécifie les permissions par défaut pour toutes les tables. Des permissions spécifiques peuvent être appliquées à une ou plusieurs tables en cliquant sur le bouton '+' sous Droits spécifiques par table.
Droits par défaut >Créer un nouvel enregistrement	Indique si le profil peut créer des enregistrements dans une table.
Droits par défaut >Surcharger des enregistrements	Indique si le profil peut remplacer des enregistrements hérités dans une table. Cette permission est utile quand on utilise l'héritage de jeu de données.
Droits par défaut >Occulter des enregistrements	Indique si le profil peut occulter des enregistrements hérités dans une table. Cette permission est utile quand on utilise l'héritage de jeu de données.
Droits par défaut >Supprimer un enregistrement	Indique si le profil peut supprimer des enregistrements dans une table.

Droits sur valeurs

Spécifie les permissions d'accès par défaut pour tous les éléments (tables, groupes et champs) d'un jeu de données, et permet de définir des permissions pour des éléments spécifiques. Les permissions d'accès par défaut sont utilisées, à condition qu'il n'y ait pas de permission spécifique affectée à un élément.

Le sélecteur de droits spécifiques permet d'attribuer des permissions d'accès spécifiques à un élément. Les liens "Lecture", "Ecriture" et "Non visible" déterminent les permissions d'accès correspondant à l'élément sélectionné.

Il est possible de retirer une permission d'accès spécifique en utilisant le lien "(par défaut)".

Droits sur les services

Spécifie les permissions d'accès sur les services. Un service barré n'est pas accessible à un profil.

CHAPITRE 22

Héritage entre jeux de données

En utilisant le concept d'héritage entre jeux de données, vous pouvez créer des jeux de données additionnels, à partir d'un jeu de données racine. Ces jeux de données enfants héritent des propriétés et des valeurs de leur parents, qui peuvent être surchargées si nécessaire. Plusieurs niveaux d'héritage peuvent être créés.

L'héritage peut être utilisé pour adapter des données de référence à divers contextes. Par exemple, il serait possible de définir des valeurs globales dans un jeu de données parent, et de créer des jeux de données enfants par zones géographiques. Ceci permettra à ces derniers d'hériter des valeurs de leur parent, et de les surcharger si besoin.

Note


Le comportement standard est d'interdire l'héritage de jeux de données. Il est donc nécessaire d'activer explicitement cette fonction au niveau du modèle de données.

Voir aussi [Configuration du modèle de données](#) [p 38]

Ce chapitre contient les sections suivantes :

1. [Structure de l'héritage entre jeux de données](#)
2. [Héritage de valeurs](#)

22.1 Structure de l'héritage entre jeux de données

Une fois le jeu de données racine créé, un jeu de données enfant peut être créé à l'aide du bouton , situé dans l'écran de sélection des jeux de données du panneau de navigation.

Note

- Un jeu de données ne peut pas être supprimé s'il a des jeux de données enfants. Ces enfants doivent être supprimés préalablement.
- Si un jeu de données enfant est dupliqué, le jeu de données nouvellement créé sera inséré dans l'arbre des jeux de données existants, au même niveau de l'arbre que le jeu de données dupliqué.

22.2 Héritage de valeurs

Quand un jeu de données enfant est créé, il hérite de toutes les valeurs des champs et des enregistrements de tables de son parent. Un champ ou un enregistrement peut soit hériter ses valeurs, soit les surcharger.


Dans une vue tabulaire, les valeurs héritées sont signalées par un repère dans le coin en haut à gauche de la cellule.

Le bouton  permet de surcharger une valeur.

Héritage d'enregistrement

Une table dans un jeu de données enfant hérite des enregistrements des tables de ses jeux de données ancêtres. La table dans le jeu de données enfant peut rajouter, éditer ou supprimer des enregistrements. Des états sont définis pour différencier les types d'enregistrement.

Racine	Un enregistrement racine est un enregistrement créé dans le jeu de données courant, qui n'existe pas dans les jeux de données ancêtres. Il sera hérité par les jeux de données enfants.
Hérité	Un enregistrement hérité est défini dans un des jeux de données ancêtres du jeu de données courant.
Surchargé	Un enregistrement surchargé est un enregistrement hérité dont les valeurs sont éditées dans le jeu de données courant. Les valeurs surchargées seront héritées par les jeux de données enfants.
Occulté	Un enregistrement occulté est un enregistrement hérité qui est supprimé du jeu de données courant. Il apparaîtra toujours dans le jeu de données courant comme un enregistrement barré, mais il ne sera pas hérité par les jeux de données enfants.

Quand le bouton  est activé, la valeur de l'enregistrement est héritée du jeu de données parent. Ce bouton peut être désactivé, afin de surcharger l'enregistrement ou la valeur. Pour un enregistrement occulté, l'activation de ce bouton restaure l'état hérité.

La table suivante résume le comportement des enregistrements lorsque l'on crée, modifie, ou supprime un enregistrement, selon son état initial.

Etat	Création	Edition	Suppression
Racine	Création normale d'un enregistrement. L'enregistrement nouvellement créé sera hérité par ses jeux de données enfant.	Edition normale d'un enregistrement. Les nouvelles valeurs seront héritées par les jeux de données enfants.	Suppression normale d'un enregistrement. L'enregistrement va disparaître du jeu de données courant ainsi que des jeux de données enfants.
Hérité	Si un enregistrement est créé à l'aide de la clé primaire d'un enregistrement hérité existant, l'état de l'enregistrement devient surchargé, et sa valeur sera celle soumise à sa création.	Un enregistrement hérité doit être déclaré comme surchargé pour que ses valeurs soient modifiables.	Supprimer un enregistrement hérité change son état à "occulté".
Surchargé	Non applicable. Il est impossible de créer un nouvel enregistrement si la clé primaire est déjà utilisée.	Un enregistrement surchargé peut être remis à l'état "hérité", mais sa valeur spécifique sera perdue. Les valeurs de l'enregistrement surchargé peuvent être héritées ou modifiées.	Supprimer un enregistrement surchargé change son état à "occulté".
Occulté	Si un enregistrement est créé en utilisant la clé primaire de l'enregistrement existant occulté, l'état de l'enregistrement devient "surchargé" et sa valeur sera celle soumise à la création.	Non applicable. Un enregistrement occulté ne peut plus être édité.	Non applicable. Un enregistrement occulté est déjà considéré comme supprimé, et ne peut donc pas être supprimé une deuxième fois.

Voir aussi [Record lookup mechanism](#) [p 312]

Modèles de workflow

CHAPITRE 23

Introduction aux modèles de workflow

Ce chapitre contient les sections suivantes :

1. [Présentation](#)
2. [Utilisation de l'interface de modélisation de workflow](#)
3. [Modèles de message génériques](#)
4. [Limitations de workflows](#)

23.1 Présentation

Définition d'un modèle de workflow

Dans EBX5, les workflows facilitent la gestion collaborative des données dans le référentiel. Un workflow peut contenir des actions utilisateurs sur les données ainsi que des tâches automatiques, tout en émettant des notifications sur différents événements.

La première étape pour réaliser un workflow est de créer un *modèle de workflow* qui définit la succession d'étapes, les implications des utilisateurs, ainsi que le comportement du workflow.

Une fois qu'un modèle de workflow est défini, il peut être validé et publié comme *publication de workflow*. Ensuite, les workflows de données peuvent être lancés à partir de la publication de workflow pour exécuter les étapes définies dans le modèle de workflow.

Voir aussi

[Modèle de workflow \(glossaire\)](#) [p 25]

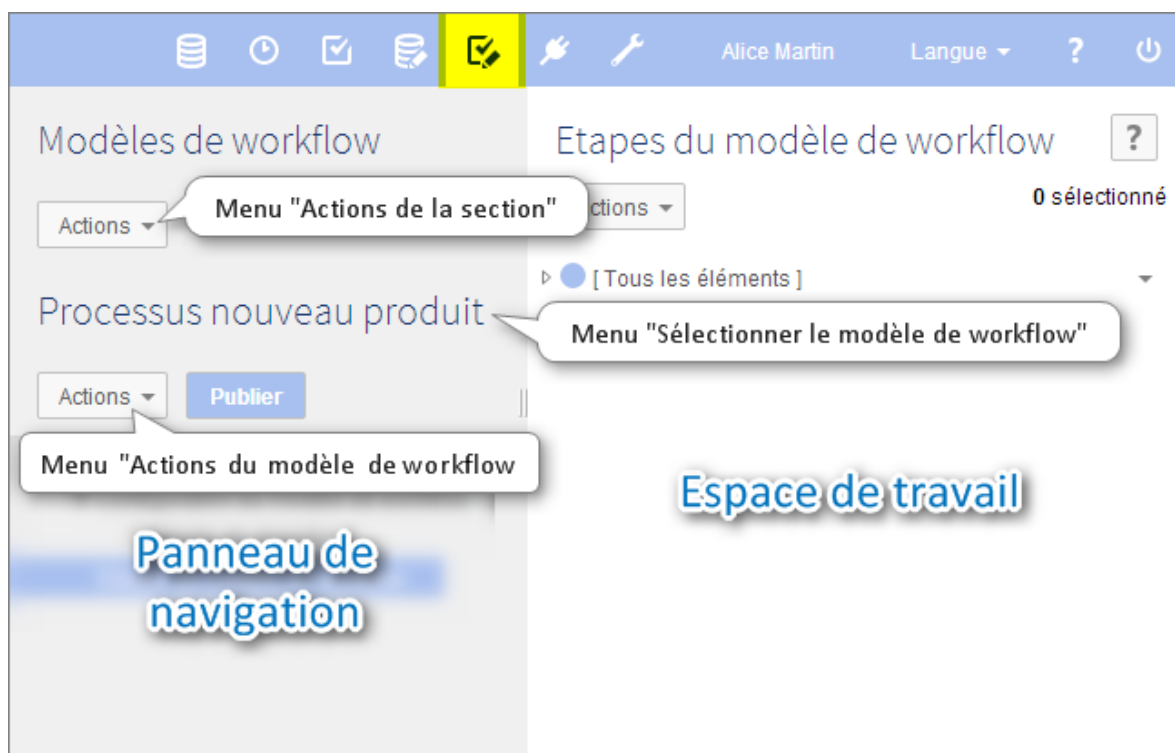
[Workflow de données \(glossaire\)](#) [p 26]

Concepts de base utilisés dans la modélisation des workflows

Une compréhension des termes suivants est nécessaire pour réaliser la création de modèles de workflows :

- [tâche automatique](#) [p 25]
- [tâche utilisateur](#) [p 25]
- [bon de travail](#) [p 27]
- [condition de workflow](#) [p 26]
- [appel à des sous-workflows](#) [p 26]
- [tâche d'attente](#) [p 26]
- [contexte des données](#) [p 26]

23.2 Utilisation de l'interface de modélisation de workflow



Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée'.
Seuls les utilisateurs autorisés peuvent accéder à ces interfaces spécifiques.

23.3 Modèles de message génériques

Des emails de notification peuvent être envoyés pour notifier les utilisateurs d'événements spécifiques pendant l'exécution d'un workflow.

Les modèles de message peuvent être définis et réutilisés dans n'importe quel modèle de workflow dans le référentiel. Pour modifier les modèles de message génériques, sélectionnez "Modèles de message" dans le menu 'Actions' de la section Modèles de Workflow.

Ces modèles, qui sont partagés par tous les modèles de workflows, sont figés et inclus dans chaque publication de workflow. Ainsi, pour prendre en compte les modifications des modèles de message, il sera nécessaire de mettre à jour les publications existantes en re-publiant les modèles de workflow concernés.

A la création d'un modèle de message générique, deux champs sont obligatoires :

- 'Libellé & Description' : spécifie les libellés et descriptions associés à ce modèle de message, localisé.
- 'Message' : spécifie l'objet de l'email et son corps de texte, localisés.

Le 'Type du message' est une donnée facultative.

Le message peut inclure des variables du contexte de données sous la forme *`${nom.variable}`*, qui seront évaluées lorsque le message sera envoyé. De plus, les variables système suivantes peuvent être incluses :

system.time	Heure système du référentiel.
system.date	Date système du référentiel.
workflow.lastComment	Dernier commentaire sur la tâche utilisateur précédente.
workflow.lastDecision	Dernières décisions sur la tâche utilisateur précédente.
user.fullName	Nom complet de l'utilisateur notifié.
user.login	Login de l'utilisateur notifié.
workflow.process.label	Libellé du workflow en cours.
workflow.process.description	Description du workflow en cours.
workflow.workItem.label	Libellé du bon de travail en cours.
workflow.workItem.description	Description du bon de travail en cours.
workflow.workItem.offeredTo	Rôle auquel le bon de travail courant a été proposé.
workflow.workItem.allocatedTo	Utilisateur, à qui le bon de travail en cours a été alloué.
workflow.workItem.link	<p>Lien d'accès au bon de travail courant dans la corbeille, au moyen de l'API du composant web.</p> <p>Pour que ce lien soit calculé, il est nécessaire qu'un bon de travail courant soit défini et que l'URL soit configurée dans Workflow-executions, dans la configuration de mail.</p>
workflow.workItem.link.allocateAndStart	Lien d'accès au bon de travail courant dans la corbeille, au moyen de l'API du composant web. Si le bon de travail cible n'est pas encore démarré, il sera automatiquement alloué à l'utilisateur qui a cliqué sur le lien, puis démarré.

Pour que ce lien soit calculé, il est nécessaire qu'un bon de travail courant soit défini et que l'URL soit configurée dans Workflow-executions, dans la configuration de mail.

workflow.currentStep.label	Libellé de l'étape courante.
workflow.currentStep.description	Description de l'étape courante.

Exemple

Modèles de message générique :

Aujourd'hui à \${system.time}, un nouveau bon de travail vous a été proposé.

Email résultant :

Aujourd'hui à 15:19, un nouveau bon de travail vous a été proposé.

23.4 Limitations de workflows

Les fonctionnalités suivantes ne sont pas supportées :

- **Tâches programmées**, tâches exécutées dès lors que leur tour vient, et dont l'exécution ne peut pas être reportée.
- **Tâches événementielles** permettant au workflow de progresser quand il reçoit un événement, du type appel web service.
- **Limitation temporelle** sur la durée d'une tâche.

Concepts apparentés [Workflows de données](#) [p 152]

CHAPITRE 24

Modélisation du workflow

Ce chapitre contient les sections suivantes :

1. [Création d'un modèle de workflow](#)
2. [Implémentation des étapes](#)
3. [Tâche utilisateur](#)
4. [Tâche autonome](#)
5. [Conditions](#)
6. [Appels à des sous-workflows](#)
7. [Tâches d'attente](#)

24.1 Création d'un modèle de workflow

Un modèle de workflow peut être créé à partir de la section 'Modèle de workflow'. La seule information requise à la création est un nom de modèle unique dans le référentiel.

Les étapes du modèle de workflow sont initialisées avec une transition initiale. Pour implémenter le modèle de workflow, il faut définir la séquence des étapes qui suivent cette transition initiale.

24.2 Implémentation des étapes

Un modèle de workflow définit des étapes correspondant à des opérations et des conditions. Les étapes possibles sont les suivantes :

- Tâche utilisateur
- Tâche automatique
- Condition
- Appel à des sous-workflows
- Tâche d'attente

Un contexte de données est lié à chaque workflow de données. Ce contexte de données peut être utilisé pour définir des variables, qui pourront être utilisées en entrée et/ou en sortie dans les différentes étapes du workflow.

Stratégie d'avancement de l'étape suivante

Pour chaque type d'étape (excepté pour les appels de sous-workflows), une propriété est disponible pour préciser la stratégie d'avancement pour l'étape suivante. A la terminaison de l'étape, cette stratégie est évaluée pour conditionner la navigation lors de l'exécution du workflow. Par défaut, la stratégie d'avancement est 'Afficher la table des bons de travail'. Dans ce cas, après l'exécution de l'étape, la table des bons de travail est automatiquement affichée pour sélectionner le prochain bon de travail à ouvrir.

Une autre stratégie est disponible : 'Ouvrir automatiquement l'étape suivante'. Cette stratégie permet à l'utilisateur de garder la main sur ce workflow et d'exécuter directement l'étape suivante. Si, à la suite de cette exécution, un bon de travail est atteint et que l'utilisateur connecté peut le démarrer, le bon de travail est automatiquement ouvert (si plusieurs bons de travail sont atteints, le premier créé est ouvert). Sinon, la stratégie d'avancement de l'étape suivante est évaluée. Si aucun bon de travail n'est finalement atteint, la table des bons de travail est affichée.

Cette stratégie est préconisée pour exécuter plusieurs étapes à la suite sans repasser par la corbeille de tâches.

Il existe actuellement plusieurs limitations qui font que cette stratégie peut être ignorée. Dans ce cas, la table des bons de travail est automatiquement affichée. Les cas où cette propriété est ignorée sont les suivants : si l'étape suivante est un sous-workflow, ou si l'étape courante est une tâche utilisateur avec plus d'un bon de travail.

Dans le cas des conditions, deux autres stratégies sont proposées : 'Si vrai, ouvrir automatiquement l'étape suivante' et 'Si faux, ouvrir automatiquement l'étape suivante'. Ces stratégies permettent de conditionner la stratégie à appliquer en fonction du résultat de la condition.

24.3 Tâche utilisateur

Une tâche utilisateur est une étape qui nécessite une action de la part d'un utilisateur humain. Son libellé et sa description peuvent être localisés.

Participants

Les participants sont les rôles ou les utilisateurs auxquels la tâche utilisateur est destinée. Si un profil est un utilisateur, le bon de travail est automatiquement alloué à cet utilisateur. Si un profil est un rôle, le bon de travail est proposé aux membres du rôle.

Voir aussi [Extension](#) [p 134]

Service

EBX5 propose les services prédéfinis suivants :

- Accéder à des données
- Accéder à l'interface de fusion des espace de données
- Accéder à un espace de données
- Comparer deux contenus
- Créer un nouvel enregistrement
- Dupliquer un enregistrement
- Exporter les données depuis une table au format CSV
- Exporter les données depuis une table au format XML
- Fusionner un espace de données
- Importer les données dans une table depuis un fichier CSV
- Importer les données dans une table depuis un fichier XML
- Valider un espace de données, une image ou un jeu de données

Voir aussi [Services prédéfinis de EBX5](#) [p 199]

Configuration

Options générales > Rejet activé

Par défaut, seulement l'action *accepter* est proposée à l'utilisateur lorsqu'il enregistre sa décision.

Il est possible d'activer l'action 'rejeter' en positionnant ce champ à 'Oui'.

Options générales > Demande de confirmation activée

Par défaut, quand un utilisateur enregistre sa décision en cliquant sur le bouton 'Accepter' ou 'Rejeter', une demande de confirmation est affichée.

Il est possible de désactiver cette demande de confirmation de la décision en positionnant ce champ à 'Non'.

Options générales > Caractère obligatoire du commentaire

Par défaut, le commentaire associé à un bon de travail est optionnel.

Il est possible de forcer l'utilisateur à saisir un commentaire avant d'enregistrer sa décision en positionnant ce champ sur le comportement attendu : 'toujours obligatoire', 'obligatoire seulement si le bon de travail a été accepté' ou 'obligatoire seulement si le bon de travail a été rejeté'.

Options générales > Libellés personnalisés

Durant l'exécution des tâches utilisateur, l'utilisateur peut accepter ou rejeter son bon de travail en cliquant sur le bouton correspondant. Dans la modélisation de workflow, il est possible pour certaines tâches utilisateur de définir un libellé et un message de confirmation personnalisés pour ces boutons. Cette fonctionnalité est particulièrement utile pour ajouter une signification particulière à l'action d'accepter ou rejeter un bon de travail.

Terminaison > Critère de fin de tâche

Une tâche utilisateur peut être assignée à plusieurs *participants* et générer plusieurs bons de travail durant l'exécution du workflow. Lors de la définition d'une tâche utilisateur dans le modèle de workflow, il est possible de sélectionner un des critères prédéfinis qui déterminent quand une tâche utilisateur est terminée, en se basant sur le statut des bons de travail associés. Lorsque la condition de sortie de la tâche utilisateur sera remplie, le workflow de données avancera jusqu'à l'étape suivante définie dans le modèle.

Par exemple, pour le cas d'une tâche utilisateur qui demande la validation de l'enregistrement d'un produit, il est possible de désigner trois participants. Le critère de fin de tâche permet de préciser si l'enregistrement du produit doit être validé par les trois participants, ou uniquement par le premier utilisateur à répondre.

Le critère de fin de tâche par défaut est 'Quand tous les bons de travail ont été acceptés'.

Remarque : Si une extension de service surcharge la méthode `UserTask.handleWorkItemCompletion` pour gérer la fin de la tâche utilisateur, c'est à la charge du développeur de l'extension de vérifier dans le code de cette méthode que le critère de fin de tâche définie dans l'interface a été satisfait. Voir `UserTask.handleWorkItemCompletionAPI` dans la JavaDoc pour plus d'informations.

Terminaison > Tolérance de rejet

Par défaut, si un utilisateur rejette un bon de travail durant l'exécution d'un workflow, la tâche utilisateur est positionnée en erreur et l'avancement du workflow est stoppé.

Pour changer ce comportement par défaut, il est possible de définir un nombre de bons de travail rejetés à tolérer. Tant que la limite de rejets tolérés n'est pas dépassée, aucune erreur ne se produit et c'est le critère de fin de tâche qui détermine quand la tâche utilisateur est terminée.

Les critères de fin de tâche suivants tolèrent automatiquement tous les rejets :

- 'Quand tous les bons de travail ont été acceptés ou rejetés'
- 'Quand tous les bons de travail ont été acceptés, ou dès qu'un bon de travail a été rejeté'

Extension

Il est possible d'étendre programmatiquement le comportement de la tâche afin que cette dernière s'insère plus finement dans le contexte du workflow. Par exemple si un comportement spécifique est nécessaire, pour la création du bon de travail ou pour terminer la tâche de l'utilisateur.

La règle spécifiée est une classe Java Bean qui doit étendre la classe `UserTaskAPI`.

Attention

Si une règle est spécifiée et la méthode `handleWorkItemCompletion` surchargée, la stratégie de fin n'est plus automatiquement contrôlée. C'est au développeur de la vérifier dans cette méthode.

Notification

Une notification par courrier électronique peut être envoyée aux utilisateurs quand des événements spécifiques se produisent. Pour chaque événement, vous pouvez spécifier un message type à utiliser.

En outre, il est possible de définir un profil, auquel une copie de chaque courrier envoyé sera transmise.

Voir aussi [Modèles de message génériques](#) [p 127]

Relance

Des courriers électroniques de relance pour les bons de travail proposés ou alloués et inachevés peuvent être envoyés aux utilisateurs concernés de manière périodique.

Le contenu des courriers de relance dépend de l'état courant du bon de travail. En effet, si le bon de travail est proposé, la notification utilisera le modèle de courrier électronique 'Bons de travail proposés' ; si le bon de travail est alloué, la notification utilisera le modèle 'Bons de travail alloués'.

Echéance

Une tâche utilisateur peut avoir une échéance. Quand cette date est atteinte et si les bons de travail associés n'ont pas été terminés, un courrier électronique spécifique est envoyé aux utilisateurs concernés. Ce courrier électronique va alors être renvoyé tous les jours jusqu'à l'achèvement de la tâche.

Il y a deux types d'échéances :

- *Echéance absolue* : une date du calendrier.
- *Echéance relative* : durée (en heures, jours ou mois). La durée est évaluée à partir d'une date de référence : début d'une tâche utilisateur ou début d'un workflow.

24.4 Tâche autonome

Il existe deux types de tâches automatiques :

Script de la bibliothèque	<p>EBX5 fournit des scripts de la bibliothèque prédéfinis, qui peuvent être utilisés directement.</p> <p>Les scripts de la bibliothèque spécifiques doivent être déclarés dans un fichier <code>module.xml</code>. Un script de la bibliothèque spécifique doit définir son libellé, sa description et ses paramètres. Quand un utilisateur sélectionne un script de la bibliothèque dans une étape d'un modèle de données, les paramètres associés sont affichés dynamiquement.</p>
Script spécifique	<p>Spécifie une classe Java qui exécute des actions spécifiques. La classe associée doit être dans le même module que celui associé au modèle de workflow. Ses libellés et ses descriptions ne sont pas affichés dynamiquement aux utilisateurs dans un modèle de workflow.</p>

[Packaging EBX5 modules](#) [p 453]

Script de la bibliothèque

EBX5 inclut les scripts de la bibliothèque prédéfinis suivants :

- Créer un espace de données
- Créer une image
- Envoyer un courrier électronique
- Fermer un espace de données
- Fusionner un espace de données
- Importer une archive

Les scripts de la bibliothèque sont des classes qui étendent la classe `ScriptTaskBeanAPI`. En complément des scripts de la bibliothèque prédéfinis, vous pouvez définir des scripts de la bibliothèque additionnels pour les utiliser dans les modèles de workflows. Leurs libellés et descriptions peuvent être localisés.

La méthode `ScriptTaskBean.executeScriptAPI` est appelée lorsque le workflow de données atteint l'étape correspondante.

Attention

La méthode `ScriptTaskBean.executeScriptAPI` ne doit créer aucun `Thread`. En effet, le moteur de workflow passera à l'étape suivante quand le script se terminera. L'exécution d'une partie du script en mode asynchrone ne garantit donc pas sa terminaison avant le passage à l'étape suivante.

Voir l'**exemple** Package `com.orchestranetworks.workflowAPI` dans l'API Java.

Il est possible de paramétrer dynamiquement les variables du script de la bibliothèque, si son implémentation suit la spécification Java Bean. Les variables doivent être déclarées comme paramètres du script de la bibliothèque dans le fichier `module.xml`. Le contexte des données n'est pas accessible depuis le Java bean.

Note

Certains scripts de bibliothèque prédéfinis sont marqués comme "obsolètes" car ils ne sont pas compatibles avec internationalisation. Il est recommandé d'utiliser les nouvelles tâches qui sont compatibles avec internationalisation.

Scripts spécifiques

Les scripts spécifiques ont la particularité d'étendre la classe `ScriptTaskAPI`.

La méthode `ScriptTask.executeScriptAPI` est appelée lorsque le workflow de données atteint l'étape correspondante.

Attention

La méthode `ScriptTask.executeScriptAPI` ne doit créer aucun `Thread`. En effet, le moteur de workflow passera à l'étape suivante quand le script se terminera. L'exécution d'une partie du script en mode asynchrone ne garantit donc pas sa terminaison avant le passage à l'étape suivante.

Voir l'**exemple** Package `com.orchestranetworks.workflowAPI` dans l'API Java.

Il n'est pas possible de paramétrer dynamiquement les variables d'un script spécifique. Toutefois, le contexte des données est accessible depuis le Java Bean.

24.5 Conditions

Il existe deux types de conditions :

Condition de la bibliothèque	<p>EBX5 fournit des conditions de la bibliothèque prédéfinies, qui peuvent être utilisées directement.</p> <p>Les conditions de la bibliothèque spécifiques doivent être déclarées dans un fichier <code>module.xml</code>. Une condition de la bibliothèque spécifique doit définir son libellé, sa description et ses paramètres. Quand un utilisateur sélectionne une condition de la bibliothèque dans une étape d'un modèle de données, les paramètres associés sont affichés dynamiquement.</p>
Condition spécifique	<p>Spécifie une classe Java qui implémente une condition spécifique. La classe associée doit être dans le même module que celui associé au modèle de workflow. Ses libellés et ses descriptions ne sont pas affichés dynamiquement aux utilisateurs dans un modèle de workflow.</p>

[Packaging EBX5 modules](#) [p 453]

Condition de la bibliothèque

EBX5 inclut les conditions de la bibliothèque prédéfinies suivantes :

- Dernière tâche utilisateur acceptée ?
- Espace de données modifié ?
- Espace de données valide ?
- Valeur nulle ou vide ?
- Valeurs égales ?

Les conditions de la bibliothèque sont des classes qui étendent la classe `ConditionBeanAPI`. En complément aux conditions de la bibliothèque prédéfinies, vous pouvez définir des conditions de la bibliothèque additionnelles pour les utiliser dans des modèles de données. Leurs libellés et descriptions peuvent être localisés.

Voir l'**exemple** `Package com.orchestranetworks.workflowAPI` dans l'API Java.

Il est possible de paramétrer dynamiquement les variables de la condition de la bibliothèque, si son implémentation suit la spécification Java Bean. Les variables doivent être déclarées comme paramètres du script de la bibliothèque dans le `module.xml`. Le contexte des données n'est pas accessible depuis le Java bean.

Conditions spécifiques

Les conditions spécifiques ont la particularité d'étendre la classe `ConditionAPI`.

Voir l'**exemple** `Package com.orchestranetworks.workflowAPI` dans l'API Java.

Il n'est pas possible de paramétrer dynamiquement les variables d'une condition spécifique. Toutefois, le contexte des données est accessible depuis le Java bean.

24.6 Appels à des sous-workflows

Les étapes du type appel à des sous-workflows positionnent le workflow de données courant dans l'état "en attente" et lancent un ou plusieurs sous-workflows.

Il est possible d'inclure un autre modèle de workflow dans le modèle de workflow courant en définissant un appel unique à ce sous-workflow dans une étape.

Si plusieurs sous-workflows sont appelés par une étape, ils sont exécutés simultanément, en parallèle. Tous les sous-workflows doivent être terminés pour que le workflow parent passe à l'étape suivante. Le libellé et la description de chaque sous-workflow peuvent être localisés.

Deux types d'appels à des sous-workflows existent :

Statique	<p>Définit un ou plusieurs sous-workflows à lancer chaque fois que cette étape est exécutée dans un workflow de données. Pour chaque sous-workflow, il est possible de spécifier ses libellés et ses descriptions, ainsi que les mappings d'entrée et de sortie dans son contexte de données.</p> <p>Ce mode est utile quand on sait à l'avance quels sous-workflows doivent être lancés, et comment le mapping de sortie doit être réalisé.</p>
Dynamique	<p>Spécifie une classe Java qui implémente un appel spécifique à des sous-workflows. Tous les workflows qui peuvent éventuellement être appelés comme sous-workflows par le code doivent être déclarés comme dépendances.</p> <p>Le contexte de données est directement accessible depuis le Java bean.</p> <p>Les appels de sous-workflows dynamiques doivent être déclarés dans un fichier <code>module.xml</code>.</p> <p>Ce mode est utile quand le lancement des sous-workflows est conditionnel (par exemple, lorsqu'il dépend d'une variable du contexte de données), ou quand le mapping de sortie dépend de l'exécution des différents sous-workflows.</p>

24.7 Tâches d'attente

Une étape d'attente dans un modèle de workflow met le workflow en pause en attendant la réception d'un événement donné.

Lorsqu'une tâche d'attente est appelée, le moteur de workflow génère un identifiant unique de réveil lié à la tâche d'attente. Cet identifiant est requis pour réveiller la tâche d'attente et, par conséquent, le workflow associé.

Une tâche d'attente spécifie quel profil est autorisé à réveiller la tâche d'attente ; et la classe Java qui implémente un bean de tâche d'attente : `WaitTaskBeanAPI`.

Le contexte de données du workflow est directement accessible à partir du Java bean.

Les beans de tâche d'attente doivent être déclarés dans un fichier `module.xml`.

Le bean de tâche d'attente est d'abord appelé lorsque le workflow est mis en pause. L'identifiant de réveil est alors disponible pour appeler un web service, par exemple. Puis, le bean de tâche d'attente est appelé lorsque la tâche d'attente est réveillée. De cette façon, le contexte de données peut être mis à jour en fonction des paramètres reçus.

Note

L'administrateur built-in est toujours autorisé à réveiller un workflow.

CHAPITRE 25

Configuration du modèle de workflow

Ce chapitre contient les sections suivantes :

1. [Informations](#)
2. [Propriétés du modèle de workflow](#)
3. [Permissions sur les workflows de données associés](#)
4. [Historique des images du modèle de workflow](#)
5. [Suppression d'un modèle de workflow](#)

25.1 Informations

Pour visualiser et éditer les informations concernant le propriétaire et la documentation du modèle de workflow, sélectionnez "Informations" dans le menu ["Actions" du modèle de workflow](#) [p 127] dans le panneau de navigation.

Propriétaire	Personne pouvant éditer les informations du modèle de workflow et définir des règles de permission.
Documentation locale	Libellé et description associés au modèle de workflow localisés.
Activation	<i>Cette propriété est obsolète.</i> Spécifie si un modèle de workflow est activé. Un modèle de workflow doit être activé pour pouvoir être publié.

25.2 Propriétés du modèle de workflow

La configuration d'un modèle de workflow est accessible depuis le panneau de navigation.

Module	Module contenant les ressources Java spécifiques (extension de tâche utilisateur, script et conditions spécifiques).
Notification de démarrage	Liste des profils qui doivent recevoir une notification (choisie dans la liste des modèles de messages) quand un workflow démarre. Voir Modèles de message génériques [p 127].
Notification de fin	Liste des profils qui doivent recevoir une notification (choisie dans la liste des modèles de messages), quand un workflow se termine. La notification n'est envoyée que si le workflow a été terminé "normalement" (pas par une action d'administration). Voir Modèles de message génériques [p 127].
Priorité	Par défaut, chaque workflow associé à ce modèle sera lancé avec cette priorité. Cette valeur est facultative. Si aucune valeur n'est définie ici, et une priorité par défaut est définie pour le référentiel, la priorité par défaut pour le référentiel sera appliquée à tous les workflows et bons de travail sans priorité. Voir Priorité de bons de travail [p 161] pour plus d'informations. Note : Seuls les utilisateurs définis en tant qu'administrateurs de workflows seront autorisés à modifier la priorité des workflows de données associés manuellement.
Activer le lancement direct	Par défaut, lorsqu'un workflow est lancé, il est proposé à l'utilisateur de saisir une documentation pour le nouveau workflow dans un formulaire intermédiaire. Cette documentation est facultative. Positionner la propriété "Activer le lancement direct" à "Oui" permet d'éviter cette étape de documentation et de lancer directement le workflow.
Ouvrir automatiquement la première étape	Permet de conditionner la navigation, suite à un lancement de workflow. Par défaut, une fois le workflow lancé, la table courante (lanceurs de workflow ou monitoring > publications) est automatiquement affichée.

Activer cette propriété permet au créateur du workflow de garder la main sur le workflow lancé. Si, suite à l'exécution de la première étape du workflow, un bon de travail est atteint, et que ce bon de travail peut être démarré par le lanceur de workflow, le bon de travail est automatiquement ouvert (si plusieurs bons de travail sont atteints, le premier créé est ouvert). Cela évite au lanceur de sélectionner le bon de travail correspondant dans la corbeille de tâches.

Si aucun bon de travail n'est atteint, la stratégie d'avancement de l'étape suivante est évaluée.

Si aucun bon de travail n'est finalement ouvert, la table à partir de laquelle le workflow a été lancé est affichée.

Limitation : Cette propriété est ignorée si la première étape est un appel de sous-workflow.

Permissions	Définit les permissions pour les actions liées au workflows de données associés au modèle de workflow.
Permissions programmatiques	Définit le composant gérant les permissions du workflow. S'il est défini, il remplace l'ensemble des permissions définies dans la propriété 'Permissions'.

25.3 Permissions sur les workflows de données associés

Administration de workflow	Définit le profil autorisé à exécuter des tâches d'administration sur les workflows. Les actions d'administration sont : rejouer une étape, réveiller un workflow, terminer un workflow, désactiver une publication et dépublier. Pour pouvoir exécuter ces actions, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". L'administrateur built-in est toujours autorisé à administrer les workflows.
Administration de workflow > Rejouer une étape	Définit le profil autorisé à rejouer une étape de workflow. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour rejouer une étape, un bouton est disponible dans la section "Monitoring > Workflows actifs". Un profil qui a la permission "Administration de workflow" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à rejouer une étape de workflow.
Administration de workflow > Terminer un workflow	Définit le profil autorisé à terminer et supprimer un workflow. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour terminer et supprimer un workflow en cours, un bouton est disponible dans la section "Monitoring > Workflows actifs". Pour supprimer un workflow terminé, un bouton est disponible dans la section "Workflows terminés". Un profil qui a la permission "Administration de workflow" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à terminer un workflow.
Administration de workflow > Forcer le réveil d'un workflow	Définit le profil autorisé à forcer le réveil d'un workflow en attente. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour réveiller un workflow, un bouton est disponible dans la section "Monitoring > Workflows actifs". Un profil qui a la permission "Administration de workflow" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à réveiller un workflow.
Administration de workflow > Désactiver une publication	Définit le profil autorisé à désactiver une publication de workflow. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour désactiver, un bouton est disponible

dans la section "Monitoring > Publications" uniquement pour les publications actives. Un profil qui a la permission "Administration de workflow" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à désactiver une publication.

Administration de workflow > Dépublier

Définit le profil autorisé à dépublier une publication de workflow. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour dépublier, un bouton est disponible dans la section "Monitoring > Publications" pour les publications désactivées uniquement. Un profil qui a la permission "Administration de workflow" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à dépublier.

Gestion de l'allocation

Définit le profil autorisé à gérer l'allocation des bons de travail. Les actions d'allocation sont : allouer les bons de travail, réallouer les bons de travail et désallouer les bons de travail. Pour pouvoir exécuter ces actions, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". L'administrateur built-in est toujours autorisé à gérer l'allocation des workflows.

Gestion de l'allocation > Allouer les bons de travail

Définit le profil autorisé à allouer les bons de travail. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour allouer un bon de travail, un bouton est disponible dans la section "Monitoring > Bons de travail" uniquement pour les bons de travail proposés. Un profil qui a la permission "Gestion de l'allocation" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à allouer les bons de travail.

Gestion de l'allocation > Réallouer les bons de travail

Définit le profil autorisé à réallouer les bons de travail. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour réallouer, un bouton est disponible dans la section "Monitoring > Bons de travail" uniquement pour les bons de travail alloués. Un profil qui a la permission "Gestion de l'allocation" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à réallouer les bons de travail.

Gestion de l'allocation > Désallouer les bons de travail

Définit le profil autorisé à désallouer les bons de travail. Pour pouvoir exécuter cette action, ce profil bénéficie automatiquement de la permission "Visualiser les workflows". Pour désallouer, un bouton est disponible dans

la section "Monitoring > Bons de travail" uniquement pour les bons de travail alloués. Un profil qui a la permission "Gestion de l'allocation" est automatiquement autorisé à effectuer cette action spécifique. L'administrateur built-in est toujours autorisé à désallouer les bons de travail.

Lancer les workflows	Définit le profil autorisé à lancer manuellement de nouveaux workflows. Cette permission permet de lancer des workflows à partir des publications actives dans la section "Lanceurs de workflow". L'administrateur built-in est toujours autorisé à lancer les workflows.
Visualiser les workflows	Définit le profil autorisé à visualiser les workflows. Par défaut, un utilisateur final peut uniquement voir les bons de travail qui lui sont proposés ou alloués dans la section "Boîte de réception". Cette permission permet en plus de visualiser les publications, workflows et bons de travail associés à ce modèle de workflow dans les sections "Monitoring" et "Workflows terminés". Ce profil bénéficie automatiquement de la permission "Visualiser les workflows terminés". L'administrateur built-in est toujours autorisé à visualiser les workflows.
Visualiser les workflows > Le créateur d'un workflow peut le visualiser	Si activé, le créateur d'un workflow a la permission de visualiser les workflows qu'il a lancés. Cette permission restreinte lui donne accès aux workflows qu'il a lancés et aux bons de travail associés dans les sections "Monitoring > Workflows actifs", "Monitoring > Bons de travail" et "Workflows terminés". La valeur par défaut est "Non".
Visualiser les workflows > Visualiser les workflows terminés	Définit le profil autorisé à visualiser les workflows terminés. Cette permission permet de visualiser les workflows terminés dans la section "Workflows terminés" et d'accéder à leur historique. Un profil qui a la permission "Visualiser les workflows" est automatiquement autorisé à effectuer cette action. L'administrateur built-in est toujours autorisé à visualiser les workflows terminés.

Note

Un utilisateur n'ayant aucun privilège particulier pourra voir un bon de travail uniquement si celui-ci lui est proposé ou personnellement alloué.

Voir aussi [Administration d'un workflow](#) [p 165]

25.4 Historique des images du modèle de workflow

L'historique des images d'un modèle de workflow peut être géré sous **Actions > Historique**.

La table d'historique affiche toutes les images contenant le modèle de workflow et indique si le modèle a été publié. Pour chaque image, il est possible d'exporter ou d'afficher le modèle de workflow correspondant en utilisant le bouton **Actions**.

25.5 Suppression d'un modèle de workflow

Un modèle de workflow peut être supprimé. Cependant, toute version publiée auparavant reste accessible dans la section Workflows de Données. D'autre part, si un modèle de workflow est recréé avec un nom identique, lors d'une publication, un message demandera la confirmation de remplacement de la publication précédente.

Voir aussi [Publication d'un modèle de workflow](#) [p 149]

CHAPITRE 26

Publication d'un modèle de workflow

Ce chapitre contient les sections suivantes :

1. [A propos des publications de workflow](#)
2. [Publication et images de modèle de données](#)
3. [Sous-workflows dans les publications](#)

26.1 A propos des publications de workflow

Dès qu'un modèle de workflow est défini, il doit être publié afin d'autoriser les utilisateurs à lancer des workflows de données associés. Une publication est réalisée en cliquant sur le bouton 'Publier' dans le panneau de navigation.

Si aucune étape d'appel à des sous-workflows n'est incluse dans le modèle courant, l'option de publier d'autres modèles en même temps vous sera proposée sur la page de publication. Si le modèle de workflow actuel contient des étapes d'appel à des sous-workflows, il doit être publié seul.

Les modèles de workflow peuvent être publiés plusieurs fois. Une publication est identifiée par son nom de publication.

26.2 Publication et images de modèle de données

Durant la publication d'un modèle de workflow, une image est enregistrée de son état actuel. Vous pouvez préciser un libellé et une description de l'image. Le libellé par défaut pour l'image est l'heure et la date au moment de publication. La description par défaut indique l'utilisateur qui a publié le modèle de workflow.

Pour chaque modèle de workflow publié, le nom de la publication doit être unique. Si un modèle de workflow a déjà été publié, il est possible de mettre à jour une publication existante en réutilisant le même nom de publication. Les noms des publications de workflow existantes sont proposés dans un menu. Dans le cas d'une mise à jour de publication, l'ancienne version ne sera plus disponible pour lancer des workflows de données ; mais elle permettra aux workflows existants de finir de s'exécuter. Le contenu des différentes images peut être consulté dans l'historique des images de modèle de workflow.

Voir aussi [Historique des images du modèle de workflow](#) [p 146]

26.3 Sous-workflows dans les publications

Quand un modèle de workflow, contenant un appel à des sous-workflows, est publié, il n'est pas nécessaire de publier séparément les sous-workflows appelés. D'un point de vue d'administration, le modèle du workflow principal (celui actuellement publié par l'utilisateur) et les modèles de sous-workflows sont publiés comme une entité unique.

Le système établit les dépendances avec les modèles de workflows utilisés comme sous-workflows et crée automatiquement une publication par modèle dépendant. Ces publications techniques sont exclusivement dédiées au moteur de workflow pour le lancement des sous-workflows et ne sont donc pas disponibles dans la section Workflows de données.

La publication multiple n'est pas disponible pour un modèle de workflow contenant un appel à des sous-workflows. C'est pourquoi la première étape de publication (sélection des modèles de workflow à publier) n'est pas proposée dans ce cas.

La republication du modèle de workflow principal met automatiquement à jour les modèles de sous-workflows appelés.

Bien qu'un sous-workflow puisse être publié séparément comme modèle de workflow principal, cela ne modifiera pas la version utilisée par un autre modèle de workflow principal publié qui utilise ce sous-workflow.

Workflows de données

CHAPITRE 27

Introduction aux workflows de données

Ce chapitre contient les sections suivantes :

1. [Présentation](#)

27.1 Présentation

Un workflow de données est un processus exécuté sous la forme d'une succession d'étapes, définie par une publication de modèle de workflow. Le workflow de données permet aux utilisateurs, ainsi qu'aux procédures automatisées, d'effectuer des actions de façon collaborative sur les données. Une fois le modèle de workflow spécifié et publié, la publication résultante peut être utilisée pour lancer un workflow de données afin d'exécuter les étapes définies.

En fonction des permissions définies par le modèle de workflow, un utilisateur peut effectuer une ou plusieurs des actions suivantes sur les workflows de données associés :

- en tant qu'utilisateur avec les permissions par défaut, effectuer les actions attendues par les bons de travail qui lui sont destinés,
- en tant qu'utilisateur avec les permissions pour lancer les workflows, créer les nouveaux workflows de données depuis une publication du modèle de workflow,
- en tant qu'utilisateur avec les permissions de monitoring de workflow, suivre l'avancement des workflows de données en cours, et consulter l'historique des workflows de données terminés.
- en tant que gestionnaire d'allocation des bons de travail, modifier les allocations des bons de travail des autres utilisateurs manuellement.
- en tant qu'administrateur de workflow, effectuer différentes actions d'administration, comme par exemple, redémarrer une étape, terminer un workflow en cours ou rendre une publication indisponible pour le lancement de workflows.

Voir aussi

[Bons de travail](#) [p 159]

[Lancement et monitoring de workflows de données](#) [p 163]

[Administration de workflows de données](#) [p 165]

[Permissions sur les workflows de données associés](#) [p 144]

Concepts apparentés [*Modèles de workflow*](#) [p 126]

CHAPITRE 28

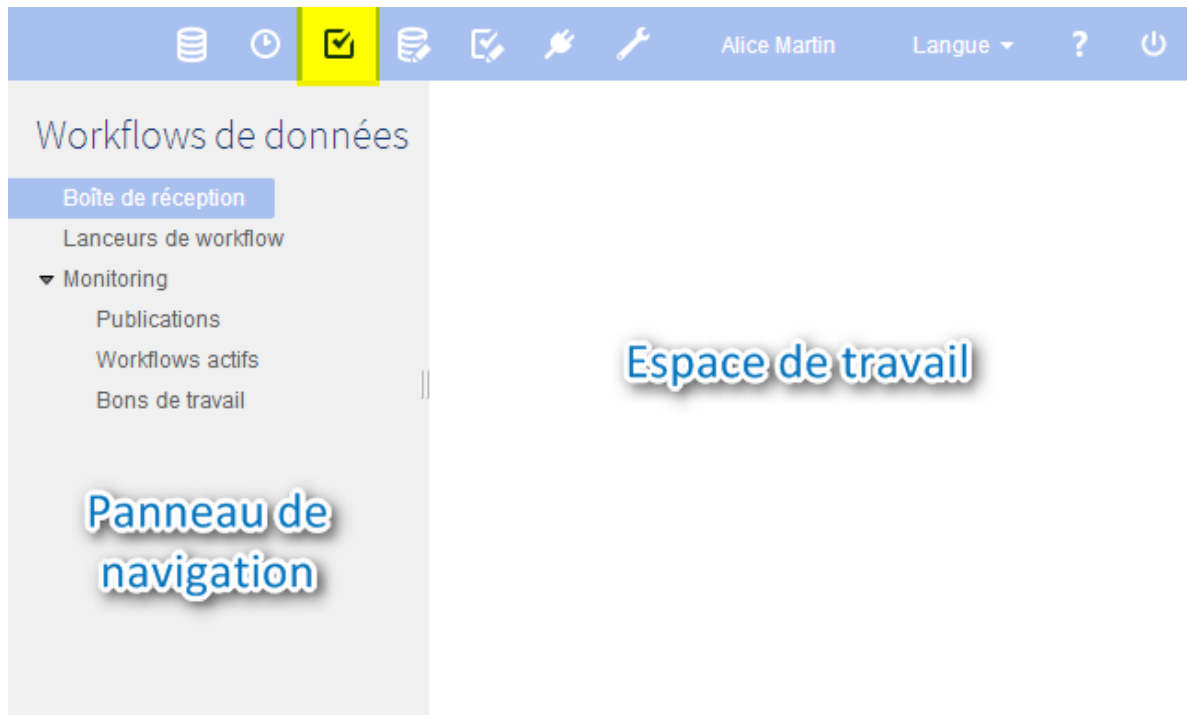
Utilisation de l'interface utilisateur de la section Workflow de données

Ce chapitre contient les sections suivantes :

1. [Navigation dans l'interface utilisateur](#)
2. [Règles de navigation](#)
3. [Filtrage d'items dans les vues](#)
4. [Vue graphique d'un workflow de données](#)

28.1 Navigation dans l'interface utilisateur

La fonctionnalité workflows de données se trouve dans la section **Workflows de données** dans l'interface utilisateur d'EBX5.



Note

Seuls les utilisateurs autorisés peuvent accéder à cet écran via la 'Perspective avancée' ou via une perspective spécifique. Seuls les utilisateurs autorisés peuvent accéder à ces interfaces spécifiques.

Le panneau de navigation est composé de plusieurs items. Ces items sont visibles en fonction des permissions globales associées. Les différents items sont :

Boîte de réception des bons de travail	Tous les bons de travail qui vous sont alloués ou proposés, pour lesquels vous devez effectuer les actions spécifiées.
Lanceurs de workflow	Liste des publications de workflow à partir desquelles vous pouvez lancer des workflows de données, en fonction de vos permissions.
Monitoring	Vues pour le monitoring des workflows de données en cours pour lesquels vous avez les permissions nécessaires de visualisation.
Publications	Publications pour lesquelles vous avez les permissions nécessaires de visualisation. Si vous disposez de permissions d'administration supplémentaires, vous pouvez également désactiver la possibilité de lancer les workflows de données depuis une publication à partir de cette vue.
Workflows actifs	Workflows de données, en cours d'exécution, pour lesquels vous avez les permissions nécessaires de visualisation. Si vous disposez de permissions d'administration supplémentaires, vous pouvez également effectuer des actions d'administration à partir de cette vue, comme par exemple redémarrer une étape, ou terminer un workflow en cours.
Bons de travail	Bons de travail pour lesquels vous avez les permissions nécessaires de visualisation. Si vous disposez de permissions d'administration supplémentaires, vous pouvez également effectuer des actions d'administration à partir de cette vue, comme par exemple allouer les bons de travail aux utilisateurs ou rôles.
Workflows terminés	Workflows de données, dont l'exécution est terminée, pour lesquels vous avez les permissions nécessaires de visualisation. Si vous disposez de permissions d'administration supplémentaires, vous pouvez également nettoyer les workflows terminés à partir de cette vue.

Note

Chaque section est accessible par composant web, par exemple pour intégration avec un portail, ou programmatiquement par la classe `ServiceKey` dans l'API Java.

Voir aussi

[Using EBX5 as a web component](#) [p 183]

ServiceKey^{API}

28.2 Règles de navigation

Corbeille de bons de travail

Par défaut, une fois qu'un bon de travail est exécuté, la corbeille de bons de travail est affichée.

Ce comportement peut être modifié en fonction de la stratégie d'avancement de l'étape suivante. Cette stratégie permet d'enchaîner plusieurs étapes à la suite sans repasser par la corbeille de tâches.

Voir [la stratégie d'avancement d'une étape de workflow](#) [p 132] dans la modélisation de workflow.

Lanceurs de workflow

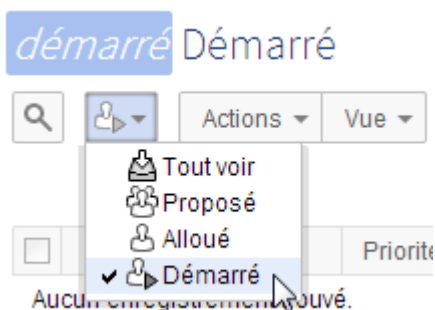
Par défaut, une fois qu'un workflow est lancé, la table des lanceurs de workflow est affichée.

Ce comportement est modifiable dans la configuration du modèle : il est possible d'ouvrir directement la première étape sans afficher la table des lanceurs de workflow.

Voir [l'ouverture automatique de la première étape d'un workflow](#) [p 142] dans la modélisation de workflow.


28.3 Filtrage d'items dans les vues

Dans certaines vues, comme la 'Boîte de réception' des bons de travail, vous pouvez filtrer les lignes affichées dans les tables en fonction de leur état. Dans ces vues, un menu est disponible à cet effet pour sélectionner l'état correspondant au filtre attendu.



28.4 Vue graphique d'un workflow de données

En tant qu'utilisateur avec un bon de travail à effectuer, ou en tant que superviseur ou administrateur de workflow de données, vous pouvez suivre l'avancement ou l'historique d'exécution d'un workflow

de données. Pour cela, cliquer sur le bouton 'Afficher'  dans la colonne 'Workflow de données' des tables de l'interface de workflows de données. Ce bouton ouvre une pop-up qui affiche une vue interactive graphique de l'exécution du workflow de données. Dans cette vue, vous pouvez suivre l'avancement global de l'exécution, et sélectionner une étape individuelle afin de consulter le détail de ses informations.

CHAPITRE 29

Bons de travail

Ce chapitre contient les sections suivantes :

1. [Présentation des bons de travail](#)
2. [Travail sur un bon de travail en tant que participant](#)
3. [Priorité de bons de travail](#)

29.1 Présentation des bons de travail

Un bon de travail est une tâche utilisateur unitaire qui doit être effectuée par un utilisateur humain. Par défaut, quand un modèle de workflow définit une tâche utilisateur, les workflows de données, lancés depuis les publications du modèle, génèrent un bon de travail individuel pour chacun des participants listés dans la tâche utilisateur.

Voir aussi [Integration of UI services with workflows or perspectives](#) [p 194]

Etats d'un bon de travail

Lorsqu'un bon de travail est émis, pendant l'exécution d'un workflow de données, pour une tâche utilisateur définie dans le modèle, le bon de travail peut prendre plusieurs états : proposé, alloué, démarré, et terminé.

Par défaut, pour chaque utilisateur défini comme participant de la tâche utilisateur, le workflow de données crée un bon de travail dans l'état *alloué*. L'utilisateur défini peut directement commencer à travailler sur le bon de travail alloué en effectuant l'action 'Prendre et démarrer'. Le bon de travail passe alors à l'état *démarré*.

Par défaut, pour chaque rôle défini comme participant de la tâche utilisateur, le workflows de données crée un bon de travail dans l'état *proposé*. Tout membre du rôle peut prendre le bon de travail en utilisant l'action 'Prendre et démarrer'. Le bon de travail passe ainsi à l'état *démarré*.

Avant qu'un utilisateur ait pris le bon de travail proposé, un gestionnaire du workflow de données peut intervenir pour affecter manuellement le bon de travail à un utilisateur spécifique, passant ainsi

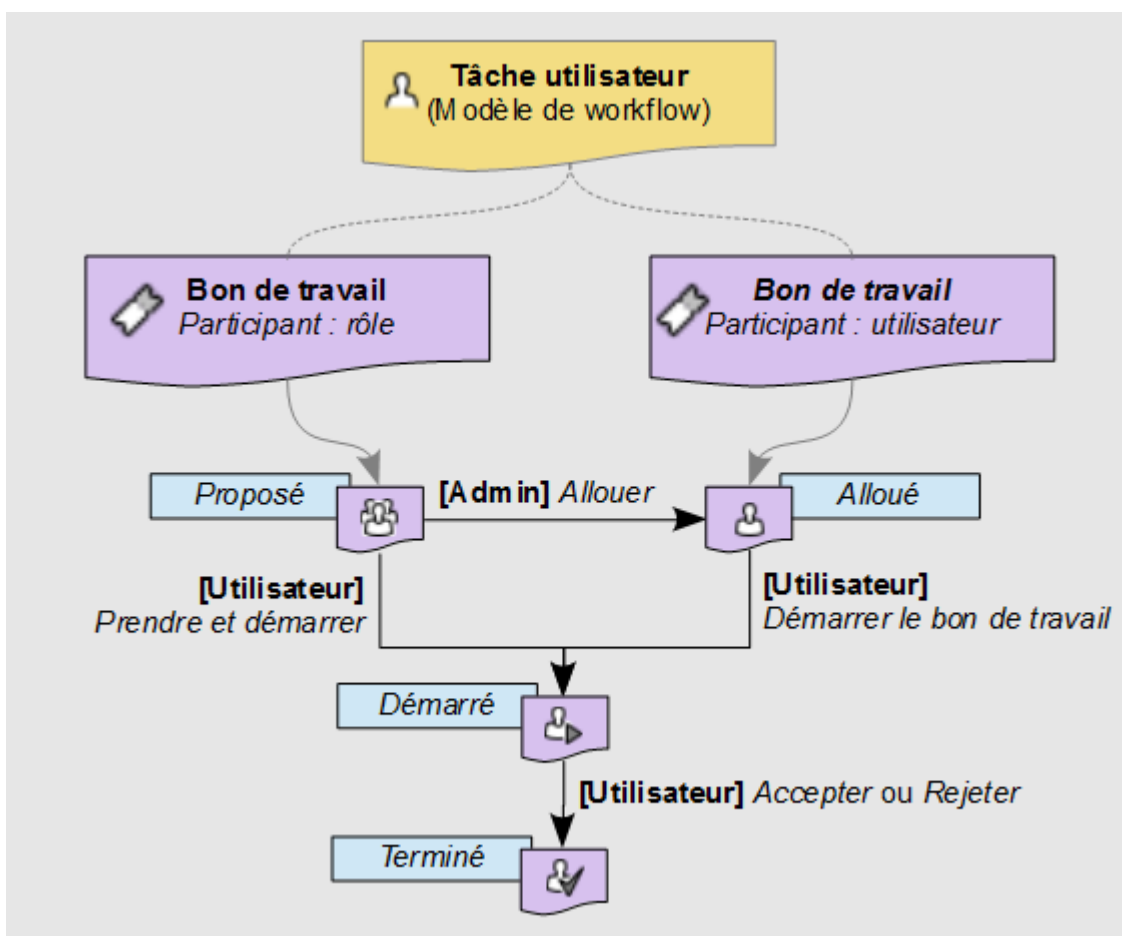
le bon de travail à l'état *alloué*. Puis, lorsque l'utilisateur commence à travailler sur le bon de travail via l'action 'Démarrer le bon de travail', le bon de travail passe à l'état *démarré*.

Note

Le comportement par défaut, décrit ci-dessus, peut être surchargé par une extension programmatique définie dans la tâche utilisateur. Dans ce cas, les bons de travail peuvent être générés programmatiquement, sans se baser obligatoirement sur la liste des participants de la tâche utilisateur.

Une fois que l'utilisateur, qui a démarré le bon de travail, a réalisé l'action demandée, l'action terminale 'Accepter' ou 'Rejeter' place le bon de travail dans l'état *terminé*. Lorsqu'un utilisateur termine un bon de travail, le workflow de données passe automatiquement à l'étape suivante définie dans le modèle de workflow.

Diagramme des états de bon de travail




29.2 Travail sur un bon de travail en tant que participant

Tous les bons de travail disponibles (qui vous sont soit proposés, soit alloués), sont affichés dans votre boîte de réception des bons de travail. Quand vous commencez à travailler sur un bon de travail, vous pouvez ajouter des commentaires associés qui seront visibles par les administrateurs, les superviseurs

du workflow et les autres participants au workflow de données. Tant que vous êtes toujours en train de travailler sur le bon de travail, vous pouvez éditer ce commentaire.

Quand vous avez réalisé toutes les actions demandées par le bon de travail, vous devez signaler la fin du travail en cliquant soit sur le bouton **Accepter**, soit sur le bouton **Rejeter**. Les libellés de ces deux boutons peuvent varier en fonction du contexte du bon de travail

Pour suivre l'avancement du workflow de données associé au bon de travail qui vous est destiné dans votre boîte de réception, cliquez sur le bouton "Afficher"  dans la colonne 'Workflow de données' de la table. Une pop-up affichera une vue graphique interactive du workflow de données jusqu'au moment actuel ainsi que les étapes à venir. Vous pouvez visualiser les détails d'une étape en sélectionnant l'étape.

Note

Si vous interrompez la session actuelle pendant un bon de travail, par exemple en fermant le navigateur ou en déconnectant, l'état actuel du bon de travail est préservé. Quand vous revenez sur le bon de travail, il continue à partir du même point.

29.3 Priorité de bons de travail

Les bons de travail peuvent porter une priorité, qui peut être utile pour trier et filtrer les bons de travail à compléter. La priorité d'un bon de travail est définie au niveau de son workflow de données, et n'est pas spécifique au bon de travail lui-même. Par conséquent, si le workflow de données est considéré comme urgent, tous les bons de travail ouverts associés sont aussi considérés comme urgent. Par défaut, il y a six niveaux de priorité, de "Très peu prioritaire" à "Urgent". Cependant, la représentation visuelle et le nommage des priorités dépendent de la configuration de votre référentiel EBX5.

Voir aussi [tâche utilisateur \(glossaire\)](#) [p 25]

Concepts apparentés [Tâche utilisateur](#) [p 132]

CHAPITRE 30

Lancement et monitoring de workflows de données

Ce chapitre contient les sections suivantes :

1. [Lancement d'un workflow de données](#)
2. [Activités de monitoring](#)
3. [Gestion de l'allocation des bons de travail](#)

30.1 Lancement d'un workflow de données

Quand un modèle de workflow vous autorise à lancer des workflows de données depuis ses publications, vous pouvez créer de nouveaux workflows en utilisant la vue 'Lanceurs de workflow' dans le panneau de navigation. Pour créer un nouveau workflow de données, depuis une publication de modèle de workflow, cliquer sur le bouton 'Lancer' de la ligne de la publication cible dans la table.

Vous pouvez alors définir des libellés et descriptions localisés pour le nouveau workflow de données que vous lancez.

30.2 Activités de monitoring

Quand un modèle de workflow vous donne les permissions de monitoring de workflow, vous avez la possibilité de suivre l'avancement des workflows de données qui sont en cours d'exécution. Vous pouvez accéder aux vues de monitoring, dans la section 'Monitoring' du panneau de navigation. Si vous disposez également de permissions d'administration de workflow, vous pouvez effectuer les actions autorisées associées depuis ces vues.

Lorsqu'un workflow de données, que vous êtes autorisé à suivre, a fini son exécution, il est affiché dans 'Workflows terminés', où vous pouvez consulter son historique.

30.3 Gestion de l'allocation des bons de travail

Lorsque vous êtes autorisé à gérer l'allocation des bons de travail, vous pouvez allouer manuellement des bons de travail durant l'exécution des workflows associés au modèle de workflow. Dans ce cas, vous pouvez effectuer une ou plusieurs des actions ci-dessous sur les bons de travail.

Sélectionnez 'Bons de travail' dans la section 'Monitoring' du panneau de navigation. Les actions que vous pouvez effectuer sont affichées dans le menu 'Actions' de l'entrée du bon de travail, selon son état actuel, dans la table.

Allouer	Allouer un bon de travail à un utilisateur spécifique. Cette action est disponible pour les bons de travail dans l'état <i>proposé</i> .
Désallouer	Remettre un bon de travail qui est actuellement dans l'état <i>alloué</i> dans l'état <i>proposé</i> .
Réallouer	Modifier l'utilisateur à qui le bon de travail est alloué. Cette action est disponible pour les bons de travail dans l'état <i>alloué</i> .

Voir aussi

[Bons de travail](#) [p 159]

[Permissions sur les workflows de données associés](#) [p 144]

Concepts apparentés [Modèles de workflow](#) [p 126]

CHAPITRE 31

Administration de workflows de données

Si vous disposez de permissions pour administrer des workflows de données, les vues 'Publications', 'Workflows actifs', et 'Bons de travail' associés seront accessibles sous le menu 'Monitoring' du panneau de navigation. Dans ces vues, sous les menus 'Actions' sur les lignes des tables, vous pourrez accéder aux actions d'administration.

Note

Quand un modèle de données vous donne des droits d'administration, vous aurez automatiquement les permissions de monitoring sur tous les objets associés à l'exécution de workflow, comme les publications, les workflows actifs, et les bons de travail.

Ce chapitre contient les sections suivantes :

1. [Présentation de l'exécution de workflow de données](#)
2. [Actions d'administration de workflow de données](#)

31.1 Présentation de l'exécution de workflow de données

Quand un workflow de données est lancé, un *jeton* qui représente l'étape en cours d'exécution est créé et positionné au début du workflow. A chaque fois qu'une étape est terminée, ce jeton se déplace sur la prochaine étape définie par le modèle de workflow associé à la publication du workflow de données.

Pendant l'exécution d'un workflow de données, le jeton est positionné sur un des types d'étape suivants:

- une tâche automatique, qui est lancée automatiquement et n'a pas besoin d'interaction utilisateur. La tâche automatique est terminée quand les actions définies finissent leur exécution.
- une tâche utilisateur, qui génère un ou plusieurs bons de travail effectués manuellement par les utilisateurs. Chaque bon de travail est terminé par une action 'Accepter' ou 'Rejeter', réalisée explicitement par l'utilisateur. La fin de la tâche utilisateur chapeau est déterminée en fonction du critère de fin de tâche défini pour la tâche utilisateur dans le modèle de workflow.
- une condition, qui est évaluée automatiquement afin de déterminer l'étape suivante de l'exécution du workflow de données.
- invocation de sous-workflows qui lance les sous-workflows associés et attend que les sous-workflows en cours soient terminés.
- tâche d'attente qui met en pause le workflow jusqu'à ce qu'un événement spécifique soit reçu.

Le jeton peut être dans les états suivants :

- **A exécuter** : Le jeton est en train de passer à la prochaine étape, en se basant sur le modèle de workflow.
- **En cours d'exécution** : Le jeton est positionné sur une tâche automatique ou une condition en train de s'exécuter.
- **Utilisateur** : Le jeton est positionné sur une tâche utilisateur et attend une action utilisateur.
- **En attente de sous-workflows** : Le jeton est positionné sur une invocation de sous-workflows et attend la terminaison de tous les sous-workflows lancés.
- **En attente d'événement** : Le jeton est positionné sur une tâche d'attente et attend de recevoir un événement donné.
- **Terminé** : Le jeton a atteint la fin du workflow de données.
- **Erreur** : Une erreur est survenue.

Voir aussi [Data workflow administration](#) [p 417]

31.2 Actions d'administration de workflow de données

Actions sur les publications

Désactivation d'une publication de workflow

Afin d'éviter que de nouveaux workflows de données soient lancés depuis une publication de workflow, vous pouvez désactiver la publication. Sélectionnez la vue 'Publications' dans la panneau de navigation, puis sélectionnez *Actions > Désactiver* sur la ligne de la publication cible.

Une fois désactivée, la publication n'apparaîtra plus dans la vue 'Lanceurs de workflow' des utilisateurs. Toutefois, les workflows de données déjà lancés vont continuer à s'exécuter.

Note

Suite à la désactivation d'une publication, il n'est pas possible de la réactiver à partir de la section 'Workflows de données'. Seul un utilisateur avec le rôle built-in 'Administrateur' peut réactiver une publication inactive dans la section 'Administration'. Cependant, il n'est pas conseillé de modifier les tables techniques manuellement, car il est important de préserver l'intégrité des données techniques des workflows.

Dépublication d'une publication de workflow

Si une publication de workflow n'est plus utilisée, vous pouvez la supprimer de toutes les vues de la section 'Workflows de données' en la dépubliant. Pour faire cela,

1. Désactivez la publication de workflow afin d'éviter que des utilisateurs continuent de lancer des nouveaux workflows de données sur cette publication. Pour cela, suivez le processus décrit dans la section [Désactivation d'une publication de workflow](#) [p 166].
2. Dépublier la publication de workflow en sélectionnant *Actions > Dépublier* de la ligne de la publication cible.

Note

A la dépublication d'une publication de workflow, une confirmation vous sera demandée pour terminer et purger tous les workflows de données en cours qui ont été lancés depuis cette publication de workflow, ainsi que tout bon de travail associé. Toute perte de données résultant d'une fin prématurée est alors définitive.

Actions sur workflows de données

Dans les vues tabulaires des workflows de données, chaque enregistrement porte un menu *Actions* qui permet d'exécuter des services sur un workflow de données.

Ré-exécution d'une étape

Dans le cas d'une erreur inattendue pendant l'exécution d'une étape, par exemple, à cause d'un problème de permissions ou de ressources non disponibles, vous pouvez "rejouer" une étape en tant qu'administrateur de workflow. En rejouant une étape, l'environnement d'exécution associé est nettoyé, notamment les bons de travail et sous-workflows liés, et le jeton est repositionné au début de l'étape courante.

Pour rejouer l'étape courante dans un workflow de données, sélectionnez *Actions > Rejouer l'étape* dans la ligne du workflow cible dans la table 'Workflows actifs'.

Terminer un workflow de données actif et le purger

Pour terminer un workflow de données en cours d'exécution, sélectionnez *Actions > Terminer et purger* dans la ligne du workflow cible dans la table 'Workflows actifs'. L'action stoppe l'exécution du workflow de données et supprime le workflow, tous les bons de travail et sous-workflows associés.

Note

Cette action n'est pas disponible pour les workflows dans l'état 'En cours d'exécution' et pour les sous-workflows lancés par d'autres workflows.

Forcer la terminaison d'un workflow de données actif

Pour forcer la terminaison d'un workflow de données en cours d'exécution, sélectionnez *Actions > Forcer la terminaison* dans la ligne du workflow cible dans la table 'Workflows actifs'. L'action

stoppe l'exécution du workflow de données et supprime les éventuels bons de travail et sous-workflows associés.

Note

Cette action est disponible pour les sous-workflows, et pour les workflows en erreur bloqués sur la dernière étape.

Forcer le réveil d'un workflow en attente

Pour réveiller un workflow qui est en attente d'événement, sélectionner **Actions > Forcer le réveil** à partir du workflow dans la table 'Workflows actifs'. Cela entraîne le réveil du workflow. Avant d'effectuer cette action, l'administrateur doit mettre à jour le contexte de données afin de s'assurer que le workflow peut exécuter les tâches suivantes.

Note

Cette action est disponible uniquement pour les workflows qui sont à l'état 'en attente d'événement'.

Purge d'un workflow de données terminé

Quand un workflow de données a terminé son exécution, son historique est visible pour ses superviseurs et administrateurs dans la vue 'Workflows terminés'. Pour purger le workflow terminé et son historique, vous pouvez effectuer un nettoyage en sélectionnant *Actions > Purger* dans la ligne du workflow cible de la table 'Workflows terminés'.

Note

Cette action n'est pas disponible pour les sous-workflows lancés par d'autres workflows.

Modification de la priorité d'un workflow de données

Suite au lancement d'un workflow de données, un administrateur du workflow peut modifier son niveau de priorité. En modifiant la priorité du workflow de données, la priorité de tous les bons de travail existants et à venir de ce workflow sera modifiée. Pour modifier la priorité d'un workflow de données, sélectionnez *Actions > Modifier la priorité* dans la ligne du workflow cible dans la table 'Workflows actifs'.

Voir aussi [Permissions sur les workflows de données associés](#) [p 144]

Services de données

CHAPITRE 32

Introduction aux services de données

Ce chapitre contient les sections suivantes :

1. [Présentation](#)
2. [Utilisation de l'interface utilisateur de la section Services de données](#)

32.1 Présentation

Fonction du service de données

Un [service de données](#) [p 27] est un web service standard qui permet d'interagir avec EBX5. Il peut être utilisé pour accéder à une partie des fonctionnalités disponibles par l'interface utilisateur.

Les services de données peuvent être générés dynamiquement à partir d'un modèle de données dans la section 'Services de données'.

Voir aussi [Manuel de référence](#) [p 213]

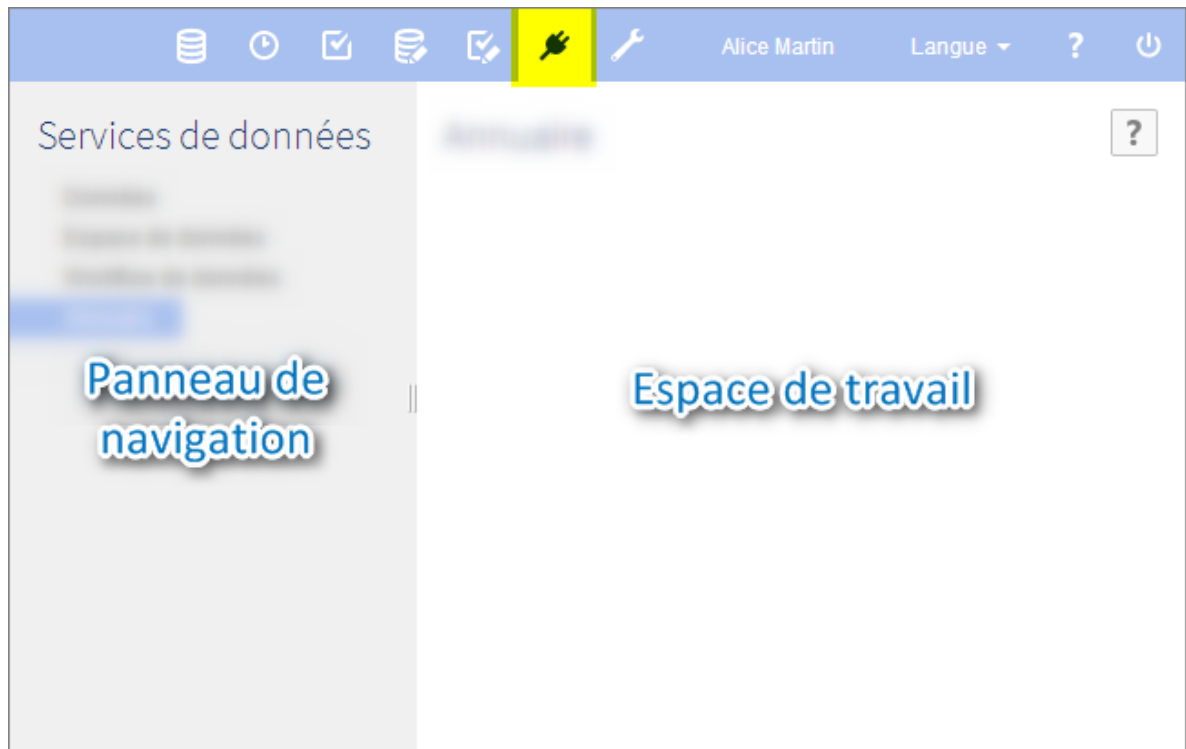
Lignage

Le [lignage](#) [p 27] établit des profils de droit d'accès utilisés par les services de données. Quand les services de données accèdent aux interfaces WSDL, ils utilisent les profils de droit d'accès définis par ce mécanisme.

Glossaire

Voir aussi [Services de données](#) [p 27]

32.2 Utilisation de l'interface utilisateur de la section Services de données



Note

Seuls les utilisateurs autorisés peuvent accéder à cette section via la 'Perspective avancée'.

Concepts apparentés

[Espaces de données](#) [p 78]

[Jeux de données](#) [p 100]

[Workflows de données](#) [p 152]

[Introduction to data services](#) [p 213]

CHAPITRE 33

Génération de WSDL pour services de données

Ce chapitre contient les sections suivantes :

1. [Générer un WSDL pour accéder aux données](#)
2. [Générer un WSDL pour accéder à un espace de données](#)
3. [Générer un WSDL pour contrôler un workflow de données](#)
4. [Générer un WSDL pour un lignage](#)
5. [Générer un WSDL pour gérer l'accès à l'interface utilisateur](#)
6. [Générer un WSDL pour modifier l'annuaire par défaut](#)

33.1 Générer un WSDL pour accéder aux données

La génération de WSDL pour l'accès aux données est disponible en sélectionnant 'Données' dans la section 'Services de données'.

Les étapes de la génération d'un WSDL sont les suivantes :

1. Sélectionner si le WSDL sera utilisé pour des opérations sur un jeu de données ou sur une table.
2. Identifier l'espace de données et le jeu de données ciblés par les opérations.
3. Sélectionner les tables sur lesquelles les opérations sont autorisées, ainsi que les opérations permises.
4. Télécharger le fichier WSDL généré en cliquant sur le bouton 'Télécharger le WSDL'.

Opérations disponibles sur un jeu de données

Les opérations suivantes sur les jeux de données sont disponibles en utilisant le WSDL généré :

- Récupérer les changements sur table(s) du jeu de données entre espaces de données ou images
- Actualiser une unité de réplication en base de données

Voir aussi

[WSDL download using a HTTP request](#) [p 220]

[Operations generated from a data model](#) [p 227]

Opérations disponibles sur une table d'un jeu de données

Si sélectionnées, les opérations suivantes sur les tables sont disponibles en utilisant le WSDL généré :

- Sélectionner un (ou plusieurs) enregistrement(s)
- Insérer un (ou plusieurs) enregistrement(s)
- Mettre à jour un (ou plusieurs) enregistrement(s)
- Supprimer un (ou plusieurs) enregistrement(s)
- Compter des enregistrements
- Récupérer les changements entre espaces de données ou images
- Obtenir les droits d'accès

Voir aussi

[*WSDL download using a HTTP request*](#) [p 220]

[*Operations generated from a data model*](#) [p 227]

Générer un WSDL pour accéder à un espace de données

La génération de WSDL pour la manipulation d'un espace de données est accessible en sélectionnant 'Espace de données' dans la section 'Services de données'. Le WSDL généré n'est pas spécifique à un espace de données et aucune information n'est requise. Il peut être téléchargé grâce au bouton **Télécharger le WSDL**.

Opérations disponibles sur un espace de données

Les opérations suivantes sur les espaces de données sont disponibles en utilisant le WSDL généré :

- Créer un espace de données
- Créer une image
- Fermer un espace de données
- Fermer une image
- Fusionner un espace de données
- Valider un espace de données ou une image
- Valider un jeu de données
- Verrouiller un espace de données
- Déverrouiller un espace de données

Voir aussi

[*WSDL download using a HTTP request*](#) [p 220]

[*Operations on data sets and data spaces*](#) [p 244]

33.3 Générer un WSDL pour contrôler un workflow de données

La génération d'un WSDL pour le contrôle d'un workflow est accessible en sélectionnant 'Workflow de données' dans la section 'Services de données'. Le WSDL généré n'est pas spécifique à une publication de workflow et aucune information n'est requise. Il peut être téléchargé grâce au bouton **Télécharger le WSDL**.

Opérations disponibles pour contrôler un workflow de données

Les opérations suivantes sur les espaces de données sont disponibles en utilisant le WSDL généré :

- Démarrer un workflow
- Réveiller un workflow
- Terminer un workflow

Voir aussi

[*WSDL download using a HTTP request*](#) [p 220]

[*Operations on data workflows*](#) [p 250]

33.4 Générer un WSDL pour un lignage

La génération d'un WSDL pour un lignage est accessible en sélectionnant 'Lignage' dans la section 'Services de données', sous réserve que des profils aient été autorisés par un profil administrateur dans *Administration > Lignage*.

Les WSDL générés pour accéder aux tables sont les mêmes que ceux utilisés pour la [génération d'un WSDL d'accès aux données](#) [p 173].

Les étapes de la génération de ce WSDL sont les suivantes :

1. Sélectionner un rôle ou un utilisateur, dont les permissions seront appliquées. Un rôle ou un utilisateur doit être autorisé à être utilisé pour le lignage par un administrateur.
2. Identifier l'espace de données et le jeu de données ciblés par les opérations.
3. Sélectionner les tables sur lesquelles les opérations sont autorisées, ainsi que les opérations permises.
4. Télécharger le fichier WSDL généré en cliquant sur le bouton **Télécharger le WSDL**.

Voir aussi [Lignage](#) [p 170]

33.5 Générer un WSDL pour gérer l'accès à l'interface utilisateur

Cette action ne peut être effectuée que par un administrateur.

La génération d'un WSDL pour modifier l'accès à l'interface utilisateur est disponible en sélectionnant 'Interface utilisateur' dans la section 'Services de données'.

Opérations disponibles pour gérer l'accès à l'interface utilisateur

- Fermer l'interface utilisateur
- Ouvrir l'interface utilisateur

Voir aussi

[WSDL download using a HTTP request](#) [p 220]

[User interface operations](#) [p 254]

33.6 Générer un WSDL pour modifier l'annuaire par défaut

Cette action ne peut être effectuée que par un administrateur et seulement si l'annuaire par défaut est utilisé.

La génération d'un WSDL pour modifier l'annuaire d'accès est accessible en sélectionnant 'Annuaire' dans la section 'Services de données'.

Opérations disponibles pour modifier l'annuaire par défaut

Les WSDL générés pour accéder aux tables sont les mêmes que ceux utilisés pour la [génération d'un WSDL d'accès aux données](#) [p 173].

Voir aussi

[WSDL download using a HTTP request](#) [p 220]

[Directory services](#) [p 253]

Manuel de référence (en anglais)

Intégration

CHAPITRE 34

Overview of integration and extension

Several service and component APIs allow you to develop custom extensions for EBX5 and integrate it with other systems.

Ce chapitre contient les sections suivantes :

1. [Using EBX5 as a web component](#)
2. [User interface customization](#)
3. [Data services](#)
4. [XML and CSV import/export services](#)
5. [Programmatic services](#)

34.1 Using EBX5 as a web component

It is possible to use EBX5 as a user interface web component, by calling it using the HTTP protocol. Such EBX5 web components can be integrated into any application that is accessible through a [supported web browser](#) [p 343].

A typical use is to integrate EBX5 views into an organization's intranet framework. Web components can also be invoked from the EBX5 user interface using [UI services](#) [p 179].

Voir aussi [Using EBX5 as a web component](#) [p 183]

34.2 User interface customization

User interface services (UI services)

User interface services (UI services) are HTTP resources (HTML pages, Java Servlets, or JavaServer Pages) that are integrated into the EBX5 interface. They allow users to access custom or advanced functionalities. Examples of UI services include:

- Importing data from an external system
- Performing mass updates on a table

Voir aussi [UI service reference page](#) [p 189]

Custom layout

A presentation layer provides the ability to override the default layout of forms in the user interface with highly customized form layouts. For example, it is possible to:

- Define field placement and ordering in an HTML layout
- Seamlessly integrate custom layout elements into the existing EBX5 user interface using the provided Java API for styles and pre-built HTML/Ajax artifacts
- Include auto-generated UI components in forms
- Automatically trigger validation and other standard EBX5 transactions
- Leverage user permissions, as in default forms
- Configure rendering parameters for each form field, such as showing or hiding icons, showing or hiding labels, aligning fields vertically or horizontally
- Rename buttons with custom labels

Voir aussi *UIForm^{API}*

UI beans

UI beans are included in the Java API to allow the development of user interface components for fields or groups of fields. When a field or group declares an associated UI bean in its data model, EBX5 automatically generates the corresponding user interface by which users interact with this element.

UI beans can be used for various purposes, such as:

- Masking characters in password fields
- Incorporating JavaScript UI components

To use a UI bean on a field or group, first extend the *UIBeanEditor^{API}* Java class. Next, declare the UI bean on the element in its data model.

Voir aussi

UIBeanEditor^{API}

[*Properties of data model elements* \[p 50\]](#)

Ajax components

Ajax components allow the asynchronous exchange of data with a server without refreshing the currently displayed page.

By using a class on the server side that inherits *UIAjaxComponent*, it is possible for a UI service, UI bean, or custom layout to communicate with the repository or a server without impacting the display of the existing page. On the client side, a JavaScript implementation sends the parameters to the Ajax component, which then returns the data to be displayed.

Ajax components can be used for a wide range of purposes, including:

- Retrieving related data from the EBX5 repository based on a user selection in a form
- Sending requests an external server

Voir aussi *UIAjaxComponent^{API}*

Specifying UI filters on a table

In addition to the default filters and search panes in the user interface, it is possible to specify additional filters dependent on the structure of the table. For that a specific class must be defined in the definition of the table and must extend `UITableFilter`.

See `UITableFilter`^{API} for more information.

34.3 Data services

The data services module provides a means for external systems to interact with EBX5 using the Web Services Description Language (WSDL) standard.

Voir aussi [Data Services reference page](#) [p 213]

34.4 XML and CSV import/export services

EBX5 includes built-in services for importing data from and export data to XML and CSV formats. Imports and exports for XML and CSV can be performed using the user interface, data services, or the Java API.

Voir aussi

[XML import and export](#) [p 255]

[CSV import and export](#) [p 261]

34.5 Programmatic services

Programmatic services allow the development of Java or JSP applications that interact with EBX5 contexts.

Some examples of programmatic services include:

- Importing data from an external source,
- Exporting data to multiple systems,
- Data historization, launched by a supervisory system
- Optimizing and refactoring data if EBX5 **built-in optimization services** `AdaptationTreeOptimizerSpec`^{API} are not sufficient.

An easy way to use it is through a JSP. This implies that the process integrates login features so that EBX5 can determine whether the user is allowed to access the data space, data set, etc., or not.

Check out the `ProgrammaticService`^{API} Java class for more information.

CHAPITRE 35

Using EBX5 as a web component

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Integrating EBX5 web components into applications](#)
3. [Repository element and scope selection](#)
4. [Request specifications](#)
5. [Example calls to an EBX5 web component](#)

35.1 Overview

EBX5 can be used as a user interface web component, called through the HTTP protocol. An EBX5 web component can be integrated into any application that is accessible through a supported web browser. This method of access offers the major benefits of EBX5, such as user authentication, data validation, and automatic user interface generation, while additionally providing the ability to focus user navigation on specific elements of the repository.

Typical uses of EBX5 web components include integrating them into the intranet frameworks of organizations or into applications that manage the assignment of specific tasks to users.

Voir aussi [Supported web browsers](#) [p 343]

35.2 Integrating EBX5 web components into applications

A web application that calls an EBX5 web component can be:

1. A non-Java application, the most basic being a static HTML page.

In this case, the application must send an HTTP request that follows the EBX5 web component [request specifications](#) [p 185].

2. A Java application, for example:

- A Java web application running on the same application server instance as the EBX5 repository it targets or on a different application server instance.
- An EBX5 [UI service](#) [p 179] or [UI bean](#) [p 180], in which case, the new session will automatically inherit from the parent EBX5 session.

Note

In Java, the recommended method for building HTTP requests that call EBX5 web components is to use the class `UIHttpManagerComponentAPI` in the API.

35.3 Repository element and scope selection

When an EBX5 web component is called, the user must first be authenticated in the newly instantiated HTTP session. The web component then selects a repository element and displays it according to the scope layout parameter defined in the request.

The parameter `firstCallDisplay` may change this automatic display according to its value.

The repository elements that can be selected are as follows:

- Data space or snapshot
- Data set
- Node
- Table or a published view
- Table record

The scope determines how much of the user interface is displayed to the user, thus defining where the user is able to navigate in the session. The default scope that the web component uses depends on the entity or service being selected or invoked by the request.

Voir aussi [Scope](#) [p 187]

Voir aussi [firstCallDisplay](#) [p 187]

35.4 Request specifications

Base URL

In a default deployment, the base URL must be of the following form:

`http://<host>[:<port>]/ebx/`

Note

The base URL must refer to the servlet `FrontServlet`, defined in the deployment descriptor `/WEB-INF/web.xml` of the web application `ebx.war`.

User authentication and session information parameters

Parameter	Description	Required
login and password, or a <i>user directory-specific token</i>	Specifies user authentication properties. If neither a login and password pair nor a user directory-specific token is provided, user will be required to authenticate using through the repository login page. See <code>Directory^{API}</code> for more information.	No
trackingInfo	Specifies the tracking information of the new session. Tracking information is logged in history tables. Additionally, it can be used to programmatically restrict access permissions. See <code>AccessRule^{API}</code> for more information.	No
redirect	The URL to which the user will be redirected at the end of the component session, when they click on the button 'Close'. The close button is always displayed for record selections, but whether or not it is displayed for all other cases must be specified using the parameter <code>closeButton</code> .	No
locale	Specifies the locale to use. Value is either <code>en-US</code> or <code>fr-FR</code> .	No, default is the locale registered for the user.

Entity and service selection parameters

Parameter	Description	Required
branch	Selects the specified data space.	No
version	Selects the specified snapshot.	No
instance	Selects the specified data set. The value must be the reference of a data set that exists in the selected data space or snapshot.	Only if xpath or viewPublication is specified.
viewPublication	<p>Specifies the publication name of the tabular view to apply to the selected content.</p> <p>This publication name is the one declared during the publication of the view. It can be found in the 'Administration' area under Views configuration > Views.</p> <p>All settings of the view, that is, its filters, sort order, and displayed columns, are applied to the result. A data space and a data set must be selected in order for this view to be applied. The target table selection is not necessary, as it can be automatically determined based on the definition of the view. This parameter can be combined with the predicate specified in the xpath parameter as a logical 'AND' operation.</p>	No
xpath	<p>Specifies a node selection in the data set.</p> <p>Value may be a valid absolute path located in the selected data set. The notation must conform to a simplified XPath, with abbreviated syntax.</p> <p>It can also be a predicate surrounded by "[" and "]" if a table can be automatically selected using other web component parameters (for example, viewPublication or workflowView).</p> <p>For XPath syntax, see XPath supported syntax [p 267]</p> <p>See <code>UIHttpManagerComponent.setPredicate^{API}</code> for more information.</p>	No
service	<p>Specifies the service to access.</p> <p>For more information on built-in UI services, see Built-in services [p 199].</p> <p>In the Java API, see <code>ServiceKey^{API}</code> for more information.</p>	No
workflowView	<p>Specifies the workflow section to be selected.</p> <p>See <code>WorkflowView^{API}</code> for more information.</p>	No.
perspectiveName	<p>Specifies the name of the perspective to be selected.</p> <p>Known limitation: the parameter scope is not supported by the perspectives.</p>	No.
perspectiveActionId	<p>Specifies the identifier of the perspective action to be selected.</p> <p>Known limitation: the parameter scope is not supported by the perspectives.</p>	No.

Layout parameters

Parameter	Description	Required
scope	Specifies the scope to be used by the web component. Value can be full, data, dataspace, dataset or node. See <code>UIHttpManagerComponent.Scope^{API}</code> for more information.	No, default will be computed according to target selection.
firstCallDisplay	Specifies which display must be used instead of the one determined by the combination of selection and scope parameter. See <code>FirstCallDisplay^{API}</code> for more information.	No, default will be computed according to the target selection.
closeButton	Specifies how to display the session close button. Value can be logout or cross. See <code>UIHttpManagerComponent.CloseButtonSpec^{API}</code> for more information.	No. If scope is not full, no close button will be displayed by default.
dataSetFeatures	Specifies which features to display in a UI service at the data set level or a form outside of a table. These options pertain only to features in the workspace. It is recommended to use this property with the smallest scope possible, namely dataset or node. Syntax: <code><prefix> ":" <feature> ["," <feature>]*</code> where <ul style="list-style-type: none"> <code><prefix></code> is hide or show, <code><feature></code> is services, title, breadcrumb, save, or revert. For example, hide:title,breadcrumb show:save,revert See <code>UIHttpManagerComponent.DataSetFeatures^{API}</code> for more information.	No.
viewFeatures	Specifies which features to display in a tabular or a hierarchy view (at the table level). These options pertain only to features in the workspace. It is recommended to use this property with the smallest scope possible, namely dataset or node. Syntax: <code><prefix> ":" <feature> ["," <feature>]*</code> where <ul style="list-style-type: none"> <code><prefix></code> is hide or show, <code><feature></code> is create, views, selection, filters, services, refresh, title, or breadcrumb. For example, hide:title,selection show:service,title,breadcrumb See <code>UIHttpManagerComponent.ViewFeatures^{API}</code> for more information.	No.
recordFeatures	Specifies which features must be displayed in a form at the record level.	No.

Parameter	Description	Required
	<p>These options pertain only to features in the workspace. It is recommended to use this property with the smallest scope possible, namely dataset or node.</p> <p>Syntax:</p> <pre><prefix> ":" <feature> [", " <feature>]*</pre> <p>where</p> <ul style="list-style-type: none"> • <prefix> is hide or show, • <feature> is services, title, breadcrumb, save, saveAndClose, close, or revert. <p>For example,</p> <pre>hide:title show:save, saveAndClose, revert</pre> <p>See <code>UIHttpManagerComponent.RecordFeatures^{API}</code> for more information.</p>	
recordsByPage	Specifies the number of records that will be displayed per page in a table view (either tabular or hierarchical).	No.
startWorkItem	<p>Specifies a work item must be automatically taken and started. Value can be true or false.</p> <p>See <code>ServiceKey^{API}</code> for more information.</p>	No. Default value is false, where the target work item state remains unchanged.

35.5 Example calls to an EBX5 web component

Minimal URI:

```
http://localhost:8080/ebx/
```

Logs in as the user 'admin' and selects the 'Reference' data space:

```
http://localhost:8080/ebx/?login=admin&password=admin&branch=Reference
```

Selects the 'Reference' data space and accesses the built-in validation service:

```
http://localhost:8080/ebx/?
login=admin&password=admin&branch=Reference&service=@validation
```

Selects the roles table in the default directory:

```
http://localhost:8080/ebx/?login=admin&password=admin&branch=ebx-
directory&instance=ebx-directory&xpath=/directory/roles
```

Selects the record 'admin' in the default directory:

```
http://localhost:8080/ebx/?login=admin&password=admin&branch=ebx-
directory&instance=ebx-directory&xpath=/directory/user[./login="admin"]
```

Accesses the interface for creating a new user in the default directory:

```
http://localhost:8080/ebx/?login=admin&password=admin&branch=ebx-
directory&instance=ebx-directory&xpath=/directory/user&service=@creation
```

Compares the record 'admin' in the default directory with the record 'jSmith':

Compares the record 'R1' in the data set 'instanceId' in the data space 'Reference' with the record 'R0':

```
http://localhost:8080/ebx/?login=admin&password=admin&branch=ebx-
directory&instance=ebx-directory&xpath=/directory/user[./
login="admin"]&service=@compare&compare.branch=ebx-
directory&compare.instance=ebx-directory&compare.xpath=/directory/user[./
login="jSmith"]
```

CHAPITRE 36

User interface services (UI services)

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [UI services on data sets](#)
3. [Global UI services on all data spaces and snapshots](#)
4. [Integration of UI services with workflows or perspectives](#)

36.1 Overview

A user interface service (UI service) is an HTTP resource, such as a JSP or Java servlet, that is integrated into EBX5. Through UI services, users can perform specific and advanced functions directly from the user interface.

The following types of UI services exist:

- [Built-in UI services](#) [p 199] provided by EBX5,
- Custom UI services on data sets declared in the data model,
- Custom UI services on data spaces or snapshots that are declared in a module and are available to all data spaces or snapshots. These can be disabled on specific data spaces or snapshots using service permissions.

Additionally, to integrate a UI service with a workflow or a perspective, it needs to be declared for this usage. See [Integration of UI services with workflows and perspectives](#) [p 194] in this chapter for more information.

Accessing UI services

Depending on the nature of UI services and their declarations, users can access them in several ways:

- Directly in the EBX5 user interface, from a **Services** or **Actions** menu,
- From a data workflow, in which case the UI service must be declared as an interaction,
- For a perspective, in which case the UI service must be declared in an action menu item,
- From the Java API, using `UIHttpManagerComponent.setServiceAPI`.

36.2 UI services on data sets

UI services can be declared in a specific data model contained in a module, which makes them available to data sets that implement that data model. These UI services can be defined to be executable on the entire data set, or on tables or record selections in the data set. They are launched in the user interface as follows:

- For data sets, from the **Services** menu in the navigation pane.
- For tables, from the **Actions** menu in the workspace.
- For record selections, from the **Actions** menu in the workspace.

Common use cases

Common uses for UI services on data sets, tables, and record selections are:

- Updating a table or a user selection of table records. For example, the field values of a column 'Product price' can be adjusted by a ratio that is specified by the user.
- Importing data from an external source into the current data set.
- Exporting the selected records of a table.
- Implementing specific events in the life cycle of MDM entities. For example, the creation of a product, which impacts several tables, or the "closure" of a product at a given date.
- Displaying statistics on a table.

Declaration and configuration

A UI service on a data set, a table, or a record selection must be declared in the associated data model under the element `xs:complexType/xs:annotation/xs:appinfo/osd:service`.

Note

The data model must be packaged in a module.

Example

The following example declares the UI service 'Distribution' on a table:

```
<xs:complexType name="Distribution">
  <xs:annotation>
    <xs:appinfo>
      <osd:service resourcePath="/Distribution/Distribution.jsp"
        activatedForPaths="/root/Product" orderInMenu="1"/>
    </xs:appinfo>
    <xs:documentation xml:lang="en-US">
      Distribute data for table 'Product'
    </xs:documentation>
  </xs:annotation>
</xs:complexType>
```

Properties

Property	Definition	Mandatory
resourcePath	<p>Attribute that identifies the JSP or servlet resource to be called. The identifier is a path relative to the root of the EBX5 module containing the data model. It must begin with '/'. Voir aussi <i>ServiceContext^{APT}</i> ServletContext.getRequestDispatcher(String)</p>	Yes
activatedForPaths	<p>Attribute that specifies where the UI service will be available. The list of paths is white-space delimited. Each path must be the absolute XPath notation of the node. It is possible to activate the UI service on nodes or on data sets. The path '/' activates the UI service globally on the data set. On <code>osd:table</code>, an additional syntax allows you to specify whether the UI service is activated globally on the table or on record selections. If the path to the table is <code>/root/myTable</code>:</p> <ul style="list-style-type: none"> <code>/root/myTable</code> activates the UI service globally on the table. <code>/root/myTable{n}</code> activates the UI service only if exactly 'n' records are selected, where 'n' is a positive integer. <code>/root/myTable{+}</code> activates the UI service if one or more records are selected (an unbounded selection). 	No. If not defined, activates as a global UI service that is not attached to a particular node.
activatedForPathRegex	<p>Attribute that specifies where the UI service will be available based on a regular expression. The UI service will be available on all nodes whose full path in the data model match the regular expression. The format of the regular expression is defined in the Java specification.</p>	No. When set, it is exclusive with <code>activatedForPaths</code> attribute.
class	<p>Attribute that specifies a Java class that implements <code>ServicePermission^{APT}</code>.</p>	No. If not defined, service has no permission restriction.
osd:confirmation	<p>Element that specifies a confirmation message or that suppresses user confirmation before the UI service is launched. See Confirmation message [p 191] below.</p>	No. If not defined, a default confirmation message is displayed.
displayInMenu	<p>Attribute that specifies if the service is displayed in UI menus and buttons.</p>	No. Default value is true.
orderInMenu	<p>Attribute that specifies the position of this UI service among the UI services defined in the schema. This position is used for the display order of UI services.</p>	No

Confirmation messages

By default, the user is shown a generic message and must confirm the execution of the UI service. The element `osd:confirmation` allows you to specify a custom localized confirmation message:

```
<osd:service...>
<osd:confirmation>
  <osd:label xml:lang="fr-FR">
    Voulez-vous lancer le service ?
  </osd:label>
  <osd:label xml:lang="en-US">
```

```

Do you want to launch the service ?
</osd:label>
</osd:confirmation>
</osd:service>

```

The element `osd:confirmation` can also be used to disable the user confirmation entirely with the attribute `disable="true"`.

36.3 Global UI services on all data spaces and snapshots

A UI service can also be declared in such a way that they are available to all data spaces and/or snapshots in the repository. The purpose of such services is to write high-level core business procedures that involve actions such as merges, imports and exports, and validations.

Common use cases

Common uses for UI services on data spaces and snapshots are:

- Importing data from an external source.
- Exporting data to multiple systems.
- Validating a data space or snapshot before exporting it
- Sending messages to monitoring systems before performing a merge

Declaration and configuration

The Java applications, JSPs, and servlets for UI services on data spaces or snapshots must be declared in a module. It is recommended to create a dedicated module for UI services on data spaces and snapshots; these UI services will be available on all data spaces and snapshots, but may have a life cycle that is independent of the data life cycle.

UI services on data spaces and snapshots must be declared in the module configuration file `module.xml`, under the element `services/service`, where `services` is the root of all UI services declared in the module.

Voir aussi [Packaging EBX5 modules](#) [p 453]

Example

The following example declares a UI service that validates and merges a data space:

```

<service name="AdvancedMerge">
  <resourcePath>/services/advancedMerge.jsp</resourcePath>
  <type>branch version</type>
  <documentation xml:lang="en-US">
    <label>Merge/Validate</label>
    <description>Merge current branch to parent if valid.</description>
  </documentation>
  <confirmation>
    <label>A default confirmation message.</label>
    <label xml:lang="en-US">An English confirmation message.</label>
    <label xml:lang="fr-FR">Un message de confirmation en français.</label>
  </confirmation>
</service>

```


Properties

Property	Definition	Mandatory
name	Attribute that specifies the name of the UI service. This name must be unique within the scope of a module.	Yes
resourcePath	<p>Element that identifies the JSP or servlet resource to be called. The identifier is a path relative to the root of the EBX5 module containing the schema. It must begin with '/'.</p> <p>Voir aussi</p> <p><i>ServiceContext</i>^{APT}</p> <p>ServletContext.getRequestDispatcher(String)</p>	Yes
type	<p>Element that defines the availability of the UI service as a whitespace-delimited list.</p> <ul style="list-style-type: none"> 'branch', which makes the UI service available to <i>data spaces</i> 'version', which makes the UI service available to <i>snapshots</i> 'workflow', which makes the UI service available to be defined for user tasks in workflow models 'perspective', which makes the UI service available to be defined for action menu items in perspective configurations. <p>If the list includes neither 'branch' or 'version', the UI service will not be available in all Services menus.</p> <p>If the value 'workflow' is specified, the UI service will be available for workflows on any data space or snapshot.</p> <p>If the value 'perspective' is specified, the UI service will be available for perspectives on any data space or snapshot.</p> <p>See Services on data spaces or snapshots for data workflows or perspectives [p 195]</p>	Yes
documentation	Localized label and description of the UI service, displayed in the menu where the UI service appears.	No
confirmation	Element that specifies a confirmation message or that suppresses user confirmation before the UI service is launched. See Confirmation message [p 193] below.	No. If not defined, a default confirmation message is displayed.

Confirmation Message

By default, the user is shown a generic message and must confirm the execution of the UI service. The element `confirmation` allows you to specify a custom localized confirmation message:

```
<service name="AdvancedMerge">
  <resourcePath>/services/advancedMerge.jsp</resourcePath>
  <type>branch version</type>
  <documentation xml:lang="en-US">
    <label>Merge/Validate</label>
    <description>Merge current branch to parent if valid.</description>
  </documentation>
  <confirmation disable="true"/>
</service>
```

The element confirmation can also be used to disable the user confirmation entirely with the attribute `disable="true"`.

36.4 Integration of UI services with workflows or perspectives

Overview

In order to make a UI service available to workflows or perspectives, an additional declaration must be made in the configuration file `module.xml` of the module that implements the UI service.

UI services that are integrated into workflows or perspectives benefit from a declarative and secure specification for a "component-oriented" approach

Workflow

Once this declaration has been made, the UI service becomes available in the Workflow Models area where the user defines user tasks. The user then maps the input parameters and output parameters that are specified in the module configuration file `module.xml`.

Once the workflow model has been published and a workflow has been started, for every user starting a work item, the corresponding user session on the application server will be associated with a **persistent interaction object** `Session.getInteractionAPI` that contains the valued input parameters. Through the Java API, the running UI service can access these input parameters and adapt its behavior accordingly. Once the corresponding work item and associated UI services are complete, the completion method must be invoked with the output parameters specifying the resulting state of the work item. For more information, the developer of the UI service should read the JavaDoc of the interface `SessionInteractionAPI`.

Perspective

Once this declaration has been made, the UI service becomes available for action menu items. The perspective administrator then maps the input parameters that are specified in the module configuration file `module.xml`. Output parameters are not used by perspectives but can be defined in `module.xml` if the UI service is also used for workflows.

Common properties for defining input and output parameters

Parameter declarations in the module configuration file `module.xml` are made under the element `services`. The element `properties` is used for declaring the input and output parameters of the UI service. These parameters will be available for the definition of user tasks that use the UI service.

For instance, the following sample specifies a service on a branch with one input parameter named `branch` and one output parameter named `choice`:

```
<properties>
  <property name="branch" input="true">
    <documentation xml:lang="en-US">
      <label>Branch</label>
      <description>
        This input parameter indicate the branch to work on.
      </description>
    </documentation>
  </property>
  <property name="choice" output="true" />
</properties>
```

The following table summarizes the XML tags to use for defining input and output parameters:

Property	Definition	Mandatory
properties	Element containing property declaration.	Yes. This element is unique for a service.
name	This attribute specifies the name of the property.	Yes
input	This attribute specifies if the property is an input one.	No
output	This attribute specifies if the property is an output one. This property is ignored by perspectives.	No
documentation	Label and description of the property.	No

Services on data spaces or snapshots for data workflows or perspectives

The declaration of an UI service on data spaces and/or snapshots can indicate if the UI service can also be used with workflows or perspectives.

If the type element contains the value `workflow` in the whitespace-delimited list, the UI service will be available to the definition of user tasks in workflow models.

If the type element contains the value `perspective` in the whitespace-delimited list, the UI service will be available for definition of action menu items in perspective configurations.

For UI services on data spaces in workflows or perspectives, an element `properties/property`, with attributes `name="branch" input="true"`, is also required. Similarly, for UI services on snapshots in workflows or perspectives, an element `properties/property`, with attributes `name="version" input="true"`, is required.

Example

This example declares a UI services that is available in the **Services** menu for data spaces, for user tasks in workflow models and for action menu items in perspective configurations. Note the mandatory inclusion of the element `property` with the attributes `name="branch" input="true"`.

```
<service name="AdvancedMerge">
  <resourcePath>/services/advancedMerge.jsp</resourcePath>
  <type>branch workflow perspective</type>
  <documentation xml:lang="en-US">
    ...Merge a branch to its parent if valid...
  </documentation>
  <properties>
    <property name="branch" input="true"/>
  </properties>
</service>
```

Service on data sets for data workflows or perspectives

UI services on data sets are declared at the data model-level. However, in order to make them definable for workflow user tasks or perspective action menu items, their declarations must include additional elements in the module configuration file `module.xml` of the module containing the data model.

Under the element `services/serviceLink`, several properties must be defined. The built-in parameters `branch` and `instance` are required, which respectively identify the data space and contained data set on which the UI service will be run.

Example

The following example declares a data set service that imports a spreadsheet into a table in a data set.

```
<serviceLink serviceName="ImportSpreadsheet">
  <importFromSchema>/WEB-INF/ebx/schema/my-ebx.xsd</importFromSchema>
  <properties>
    <property name="branch" input="true"/>
    <property name="instance" input="true"/>
    <property name="pathToTable" input="true"/>
  </properties>
</serviceLink>
```

Properties

The following table summarizes the elements and attributes under the element `services/serviceLink`:

Property	Definition	Mandatory
<code>serviceName</code>	Attribute of <code>serviceLink</code> that defines the name of the UI service as it is specified in the data model.	Yes
<code>importFromSchema</code>	Element that specifies the path to the data model, relative to the current module (web application).	Yes
<code>properties</code>	Element that defines input and output parameters.	Yes. At least the input properties for "branch" and "instance" must be defined.

Service extensions

It is possible to extend UI services that have already been declared, by defining labels and descriptions and adding properties. You can extend built-in UI services, data space or snapshot UI services and data set UI services. However, it is not possible to further extend a service extension.

Service extensions must be declared in the module configuration file `module.xml`, under the element `services/serviceExtension`.

Extending a built-in UI service

```
<serviceExtension name="BuiltinCreationServiceExtension" extends="@creation" >
  <documentation xml:lang="en-US">
    <label>Built in creation service extension</label>
  </documentation>
  <properties>
    ...
  </properties>
```

```
</serviceExtension>
```

Property	Definition	Mandatory
name	Attribute that specifies the name of the service extension.	Yes
extends	Attribute that specifies the name of the built-in UI service being extended. The value is the ServiceKey ^{APT} of the built-in UI service. For the default built-in UI service 'Access data', this attribute must be empty.	Yes
properties	Element that declares properties being added to the built-in UI service.	No
documentation	Localized labels and descriptions to add to the built-in UI service.	No

Extending data space and snapshot UI services

The service extension and the UI service being extended must be defined in the same `module.xml` module configuration file.

```
<serviceExtension name="AdvancedMergeExtension" extends="AdvancedMerge">
  <documentation xml:lang="en-US">
    ...
  </documentation>
  <properties>
    ...
  </properties>
</serviceExtension>
```

Property	Definition	Mandatory
name	Attribute that specifies the name of the service extension.	Yes
extends	Attribute that specifies the name of the UI service being extended.	Yes
properties	Element that declares properties being added to the UI service.	No
documentation	Localized labels and descriptions to add to the UI service.	No

Extending data set UI services

Since the service extension and the UI service being extended must be defined in the same `module.xml` module configuration file, you must first declare the UI service as an interaction using element `serviceLink`.

```
<serviceExtension name="ImportSpreadsheetExtension" extends="ImportSpreadsheet" fromSchema="/WEB-INF/ebx/schema/
my-ebx.xsd">
  <documentation xml:lang="en-US">
    ...
  </documentation>
</serviceExtension>
```

```

</documentation>
<properties>
...
</properties>
</serviceExtension>

```

Property	Definition	Mandatory
name	Attribute that specifies the name for the service extension.	Yes
extends	Attribute that specifies the name of the UI service being extended.	Yes
fromSchema	Attribute that specifies the path to the schema that defines the UI service.	Yes
properties	Element that declares properties being added to the UI service.	No
documentation	Localized labels and descriptions to add to the UI service.	No

CHAPITRE 37

Built-in UI services

EBX5 includes a number of built-in UI services. Built-in UI services can be used:

- when defining [workflow model tasks](#) [p 132]
- when defining [perspective action menu items](#) [p 16]
- as [extended UI services](#) [p 196] when used with [service extensions](#) [p 196]
- when using EBX5 as a [web component](#) [p 183]

This reference page describes the built-in UI services and their parameters.

Ce chapitre contient les sections suivantes :

1. [Access a data space](#)
2. [Access data \(default service\)](#)
3. [Access the data space merge view](#)
4. [Compare contents](#)
5. [Create a new record](#)
6. [Data workflows](#)
7. [Duplicate a record](#)
8. [Export data to a CSV file](#)
9. [Export data to an XML file](#)
10. [Import data from a CSV file](#)
11. [Import data from an XML file](#)
12. [Merge a data space](#)
13. [Validate a data space, a snapshot or a data set](#)

37.1 Access a data space

A workflow automatically considers that the data space selection service is complete.

Service name parameter: `service=@selectDataSpace`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
version	Snapshot	The identifier of the specified snapshot - A data space is required for this service.

37.2 Access data (default service)

The default service. By default, workflows automatically consider this service as complete. That is, the 'Accept' button is always available.

This is the default service used if no service is specified.

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
disableAutoComplete	Disable Accept at start	By default, the interaction associated with this service is directly considered as complete. Therefore, the Accept button is automatically displayed at the opening of the work item. This parameter is useful to disable this behavior. If the value is 'true', the developer will be in charge of completing the interaction by using SessionInteraction in a UI service or a trigger, for example. The default value is 'false'. Perspectives do not use this parameter.
instance	Data set	The value must be the reference of a data set that exists in the selected data space.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
version	Snapshot	The identifier of the specified snapshot - A data space is required for this service.
viewPublication	View	The publication name of the view to display. The view must be configured for the selected table.
xpath	Data set node (XPath)	The value must be a valid absolute location path in the selected data set. The notation must conform to a simplified XPath, in its abbreviated syntax.

37.3 Access the data space merge view

The merge.view service is automatically considered complete.

Service name parameter: `service=@merge.view`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.

37.4 Compare contents

Workflows automatically consider the compare service as complete.

Service name parameter: `service=@compare`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space or snapshot and a data space or snapshot to compare to are required for this service.
compare.branch	Data space to compare	The identifier of the data space to compare - A data space or snapshot and a data space or snapshot to compare to are required for this service.
compare.filter	Comparison filter	To ignore inheritance and computed values fields in the comparison (disable resolved mode), the filter "persistedValuesOnly" must be specified. By default, when no filter is defined, the comparison uses resolved mode.
compare.instance	Data set to compare	The value must be the reference of a data set that exists in the selected data space to compare.
compare.version	Snapshot to compare	The identifier of the snapshot to compare - A data space or snapshot and a data space or snapshot to compare to are required for this service.
compare.xpath	Table or record to compare (XPath)	The value must be a valid absolute location path of a table or a record in the selected data set to compare. The notation must conform to a simplified XPath, in its abbreviated syntax.
instance	Data set	The value must be the reference of a data set that exists in the selected data space.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
version	Snapshot	The identifier of the specified snapshot - A data space or snapshot and a data space or snapshot to compare to are required for this service.
xpath	Table or record (XPath)	The value must be a valid absolute location path of a table or a record in the selected data set. The notation must

Parameter	Label	Description
		conform to a simplified XPath, in its abbreviated syntax.

37.5 Create a new record

For a workflow, the creation service is considered complete when the first successful submit is performed (record has been created). If this service is called whereas it is already complete, the created record is displayed in update or read-only mode (depending on the user rights).

Service name parameter: `service=@creation`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - This field is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space - This field is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
xpath	Data set table (XPath)	The value must be a valid absolute location path of a table in the selected data set. The notation must conform to a simplified XPath, in its abbreviated syntax - This field is required for this service.

Output parameters

Parameter	Label	Description
created	Created record	Contains the XPath of the created record.

37.6 Data workflows

This service provides access to the data workflows user interfaces.

Service name parameter: `service=@workflow`

Note

This service is for perspective only.

Input parameters

Parameter	Label	Description
scope	Scope	Defines the scope of the user navigation for this service.
workflowView	Workflow view	Specifies the workflow view type. Value can be one of the following: "inbox", "launcher", "monitoringPublications", "monitoringWorkflows", "monitoringWorkItems" or "completedWorkflows".
xpath	Filter (XPath)	An optional filter. The syntax should conform to a XPath predicate surrounded by "[" and "]".

37.7 Duplicate a record

For a workflow, the duplicate service is considered complete when the first successful submit is performed (record has been created). If this service is called whereas it is already complete, the created record is displayed in update or read-only mode (depending on the user rights).

Service name parameter: `service=@duplicate`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - This field is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space - This field is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
xpath	Record to duplicate (XPath)	The value must be a valid absolute location path of an existing record. The notation must conform to a simplified XPath, in its abbreviated syntax - This field is required for this service.

Output parameters

Parameter	Label	Description
created	Created record	Contains the XPath of the created record.

37.8 Export data to a CSV file

Workflows consider the exportToCSV service as complete when export is done and file downloaded.

Service name parameter: `service=@exportToCSV`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space - This field is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
version	Snapshot	The identifier of the specified snapshot - A data space is required for this service.
xpath	Data set table to export (XPath)	The value must be a valid absolute location path of a table in the selected data set. The notation must conform to a simplified XPath, in its abbreviated syntax - This field is required for this service.

37.9 Export data to an XML file

The exportToXML service is considered complete when export is done and file downloaded.

Service name parameter: service=@exportToXML

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space - This field is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
version	Snapshot	The identifier of the specified snapshot - A data space is required for this service.
xpath	Data set table to export (XPath)	The value must be a valid absolute location path of a table in the selected data set. The notation must conform to a simplified XPath, in its abbreviated syntax - This field is required for this service.

37.10 Import data from a CSV file

Workflows consider the importFromCSV service as complete when import is performed.

Service name parameter: service=@importFromCSV

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space - This field is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
xpath	Data set table to import (XPath)	The value must be a valid absolute location path of a table in the selected data set. The notation must conform to a simplified XPath, in its abbreviated syntax - This field is required for this service.

37.11 Import data from an XML file

Workflows consider the importFromXML service as complete when import is performed.

Service name parameter: service=@importFromXML

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space - This field is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
xpath	Data set table to import (XPath)	The value must be a valid absolute location path of a table in the selected data set. The notation must conform to a simplified XPath, in its abbreviated syntax - This field is required for this service.

37.12 Merge a data space

Workflows consider the merge service as complete when merger is performed and data space is closed.

Service name parameter: `service=@merge`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space is required for this service.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.

Output parameters

Parameter	Label	Description
mergeResult	Merge success	Contains 'true' if merge succeeded, otherwise 'false'.
mergeState	Merge state	Contains the return code of the merge. It is strongly recommended to parse this value by using the InteractionMergeState UIHttpManagerComponentReturnCode.

37.13 Validate a data space, a snapshot or a data set

Workflows automatically consider the validation service as complete.

Service name parameter: `service=@validation`

Input parameters

Parameter	Label	Description
branch	Data space	The identifier of the specified data space - A data space or snapshot is required for this service.
instance	Data set	The value must be the reference of a data set that exists in the selected data space.
scope	Scope	Defines the scope of the user navigation for this service, namely, the entities that the user is able to select during their session. If left blank, the default value will be used. For perspectives, the default value is always 'node'. For workflows, the default value depends on the selected entities or service.
trackingInfo	Tracking information	Tracking information is logged into 'history' logs. It may also be used for any other purpose like access control or additional export information.
version	Snapshot	The identifier of the specified snapshot - A data space or snapshot is required for this service.

Output parameters

Parameter	Label	Description
hasError	Found errors	Contains 'true' if validation has produced errors.
hasFatal	Found fatal errors	Contains 'true' if validation has produced fatal errors.
hasInfo	Found informations	Contains 'true' if validation has produced informations.
hasWarning	Found warnings	Contains 'true' if validation has produced warnings.

CHAPITRE 38

Introduction

Ce chapitre contient les sections suivantes :

1. [Overview of data services](#)
2. [Enabling and configuring data services](#)
3. [SOAP interactions](#)
4. [Data services security](#)
5. [Monitoring](#)
6. [Known limitations](#)

38.1 Overview of data services

Data services allow external systems to interact with the data governed in the EBX5 repository using the SOAP and Web Services Description Language (WSDL) standards.

A number of WSDLs can be dynamically generated from data models, then used to perform operations such as:

- Selecting, inserting, updating, deleting, or counting records
- Getting the differences on a table between data spaces or snapshots, or between two data sets based on the same data model
- Getting the credentials of records

Other generic operations for:

- Creating, merging, or closing a data space
- Creating or closing a snapshot
- Validating a data set, data space, or a snapshot
- Starting, resuming or ending a data workflow

38.2 Enabling and configuring data services

Data services are enabled by deploying the module `ebx-dataservices` along with the other EBX5 modules. See [Java EE deployment overview](#) [p 347] for more information.

For specific deployment, for example using reverse-proxy mode, the URL to `ebx-dataservices` must be configured through lineage administration.

The default method for accessing data services is over HTTP, although it is also possible to use JMS. See [JMS configuration](#) [p 368] and [Using JMS](#) [p 214] for more information.

38.3 SOAP interactions

Input and output message encoding

All input messages must be *exclusively* in UTF-8. All output messages are in UTF-8.

Tracking information

Depending on the data services operation being called, it may be possible to specify session tracking information in an optional SOAP header. For example:

```
<SOAP-ENV:Header>
<!-- optional security header here -->
<m:session xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <trackingInformation>String</trackingInformation>
</m:session>
</SOAP-ENV:Header>
```

For more information, see `Session.getTrackingInfoAPI` in the Java API.

Exceptions handling

Exceptions are re-thrown to the consumer through the `soap:fault` element within a standard exception. For example:

```
<soapenv:Fault>
  <faultcode>soapenv:java.lang.IllegalArgumentException</faultcode>
  <faultstring />
  <faultactor>admin</faultactor>
  <detail>
    <m:StandardException xmlns:m="urn:ebx-schemas:dataservices_1.0">
      <code>java.lang.IllegalArgumentException</code>
      <label/>
      <description>java.lang.IllegalArgumentException:
        Parent home not found at
        com.orchestranetworks.XX.YY.ZZ.AA.BB(AA.java:44) at
        com.orchestranetworks.XX.YY.ZZ.CC.DD(CC.java:40) ...
      </description>
    </m:StandardException>
  </detail>
</soapenv:Fault>
```

Using JMS

It is possible to access data services using JMS, instead of HTTP. The JMS architecture relies on one mandatory queue and one optional queue, see configuration [JMS](#) [p 368]. The mandatory queue is the input queue. Request messages must be put in the input queue, and response messages are put by EBX5 in the `replyTo` queue of the JMS request. The optional queue is the failure queue which allows you to replay an input message if necessary. If the queue is set in the configuration file and an exception occurs while handling a request message, this input message will be copied in the failure queue.

The relationship between a request and a response is made by copying the `messageId` message identifier field of the JMS request into the `correlId` correlation identifier field of the response.

JMS endpoints must be defined in the Lineage administration to provide them in the WSDL. If no specific endpoint is given, the default value will be `jms:queue:jms/EBX_QueueIn`.

38.4 Data services security

Authentication

Authentication is mandatory. Several authentication methods are available and described below. The descriptions are ordered by priority (EBX5 applies the highest priority authentication method first).

- Specific authentication based on HTTP Request see `Directory.authenticateUserFromHttpRequest`^{APT} for more information. An implementation of this method can, for example, extract a password-digest or a ticket from the HTTP request.
- 'Basic Authentication Scheme' method based on the HTTP-Header Authorization in base 64, as described in [RFC 2324](#).

If the user agent wishes to send the userid "Alibaba" and password "open sesame", it will use the following header field: Authorization: Basic QWxpcYmFiYTpvGvUHNlc2FtZQ==

- A simple authentication based on the specification [Web Services Security UsernameToken Profile 1.0](#).

Only the mode PasswordText is supported. This is done with the following SOAP header defined in the WSDL:

```
<SOAP-ENV:Header>
  <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
    <wsse:UsernameToken>
      <wsse:Username>String</wsse:Username>
      <wsse:Password Type="wsse:PasswordText">String</wsse:Password>
    </wsse:UsernameToken>
  </wsse:Security>
</SOAP-ENV:Header>
```

Note

Only available for [Operations](#) [p 227] from HTTP request.

- Standard authentication based on HTTP Request. User and password are extracted from request parameters. For more information, see [Request parameters](#) [p 221].
See `Directory.authenticateUserFromLoginPassword`^{APT} for more information.

Note

Only available for [WSDL generation](#) [p 219].

Overriding the SOAP security header

It is possible to override the default WSS header in order to define another security authentication mechanism. Such an override is taken into account for both HTTP and JMS. To define and override,

use the 'SOAP Header Security declaration' configuration settings under Administration > Lineage, which includes the following fields:

Schema location	The URI of the Security XML Schema to import into the WSDL.
Target namespace	The target namespace of elements in the schema.
Namespace prefix	The prefix for the target namespace.
Message name	The message name to use in the WSDL.
Root element name	The root element name of the security header. The name must be the same as the one declared in the schema.
wSDL:part element name	The name of the wSDL:part of the message.

The purpose of overriding the default security header is to change the declaration of the WSDL message matching the security header so that it contains the following:

```
<wSDL:definitions ... xmlns:MyPrefix="MyTargetNameSpace" ...
...
<xs:schema ...>
  <xs:import namespace="MyTargetNameSpace" schemaLocation="MySchemaURI"/>
  ...
</xs:schema>
...
<wSDL:message name="MySecurityMessage">
  <wSDL:part name="MyPartElementName" element="MyPrefix:MySecurityRootElement"/>
</wSDL:message>
...
<wSDL:operation name="...">
  <soap:operation soapAction="..." style="document"/>
  <wSDL:input>
    <soap:body use="literal"/>
    <soap:header message="impl:MySecurityMessage" part="MyPartElementName" use="literal"/>
  ...
</wSDL:operation>
</wSDL:definitions>
```

The corresponding XML Schema header declaration would be as follows:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="MyNameSpace"
  xmlns:MyPrefix="MyNameSpace">
  <element name="MySecurityRootElement" type="MyPrefix:SpecificSecurity"/>
  <complexType name="SpecificSecurity">
    <sequence>
      <element name="AuthToken" type="string"/>
    </sequence>
  </complexType>
</schema>
```

A SOAP message using the XML schema and configuration above would have the following header:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Header>
    <m:MySecurityRootElement xmlns:m="MyNameSpace">
      <AuthToken>String</AuthToken>
    </m:MySecurityRootElement>
    ...
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    ...
  </SOAP-ENV:Body>
```



```
</SOAP-ENV:Envelope>
```

To handle this non-default header, you must implement the method: `Directory.authenticateUserFromSOAPHeaderAPI`.

Note

Only available for [Operations](#) [p 227].

Lookup mechanism

Using HTTP, the web service connector attempts authentication in the following order:

1. Using an HTTP Request
2. Using the HTTP Header Authorization
3. Looking for a security header (WSS or custom)

When using JMS, the authentication process only looks for a security header (WSS or custom).

38.5 Monitoring

Data service events can be monitored through the log category `ebx.dataServices`, as declared in the EBX5 main configuration file. For example, `ebx.log4j.category.log.dataServices= INFO, ebxFile:dataservices`.

Voir aussi

[Configuring the EBX5 logs](#) [p 366]

[EBX5 main configuration file](#) [p 361]

38.6 Known limitations

Naming convention

Due to the naming convention of the data service operations, each table defined within a data model must have a unique name for WSDL generation.

Date, time & dateTime format

Data services only support the following date and time formats:

Type	Format	Example
xs:date	yyyy-MM-dd	2007-12-31
xs:time	HH:mm:ss or HH:mm:ss.SSS	11:55:00
xs:dateTime	yyyy-MM-ddTHH:mm:ss or yyyy-MM-ddTHH:mm:ss.SSS	2007-12-31T11:55:00

CHAPITRE 39

WSDL generation

Ce chapitre contient les sections suivantes :

1. [Supported standard](#)
2. [Operation types](#)
3. [Supported access methods](#)
4. [WSDL download from the data services user interfaces](#)
5. [WSDL download using a HTTP request](#)

39.1 Supported standard

EBX5 generates a WSDL that complies with the [W3C Web Services Description Language 1.1](#) standard.

39.2 Operation types

A WSDL can be generated for different types of operations:

Operation type	WSDL description
custom	WSDL for EBX5 add-ons.
dataset	WSDL for data set and replication operations.
directory	WSDL for default EBX5 directory operations. It is also possible to filter data using the tablePaths [p 222] or operations [p 222] parameters.
repository	WSDL for data space or snapshot management operations.
tables	WSDL for operations on the tables of a specific data set.
userInterface	WSDL for user interface management operations (these operations can only be accessed by administrators).
workflow	WSDL for EBX5 workflow management operations.

39.3 Supported access methods

EBX5 supports the following downloading methods:

- From the data services user interface
- From an HTTP request

Global access permissions can be independently defined for each method. For more information see [Global permissions](#) [p 392].

A WSDL can only be downloaded by authorized profiles:

Operation type	Access right permissions
custom	All profiles, if at least one web service is registered.
dataset	All profiles.
directory	All profiles, if the following conditions are valid: <ul style="list-style-type: none"> • No specific directory implementation is used. (The built-in Administrator role is only subject to this condition). • Global access permissions are defined for the administration. • 'Directory' data set permissions have writing access for the current profile.
repository	All profiles.
tables	All profiles.
userInterface	Built-in Administrator role or delegated administrator profiles, if all conditions are valid: <ul style="list-style-type: none"> • Global access permissions are defined for the administration. • 'User interface' data set permissions have writing access for the current profile.
workflow	All profiles.

39.4 WSDL download from the data services user interfaces

An authorized user can download an EBX5 WSDL from the data services administration area.

Note

See [generating data service WSDLs](#) [p 173] in the user guide for more information.

39.5 WSDL download using a HTTP request

An application can download an EBX5 WSDL using an HTTP GET or POST request. The application has to be authenticated using a profile with appropriate rights.

Request details

- HTTP(S) URL format:

`http[s]://<host>[:<port>]/ebx-dataservices/<pathInfo>?<key - values>`

Both <pathInfo> and <key - values> are mandatory. For more information on possible values see: [Request parameters](#) [p 221]. An error is returned for incorrect parameters.

- HTTP(S) response status codes:

Status code	Information
200	The WSDL content was successfully generated and is returned by the request (optionally in an attachment [p 223]).
400	Bad request: request argument is incorrect.
401	Unauthorized: request requires an authenticated user.
403	Forbidden: request is not allowed for the authenticated user.
405	Method not allowed: request is not allowed in this configuration.
500	An internal error occurred: request generates an error (a stack trace and a detailed error message are returned).

Note

A detailed error message is returned for the HTTP response with status code 4xx.

- HTTP(S) parameters size restrictions:

Request type	Information
GET	2048 octets or more (HTTP Protocol Parameters). <i>Note: Servers ought to be cautious about depending on URI lengths above 255 bytes, because some older client or proxy implementations might not properly support these lengths.</i>
POST	2 mega octets or more (depending on the servlet/JSP container). Each key - value parameter is limited to 1024 characters.

Request parameters

A request parameter can be specified by one of the following methods:

- a path info on the URL (recommended)
- key values in a standard HTTP parameter.

For more detail, refer to the following table (some parameters do not have a path info representation):

Parameter name	As path info	As key - values	Required	Description
WSDL	no	yes	yes	Used to indicate the WSDL download. Empty value.
login	no	yes	no	A user identifier. Required when the standard authentication method is used. String type value.
password	no	yes	no	A password. Required when the standard authentication method is used. String type value.
type	yes	no	yes	An operation type [p 219]. Possible values are: custom, dataset, directory, userInterface, repository, tables or workflow. String type value.
branch version	yes	yes	(*)	A data space or a snapshot identifier. (*) required for tables and dataset types, otherwise ignored. String type value.
instance	yes	yes	(*)	A data set identifier. String type value.
tablePaths	no	yes	no	A list of table paths. Optional for tables or directory types, otherwise ignored. If not defined, all tables are selected. Each table path is separated by a comma character. String type value.
operations	no	yes	no	Allows generating a WSDL for a subset of operations. Optional for tables or directory operation types, otherwise ignored. If not defined, all operations for the given type are generated. This parameter's value is a concatenation of one or more of the following characters: <ul style="list-style-type: none"> • C = Count record(s) • D = Delete record(s) • E = Get credentials • G = Get changes

Parameter name	As path info	As key - values	Required	Description
				<ul style="list-style-type: none"> • I = Insert record(s) • U = Update record(s) • R = Read operations (equivalent to CEGS) • S = Select record(s) • W = Write operations (equivalent to DIU) String type value.
namespaceURI	yes	yes	(**)	<p>Unique name space URI of the custom web service.</p> <p>(**)Is required when type parameter is set to custom types. Otherwise is ignored.</p> <p>URI type value.</p>
attachmentFilename	no	yes	(***)	<p>The attachment file name.</p> <p>(***) optional if isContentInAttachment parameter is defined and set to true. Otherwise is ignored.</p> <p>String type value.</p>
isContentInAttachment	no	yes	no	<p>If value is true, the WSDL is downloaded as an attachment.</p> <p>Boolean type value.</p> <p>Default value is false.</p>

Request examples

Some of the following examples are displayed in two formats: *path info* and *key - values*.

- The WSDL will contain all repository operations.

```
http[s]://<host>[:<port>]/ebx-dataservices/repository?
WSDL&login=<login>&password=<password>
```

- The WSDL will contain all workflow operations.

```
http[s]://<host>[:<port>]/ebx-dataservices/workflow?
WSDL&login=<login>&password=<password>
```

- The WSDL will contain all tables operations for the data set 'dataset1' in data space 'dataspace1'.

Path info

```
http[s]://<host>[:<port>]/ebx-dataservices/tables/dataspace1/dataset1?
WSDL&login=<login>&password=<password>
```

Key - values

```
http[s]://<host>[:<port>]/ebx-dataservices/tables?
WSDL&login=<login>&password=<password>&
branch=<dataspace1>&instance=<dataset1>
```

- The WSDL will contain all tables with only readable operations for the data set 'dataset1' in data space 'dataspace1'.

Path info

```
http[s]://<host>[:<port>]/ebx-dataservices/tables/dataspace1/dataset1?
WSDL&login=<login>&password=<password>&operations=R
```

Key - values

```
http[s]://<host>[:<port>]/ebx-dataservices/tables?
WSDL&login=<login>&password=<password>&
branch=dataspace1&instance=dataset1&operations=R
```

- The WSDL will contain two selected tables operations for the data set 'dataset1' in data space 'dataspace1'.

Path info

```
http[s]://<host>[:<port>]/ebx-dataservices/tables/dataspace1/dataset1?
WSDL&login=<login>&password=<password>&tablePaths=/root/table1,/root/
table2
```

Key - values

```
http[s]://<host>[:<port>]/ebx-dataservices/tables?
WSDL&login=<login>&password=<password>&
branch=dataspace1&instance=dataset1&tablePaths=/root/table1,/root/table2
```

- The WSDL will contain custom web service operations for the dedicated URI.

Path info

```
http[s]://<host>[:<port>]/ebx-dataservices/custom/urn:ebx-
test:com.orchestranetworks.dataservices.WSDemo?
WSDL&login=<login>&password=<password>
```


Key - values

```
http[s]://<host>[:<port>]/ebx-dataservices/custom?  
WSDL&login=<login>&password=<password>&namespaceURI=urn:ebx-  
test:com.orchestranetworks.dataservices.WSDemo
```


CHAPITRE 40

Operations

Ce chapitre contient les sections suivantes :

1. [Operations generated from a data model](#)
2. [Operations on data sets and data spaces](#)
3. [Operations on data workflows](#)
4. [Administrative services](#)

40.1 Operations generated from a data model

For a data model used in an EBX5 repository, it is possible to dynamically generate a corresponding WSDL defining operations used to access the data sets that are an instance of this data model. For example, for a table located at the path /root/XX/exampleTable, the generated requests would follow the structure of its underlying data model and include the name of the table <m:{operation}_exampleTable xmlns:m="urn:ebx-schemas:dataservices_1.0">.

Attention

Since the WSDL and the SOAP operations tightly depend on the data model structure, it is important to redistribute the up-to-date WSDL after any data model change.

Content policy

Access to the content of records, the presence or absence of XML elements, depend on the [resolved permissions](#) [p 317] of the authenticated user session. Additional aspects, detailed below, can impact the content.

Disabling fields from data model

The hiddenInDataServices property, defined in the data model, allows always hiding fields in data services, regardless of the user profile. This parameter has an impact on the generated WSDL: any hidden field or group will be absent from the request and response structure.

Modifying the `hiddenInDataServices` parameter value has the following impact on a client which would still use the former WSDL:

- On request, if the data model property has been changed to `true`, and if the concerned field is present in the WSDL request, an exception will be thrown.
- On response, if the schema property has been changed to `false`, WSDL validation will return an error if it is activated.

This setting of "Default view" is defined inside data model.

Voir aussi

[*Hiding a field in Data Services*](#) [p 512]

[*Permissions*](#) [p 317]

Association field

Read-access on table records can export the association fields as displayed in UI Manager. This feature can be coupled with the 'hiddenInDataServices' model parameter.

Note

Limitations: change and update operations do not manage association fields. Also, the select operation only exports the first level of association elements (the content of associated objects cannot contain association elements).

Common request parameters

Several parameters are common to several operations and are detailed below.

Element	Description	Required
branch	The identifier of the data space to which the data set belongs.	Either this parameter or the 'version' parameter must be defined. Required for the 'insert', 'update' and 'delete' operations.
version	The identifier of the snapshot to which the data set belongs.	Either this parameter or the 'branch' parameter must be defined
instance	The unique name of the data set which contains the table to query.	Yes
predicate	XPath predicate [p 267] defines the records on which the request is applied. If empty, all records will be retrieved, except for the 'delete' operation where this field is mandatory.	Only required for the 'delete' operation
data	Contains the records to be modified, represented within the structure of their data model. The whole operation is equivalent to an XML import. The details of the operations performed on data content are specified in the section Import [p 255].	Only required for the insert and update operations
viewPublication	This parameter can be combined with the predicate [p 229] parameter as a logical AND operation. The behavior of this parameter is described in the section EBX5 as a web component [p 186]. It cannot be used if the 'viewId' parameter is used, and cannot be used on hierarchical views.	No
viewId	<i>Deprecated since version 5.2.3.</i> This parameter has been replaced by the parameter 'viewPublication'. While it remains available for backward compatibility, it will eventually be removed in a future version. This parameter cannot be used if the 'viewPublication' parameter is used.	No
blockingConstraintsDisabled	This property is available for all table updates data service operations. If <code>true</code> , the validation process disables blocking constraints defined in the data model. If this parameter is not present, the default is <code>false</code> .	No
disableRedirectionToLastBroadcast	This property is available for all data service operations. If <code>true</code> , access to a delivery data space on a D3 master node is not redirected to the last broadcast snapshot. Otherwise, access to such a data space is always redirected to the last snapshot broadcast.	No

Element	Description	Required
	<p>If this parameter is not present, the default is <code>false</code> (redirection on a D3 master enabled), unless the configuration property ebx.dataservices.disableRedirectionToLastBroadcast.default [p 368] has been set.</p> <p>If the specified data space is not a delivery data space on a D3 master node, this parameter is ignored.</p>	

Select operation

Select request

```
<m:select_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <predicate>String</predicate>
  <viewPublication>String</viewPublication>
  <exportCredentials>boolean</exportCredentials>
  <pagination>
    <previousPageLastRecordPredicate>String</previousPageLastRecordPredicate>
    <pageSize>Integer</pageSize>
  </pagination>
</m:select_{TableName}>
```

with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
version	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
predicate	See the description under Common parameters [p 229]. This parameter can be combined with the viewPublication [p 229] parameter as a logical AND operation.	
viewPublication	See the description under Common parameters [p 229].	
includesTechnicalData	The response will contain technical data if true. See also the optimistic locking [p 243] section. Each returned record will contain additional attributes for this technical information, for instance: <pre>...<table ebxd:lastTime="2010-06-28T10:10:31.046" ebxd:lastUser="Uadmin" ebxd:uuid="9E7D0530-828C-11DF-B733-0012D01B6E76">...</pre>	No
exportCredentials	If true the select will also return the credentials for each record.	No
pagination	Enables pagination, see child elements below.	No
pageSize (nested under the pagination element)	When pagination is enabled, defines the number of records to retrieve.	When pagination is enabled, yes
previousPageLastRecordPredicate (nested under the pagination element)	When pagination is enabled, XPath predicate that defines the record after which the page must fetched, this value is provided by the previous response, as the element lastRecordPredicate. If the passed record is not found, the first page will be returned.	No
disableRedirectionToLastBroadcast	See the description under Common parameters [p 229].	

Select response

```
<ns1:select_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <data>
    <XX>
      <TableName>
        <a>key1</a>
        <b>valueb</b>
        <c>1</c>
        <d>1</d>
      </TableName>
    </XX>
  </data>
  <credentials>
    <XX>
      <TableName predicate="./a='key1'">
        <a>W</a>
        <b>W</b>
      </TableName>
    </XX>
  </credentials>
</ns1:select_{TableName}Response>
```

```

<c>W</c>
<d>W</d>
</TableName>
</XX>
</credentials>
<lastRecordPredicate>./a='key1'</lastRecordPredicate>
</ns1:select_{TableName}Response>

```

See also the [optimistic locking](#) [p 243] section.

Delete operation

Delete request

```

<m:delete_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <predicate>String</predicate>
  <occultIfInherit>boolean</occultIfInherit>
  <checkNotChangedSinceLastTime>dateTime</checkNotChangedSinceLastTime>
  <blockingConstraintsDisabled>boolean</blockingConstraintsDisabled>
</m:delete_{TableName}>

```

with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
predicate	See the description under Common parameters [p 229].	
occultIfInherit	Occults the record if it is in inherit mode. Default value is false.	No
checkNotChangedSinceLastTime	Timestamp used to ensure that the record has not been modified since the last read. Also see the optimistic locking [p 243] section.	No
blockingConstraintsDisabled	See the description under Common parameters [p 229].	
disableRedirectionToLastBroadcast	See the description under Common parameters [p 229].	

Delete response

```

<ns1:delete_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:delete_{TableName}Response>

```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully. '95' indicates that at least one operation has violated a blocking constraint, resulting in the overall operation being aborted.

Count operation

Count request

```
<m:count_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <predicate>String</predicate>
</m:count_{TableName}>
```

with:

Element	Description
branch	See the description under Common parameters [p 229].
version	See the description under Common parameters [p 229].
instance	See the description under Common parameters [p 229].
predicate	See the description under Common parameters [p 229].
disableRedirectionToLastBroadcast	See the description under Common parameters [p 229].

Count response

```
<ns1:count_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <count>Integer</count>
</ns1:count_{TableName}Response>
```

with:

Element	Description
count	The number of records that correspond to the predicate in the request.

Update operation

Update request

```
<m:update_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <updateOrInsert>boolean</updateOrInsert>
  <byDelta>boolean</byDelta>
  <blockingConstraintsDisabled>boolean</blockingConstraintsDisabled>
  <data>
    <XX>
      <TableName>
        <a>String</a>
        <b>String</b>
        <c>String</c>
        <d>String</d>
        ...
      </TableName>
    </XX>
  </data>
```

```
</m:update_{TableName}>
```

with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
updateOrInsert	If true and the record does not currently exist, the operation creates the record.	No
byDelta	If true and an element does not currently exist in the incoming message, the target value is not changed. If false and node is declared hiddenInDataServices, the target value is not changed. The complete behavior is described in the sections Insert and update operations [p 256].	No
blockingConstraintsDisabled	See the description under Common parameters [p 229].	
disableRedirectionToLastBroadcast	See the description under Common parameters [p 229].	
data	See the description under Common parameters [p 229].	

Voir aussi [Optimistic locking](#) [p 243]

Update response

```
<ns1:update_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:update_{TableName}Response>
```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully. '95' indicates that at least one operation has violated a blocking constraint, resulting in the overall operation being aborted.

Insert operation

Insert request

```
<m:insert_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <byDelta>boolean</byDelta>
  <blockingConstraintsDisabled>boolean</blockingConstraintsDisabled>
  <data>
    <XX>
      <TableName>
        <a>String</a>
        <b>String</b>
```

```

    <c>String</c>
    <d>String</d>
    ...
  </TableName>
</XX>
</data>
</m:insert_{TableName}>

```

with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
byDelta	<p>If true and an element does not currently exist in the incoming message, the target value is not changed.</p> <p>If false and node is declared hiddenInDataServices, the target value is not changed.</p> <p>The complete behavior is described in the sections Insert and update operations [p 256].</p>	No
blockingConstraintsDisabled	See the description under Common parameters [p 229].	
disableRedirectionToLastBroadcast	See the description under Common parameters [p 229].	
data	See the description under Common parameters [p 229].	

Insert response

```

<ns1:insert_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
  <inserted>
    <predicate>./a='String'</predicate>
  </inserted>
</ns1:insert_{TableName}Response>

```

with:

Element	Description
status	<p>'00' indicates that the operation has been executed successfully.</p> <p>'95' indicates that at least one operation has violated a blocking constraint, resulting in the overall operation being aborted.</p>
predicate	A predicate matching the primary key of the inserted record. When several records are inserted, the predicates follow the declaration order of the records in the input message.

Get changes operations

Get changes requests

Changes between two data sets:

```

<m:getChangesOnDataSet_{schemaName} xmlns:m="urn:ebx-schemas:dataservices_1.0">

```

```
<branch>String</branch>
<version>String</version>
<instance>String</instance>
<compareWithBranch>String</compareWithBranch>
<compareWithVersion>String</compareWithVersion>
<compareWithInstance>String</compareWithInstance>
<resolvedMode>boolean</resolvedMode>
</m:getChangesOnDataSet_{schemaName}>
```

Changes between two tables:

```
<m:getChanges_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <compareWithBranch>String</compareWithBranch>
  <compareWithVersion>String</compareWithVersion>
  <resolvedMode>boolean</resolvedMode>
</m:getChanges_{TableName}>
```

with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
version	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
compareWithBranch	The identifier of the data space with which to compare.	One of either this parameter or the ' compareWithVersion [p 237]' parameter must be defined.
compareWithVersion	The identifier of the snapshot with which to compare.	One of either this parameter or the ' compareWithBranch [p 237]' parameter must be defined.
compareWithInstance	The identifier of the data set with which to compare. If it is undefined, instance [p 237] parameter is used.	No
resolvedMode	Defines whether or not the difference is calculated in resolved mode. Default is true. See Resolved mode DifferenceHelper^{API} for more information.	No
pagination	Enables pagination context for the operations <code>getChanges</code> and <code>getChangesOnDataSet</code> . Allows client to define pagination context size. Each page contains a collection of inserted, updated and/or deleted records of tables according to the maximum size. Get changes persisted context is built at first call according to the page size parameter in request. The pagination context is persisted on the server file system [p 389] and allows invoking the next page until last page or when a timeout is reached. For creation: Defines <code>pageSize</code> parameter. For next: Defines context element with <code>identifier</code> from previous response. Enables pagination, see child elements below.	No
pageSize (nested under pagination element)	Defines maximum number of records in each page. Minimal size is 50.	No (Only for creation)
context (nested under pagination element)	Defines content of pagination context.	No (Only for next)
identifier (nested under context element)	Pagination context identifier.	Yes

Element	Description	Required
disableRedirectionToLastBroadcast	See the description under Common parameters (p 229).	

Note

If neither *compareWithBranch* nor *compareWithVersion* are specified, the comparison will be made with its parent:

- if the current data space or snapshot is a data space, the comparison is made with its initial snapshot (includes all changes made in the data space);
- if the current data space or snapshot is a snapshot, the comparison is made with its parent data space (includes all changes made in the parent data space since the current snapshot was created);
- returns an exception if the current data space is the 'Reference' data space.

Voir aussi *DifferenceHelper*^{API}

Get changes responses

Changes between two data sets:

```
<ns1:getChangesOnDataSet_{schemaName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <getChanges_{TableName1}>
    ... see the getChanges between tables response example ...
  </getChanges_{TableName1}>
  <getChanges_{TableName2}>
    ... see the getChanges between tables response example ...
  </getChanges_{TableName2}>
  ...
</ns1:getChangesOnDataSet_{schemaName}Response>
```

Changes between two tables:

```
<ns1:getChanges_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <inserted>
    <XX>
      <TableName>
        <a>AVALUE3</a>
        <b>BVALUE3</b>
        <c>CVALUE3</c>
        <d>DVALUE3</d>
      </TableName>
    </XX>
  </inserted>
  <updated>
    <changes>
      <change predicate="./a='AVALUE2'">
        <path>/b</path>
        <path>/c</path>
      </change>
    </changes>
  <data>
    <XX>
      <TableName>
        <a>AVALUE2</a>
        <b>BVALUE2.1</b>
        <c>CVALUE2.1</c>
        <d>DVALUE2</d>
      </TableName>
    </XX>
  </data>
</updated>
<deleted>
  <predicate>./a='AVALUE1'</predicate>
</deleted>
</ns1:getChanges_{TableName}Response>
```

with:

Element	Description	Required
inserted	Contains inserted record(s) from choice <code>compareWithBranch</code> or <code>compareWithVersion</code> . Content under this element corresponding to an XML export of inserted records.	No
updated	Contains updated record(s).	No
changes (nested under context updated)	Only the group of field have been updated.	Yes
change (nested under context changes)	Group of fields have been updated with own XPath predicate attribute of the record.	Yes
data (nested under context changes)	Content under this element corresponding to an XML export of updated records under changes element.	No
deleted	Records have been deleted from context of request. Content corresponding to a list of predicate element who contains the XPath predicate of record.	No
pagination	When pagination is enabled on request. Get changes persisted context allows invoking the next page until last page or when a timeout is reached. For next: Defines context element with <code>identifier</code> . For last: Defines context element without <code>identifier</code> . Enables pagination, see child elements below.	No
context (nested under pagination element)	Defines content of pagination context.	Yes (Only for next and last)
identifier (nested under context element)	Pagination context identifier. Not defined at last returned page.	No
pageNumber (nested under context element)	Current page number in pagination context.	Yes
totalPages (nested under context element)	Total pages in pagination context.	Yes

Get changes operation with pagination enabled

Only pagination element and sub elements have been described.

For creation:

Extract of request:

```
...
<pagination>
  <!-- on first request for creation -->
  <pageSize>Integer</pageSize>
```

```
</pagination>
...
```

Extract of response:

```
...
<pagination>
  <!-- on next request to continue -->
  <context>
    <identifier>String</identifier>
    <pageNumber>Integer</pageNumber>
    <totalPages>Integer</totalPages>
  </context>
</pagination>
...
```

For next:

Extract of request:

```
...
<pagination>
  <context>
    <identifier>String</identifier>
  </context>
</pagination>
...
```

Extract of response:

```
...
<pagination>
  <!-- on next request to continue -->
  <context>
    <identifier>String</identifier>
    <pageNumber>Integer</pageNumber>
    <totalPages>Integer</totalPages>
  </context>
</pagination>
...
```

For last:

Extract of request:

```
...
<pagination>
  <context>
    <identifier>String</identifier>
  </context>
</pagination>
...
```

Extract of response:

```
...
<pagination>
  <context>
    <pageNumber>Integer</pageNumber>
    <totalPages>Integer</totalPages>
  </context>
</pagination>
...
```

Get credentials operation

Get credentials request

```
<m:getCredentials_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <predicate>String</predicate>
  <viewPublication>String</viewPublication>
</m:getCredentials_{TableName}>
```


with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
version	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
viewPublication	See the description under Common parameters [p 229].	

Get credentials response

```
<ns1:getCredentials_{TableName}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <XX>
    <TableName>
      <a>R</a>
      <b>W</b>
      <c>H</c>
      <d>W</d>
      ...
    </TableName>
  </XX>
</ns1:getCredentials_{TableName}Response>
```

With the following possible values:

- R: for read-only
- W: for read-write
- H: for hidden

Multiple chained operations

Multiple operations request

It is possible to run multiple operations across tables in the data set, while ensuring a consistent response. The operations are executed sequentially, according to the order defined on the client side.

All operations are executed in a single transaction with a `SERIALIZABLE` isolation level. If all requests in the multiple operation are read-only, they are allowed to run fully concurrently along with other read-only transactions, even in the same data space.

When an error occurs during one operation in the sequence, all updates are rolled back and the client receives a `StandardException` error message with details.

See [Concurrency and isolation levels](#) [p 519].

```
<m:multi xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <blockingConstraintsDisabled>boolean</blockingConstraintsDisabled>
  <request id="id1">
    <{operation}_{TableName}>
      ...
    </{operation}_{TableName}>
  </request>
  <request id="id2">
    <{operation}_{TableName}>
      ...
    </{operation}_{TableName}>
  </request>
```

</m:multi_>

with:

Element	Description	Required
branch	See the description under Common parameters [p 229].	
version	See the description under Common parameters [p 229].	
instance	See the description under Common parameters [p 229].	
blockingConstraintsDisabled	See the description under Common parameters [p 229].	
disableRedirectionToLastBroadcast	See the description under Common parameters [p 229].	
request	<p>This element contains one operation, like a single operation without branch, version and instance parameters. This element can be repeated multiple times for additional operations. Each request can be identified by an 'id' attribute. In a response, this 'id' attribute is returned for identification purposes.</p> <p>Operations such as count, select, getChanges, getCredentials, insert, delete or update.</p>	Yes

Note:

- Does not accept a limit on the number of request elements.
- The request id attribute must be unique in multi-operation requests.
- If all operations are read only (count, select, getChanges, or getCredentials) then the whole transaction is set as read-only for performance considerations.

Limitation:

- The multi operation applies to one model and one data set (parameter instance).
- The select operation cannot use the pagination parameter.

Voir aussi

Procedure^{API}

Repository^{API}

Multiple operations response

See each response operation for details.

```
<ns1:multi_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <response id="id1">
    <{operation}_{TableName}Response>
      ...
    </{operation}_{TableName}Response>
  </response>
  <response id="id2">
    <{operation}_{TableName}Response>
      ...
    </{operation}_{TableName}Response>
  </response>
</ns1:multi_Response>
```

with:

Element	Description
response	<p>This element contains the response of one operation. It is repeated multiple times for additional operations. Each response is identified by an 'id' attribute set in the request or automatically generated.</p> <p>The content of the element corresponds to the response of a single operation, such as count, select, getChanges, getCredentials, insert, delete or update.</p>

Optimistic locking

To prevent an update or a delete operation from being performed on a record that was read earlier but may have changed in the meantime, an optimistic locking mechanism is provided.

In the select request, it is possible to ask for technical information by adding the element `includesTechnicalData`:

```
<m:select_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <predicate>String</predicate>
  <includesTechnicalData>boolean</includesTechnicalData>
</m:select_{TableName}>
```

The value of the `lastTime` attribute can then be used in the update request. If the record has been changed since the specified time, the update will be cancelled. The attribute `lastTime` has to be added on the record to be updated.

```
<m:update_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <updateOrInsert>true</updateOrInsert>
  <data>
    <XX>
      <TableName ebxd:lastTime="2010-06-28T10:10:31.046">
        <a>String</a>
        <b>String</b>
        <c>String</c>
        <d>String</d>
        ...
      </TableName>
    </XX>
  </data>
</m:update_{TableName}>
```

The value of the `lastTime` attribute can also be used to prevent deletion on a modified record:

```
<m:delete_{TableName} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <predicate>String</predicate>
  <checkNotChangedSinceLastTime>2010-06-28T10:10:31.046</checkNotChangedSinceLastTime>
</m:delete_{TableName}>
```

The element `checkNotChangedSinceLastTime` can be used for one and only one record upon request. This means that if the predicate returns more than one record, the request will fail if the element `checkNotChangedSinceLastTime` is set.

40.2 Operations on data sets and data spaces

Parameters for operations on data spaces and snapshots are as follows:

<i>Element</i>	<i>Description</i>	<i>Required</i>
branch	Identifier of the target data space on which the operation is applied. When not specified, the 'Reference' data space is used except for the merge data space operation where it is required.	One of either this parameter or the 'version' parameter must be defined. Required for the data space merge, locking, unlocking and replication refresh operations.
version	Identifier of the target snapshot on which the operation is applied.	One of either this parameter or the 'branch' parameter must be defined
versionName	Identifier of the snapshot to create. If empty, it will be defined on the server side.	No
childBranchName	Identifier of the data space child to create. If empty, it will be defined on the server side.	No
instance	The unique name of the data set on which the operation is applied.	Required for the replication refresh operation.
ensureActivation	Defines if validation must also check whether this instance is activated.	Yes
details	<p>Defines if validation returns details.</p> <p>The optional attribute <code>severityThreshold</code> defines the lowest severity level of message to return. The possible values are, in descending order of severity, 'fatal', 'error', 'warning', or 'info'. For example, setting the value to 'error' will return error and fatal validation messages. If this attribute is not defined, all levels of messages are returned by default.</p> <p>The optional attribute <code>locale</code> (default 'en-US') defines the language in which the validation messages are to be returned.</p>	No. If not specified, no details are returned.
owner	<p>Defines the owner.</p> <p>Must respect the inner format as returned by <code>Profile.format^{API}</code>.</p>	No
branchToCopyPermissionFrom	Defines the identifier of the data space from which to copy the permissions.	No
documentation	Documentation for a dedicated language.	No

<i>Element</i>	<i>Description</i>	<i>Required</i>
	Multiple documentation elements may be used for several languages.	
locale (nested under the documentation element)	Locale of the documentation.	Only required when the documentation element is used
label (nested under the documentation element)	Label for the language.	No
description (nested under the documentation element)	Description for the language.	No

Validate a data space

Validate data space request

```
<m:validate xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
</m:validate>
```

Validate data space response

```
<ns1:validate_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <validationReport>
    <instanceName>String</instanceName>
    <fatals>boolean</fatals>
    <errors>boolean</errors>
    <infos>boolean</infos>
    <warnings>boolean</warnings>
  </validationReport>
</ns1:validate_Response>
```

Validate a data set

Validate data set request

```
<m:validateInstance xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <version>String</version>
  <instance>String</instance>
  <ensureActivation>boolean</ensureActivation>
  <details severityThreshold="fatal|error|warning|info" locale="en-US"/>
</m:validateInstance>
```

Validate data set response

```
<ns1:validateInstance_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <validationReport>
    <instanceName>String</instanceName>
    <fatals>boolean</fatals>
    <errors>boolean</errors>
    <infos>boolean</infos>
    <warnings>boolean</warnings>
    <details>
      <reportItem>
        <severity>{fatal|error|warning|info}</severity>
        <message>
          <internalId />
          <text>String</text>
        </message>
        <subject>
          <table>Path</table>
          <predicate>String</predicate>
        </subject>
      </reportItem>
    </details>
  </validationReport>
</ns1:validateInstance_Response>
```

```
<path>Path</path>
</subject>
</reportItem>
</details>
</validationReport>
</ns1:validateInstance_Response>
```

Create a data space

Create data space request

```
<m:createBranch xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <owner>String</owner>
  <branchToCopyPermissionFrom>String</branchToCopyPermissionFrom>
  <documentation>
    <locale>Locale</locale>
    <label>String</label>
    <description>String</description>
  </documentation>
  <childBranchName>String</childBranchName>
</m:createBranch>
```

Create data space response

```
<ns1:createBranch_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
  <childBranchName>String</childBranchName>
</ns1:createBranch_Response>
```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully.

Create a snapshot

Create snapshot request

```
<m:createVersion xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <versionName>String</versionName>
  <owner>String</owner>
  <documentation>
    <locale>Locale</locale>
    <label>String</label>
    <description>String</description>
  </documentation>
</m:createVersion>
```

Create snapshot response

```
<ns1:createVersion_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
  <versionName>String</versionName>
</ns1:createVersion_Response>
```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully.

Locking a data space

Lock data space request

```
<m:lockBranch xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <durationToWaitForLock>Integer</durationToWaitForLock>
  <message>
    <locale>Locale</locale>
    <label>String</label>
  </message>
</m:lockBranch>
```

with:

Element	Description	Required
durationToWaitForLock	This parameter defines the maximum duration (in seconds) that the operation waits for a lock before aborting.	No, does not wait by default
message	User message of the lock. Multiple message elements may be used.	No
locale (nested under the message element)	Locale of the user message.	Only required when the message element is used
label (nested under the message element)	The user message.	No

Lock data space response

```
<ns1:lockBranch_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:lockBranch_Response>
```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully. '94' indicates that the data space has been already locked by another user. Otherwise, a SOAP exception is thrown.

Unlocking a data space

Unlock data space request

```
<m:unlockBranch xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
</m:unlockBranch>
```

Unlock data space response

```
<ns1:unlockBranch_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:unlockBranch_Response>
```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully. Otherwise, a SOAP exception is thrown.

Merge a data space

Merge data space request

```
<m:mergeBranch xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <deleteDataOnMerge>boolean</deleteDataOnMerge>
  <deleteHistoryOnMerge>boolean</deleteHistoryOnMerge>
</m:mergeBranch>
```

with:

Element	Description	Required
deleteDataOnMerge	This parameter is available for the merge data space operation. Sets whether the specified data space and its associated snapshots will be deleted upon merge. When this parameter is not specified in the request, the default value is false. It is possible to redefine the default value by specifying the property <code>ebx.dataservices.dataDeletionOnCloseOrMerge.default</code> in the EBX5 main configuration file [p 368]. See Deleting data and history [p 386] for more information.	No
deleteHistoryOnMerge	This parameter is available for the merge data space operation. Sets whether the history associated with the specified data space will be deleted upon merge. Default value is false. When this parameter is not specified in the request, the default value is false. It is possible to redefine the default value by specifying the property <code>ebx.dataservices.historyDeletionOnCloseOrMerge.default</code> in the EBX5 main configuration file [p 368]. See Deleting data and history [p 386] for more information.	No

Note

The merge decision step is bypassed during merges performed through data services. In such cases, the data in the child data space automatically overrides the data in the parent data space.

Merge data space response

```
<ns1:mergeBranch_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:mergeBranch_Response>
```


with:

Element	Description
status	'00' indicates that the operation has been executed successfully.

Close a data space or snapshot

Close data space or snapshot request

Close data space request:

```
<m:closeBranch xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <deleteDataOnClose>boolean</deleteDataOnClose>
  <deleteHistoryOnClose>boolean</deleteHistoryOnClose>
</m:closeBranch>
```

Close snapshot request:

```
<m:closeVersion xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <version>String</version>
  <deleteDataOnClose>boolean</deleteDataOnClose>
</m:closeVersion>
```

with:

Element	Description	Required
deleteDataOnClose	<p>This parameter is available for the close data space and close snapshot operations. Sets whether the specified snapshot, or data space and its associated snapshots, will be deleted upon closure.</p> <p>When this parameter is not specified in the request, the default value is false. It is possible to redefine this default value by specifying the property <code>ebx.dataservices.dataDeletionOnCloseOrMerge.default</code> in the EBX5 main configuration file [p 368].</p> <p>See Deleting data and history [p 386] for more information.</p>	No
deleteHistoryOnClose	<p>This parameter is available for the close data space operation. Sets whether the history associated with the specified data space will be deleted upon closure. Default value is false.</p> <p>When this parameter is not specified in the request, the default value is false. It is possible to redefine the default value by specifying the property <code>ebx.dataservices.historyDeletionOnCloseOrMerge.default</code> in the EBX5 main configuration file [p 368].</p> <p>See Deleting data and history [p 386] for more information.</p>	No

Close data space or snapshot response

Close data space response:

```
<ns1:closeBranch_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:closeBranch_Response>
```

Close snapshot request:

```
<ns1:closeVersion_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:closeVersion_Response>
```

Replication refresh

Replication refresh request

```
<m:replicationRefresh_{$schema} xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>String</branch>
  <instance>String</instance>
  <unitName>String</unitName>
</m:replicationRefresh_{$schema}>
```

with:

Element	Description	Required
branch	See the description under Common parameters [p 244].	Yes
instance	See the description under Common parameters [p 244].	Yes
unitName	Name of the replication unit. Voir aussi Replication refresh information [p 298]	Yes

Replication refresh response

```
<ns1:replicationRefresh_{$schema}Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:replicationRefresh_{$schema}Response>
```

with:

Element	Description
status	'00' indicates that the operation has been executed successfully.

40.3 Operations on data workflows

Parameters for operations on data workflows are as follows:

Element	Description	Required
parameters	Input parameters for the process instance to be created.	No
parameter (nested under the parameters element). Multiple parameter elements may be used.	An input parameter for the process instance.	No
name (nested under the parameter element)	Name of the parameter.	Yes
value (nested under the parameter element)	Value of the parameter.	No

Start a workflow

Start a workflow process instance from a published process key. It is possible to start a workflow with localized documentation and specific input parameters (with name and optional value).

Sample request:

```
<m:workflowProcessInstanceStart xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <publishedProcessKey>String</publishedProcessKey>
  <documentation>
    <locale>Locale</locale>
    <label>String</label>
    <description>String</description>
  </documentation>
  <parameters>
    <parameter>
      <name>String</name>
      <value>String</value>
    </parameter>
  </parameters>
</m:workflowProcessInstanceStart>
```

with:

Element	Description	Required
publishedProcessKey	Key of the published workflow to start.	Yes
documentation	See the description under Common parameters [p 244].	No
parameters	See the description under Common parameters [p 250].	No

Sample response:

```
<m:workflowProcessInstanceStart_Response xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <processInstanceKey>String</processInstanceKey>
</m:workflowProcessInstanceStart_Response>
```

with:

Element	Description	Required
processInstanceId	<i>Deprecated since version 5.6.1</i> This parameter has been replaced by the parameter 'processInstanceKey'. While it remains available for backward compatibility, it will eventually be removed in a future major version.	No
processInstanceKey	Key of the workflow process instance.	No

Resume a workflow

Resume a workflow process instance in a wait step from a resume identifier. It is possible to defined specific input parameters (with name and optional value).

Sample request:

```
<m:workflowProcessInstanceResume xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <resumeId>String</resumeId>
  <parameters>
    <parameter>
      <name>String</name>
      <value>String</value>
    </parameter>
  </parameters>
```

```
</m:workflowProcessInstanceResume>
```

with:

Element	Description	Required
resumeId	Resume identifier of the waiting task.	Yes
parameters	See the description under Common parameters [p 250].	No

Sample response:

```
<m:workflowProcessInstanceResume_Response xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
  <processInstanceKey>String</processInstanceKey>
</m:workflowProcessInstanceResume_Response>
```

with:

Element	Description	Required
status	'00' indicates that the operation has been executed successfully. '20' indicates that the process instance has not been found. '21' indicates that the event has already been received.	Yes
processInstanceKey	Key of the workflow process instance. This parameter is returned if the operation has been executed successfully.	No

End a workflow

End a workflow process instance from its key representation.

Sample request:

```
<m:workflowProcessInstanceEnd xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <processInstanceKey>String</processInstanceKey>
</m:workflowProcessInstanceEnd>
```

with:

Element	Description	Required
processInstanceKey	Key of the workflow process instance.	Either this parameter or 'publishedProcessKey' and 'processInstanceId' parameters must be defined.
publishedProcessKey	<i>Deprecated since version 5.6.1</i> Due to a limitation this parameter has been replaced by the 'processInstanceKey' parameter. While it remains available for backward compatibility, it will eventually be removed in a future major version.	No
processInstanceId	<i>Deprecated since version 5.6.1</i> Due to a limitation this parameter has been replaced by the 'processInstanceKey' parameter. While it remains available for backward compatibility, it will eventually be removed in a future major version.	No

Sample response:

```
<m:workflowProcessInstanceEnd_Response xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</m:workflowProcessInstanceEnd_Response>
```

with:

Element	Description	Required
status	'00' indicates that the operation has been executed successfully.	Yes

40.4 Administrative services

Directory services

The services on directory provide operations on the 'Users' and 'Roles' tables of the default directory. To execute an operation related to these services, the authenticated user must be a member of the built-in role 'Administrator'.

The technical data space and data set must be set to ebx-directory. For all SOAP operation syntaxes, see [Operations generated from a data model](#) [p 227] for more information.

Create a directory user

This example of a SOAP insert request adds a user to the EBX5 directory.

```
<m:insert_user xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <branch>ebx-directory</branch>
  <instance>ebx-directory</instance>
  <data>
    <directory>
      <user>
        <login>login</login>
        <lastName>lastname</lastName>
        <firstName>firstname</firstName>
        <email>firstname.lastname@email.com</email>
        <password>***</password>
```

```
<passwordMustChange>true</passwordMustChange>
<builtInRoles>
  <administrator>false</administrator>
  <readOnly>false</readOnly>
</builtInRoles>
<specificRoles>R1</specificRoles>
<specificRoles>R2</specificRoles>
<comments>a comment</comments>
</user>
</directory>
</data>
</m:insert_user>
```

For the insert SOAP response syntax, see [insert response](#) [p 235] for more information.

User interface operations

See [Application locking](#) [p 398] for more information.

Parameters for operations on the user interface are as follows:

Element	Description	Required
closedMessage	Message to be displayed to users when the user interface is closed to access.	No

Close user interface request

The close operation removes all user sessions that are not acceptable in this mode.

```
<m:close xmlns:m="urn:ebx-schemas:dataservices_1.0">
  <closedMessage>Access is temporarily forbidden.</closedMessage>
</m:close>
```

Close user interface response

```
<ns1:close_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:close_Response>
```

Open user interface request

```
<m:open xmlns:m="urn:ebx-schemas:dataservices_1.0"/>
```

Open user interface response

```
<ns1:open_Response xmlns:ns1="urn:ebx-schemas:dataservices_1.0">
  <status>String</status>
</ns1:open_Response>
```

CHAPITRE 41

XML import and export

XML imports and exports can be performed on tables through the user interface using the **Actions** menu in the workspace.

Both imports and exports are performed in the context of a data set.

Imports and exports can also be done programmatically.

Default import and export option values can be set in the Administration area, under **User interface > Graphical interface configuration > Default option values > Import/Export**.

Ce chapitre contient les sections suivantes :

1. [Imports](#)
2. [Exports](#)
3. [Handling of field values](#)
4. [Known limitations](#)

41.1 Imports

Attention

Imported XML documents must be encoded in UTF-8 and its structure must conform to the underlying data model of the target data set.

Import mode

When importing an XML file, you must specify one of the following import modes, which will dictate how the import procedure handles the source records.

Insert mode	Only record creations are allowed. If a record exists in the target table with the same primary key as the source record, an error is returned and the whole import operation is cancelled.
Update mode	Only modifications of existing records are allowed. If no record exists in the target table with the same primary key as the source record, an error is returned and the whole import operation is cancelled.
Update or insert mode	If a record with the same primary key as the source record already exists in the target table, that record is updated. Otherwise, a new record is created.
Replace (synchronization) mode	If a record with the same primary key as the source record already exists in the target table, that record is updated. Otherwise, a new record is created. If a record exists in the target table but is not present in the source XML file, that record is deleted from the table.

Insert and update operations

The mode *'by delta'* allows ignoring data model elements that are missing from the source XML document. This mode can be enabled through data services or the Java API. The following table summarizes the behavior of insert and update operations when elements are not present in the source document.

See the data services operations [update](#) [p 233] and [insert](#) [p 234], as well as `ImportSpec.setByDelta`^{API} in the Java API for more information.

State in source XML document	Behavior
Element does not exist in the source document	<p>If 'by delta' mode is disabled (default):</p> <p>Target field value is set to one of the following:</p> <ul style="list-style-type: none"> • If the element defines a default value, the target field value is set to that default value. • If the element is of a type other than a string or list, the target field value is set to null. • If the element is an aggregated list, the target field value is set to an empty list. • If the element is a string that distinguishes null from an empty string, the target field value is set to null. If it is a string that does not distinguish between the two, an empty string. • If the element (simple or complex) is hidden in data services, the target value is not changed. <p>Voir aussi Hiding a field in Data Services [p 512]</p> <p>Note: The user performing the import must have the permissions necessary to create or change the target field value. Otherwise, the value will remain unchanged.</p> <p>If 'by delta' mode has been enabled through data services or the Java API:</p> <ul style="list-style-type: none"> • For the update operation, the field value remains unchanged. • For the insert operation, the behavior is the same as when byDelta mode is disabled.
Element exists but is empty (for example, <fieldA/>)	<ul style="list-style-type: none"> • For nodes of type <code>xs:string</code> (or one of its sub-types), the target field's value is set to null if it distinguishes null from an empty string. Otherwise, the value is set to empty string. • For non-<code>xs:string</code> type nodes, an exception is thrown in conformance with XML Schema. <p>Voir aussi EBX5 whitespace management for data types [p 497]</p>
Element is present and null (for example, <fieldA xsi:nil="true"/>)	<p>The target field is always set to null except for lists, for which it is not supported.</p> <p>In order to use the <code>xsi:nil="true"</code> attribute, you must import the namespace <code>xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"</code>.</p>

It may happen that the XML document contains elements that do not exist in the target data model. By default, in this case, the import procedure will fail. It is possible, however, to allow users to launch import procedures that will ignore the extra columns defined in the XML files. This can be done in the configuration parameters of the import wizard for XML. The default value of this parameter can be configured in the 'User interface' configuration under the 'Administration' area.

Optimistic locking

If the technical attribute `x:lastTime` exists in the source XML file, the import mechanism performs a verification to prevent an update operation on a record that may have changed since the last read. The timestamp associated with the current record will be compared to this timestamp. If they are different, the update is rejected.

41.2 Exports

Note

Exported XML documents are always encoded in UTF-8.

When exporting to XML, if the table has filters applied, only the records that correspond to the filter are included in the exported file.

The XML export options are as follows:

Download file name	Specifies the name of the XML file to be exported. This field is pre-populated with the name of the table from which the records are being exported.
User-friendly mode	Specifies whether exported values will be represented in a user-friendly way, or in the standard XML raw format. For example, in user-friendly mode, dates and numbers are formatted according to the user's locale, and foreign keys and enumerated values display their associated labels. Note: If this option is selected, the exported file will not be able to be re-imported.
Include technical data	Specifies whether internal technical data will be included in the export. Note: If this option is selected, the exported file will not be able to be re-imported.
Include computed values	Specifies whether computed values will be exported.
Is indented	Specifies whether the file should be indented to improve readability.

41.3 Handling of field values

Date, time & dateTime format

The following date and time formats are supported:

Type	Format	Example
xs:date	yyyy-MM-dd	2007-12-31
xs:time	HH:mm:ss or HH:mm:ss.SSS	11:55:00
xs:dateTime	yyyy-MM-ddTHH:mm:ss or yyyy-MM-ddTHH:mm:ss.SSS	2007-12-31T11:55:00

41.4 Known limitations

Association fields

The XML import and export services do not support association values.

Exporting such nodes will not cause any error, however, no value will be exported.

Importing such nodes will cause an error, and the import procedure will be aborted.

CHAPITRE 42

CSV import and export

CSV imports and exports can be performed on tables through the user interface using the **Actions** menu in the workspace. Both imports and exports are performed in the context of a data set.

Imports and exports can also be done programmatically.

Default import and export option values can be set in the Administration area, under **User interface > Graphical interface configuration > Default option values > Import/Export**.

Voir aussi [Default option values](#) [p 400]

Ce chapitre contient les sections suivantes :

1. [Exports](#)
2. [Imports](#)
3. [Handling of field values](#)
4. [Known limitations](#)

42.1 Exports

When exporting to CSV, if the table has filters applied, only the records that correspond to the filter are included in the exported file.

The CSV export options are as follows:

Download file name	Specifies the name of the CSV file to be exported. This field is pre-populated with the name of the table from which the records are being exported.
File encoding	Specifies the character encoding to use for the exported file. The default is UTF-8.
Enable inheritance	<p>Specifies if inheritance will be taken into account during a CSV export. If inheritance is enabled, resolved values of fields are exported with the technical data that define the possible inheritance mode of the record or the field. If inheritance is disabled, resolved values of fields are exported and occluded records are ignored. By default, this option is disabled.</p> <p>Note: Inheritance is always ignored, if the table data set has no parent or if the table has no inherited field.</p>
User-friendly mode	<p>Specifies whether exported values will be represented in a user-friendly way, or in a raw format. For example, in user-friendly mode, dates and numbers are formatted according to the user's locale, and foreign keys and enumerated values display their associated labels.</p> <p>Note: If this option is selected, the exported file will not be able to be re-imported.</p>
Include technical data	<p>Specifies whether internal technical data will be included in the export.</p> <p>Note: If this option is selected, the exported file will not be able to be re-imported.</p>
Include computed values	Specifies whether computed values will be exported.
Column header	<p>Specifies the whether or not to include column headers in the CSV file.</p> <ul style="list-style-type: none"> • No header • Label: For each column in the spreadsheet, the CSV displays its label. Each label is localized according to the locale preference of the current session. If no user-friendly label is defined for a node, the technical name of the node is used. • XPath: For each column in the spreadsheet, the CSV displays the path to the node in the table.

Field separator	Specifies the field separator to use for exports. The default separator is comma.
List separator	Specifies the separator to use for values lists. The default separator is line return.

Programmatic CSV exports are performed using the classes `ExportSpecAPI` and `ExportImportCSVSpecAPI` in the Java API.

42.2 Imports

When importing a CSV file, you must specify one of the following import modes, which will determine how the import procedure handles the source records.

Insert mode	Only record creation is allowed. If a record exists in the target table with the same primary key as the source record, an error is returned and the whole import operation is cancelled.
Update mode	Only modifications of existing records are allowed. If no record exists in the target table with the same primary key as the source record, an error is returned and the whole import operation is cancelled.
Update or insert mode	If a record with the same primary key as the source record already exists in the target table, that record is updated. Otherwise, a new record is created.
Replace (synchronization) mode	If a record with the same primary key as the source record already exists in the target table, that record is updated. Otherwise, a new record is created. If a record exists in the target table but is not present in the source XML file, that record is deleted from the table.

In order to consider the inheritance during a CSV import, the following option has to be specified.

Enable inheritance	<p>Specifies whether the inheritance will be taken into account during a CSV import. If technical data in the CSV file define an inherit mode, corresponding fields or records are forced to be inherited. If technical data define an occult mode, corresponding records are forced to be occulted. Otherwise, fields are overwritten with values read from the CSV file. By default, this option is disabled.</p> <p>Note: Inheritance is always ignored if the data set of the table has no parent or if the table has no inherited field.</p>
---------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Programmatic CSV imports are performed using the classes `ImportSpecAPI` and `ExportImportCSVSpecAPI` in the Java API.

42.3 Handling of field values

Aggregated lists

The CSV import and export services support multi-valued fields, namely aggregated lists. This is only supported for simple typed lists, such as lists of `string`, `date`, or `int`, and for table references. If a table reference is linked to a composite primary key, each item in the list is a formatted string, for example, "true|99". Aggregated lists of groups are not exported.

At export, the items in the list are separated using line separators. In cases where the exported field already contains a line separator, for example in an `osd:html` or an `osd:text`, the code `_crnl_` is inserted in place of the field value's line separators. The same formatting is expected at import, with the whole field value surrounded by quotes.

Hidden fields

Hidden fields are exported as `ebx-csv:hidden` strings. An imported hidden string will not modify a node's content.

'Null' value for strings

Using CSV import and export services, a string with a value set to `null` is exported as an empty string. Therefore, a round trip export-import procedure will end up replacing `null` string values with empty strings.

Using programmatic services, the specific value `ebx-csv:nil` can be assigned to strings with values set to `null`. If this is done, the `null` string values will not be replaced by empty strings during round trip export-import procedures. See `ExportImportCSVSpec.setNullStringEncodedAPI` in the Java API for more information.

Date, time & dateTime format

The following date and time formats are supported:

Type	Format	Example
xs:date	yyyy-MM-dd	2007-12-31
xs:time	HH:mm:ss or HH:mm:ss.SSS	11:55:00
xs:dateTime	yyyy-MM-ddTHH:mm:ss or yyyy-MM-ddTHH:mm:ss.SSS	2007-12-31T11:55:00

42.4 Known limitations

Aggregated lists of groups

The CSV import and export services do not support importing multi-valued groups, that is, aggregated lists of complex type elements. Exporting such nodes will not cause any error, however, no value will be exported.

Terminal groups

In a CSV file, it is impossible to differentiate a created terminal group that contains only empty fields from a non-created one.

As a consequence, some differences may appear during comparison after performing an export followed by an import. To ensure the symmetry of import and export, use XML import and export instead. See [XML import and export](#) [p 255].

Column label headers

If two columns share the same label header, an export of the table can be performed successfully, but exported data cannot later be re-imported.

Association fields

The CSV import and export services do not support importing association values, that is multi-valued of complex type elements.

Exporting such nodes will not cause any error, however, no value will be exported.

Importing such nodes will cause an error and the import procedure will be aborted.

CHAPITRE 43

Supported XPath syntax

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Example expressions](#)
3. [Syntax specifications for XPath expressions](#)

43.1 Overview

The XPath notation used in EBX5 must conform to the *abbreviated syntax* of the [XML Path Language \(XPath\) Version 1.0](#) standard, with certain restrictions. This document details the abbreviated syntax that is supported.

43.2 Example expressions

The general XPath expression is:

path[predicate]

Absolute path

/library/books/

Relative paths

./Author

../Title

Root and descendant paths

//books

Table paths with predicates

../..books/[author_id = 0101 and (publisher = 'harmattan')]

/library/books/[not(publisher = 'dumesnil')]

Complex predicates

starts-with(col3, 'xxx') and ends-with(col3, 'yyy') and osd:is-not-null(./col3))

```
contains(col3 , 'xxx') and ( not(col1=100) and date-greater-  
than(col2, '2007-12-30') )
```

Predicates on label

```
osd:label(./delivery_date)='12/30/2014' and ends-with(osd:label(../  
adress), 'Beijing - China')
```

Syntax specifications for XPath expressions

Overview

Expression	Format	Example
XPath expression	<container path>[predicate]	/books[title='xxx']
<container path>	<absolute path> or <relative path>	
<absolute path>	/a/b or //b	//books
<relative path>	../b, ./b or b	../b

Predicate specification

Expression	Format	Notes/Example
<predicate>	Example: A and (B or not(C)) A,B,C: <atomic expression>	Composition of: logical operators braces, not() and atomic expressions.
<atomic expression>	<path><comparator><criterion> or method(<path>,<criterion>)	royalty = 24.5 starts-with(title, 'Johnat')booleanValue = true
<path>	<relative path> or osd:label(<relative path>)	Relative to the table that contains it: ../authorstitle
<comparator>	<boolean comparator>, <numeric comparator> or <string comparator>	
<boolean comparator>	= or !=	
<numeric comparator>	= , != , < , > , <= , or >=	
<string comparator>	=	
<method>	<date method>, <string method>, osd:is-null method or osd:is-not- null method	
<date, time & dateTime method>	date-less-than, date-equal or date- greater-than	
<string method>	matches, starts-with, ends-with, contains, osd:is-empty, osd:is- not-empty, osd:is-equal-case- insensitive, osd:starts-with-case- insensitive, osd:ends-with-case- insensitive, or osd:contains-case- insensitive	
<criterion>	<boolean criterion>, <numeric criterion>, <string criterion>, <date criterion>, <time criterion>, or <dateTime criterion>	
<boolean criterion>	true , false	
<numeric criterion>	An integer or a decimal	-4.6
<string criterion>	Quoted character string	'azerty'
<date criterion>	Quoted and formatted as 'yyyy-MM-dd'	'2007-12-31'

Expression	Format	Notes/Example
<time criterion>	Quoted and formatted as 'HH:mm:ss' or 'HH:mm:ss.SSS'	'11:55:00'
<dateTime criterion>	Quoted and formatted as 'yyyy-MM-ddTHH:mm:ss' or 'yyyy-MM-ddTHH:mm:ss.SSS'	'2007-12-31T11:55:00'

XPath 1.0 formula

It is possible to use an XPath 1.0 formula in the criterion value part of an atomic predicate expression (right-hand side).

For example, instead of `[./a=3]`, you may use the expression `[./a=(floor(./d)+ 2.0)]`.

Due to the strong dependence of predicates on the data model node and the node type of the criterion, the path portion of the atomic predicate expression (left-hand side) must be a node path and cannot be an XPath formula. For example, the expression `/table[floor(./a) > ceiling(./d)]` is not valid.

Predicate on label

The `osd:label()` function can be applied to the path portion of the atomic predicate, in order to resolve the predicate on the label instead of the value. In this case, only string operators and string criteria can be used, i.e. `ends-with(osd:label(./price), '99')`.

A predicate on label is localized, so the criterion must be expressed in the same locale as the predicate-filtered request. For example: `request.setLocale(Locale.FRENCH); request.setXPathFilter("osd:label(./delivery_date)='30/12/2014'");`

Note

It is forbidden to use the `osd:label` function if the right part of the predicate is a contextual value.

Note

If the `osd:label` function is used in a data model, for example on a selection or in the filter predicate of a table reference node, the default locale of the data model (as defined in its module declaration) must be used for the criterion format (even though this is generally not recommended).

Voir aussi `SchemaNode.displayOccurrenceAPI`

Contextual values

For predicates that are relative to a selected node, the criterion value (that is, the right-hand side of the predicate) can be replaced with a contextual path using the syntax `${<relative-path>}` where `<relative-path>` is the location of the element relative to the selected node.

Note

When calling a method, the criterion is the second parameter, and the first parameter cannot be a relative value.

Aggregated lists

For predicates on aggregated lists, the predicate returns true regardless of the comparator if one of the list elements verifies the predicate.

Note

Special attention must be paid to the comparator `!=`. For example, for an aggregated list, `./list != 'a'` is not the same as `not(./list = 'a')`. Where the list contains the elements (`e1, e2, . . .`), the first predicate is equivalent to `e1 != 'a' or e2 != 'a' . . .`, while the second is equivalent to `e1 != 'a' and e2 != 'a' . . .`.

'Null' values

Null values must be explicitly treated in a predicate using the operators `osd:is-null` and `osd:is-not-null`.

For example, `/root/products[./price<100]` or `/root/products[./price!=100]` will not return any products whose prices are not set (null). For the latter case to return unset values as well, the predicate must instead be: `/root/products[./price!=100 or osd:is-null(./price)]`.

How to manage single and double quotes in literal expressions

By default, a literal expression is delimited by single quotes (`'`). If the literal expression contains single quotes and no double quotes, the expression must be delimited by double quotes (`"`). If the literal expression contains both single and double quotes, the single quotes must be doubled.

The method `XPathExpressionHelper.encodeLiteralStringWithDelimiters`^{API} in the Java API handles this.

Examples of using `encodeLiteralStringWithDelimiters`

Value of Literal Expression	Result of this method
Coeur	'Coeur'
Coeur d'Alene	"Coeur d'Alene"
He said: "They live in Coeur d'Alene".	'He said: "They live in Coeur d''Alene".'

Extraction of foreign keys

In EBX5, the foreign keys are grouped into a single field with the [osd:tableRef](#) [p 471] declaration.

The standard XPath syntax has been extended so as to extract the value of any targeted primary key field.

Example

If the table `/root/tableA` has an `osd:tableRef` field named 'fkB' whose target is `/root/tableB` and the primary key of `tableB` has two fields, `id` of type `xs:int` and `date` of type `xs:date`, then the following expressions would be valid:

- `/root/tableB[fkB = '123|2008-01-21']`, where the string "123|2008-01-21" is a representation of the entire primary key value.

See **Syntax of the internal String representation of primary keys** `PrimaryKeyAPI` for more information.

- `/root/tableB[fkB/id = 123 and date-equal(fkB/date, '2008-01-21')]`, where this predicate is a more efficient equivalent to the one in the previous example.
- `/root/tableB[fkB/id >= 123]`, where any number operator could be used, as the targeted primary key field is of type `xs:int`.
- `/root/tableB[date-greater-than(./fkB/date, '2007-01-01')]`, where any date operator could be used, as the targeted primary key field is of type `xs:date`;
- `/root/tableB[fkB = ""]` is not valid as the targetted primary key has two columns.
- `/root/tableB[osd:is-null(fkB)]` checks if a foreign key is null (notdefined).

Localisation

CHAPITRE 44

Labeling and localization

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Value formatting policies](#)
3. [Syntax for locales](#)

44.1 Overview

EBX5 offers the ability to handle the labeling and the internationalization of data models.

Localizing user interactions

In EBX5, language preferences can be set for two scopes:

1. Session: Each user can select a default locale on the user profile page.
2. Data model: If a data model has been localized into other languages than those natively supported by EBX5, the user can select one of those languages for that particular data model. See [Extending EBX5 internationalization](#) [p 277] for more information.

The languages supported by a particular data model must be specified in its [module declaration](#) [p 453].

Textual information

In EBX5, most master data entities can have a label and a description, or can correspond to a user message. For example:

- Data spaces, snapshots and data sets can have their own label and description. The label is independent of the unique name, so that it remains localizable and modifiable;
- Any node in the data model can have a static label and description;
- Values can have a static label when they are enumerated;
- Validation messages can be customized, and permission restrictions can provide text explaining the reason;
- Each record is dynamically displayed according to its content, as well as the context in which it is being displayed (in a hierarchy, as a foreign key, etc.);

All this textual information can be localized into the locales that are declared by the module.

Voir aussi[Labels and messages](#) [p 503][Tables declaration](#) [p 467][Foreign keys declaration](#) [p 471]

44.2 Value formatting policies

When a value is displayed to the user, it is formatted according to its type and the formatting policy of the current locale. For example, a date will be displayed in some locales as "dd/MM/yyyy" and "MM/dd/yyyy" in others.

A formatting policy is used to define how to display the values of [simple types](#) [p 458].

For each locale declared by the module, its formatting policy is configured in a file located at /WEB-INF/ebx/{locale}/frontEndFormattingPolicy.xml. For instance, to define the formatting policy for Greek (el), the engine looks for the following path in the module:

```
/WEB-INF/ebx/el/frontEndFormattingPolicy.xml
```

If the corresponding file does not exist, the formatting policy is looked up in the root module, ebx-root-1.0. If the locale-specific formatting policy is not declared in root module, the formatting policy of en_US is applied.

The content of the file frontEndFormattingPolicy.xml is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<formattingPolicy xmlns="urn:ebx-schemas:formattingPolicy_1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ebx-schemas:formattingPolicy_1.0 ../schema/ebx-reserved/formattingPolicy_1.0.xsd">
  <date pattern="dd/MM" />
  <time pattern="HH:mm:ss" />
  <dateTime pattern="dd/MM/yyyy HH:mm" />
  <decimal pattern="00,00,00.000" />
  <int pattern="000,000" />
</formattingPolicy>
```

The elements date, dateTime and time are mandatory.

44.3 Syntax for locales

There are two ways to express a locale:

1. The XML recommendation follows the [IETF BCP 47](#) recommendation, which uses a hyphen '-' as the separator.
2. The Java specification uses an underscore '_' instead of a hyphen.

In any XML file (XSD, formatting policy file, etc.) read by EBX5, either syntax is allowed.

For a web path, that is, a path within the web application, only the Java syntax is allowed. Thus, formatting policy files must be located in directories whose locale names respect the Java syntax.

Voir aussi [Extending EBX5 internationalization](#) [p 277]

CHAPITRE 45

Extending EBX5 internationalization

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Extending EBX5 user interface localization](#)
3. [Known limitations](#)

45.1 Overview

EBX5 offers the following localization capabilities:

1. The ability to internationalize of labels and specific data models, as described in section [Labeling and Localization](#) [p 274],
2. The extensibility of EBX5 user interface localization

EBX5 native localization

By default, EBX5 provides its built-in user interface and messages in:

1. English (en-US),
2. French (fr-FR).

The built-in localized contents are provided 'as-is', and cannot be changed.

45.2 Extending EBX5 user interface localization

EBX5 now supports the localization of its user interfaces into any compatible language and region.

Note

Currently, Latin & Cyrillic characters are supported. Locales that use other character sets may be usable, but are not supported.

Adding a new locale

In order to add a new locale, the following steps must be performed:

- Declare the new locale in the EBX5 main configuration file. For example:

`ebx.locales.available=en-US, fr-FR, xx`

- The first locale is always considered the default.
- The built-in locales, en-US and fr-FR, may be removed if required.

See [EBX5 main configuration file](#) [p 361].

- Deploy the following files in the EBX5 class-path:

- A formatting policy file, named `com.orchestranetworks.i18n.frontEndFormattingPolicy_xx.xml`,
- A set of localized message files in a resource bundle.

Specific localization deployment

Specific localized files can be deployed as a JAR file alongside `ebx.jar`.

45.3 Known limitations

Default localization

The EBX5 'en-US' and 'fr-FR' localizations cannot be modified.

Non extendable materials

Localization of the following cannot be extended:

- EBX5 product documentation,
- EBX5 HTML editor and viewer,

Persistence

CHAPITRE 46

Overview of persistence

This chapter describes how master data, history, and replicated tables are persisted. A given table can employ any combination of master data persistence mode, historization, and replication.

While all persisted information in EBX5 is ultimately stored as relational tables in the underlying database, whether it is in a form that is accessible outside of EBX5 depends on if it is in mapped mode.

Note

The term *mapped mode* [p 281] refers to any tables that are stored as-is, and thus whose contents can be accessed directly in the database.

Ce chapitre contient les sections suivantes :

1. [Persistence of managed master data](#)
2. [Historization](#)
3. [Replication](#)
4. [Mapped mode](#)

46.1 Persistence of managed master data

Data that is modeled in and governed by the EBX5 repository can be persisted in one of two modes, semantic (default) or relational, as specified in its underlying data model. Distinct tables defined in either mode can co-exist and collaborate within the same EBX5 repository.

For a comparison between relational mode and semantic mode, see the chapter [Overview of modes](#) [p 283]

46.2 Historization

Master data tables can activate historization in order to track modifications to their data, regardless of whether they are persisted in semantic or relational mode, and whether they are replicated.

The history itself is in mapped mode, meaning that it can potentially be consulted directly in the underlying database.

Voir aussi [History](#) [p 289]

46.3 Replication

Replication enables direct SQL access to tables of master data, by making a copy of data in the repository to relational table replicas in the database. Replication can be enabled any table regardless of whether it is persisted in semantic or relational mode, and whether it has history activated.

The replica tables are persisted in mapped mode, as their primary purpose is to make master data accessible to direct queries outside of EBX5.

Voir aussi [Replication](#) [p 297]

46.4 Mapped mode

Overview of mapped mode

Mapped mode refers to cases where tables are persisted in the underlying relational database in a format that allows their data to be accessed directly, outside of EBX5. Master data modeled in relational mode, history, and replica tables are all examples of tables in mapped mode.

All cases of mapped mode involve automatic alterations of the database schema (the database tables, indexes, etc.) when necessary, by automatically executing required DDL statements in the background. Such procedures are always triggered at data model compilation time and the data model compilation report notifies of any resulting errors.

Another general consideration regarding mapped modes is that, in most cases, when a data model entity is removed, its corresponding database object is not deleted immediately. Instead, it is marked as disabled, which leaves the possibility of later re-enabling the object. In order to definitively drop the object and its associated data and resources from the database, it must be marked for purge. The removal then takes place during the next global purge.

Voir aussi

[Purging database resources of mapped tables](#) [p 388]

[Data model evolutions](#) [p 303]

Data model restrictions due to mapped mode

Due to the nature of persisting directly in the underlying database, some restrictions apply to all tables stored in mapped mode:

- [Limitations of supported databases](#) [p 345]
- Unlimited-length strings: All string fields, except foreign keys, of type `xs:string`, its derived types, and `xs:anyURI` must define a 'maxLength' or 'length' facet. Since a foreign key field is composed of the final primary key field(s) of its target table(s), this facet requirement applies to each of those final primary key fields instead of the foreign key field itself. Additionally, limitations of the underlying database concerning the maximum length of its character types apply, such as *VARCHAR* and *NVARCHAR2*.
- Large lists of columns might not be indexable. Example for Oracle: the database enforces a limit on the maximum cumulated size of the columns included in an index. For strings, this size also depends on the character set. If the database server fails to create the index, you should consider redesigning your indexes, typically by using a shorter length for the concerned columns, or by including fewer columns in the index. The reasoning is that an index leading to this situation would have headers so large that it could not be efficient anyway.
- The attribute `type="osd:password"` is ignored.
- Terminal complex types are supported, however at record-level, they cannot be globally set to `null`.

More generally, tables in mapped mode are subject to any limitations of the underlying RDBMS. For example, the maximum number of columns in a table applies (1000 for Oracle 11g R2, 1600 for PostgreSQL). Note that a history table contains twice as many fields as declared in the schema (one functional field, plus one generated field for the operation code).

Data model evolutions may also be constrained by the underlying RDBMS, depending on the existing data model.

Voir aussi [Data model evolutions](#) [p 303]

CHAPITRE 47

Relational mode

Ce chapitre contient les sections suivantes :

1. [Overview of modes](#)
2. [Enabling relational mode for a data model](#)
3. [Validation](#)
4. [SQL access to data in relational mode](#)
5. [Limitations of relational mode](#)

47.1 Overview of modes

Semantic mode explained

Semantic mode offers all EBX5 advanced features of master data management, in particular, data spaces, data set inheritance, and inherited fields.

Semantic mode is the default mode for persisting the data governed by the EBX5 repository. Data models are in semantic mode unless [relational mode](#) [p 284] is explicitly [specified](#) [p 284].

Internally, the master data managed in semantic mode is represented as standard XML, which complies with the XML Schema Document of its data model. The XML representation is additionally compressed and segmented for storage into generic relational database tables. This mode provides efficient data storage and access, including for:

- Data spaces: no data is duplicated when creating a child data space, and
- Inheritance: no data is duplicated when creating an inherited instance.

Semantic mode also makes it possible to maintain an unlimited number of data sets for each data model, organized into an unlimited number of data spaces or snapshots. This can be done with no impact on the database schema.

As this mode only uses common, generic internal tables, modifications to the structure of the data model also never impact the database schema. Data model evolutions only impact the content of the generic database tables.

Voir aussi

[data spaces](#) [p 78]

[data set inheritance](#) [p 311]

[inherited fields](#) [p 313]

Relational mode explained

Relational mode, which is a mapped mode, persists master data directly into the database. The primary function of relational mode is to be able to benefit from the performance and scalability capabilities of the underlying relational database. However, relational mode does not support the advanced governance features offered by semantic mode.

For some cases where the management advantages of semantic mode are not necessary, such as "current time" tables, or tables that are regularly updated by external systems, the performance gains offered by relational mode may be more valuable.

Generally, when a data set is in relational mode, every table in this data set has a corresponding table in the database and every field of its data model is mapped to a relational table column.

Direct comparison of semantic and relational modes

This table summarizes the differences between the two persistence modes:

	Semantic mode	Relational mode
Data spaces	Yes	No
Data set inheritance	Yes	No
Inherited fields	Yes	No
Data model	All features are supported.	Some restrictions, see Data model restrictions for tables in relational mode [p 287].
Direct SQL reads	No	Yes, see SQL reads [p 287].
Direct SQL writes	No	Yes, but only under precise conditions, see SQL writes [p 287].
Data validation	Yes, enables tolerant mode.	Yes, some constraints become blocking, see Validation [p 285].
Transactions	See Concurrency and isolation levels [p 519].	See Concurrency and isolation levels [p 519].

47.2 Enabling relational mode for a data model

The data model declares that it is in relational mode. Due to the necessary restrictions of relational mode, such as not having data spaces or snapshots, a specific relational data space must be provided, to which the data model will be published. Relational data spaces do not allow creating sub-data spaces or snapshots.

Example of a relational mode declaration:

```
<xs:schema>
<xs:annotation>
```

```

<xs:appinfo>
  <osd:relationalMode>
    <dataSpace>aDataSpaceKey</dataSpace>
    <dataSet>aDataSetReference</dataSet>
    <tablesPrefix>aPrefixForTablesInRDBMS</tablesPrefix>
  </osd:relationalMode>
</xs:appinfo>
</xs:annotation>
...
</xs:schema>

```

with the elements:

Element	Description	Required
dataSpace	Specifies the data space where the data model must be published. This data space must itself be in relational mode. No data space or snapshot can be created from a data space declared in such a mode.	Yes
dataSet	Specifies the data set where the data model must be published.	Yes
tablesPrefix	Specifies the common prefix used for naming the generated tables in the database.	Yes

47.3 Validation

This section details the impact of relational mode on data validation.

Structural constraints

Some EBX5 data model constraints will generate a "structural constraint" on the underlying RDBMS schema for relational mode and also if [table history is activated](#) [p 289]. This concerns the following facets:

- facets `xs:maxLength` and `xs:length` on string elements;
- facets `xs:totalDigits` and `xs:fractionDigits` on `xs:decimal` elements.

Databases do not support as tolerant a validation mode as EBX5. Hence, the above constraints become *blocking constraints*. A blocking constraint means that updates are rejected if they do not comply. Additionally, such constraints are no longer checked during validation process, except for foreign key constraints under some circumstances (see [Foreign key blocking mode](#) [p 286]). When a transaction does not comply with a blocking constraint, it is cancelled and a `ConstraintViolationException`^{API} is thrown.

Voir aussi [Blocking and non-blocking constraints](#) [p 494]

Foreign key blocking mode

The foreign key constraints defined on a table in relational mode and referencing a table in relational mode are also in blocking mode, so as to reduce validation time. For this constraint, blocking mode implies that attempting the following actions will result in a `ConstraintViolationException`^{API}:

- Deleting a record referenced by a foreign key constraint,
- Deleting an instance referenced by a foreign key constraint,
- Closing a data space referenced by a foreign key constraint.

However, in order to ensure the integrity of foreign key constraints after direct SQL writes that bypass the EBX5 governance framework, the foreign key constraints will be validated on the following cases:

- On the first explicit validation through the user interface or API,
- On the first explicit validation through the user interface or API after refreshing the schema,
- On the first explicit validation through the user interface or API after resetting the validation report of a data set in the user interface.

Important:

- Blocking aspect of the foreign key constraint does not concern filters that may be defined. That is, a foreign key constraint is non-blocking if a referenced record exists but does not satisfy a foreign key filter. In this case, updates are not rejected and thus an error will be added to the validation report.
- Foreign key constraints are not in blocking mode upon archive import. Indeed, all blocking constraints, excepted structural constraints, are always disabled when importing archives. This allows flexibility upon archive import where under certain circumstances the import of foreign keys referencing records that are not yet imported must be tolerant.

Constraints on the whole table

Programmatic **constraints** `Constraint`^{API} are checked on each record of the table at validation time. If the table defines millions of records, this becomes a performance issue. It is then recommended to define a **table-level constraint** `ConstraintOnTable`^{API}.

In the case where it is not possible to define such a table-level constraint, it is recommended to at least define a **local or explicit dependency** `DependenciesDefinitionContext`^{API}, so as to reduce the cost of incremental validation.

Voir aussi `ConstraintOnTable`^{API}

47.4 SQL access to data in relational mode

This section describes how to directly access the data in relational mode, through SQL.

Voir aussi [SQL access to history](#) [p 292]

Finding the table in the database

For every EBX5 table in relational mode, a corresponding table is generated in the RDBMS. Using the EBX5 user interface, you can find the name of this database table by clicking on the documentation pane of the table.

SQL reads

Direct SQL reads are possible in well-managed, preferably short-lived transactions. However, for such accesses, EBX5 permissions are not taken into account. As a result, applications given allowed to perform reads must be trusted through other authentication processes and permissions.

SQL writes

Direct SQL writes bypass the governance framework of EBX5. Therefore, they must be used with **extreme caution**. They could cause the following situations:

- failure to historize EBX5 tables;
- failure to execute EBX5 triggers;
- failure to verify EBX5 permissions and constraints;
- modifications missed by the incremental validation process;
- losing visibility on EBX5 semantic tables, which might be referenced by foreign keys.

Consequently, direct SQL writes are to be performed *if, and only if, all the following conditions are verified*:

- The written tables are not historized and have no EBX5 triggers.
- The application performing the writes can be fully trusted with the associated permissions, to ensure the integrity of data. Specifically, the integrity of foreign keys (`osd:tableRef`) must be preserved at all times. See [Foreign key blocking mode](#) [p 286] for more information.
- The application server running EBX5 is shut down *whenever writes are performed*. This is to ensure that incremental validation does not become out-of-date, which would typically occur in a batch context.

47.5 Limitations of relational mode

The relational mode feature is fully functional, but has some known limitations, which are listed below. If using relational mode, it is strongly recommended to read these limitations carefully and to contact Orchestra Networks support in case of questions.

See [Supported databases](#) [p 345] for the databases on which relational mode is supported.

Data model restrictions for tables in relational mode

Some restrictions apply to data models in relational mode:

- [Data model restrictions due to mapped mode](#) [p 282]
- [Aggregated lists](#) [p 463] are not supported in relational tables. Such a schema will cause a compilation error.
- User-defined attributes on relational tables result in data model compilation errors.
- [Data set inheritance](#) [p 311].
- [Inherited fields](#) [p 313].
- Programmatic constraints, since the computation cost of validation would be too high. However, **constraints on tables** `ConstraintOnTableAPT` remain available.

Schema evolutions may also be constrained by the underlying RDBMS, depending on the data already contained in the concerned tables.

Voir aussi [Data model evolutions](#) [p 303]

Other limitations of relational mode

- [Limitations of mapped mode](#) [p 282]
- From a data space containing data sets in relational mode, it is not possible to create child data spaces and snapshots.
- For D3, it is not possible to broadcast a data space defined in relational mode.
- For very large volumes of data, the validation will show poor performance if the relational table declares any of these features: `osd:function`, `osd:select`, `osd:uiFilter`, `osd:tableRef/filter`. Additionally, a sort cannot be applied on a `osd:function` column.
- It is not possible to set the `AdaptationValue.INHERIT_VALUE` to a node belonging to a data model in relational mode.

CHAPITRE 48

History

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Configuring history](#)
3. [History views and permissions](#)
4. [SQL access to history](#)
5. [Impacts and limitations of historized mode](#)

48.1 Overview

History is a feature allowing to track all data modifications on a table (records creation, update and deletion).

Some RDBMS are not supported yet (see [Limitations of the historized mode](#) [p 294] for more details). [XML audit trail](#) [p 411] can then be used as an alternative. XML audit trail is activated by default; it can be safely deactivated if your RDBMS is supported.

Voir aussi

[Historique](#) [p 24]

[Relational mode](#) [p 283]

[Replication](#) [p 297]

[Data model evolutions](#) [p 303]

48.2 Configuring history

In order to activate historization for a table, a history profile has to be set for the table in the data model. This section describes history profiles and the way they are associated with tables.

Configuring history in the repository

A history profile specifies when the historization is to be created. In order to edit history profiles, select **Administration > History and logs**.

A history profile is identified by a name and defines the following information:

- An internationalized label.
- A list of data spaces (branches) for which history is activated. It is possible to specify whether direct children and/or all descendants should also be concerned.

Some profiles are already created when installing the repository. These profiles can neither be deleted nor modified.

Profile Id	Description
ebx-referenceBranch	This profile is activated only on the reference data space.
ebx-allBranches	This profile is activated on all data spaces.
ebx-instanceHeaders	This profile historizes data set headers. However, this profile will only be setup in a future version, given that the internal data model only defines data set nodes.

Configuring history in the data model

Activating table history

History can be activated on a table either through the data model assistant, or by editing the underlying data model.

To activate history by editing the data model, a history profile should be declared on the table using the `historyProfile` element.

```
<osd:table>
  <primaryKeys>/key</primaryKeys>
  <historyProfile>historyProfileForProducts</historyProfile>
</osd:table>
```

The data model assistant allows you to view the historization profiles defined in the repository.

Historization must be activated for each table separately. See [model design](#) [p 450] documentation for more details.

Disabling history on a specific field or group

For a historized table, the default behavior is to historize all its supported elements (see [Impacts and limitations of historized mode](#) [p 294]).

It is possible to disable history for a given field or group, either through the data model assistant, or by editing the underlying data model.

To disable the history of a field or group by editing the data model, use the element `osd:history` with the attribute `disable="true"`.

```
<xs:element name="longDescription" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:history disable="true" />
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

To disable the history of a field or group through the data model assistant, use the `History` property in the `Advanced` properties of the element.

When this property is defined on a group, history is disabled recursively for all its descendents. Once a group disables history, it is not possible to specifically re-enable history on a descendant.

Note

If the table containing the field or group is not historized, this property will not have any effect.
It is not possible to disable history for primary key fields.

Integrity

If problems are detected at data model compilation, warning messages or error messages will be added to the validation report associated with this data model. Furthermore, if any error is detected, each associated instance (data set) will be inaccessible. The most common error cases are the following:

- A table references a profile that is not defined in the repository.
- A history profile that is referenced in the data model mentions a non-defined or closed data space in the current repository.

Note

Deploying a data model on a repository that does not have the expected profiles requires the administrator to add them.

48.3 History views and permissions

Table history view

When the history has been activated on a table in the data model, it is possible to access the history view from various locations in the user interface: record, selection of records, table and data set.

The next section explains how permissions are resolved.

For more information, see [vue historique de table](#) [p 25] section. To access the table history view from Java, the method `AdaptationTable.getHistoryAPI` must be invoked.

Permissions for table history

Data permissions are also applied to data history. History permissions are resolved automatically as the most restricted permission between data permissions and *read-only* access right.

This is true for user-defined permission rules and also for programmatic permission rules.

When defining a programmatic rule, it may be required to distinguish between the functional data set context and the history view context, either because the expected permissions are not the same, or because some fields are not present in the history structure. This is the case for data set fields, computed values and [fields for which history has been disabled](#) [p 290]. The methods `Adaptation.isHistoryAPI` and `AdaptationTable.getHistoryAPI` can then be used in the programmatic rule in order to implement specific behavior for history.

Transaction history views

The transaction history view gives access to the executed transactions, independently of a table, a data set or a data model, directly from the user interface.

To see the 'Transaction history' table, navigate to the Administration area and select 'History and logs' using the down arrow menu in the navigation pane. Transaction history can also be accessed from the Data Spaces area by selecting a historized data space and using the **Actions** menu in the workspace.

For more information, see [vue historique des transactions](#) [p 25].

48.4 SQL access to history

This section describes how to directly access the history data by means of SQL.

Voir aussi [SQL access to data in relational mode](#) [p 286]

Access restrictions

The database tables must be accessed only in read-only mode. It is up to the database administrator to forbid write access except for the database user used by EBX5, as specified in the section [Rules for the access to the database and user privileges](#) [p 381].

Relational schema overview

Here is a description of the history tables in the database.

The database schema contains (see also the diagram in the next section):

Common and generic tables	<p>The main table is HV_TX; each record of this table represents a transaction. Only transactions that involve at least one historized table are recorded.</p> <p>These common tables are all prefixed by "HV".</p>
Specific generated tables	<p>For each historized table, a specific history table is generated. This table contains the history of the data modifications on the table.</p> <p>In the EBX5 user interface, the name of this table in database can be obtained by clicking on the table documentation pane (advanced mode). All the specific history tables are prefixed with "HG".</p>

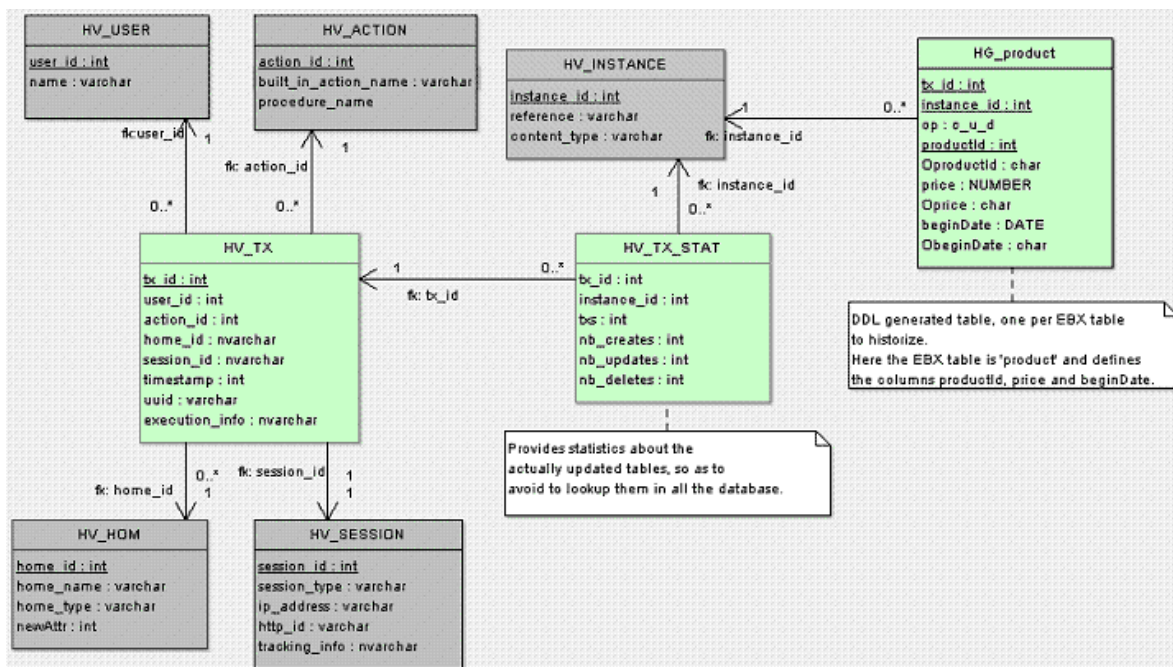
Example of a generated history table

In the following example, we are historizing a table called product. Let us assume this table declares three fields in EBX5 data model:

Product

- productId: int
- price: int
- beginDate: Date

The diagram below shows the resulting relational schema:



Activating history on this table generates the HG_product table shown in the history schema structure above. Here is the description of its different fields:

- tx_id: transaction ID.
- instance: instance ID.
- op: operation type - C (create), U (update) or D (delete).
- productid: productid field value.
- Oproductid: operation field for productid, see next section.
- price: price field value.
- Oprice: operation field for price, see next section.
- beginDate: date field value.
- ObeginDate: operation field for beginDate, see next section.

Operation field values

For each functional field, an additional operation field is defined, composed by the field name prefixed by the character O. This field specifies whether the functional field has been modified. It is set to one of the following values:

- null: if the functional field value has not been modified (and its value is not INHERIT).
- M: if the functional field value has been modified (not to INHERIT).
- D: if record has been deleted.

If [inheritance](#) [p 310] is enabled, the operation field can have three additional values:

- T: if the functional field value has not been modified and its value is INHERIT.
- I: if the functional field value has been set to INHERIT.
- O: if the record has been set to OCCULTING mode.

48.5 Impacts and limitations of historized mode

The history feature has some impacts and known limitations, which are listed in this section. If using historized mode, it is strongly recommended to read these limitations carefully and to contact Orchestra Networks support in case of questions.

See [Supported databases](#) [p 345] for the databases on which historized mode is supported.

Validation

Some EBX5 data model constraints become blocking constraints when table history is activated. For more information, see the section [Structural constraints](#) [p 285].

Data model restrictions for historized tables

Some restrictions apply to data models containing historized tables:

- [Data model restrictions due to mapped mode](#) [p 282]
- Limitations exist for two types of aggregated lists: aggregated lists under another aggregated list, and aggregated lists under a terminal group. Data models that contain such aggregated lists can be used, however these list will be ignored (not historized).
- Computed values are ignored.
- User-defined attributes on historized tables result in data model compilation errors.

Data model evolutions may also be constrained by the underlying RDBMS, depending on the data already contained in the concerned tables.

Voir aussi [Data model evolutions](#) [p 303]

Other limitations of historized mode

- No data copy is performed when a table with existing data is activated for history.
- Global operations on data sets are not historized (create an instance and remove an instance), even if they declare an historized table.
- Default labels referencing a non-historized field are not supported for historized tables.

As a consequence, default labels referencing a computed field are not supported for historized tables.

The workaround is to implement the `UILabelRenderer` interface and adapt the label computation for history.

- D3: the history can be enabled in the delivery data space of a master node, but in the delivery data space of the slave nodes, the historization features are always disabled.
- Recorded user in history: for some specific operations, the user who performs the last operation and the one recorded in the corresponding history record may be different.

This is due to the fact that these operations are actually a report of the data status at a previous state:

- Archive import: when importing an archive on a data space, the time and user of the last operation performed in the child data space are preserved, while the user recorded in history is the user who performs the import.
- Programmatic merge: when performing a programmatic merge on a data space, the time and user of the last operation performed in the child data space are preserved, while the user recorded in history is the user who performs the merge.
- D3: for distributed data delivery feature, when a broadcast is performed, the data from the master node is reported on the slave node and the time and user of the last operation performed in the child data space are preserved, while the user recorded in history is 'ebx-systemUser' who performs the report on the slave node upon the broadcast.

CHAPITRE 49

Replication

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Configuring replication](#)
3. [Accessing a replica table using SQL](#)
4. [Requesting an 'onDemand' replication refresh](#)
5. [Impact and limitations of replication](#)

49.1 Overview

Data stored in the EBX5 repository can be mirrored to dedicated relational tables to enable direct access to the data by SQL requests and views.

Like history and relational mode, this data replication is transparent to end-users and client applications. Certain actions trigger automatic changes to the replica in the database:

- Activating replication at the model-level updates the database schema by automatically executing the necessary DDL statements.
- Data model evolutions that impact replicated tables, such as creating a new column, also automatically update the database schema using DDL statements.
- When using the 'onCommit' refresh mode: updating data in the EBX5 repository triggers the associated inserts, updates, and deletions on the replica database tables.

Voir aussi

[Relational mode](#) [p 283]

[History](#) [p 289]

[Data model evolutions](#) [p 303]

[Repository administration](#) [p 380]

49.2 Configuring replication

Enabling replication

To define a *replication unit* on a data model, use the element `osd:replication` under the elements `annotation/appinfo`. Each replication unit specifies tables in a single data set in a specific data space.

The nested elements are as follows:

Element	Description	Required
<code>name</code>	Nom de l'unité de réplication. Ce nom identifie l'unité de réplication dans le modèle de données. Ce nom doit être unique.	Yes
<code>dataSpace</code>	Indique l'espace de données concerné par la réplication. Cet espace de données ne peut ni être une version ni être relationnel.	Yes
<code>dataSet</code>	Indique le jeu de données concerné par la réplication.	Yes
<code>refresh</code>	Specifies the data synchronization policy. The possible policies are: <ul style="list-style-type: none"> <code>onCommit</code>: The replica table content in the database is always up to date with respect to its source table. Every transaction that updates the EBX5 source table triggers the corresponding insert, update, and delete statements on the replica table. <code>onDemand</code>: The replication of specified tables is only done when an explicit refresh operation is performed. See Requesting an 'onDemand' replication refresh [p 300]. 	Yes
<code>table/path</code>	Indique le chemin de la table dans le modèle de données qui doit être répliquée dans la base de données.	Yes
<code>table/nameInDatabase</code>	Indique le nom de la table dans la base de données qui contiendra les données répliquées. Ce nom doit être unique par rapport à toutes les unités de réplication.	Yes

For example:

```
<xs:schema>
<xs:annotation>
  <xs:appinfo>
    <osd:replication>
      <name>ProductRef</name>
      <dataSpace>ProductReference</dataSpace>
      <dataSet>productCatalog</dataSet>
      <refresh>onCommit</refresh>
      <table>
        <path>/root/domain1/tableA</path>
        <nameInDatabase>PRODUCT_REF_A</nameInDatabase>
      </table>
      <table>
        <path>/root/domain1/tableB</path>
        <nameInDatabase>PRODUCT_REF_B</nameInDatabase>
      </table>
    </osd:replication>
  </xs:appinfo>
</xs:annotation>
...
```

```
</xs:schema>
```

Notes:

- See [Data model restrictions for replicated tables](#) [p 301]
- If, at data model compilation, the specified data set and/or data space does not exist in the current repository, a warning is reported, but the replica table is created in the database. Once the specified data space and data set are created, the replication becomes active.
- At data model compilation, if a table replication is removed, or if some of the above properties has changed, the replica table is dropped from the database, and then recreated with the new definition if needed.

Disabling replication on a specific field or group

For a replicated table, the default behavior is to replicate all its supported elements (see [Data model restrictions for replicated tables](#) [p 301]).

It is possible to disable replication for a specific field or group, either through the data model assistant, or by editing the underlying data model.

To disable the replication of a field or group by editing the data model, use the element `osd:replication` with the attribute `disable="true"`.

```
<xs:element name="longDescription" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:replication disable="true" />
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

To disable the replication of a field or group through the data model assistant, use the Replication property in the Advanced properties of the element.

When this property is defined on a group, replication is disabled recursively for all its descendents. Once a group disables replication, it is not possible to specifically re-enable replication on a descendant.

Note

If the table containing the field or group is not replicated, this property will not have any effect.
It is not possible to disable replication for primary key fields.

49.3 Accessing a replica table using SQL

This section describes how to directly access a replica table using SQL.

Voir aussi [SQL access to history](#) [p 292]

Finding the replica table in the database

For every replicated EBX5 table, a corresponding table is generated in the RDBMS. Using the EBX5 user interface, you can find the name of this database table by clicking on the documentation pane of the table.

Access restrictions

The replica database tables must only be directly accessed in read-only mode. It is the responsibility of the database administrator to block write-access to all database users except the one that EBX5 uses.

Voir aussi [Rules for the access to the database and user privileges](#) [p 381]

SQL reads

Direct SQL reads are possible in well-managed, preferably short-lived transactions. However, for such accesses, EBX5 permissions are not taken into account. As a result, applications given the privilege to perform reads must be trusted through other authentication processes and permissions.

49.4 Requesting an 'onDemand' replication refresh

The 'onDemand' refresh policy requires an explicit request to refresh the replicated table data.

There are several ways to request a replication refresh:

- **User interface:** In the data set actions menu, use the action 'Rafraîchir les répliques' under the group 'Réplication' to launch the replication refresh wizard.
- **Data services:** Use the replication refresh data services operation. See [Replication refresh](#) [p 250] for data services for more information.
- **Java API:** Call the `ReplicationUnit.performRefreshAPI` methods in the `ReplicationUnit` API to launch a refresh of the replication unit.

49.5 Impact and limitations of replication

The replication feature has some known limitations and side-effects, which are listed below. If using replication, it is strongly recommended to read this section carefully and to contact Orchestra Networks support in case of questions.

See [Supported databases](#) [p 345] for the databases for which replication is supported.

Validation

Some EBX5 data model constraints become blocking constraints when replication is enabled. For more information, see [Structural constraints](#) [p 285].

Data model restrictions for replicated tables

Some restrictions apply to data models containing tables that are replicated:

- [Data model restrictions due to mapped mode](#) [p 282]
- Data set inheritance is not supported for the 'onCommit' refresh policy if the specified data set is not a root data set or has not yet been created. See [data set inheritance](#) [p 311] for more information.
- Field inheritance is also only supported for the 'onDemand' refresh policy. This means that, at data model compilation, an error is reported if the refresh mode is 'onCommit' and the table to be replicated has an inherited field. See [inherited fields](#) [p 313] for more information.
- Computed values are ignored.
- Limitations exist for two types of aggregated lists: aggregated lists under another aggregated list, and aggregated lists under a terminal group. Data models that contain such aggregated lists can be used, however these list will be ignored (not replicated).
- User-defined attributes are not supported. A compilation error is raised if they are included in a replication unit.
- It is currently not supported to include the same table in several replication units. In a future version, it will be possible to include a table in at most two units, one unit having the refresh policy 'onDemand' and the other having 'onCommit'.

Data model evolutions may also be constrained by the underlying RDBMS, depending on the data already contained in the concerned tables.

Voir aussi [Data model evolutions](#) [p 303]

Database configuration

The refresh operation is optimized to transmit only the rows of the source table that have been modified (with respect to creation and deletion) since the last refresh. However, depending on the volume of data exchanged, this can be an intensive operation, requiring large transactions. In particular, the first refresh operation can concern a large number of rows. It is necessary for the database to be configured properly to allow such transactions to run under optimal conditions.

For instance, with Oracle:

- It is mandatory for the whole replica table to fit into the 'UNDO' tablespace.
- It is recommended to provide enough space in the buffer cache to allow those transactions to run with minimal disk access.
- It is recommended to provision 'REDO' log groups big enough to avoid those transactions to wait for the 'db_writer' process.

Distributed data delivery (D3)

Replication is available on both D3 master and slave delivery data spaces. On master, the replication behavior is the same as in a standard semantic data space, but on slaves, the replicated content is that of the last broadcast snapshot.

In a slave delivery data space, some restrictions occur:

- The refresh policy defined in the data model has no influence on the behavior described above: replication always happens on snapshot.
- The action item Refresh replicas is not available.
- It is not allowed to invoke the `ReplicationUnit.performRefreshAPI` method.

Voir aussi [D3 overview](#) [p 428]

Other limitations of replication

- [Limitations of supported databases](#) [p 345]
- For inheritance, a replica record field cannot hold the "inherit value" flag (`AdaptationValue.INHERIT_VALUE`). It only holds the inherited value in such cases. More generally, it is not possible to distinguish inheriting state from overwriting state.

CHAPITRE 50

Data model evolutions

This chapter describes the modifications that are possible on data models, as well as potential limitations. Most limitations apply to the mapped modes.

Note

Certain types of data model evolutions cannot be performed directly in the user interface, and thus the data model must be exported, modified in XSD format, then re-imported. For changes to a data model that impact its configuration, not just its structure, the XSD must be imported into EBX5 from a module. Otherwise, the configuration modifications are not taken into account.

Voir aussi [Mapped mode](#) [p 281]

Ce chapitre contient les sections suivantes :

1. [Types of permitted evolutions](#)
2. [Limitations/restrictions](#)

50.1 Types of permitted evolutions

This section describes the possible modifications to data models after their creation.

Model-level evolutions

The following modifications can be made to existing data models:

- A data model in semantic mode can be declared to be in relational mode. Data should be manually migrated, by exporting then re-importing an XML or archive file.
- Relational mode can be disabled on the data model. Data should be manually migrated, by exporting then re-importing an XML or archive file.
- Replication units can be added to the data model. If their refresh policy is 'onCommit', the corresponding replica tables will be created and refreshed on next schema compilation.
- Replication units can be removed from the data model. The corresponding replica tables will be dropped immediately.
- The data model can be deleted. If it declares replication units, the corresponding replica tables will be dropped immediately. If it is relational or contains historized tables, this change marks the associated mapped tables as disabled. See [Purging database resources of mapped tables](#) [p 388] for the actual removal of associated database objects.

Table-level evolutions

The following modifications can be made to a data model at the table-level:

- A new table can be added. Upon creation, the table can also declare one or more mapped modes.
- An existing table can be deleted. If it declares replication units, the corresponding replica tables will be dropped immediately. If it historized or relational, this change marks the mapped table as disabled. See [Purging database resources of mapped tables](#) [p 388] for the actual removal of associated database objects.
- An existing table in semantic mode can be declared to be in relational mode. Data should be manually migrated, by exporting then re-importing an XML or archive file.
- History can be enabled or disabled on a table. History will not take into account the operations performed while it is disabled.
- A table can be renamed. Data should be manually migrated, by exporting then re-importing an XML or archive file, because this change are considered to be a combination of deletion and creation.

Field-level evolutions

The following modifications can be made to a data model at the field-level:

- A new field can be added.
- An existing field can be deleted. In semantic mode, the data of the deleted field will be removed from each record upon its next update. For a replica table, the corresponding column is automatically removed. In history or relational mode, the field is marked as disabled.
- A field can be specifically disabled from the history or replication which applies to its containing table, by using the attribute `disable="true"`. For a replica table, the corresponding column is automatically removed. For a history table, the column remains but is marked as disabled. See [Disabling history on a specific field or group](#) [p 290] and [Disabling replication on a specific field or group](#) [p 299].
- The facets of a field can be modified, except for the facets listed under [Limitations/restrictions](#) [p 305].

The following changes are accepted, but they can lead to loss of data. Data should be manually migrated, by exporting then re-importing an XML or archive file, because these changes are considered to be a combination of deletion and creation.

- A field can be renamed.
- The type of a field can be changed.

Index-level evolutions

- An index can be added or renamed.
- An index can be modified, by changing or reordering its fields. In mapped mode, the existing index is deleted and a new one is created.
- An index can be deleted. In mapped mode, a deleted index is also deleted from the database.

50.2 Limitations/restrictions

Note

All limitations listed in this section that affect mapped mode can be worked around by purging the mapped table database resources. For the procedure to purge mapped table database resources, see [Purging database resources of mapped tables](#) [p 388].

Limitations related to primary key evolutions

When a primary key definition is modified:

- In semantic mode, the existing records are only loaded into the cache if they:
 - Respect the uniqueness constraint of the primary key,
 - Comply with the new structure of the primary key.
- In mapped mode, the underlying RDBMS only accepts a primary key modification if all table records respect its uniqueness and non-nullity constraints. In particular, if a table already has existing records:
 - Adding a new field to the primary key requires assigning a default value to this field. Workaround: first add the field, value it for the existing records, then add the field to the primary key.
 - Removing an existing field from the primary key will be rejected if it would cause the existing records to no longer have a unique primay key (assigning a default value makes no change in this case).
 - It is generally not possible to rename a field of the primary key; more formally, it is only possible if the field was not needed for making all primary keys unique. Indeed, renaming a field translates to a combination of deletion and creation; consequently, the operation will be rejected if it would cause the existing records to no longer have a unique primay key (assigning a default value makes no change in this case).

Limitations related to foreign key evolutions

- When the declaration of a facet `osd:tableRef` is added or modified, or the primary key of the target table of a facet `osd:tableRef` is modified:
 - In semantic mode, the existing values for this field are only loaded into the cache if they comply with the new structure of the target primary key.
 - In mapped mode, the structure of a foreign key field is set to match that of the target primary key. A single field declaring an `osd:tableRef` constraint may then be split into a number of columns, whose number and types correspond to that of the target primary key. Hence, the following cases of evolutions will have an impact on the structure of the mapped table:
 - declaring a new `osd:tableRef` constraint on a table field;
 - removing an existing `osd:tableRef` constraint on a table field;
 - adding (resp. removing) a column to (resp. from) a primary key referenced by an existing `osd:tableRef` constraint;
 - modifying the type or path for any column of a primary key referenced by an existing `osd:tableRef` constraint.

These cases of evolution will translate to a combination of field deletions and/or creations. Consequently, the existing data should be migrated manually.

Limitations related to field-level evolutions

- In mapped mode, when a `maxLength`, `length`, `totalDigits` or `fractionDigits` facet is modified:

Whether or not this modification is accepted depends on the underlying DBMS, as well as the field type and the contents of the table.

For example, Oracle will accept changing a `VARCHAR(20)` to a `VARCHAR(50)`, but will only change a `VARCHAR(50)` to a `VARCHAR(20)` if the table does not contain any values over 20 characters long.

PostgreSQL has the same limitations, but additionally, any modification of a field type (including modifications of its length) will invalidate all related prepared statements, and require restarting the application server.

- When a cardinality of an element is modified:
 - In semantic mode, this change is supported. However, two cases are distinguished:
 - When changing a single element to an aggregated list, the previous single value is preserved and added to the new aggregated list.
 - When changing an aggregated list to a single element, only the last value of the aggregated list is preserved in the single element. The other values are lost.
 - In relational mode, aggregated lists are not supported. An error message is added to the compilation report of the data model if an element is changed to an aggregated list.
 - In historized mode, when changing a single element to an aggregated list, the modification is taken into account, but the previous single value is lost.

Divers

CHAPITRE 51

Inheritance and value resolution

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Data set inheritance](#)
3. [Inherited fields](#)
4. [Optimize & Refactor service](#)

51.1 Overview

The principle of inheritance is to mutualize resources that are shared by multiple contexts or entities. EBX5 offers mechanisms for defining, factorizing and resolving data values: *data set inheritance* and *inherited fields*.

Furthermore, *functions* can be defined to compute values.

Note

Inheritance mechanisms described in this chapter should not be confused with "structural inheritance", which usually applies to models and is proposed in UML class diagrams, for example.

Voir aussi [Inheritance \(glossary\)](#) [p 23]

Data set inheritance

Data set inheritance is particularly useful when data applies to global enterprise contexts, such as subsidiaries or business partners.

Based on a hierarchy of data sets, it is possible to factorize common data into the root or intermediate data sets and define specialized data in specific contexts.

The data set inheritance mechanisms are detailed below in [Data set inheritance](#) [p 311].

Inherited fields

Contrary to data set inheritance, which exploits a global built-in relationships between data sets, inherited fields exploit finer-grained dependencies that are specific to the data structure. It allows factorizing and specializing data at the business entities-level.

For example, if the model specifies that a 'Product' is associated with a 'FamilyOfProducts', it is possible that some attributes of 'Product' inherit their values from the attributes defined in its associated 'FamilyOfProducts'.

Note

It is not possible to use both attribute inheritance and data set inheritance in the same data set.

Computed values (functions)

In the data model, it is also possible to specify that a node holds a *computed value*. In this case, the specified JavaBean function will be executed each time the value is requested.

The function is able to take into account the current context, such as the values of the current record or computations based on another table, and to make requests to third-party systems.

Voir aussi [Computed values](#) [p 498]

51.2 Data set inheritance

Data set inheritance declaration

The data set inheritance mechanism is declared as follows in a data model:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ebxnd="urn:ebx-schemas:binding_1.0">
  <xs:annotation>
    <xs:appinfo>
      <osd:inheritance>
        <dataSetInheritance>all</dataSetInheritance>
      </osd:inheritance>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:schema>
```

The element `osd:inheritance` defines the property `dataSetInheritance` to specify the use of inheritance on data sets based on this data model. The following values can be specified:

- `all`, indicates that inheritance is enabled for all data sets based on the data model.
- `none`, indicates that inheritance is disabled for all data sets based on the data model.

If not specified, the inheritance mechanism is disabled.

Value lookup mechanism

The data set inheritance lookup mechanism for values proceeds as follows:

1. If the value is locally defined, it is returned.
It can be explicitly `null`.
2. Otherwise, looks up the first locally defined value according to the built-in child-to-parent relationship of the data set in the hierarchy of data sets.
3. If no locally defined value is found, the default value is returned.
If no default value is defined, `null` is returned.

Note: Default values cannot be defined on:

- A single primary key node
- Auto-incremented nodes
- Nodes defining a computed value

Record lookup mechanism

Like values, table records can also be inherited as a unit by multiple contexts, but they can also be partially redefined (*overwritten*), defined for a specific context (*root mode*), or be *occulted*.

Formally, a table record has one of four distinct definition modes:

<i>root record</i>	Locally defined in the table and has no parent. This means that no record with the same primary key exists in the parent table, or that this parent is an occulting record.
<i>overwriting record</i>	Locally defined in the table and has a parent record. This means that a record with the same primary key exists in the parent table, and that this parent is not an occulting record. The overwriting record inherits its values from its parent, except for the values that it explicitly redefines.
<i>inherited record</i>	Not locally defined in the current table and has a parent record. All values are inherited. Functions are always resolved in the current record context and are not inherited.
<i>occulting record</i>	Specifies that if a parent with same primary key is defined, this parent will not be visible in table descendants.

Voir aussi [Héritage entre jeux de données](#) [p 121]

Defining inheritance behavior at the table level

It is also possible to specify management rules in the declaration of a table in the data model.

Voir aussi [Properties related to data set inheritance](#) [p 471]

51.3 Inherited fields

The specific inheritance mechanism allows fetching a value of a field according to its relationship to other tables.

Field inheritance declaration

Specific inheritance must be specified on terminal nodes in the underlying data model and is declared as follows:

```
<xs:element name="sampleInheritance" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:inheritance>
        <sourceRecord>
          /root/table1/fkTable2, /root/table2/fkTable3
        </sourceRecord>
        <sourceNode>color</sourceNode>
      </osd:inheritance>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

The element `sourceRecord` is an expression that describes how to look up the record from which the value is inherited. It is a foreign key, or a sequence of foreign keys, from the current element to the source table.

If `sourceRecord` is not defined in the data model, the inherited fields are fetched from the current record.

The element `sourceNode` is the path of the node from which to inherit in the source record.

The following conditions must be satisfied for specific inheritance:

- The element `sourceNode` is mandatory.
- The expression for the path to the source record must be a consistent path of foreign keys, from the current element to the source record. This expression must involve only one-to-one and zero-to-one relationships.
- The `sourceRecord` cannot contain any aggregated list elements.
- Each element of the `sourceRecord` must be a foreign key.
- If the inherited field is also a foreign key, the `sourceRecord` cannot refer to itself to get the path to the source record of the inherited value.
- Every element of the `sourceRecord` must exist.
- The source node must belong to the table containing the source record.
- The source node must be terminal.
- The source node must be writeable.
- The source node type must be compatible with the current node type
- The source node cardinalities must be compatible with those of the current node.
- The source node cannot be the same as the inherited field if the fields to inherit from are fetched into the same record.

Value lookup mechanism

The lookup mechanism for inherited fields values proceeds as follows:

1. If the value is locally defined, it is returned.
It can be explicitly null
2. Otherwise, looks up the source record and value to inherit from, according to the properties that are defined in the data model.
3. The process is recursive; if the source node does not locally define a value, it is then looked up according to the inheritance behavior of the source node.

51.4 Optimize & Refactor service

EBX5 provides a built-in UI service for optimizing data set inheritance in the hierarchy of data sets. This service performs the following functions:

- **Handles duplicated values:** Detects and removes all parameter values that are duplicates of the inherited value.
- **Mutualizes common values:** Detects and mutualizes the common values among the descendants of a common ancestor.

Procedure details

Data sets are processed from the bottom up, which means that if the service is run on the data set at level N , with $N+1$ being the level of its children and $N+2$ being the level of its children's children, the service will first process the data sets at level $N+2$ to determine if they can be optimized with respect to the data sets at level $N+1$. Next, it would proceed with an optimization of level $N+1$ against level N .

Note

- These optimization and refactoring functions do not handle default values that are declared in the data model.
- The highest level considered during the optimization procedure is always the data set on which the service is run. This means that optimization and refactoring is not performed between the target data set and its own ancestors.
- Table optimization is performed on records with the same primary key.
- Inherited fields are not optimized.
- *The optimization and refactoring functions do not modify the resolved view of a data set, if it is activated.*

Service availability

The 'Optimize & Refactor' service is available on data sets that have child data sets and also have the property 'Activated' set to 'No' in its data set information.

The service is available to any profile with write access on current data set values. It can be disabled by setting restrictive access rights on a profile.

Note

For performance reasons, access rights are not verified on every node and table record.

CHAPITRE 52

Permissions

Permissions dictate the access each user has to data and actions.

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Defining permissions and access rights in the user interface](#)
3. [Defining permissions and access rights with programmatic rules](#)
4. [Resolving permissions](#)
5. [Resolving access rights](#)
6. [Resolving permissions for actions and services](#)

52.1 Overview

Permissions are related to whether actions are authorized or not. They are also related to access rights, that is, whether an entity is hidden, read, or read-write. The main entities controlled by permissions are:

- Data space
- Data set
- Table
- Group
- Field

Users, roles and profiles

The definition and resolution of permissions make extensive use of the notion of *profiles*, which is the generic term applied to users or roles.

Each user can participate in several roles, and a role can be shared by several users.

These relationships are defined in the user and roles directory. See [Users and roles directory](#) [p 407].

Special definitions:

- An *administrator* is a member of the built-in role 'ADMINISTRATOR'.
- An *owner of a data set* is a member of the *owner* attribute specified in the information of a root data set. In this case, the built-in role 'OWNER' is activated when permissions are resolved in the context of the data set.
- An *owner of a data space* is a member of the *owner* attribute specified for a data space. In this case, the built-in role 'OWNER' is activated when permissions are resolved in the context of the data space.

Permission rules

A permission rule defines the authorization granted to a profile for a particular entity.

User-defined permission rules are created through the user interface. See the section [Defining permissions and access rights in the user interface](#) [p 320].

Programmatic permission rules can be created by developers. See the section [Defining permissions and access rights with programmatic rules](#) [p 323].

Resolution of permissions

Permissions are always resolved in the context of an authenticated user session. Thus, the defined permission rules can take user's roles into account.

Note

In the Java API, the class `SessionPermissionsAPI` provides access to the resolved permissions.

Voir aussi [Resolving permissions](#) [p 323]

Owner and administrator permissions on a data set

An administrator or owner of a data set can perform the following actions:

- Manage its permissions
- Change its owner, if the data set is a root data set
- Change its localized labels and descriptions

Attention

While the definition of permissions can restrict an administrator or data set owner's right to view data or perform certain actions, it remains possible for them to modify their own access, as they will always have access to permissions management.

Owner and administrator permissions on a data space

To be a *super owner* of a data space, a user must either:

- Own the data space and be allowed to manage its permissions, or
- Own a data space that is an ancestor of the current data space and be allowed to manage the permissions of that ancestor data space.

An administrator or super owner of a data space can perform the following actions:

- Manage its permissions of data space.
- Change its owner
- Lock it or unlock it
- Change its localized labels and descriptions

Attention

While the definition of permissions can restrict an administrator or data space owner's right to view data or perform certain actions, it remains possible for them to modify their own access, as they will always have access to permissions management.

Impact of merge on permissions

When a data space is merged, the permissions of the child data set are merged with those of the parent data space if and only if the user specifies to do so during the merge process. The permissions of its parent data space are never impacted.

If some elements are hidden for the profile attempting to perform a merge, it will not be possible to proceed as the impacts of the merge on data will not be fully visible.

Important considerations about permissions

The following statements should be kept in mind while working with permissions:

- Resolution of custom display labels for tables ('defaultLabel' property) and relationships ('display' property) ignores permission, and fields usually hidden due to access rights restrictions will be displayed in such labels. As a result, these labels should not contain any confidential field. Otherwise, a permission strategy should also be defined to restrict the display of the whole label.
- When a procedure disables all permission checks by using `ProcedureContext.setAllPrivilegesAPI`, the client code must check that the current user session is allowed to run the procedure.
- To increase permissions resolution, a dedicated cache is implemented at the session-level; it only takes user-defined permission rules into account, not programmatic rules (which are not cached since they are contextual and dynamic). The session cache life cycle depends on the context, as described hereafter:
 - In the UI, the cache is cleared for every non-ajax event (i.e on page display, pop-up opening, etc.).
 - In programmatic procedures, the cache lasts until the end of the procedure, unless explicitly cleared (see below).

Attention

When modifying permissions in a procedure context (by importing an EBX5 archive or merging a data space programmatically), the session cache **must** be cleared via a call to `Session.clearCacheAPI`. Otherwise, these modifications will not be reflected until the end of the procedure.

52.2 Defining permissions and access rights in the user interface

Each level has a similar schema, which allows defining permission rules for profiles.

Data space permissions

For a given data space, the allowable permissions for each profile are as follows:

Data space access	Defines access rights as read-write, read-only or hidden.
Restriction	Indicates whether this data space profile-permission association should have priority over other permissions rules.
Create a child data space	Indicates whether the profile can create child data spaces from the current data space.
Create a child snapshot	Indicates whether the profile can create snapshots of the current data space.
Initiate merge	Indicates whether the profile can merge the current data space with its parent data space.
Export archive	Indicates whether the profile can export the current data space as an archive.
Import archive	Indicates whether the profile can import an archive into the current data space.
Close a data space	Indicates whether the profile can close the current data space.
Close a snapshot	Indicates whether the profile can close a snapshot of the current data space.
Rights on services	Indicates if a profile has the right to execute services on the data space. By default, all data space services are allowed. An administrator or super owner of the current data space or a given user who is allowed to modify permissions on the current data space can modify these permissions to restrict data space services for certain profiles.
Permissions of child data space when created	Defines the same data space permissions as the current data space on child data spaces.

Access rights on data spaces

Mode	Authorization
Write	<ul style="list-style-type: none"> • Can view the data space. • Can access data sets according to data set permissions.
Read-only	<ul style="list-style-type: none"> • Can view the data space and its snapshots. • Can view child data spaces, if allowed by permissions. • Can view contents of the data space, though cannot modify them.
Hidden	<ul style="list-style-type: none"> • Can neither see the data space nor its snapshots. • If allowed to view child data space, can see the current data space but cannot select it. • Cannot access the data space contents, including data sets. • Cannot perform any actions on the data space.

Permissions on a new data space

When a user creates a child data space, the permissions of this new data space are automatically assigned to the profile's owner, based on the permissions defined under 'Permissions of child data space when created' in the parent data space. If multiple permissions are defined for the owner through different roles, [resolved permissions](#) [p 323] are applied.

Attention

Only the administrator and owner of a data space can define a new owner for the data space or modify associated permissions. They can also modify the general information on the data space and permissions of the users.

Furthermore, in a workflow, when using a "Create a data space" or "Create a snapshot" built-in script task, resolved permissions are computed using the owner defined in the script task's configuration, rather than the current session. This is because, in these cases, the current session is associated with a system user.

Permissions on data set

For a given data set, the allowable permissions for each profile are as follows:

Actions on data sets

Restricted mode	Indicates whether this data set profile-permission association should have priority over other permissions rules.
Create a child data set	Indicates whether the profile has the right to create a child data set of the current data set.
Duplicate data set	Indicates whether the profile has the right to duplicate the current data set.
Change the data set parent	Indicates whether the profile has the right to change the parent data set of a given child data set.

Actions on tables

For a specific table, permissions can be defined, which override the default permissions defined in its containing data set. The allowable permissions for each profile are as follows:

Create a record	Indicates whether the profile has the right to create records in the table.
Modify a record	Indicates whether the profile has the right to modify records in the table.
Hide a record	Indicates whether the profile has the right to hide records in the table.
Delete a record	Indicates whether the profile has the right to delete records in the table.

Access rights on node values

Permissions defined on specific terminal nodes override their default access rights.

Read-write	Can view and modify node values.
Read	Can view nodes, but cannot modify their values.
Hidden	Cannot view nodes.

Permissions on services

By default, all data set services are allowed. An administrator or an owner of the current data space can modify these permissions to restrict data set services for certain profiles.

52.3 Defining permissions and access rights with programmatic rules

Access rights can be defined on a data set using programmatic rules on target nodes. This can be done for all data set nodes, table records, a specific node, or a complex node and its child nodes. To define programmatic rules for access rights, create a class that implements the Java interface `SchemaExtensions`.

Voir aussi `SchemaExtensions`^{API}

It is also possible to define a permission for a service using a programmatic rule.

To define programmatic rules for service permissions, create a class that implements the Java interface `ServicePermission`.

Voir aussi `ServicePermission`^{API}

One example of using programmatic rules is updating a field according to the value of another field.

Attention

Only one programmatic access right can be defined for each node, data set or record. Thus, defining a new programmatic access right will replace an existing one.

52.4 Resolving permissions

The resolution of permissions is always performed in the context of a user session according to the associated roles. In general, resolution of restrictive permissions is performed between a given level and its parent level. Thus, at any given level, a user cannot have a higher permission than the one resolved at a parent level.

Programmatic permissions are always considered to be restrictive.

Restriction policies

When a restrictive permission is defined for a profile, it takes precedence over the other permissions potentially accorded by the user's other roles. Generally:

- If restrictions are defined, the minimum permissions based on all the restricted profiles is applied.
- If no restrictions are defined, the maximum permissions based on all profiles is applied.

Permission resolution examples

Given two profiles *P1* and *P2* concerning the same user, the following table lists the possibilities when resolving that user's access to a service.

P1 authorization	P2 authorization	Permission resolution
Allowed	Allowed	Allowed. Restrictions do not make any difference.
Forbidden	Forbidden	Forbidden. Restrictions do not make any difference.
Allowed	Forbidden	Allowed, unless P2's authorization is a restriction.
Forbidden	Allowed	Allowed, unless P1's authorization is a restriction.

In another example, a data space can be hidden from all users by defining a restrictive association between the built-in profile "Profile.EVERYONE" and the access right "hidden".

52.5 Resolving access rights

Resolving access rights defined using the user interface

Access rights defined using the user interface are resolved on three levels: data space, data set and node.

If a profile is associated with restrictive access rights at a given level, the minimum of all restrictive rights defined at that level is resolved. If no restrictions are defined at that level, the maximum of all access rights defined at that level is resolved.

At any given level, the most restrictive access rights between those resolved at this level and higher levels are applied. For instance, if a user's data set access permissions resolve to read-write access, but the container data space only allows read access, the user will only have read-only access to this data set.

Note

The data set inheritance mechanism applies to both values and access rights. That is, access rights defined on a data set will be applied to its child data sets. It is possible to override these rights in the child data set.

Access rights resolution example

In this example, there are three users who belong to the following defined roles and profiles:

User	Profile
User 1	<ul style="list-style-type: none"> • user1 • role A • role B
User 2	<ul style="list-style-type: none"> • role A • role B • role C
User 3	<ul style="list-style-type: none"> • user3 • role A • role C

The access rights of the profiles on a given element are as follows:

Profile	Access rights	Restriction policy
user1	Hidden	Yes
user3	Read	No
Role A	Read/Write	No
Role B	Read	Yes
Role C	Hidden	No

After resolution based on the role and profile access rights above, the right that are applied to each user are as follows:

User	Resolved access rights
User 1	Hidden
User 2	Read
User 3	Read/Write

Resolving data space and snapshot access rights

At data space level, access rights are resolved as follows:

- If a user has several rights defined through multiple profiles:
 - If the rights include restrictions, the minimum of the restrictive profile-rights associations is applied.
 - Otherwise, the maximum of the profile-rights associations is applied.
- If the user has no rights defined:
 - If the user is an administrator or is the owner of the data space, read-write access is given for this data space.
 - Otherwise, the data space will be hidden.

Resolving data set access rights

At the data set level, the same principle applies as at the data space level. After resolving the access rights at the data set level alone, the final access rights are determined by taking the minimum rights between the resolved data space rights and the resolved data set rights. For example, if a data space is resolved to be read-only for a user and one of its data sets is resolved to be read-write, the user will only have read-only access to that data set.

Resolving node access rights

At the node level, the same principle applies as at the data space and data set levels. After resolving the access rights at the node level alone, the final access rights are determined by taking the minimum rights between the resolved data set rights and the resolved node rights.

Specific access rights can be defined at the node level. If no specific access right is defined, the default access right is used for the resolution process.

Note

The resolution procedure is slightly different for table and table child nodes.

Special case for table and table child nodes

This describes the resolution process used for a given table node or table record *N*.

For each user-defined permission rule that matches one of the user's profiles, the access rights for *N* are either:

1. The locally defined access rights for *N*;
2. Inherited from the access rights defined on the table node;
3. Inherited from the default access rights for data set values.

All matching user-defined permission rules are used to resolve the access rights for *N*. Resolution is done according to the [restriction policy](#) [p 323].

The final resolved access rights will be the minimum between the data space, data set and the resolved access right for *N*.

Resolving access rights defined with programmatic rules

There are three levels of resolution for programmatic access right rules: data set, record and node. Since only one programmatic access rule can be set for a given level, the last rule set is the one used by the resolution procedure.

Rule resolution on data set

For a data set, the last rule set is considered as the resolved rule

Rule resolution on record

For a record, the resolved rule is the minimum between the resolved rule set on the data set and the rule set on this record. See `SchemaExtensionsContext.setAccessRuleOnOccurrenceAPI` for more details.

Access rule resolution on node

For a node that is a child node of a record, the resolved rule is the minimum between the resolved rule the record and the rule set on this node.

Otherwise, the resolved rule is the minimum between the resolved rule set on the data set and the rule set on this node. See `SchemaExtensionsContext.setAccessRuleOnNodeAPI` for more details.

Display policy for foreign key drop-down menus

If a record is hidden due to access rules, it will not appear in foreign key drop-down menus.

Attention

The resolved access rights on a data set or data set node is the minimum between the resolved access rights defined in the user interface and the resolved programmatic rules, if any.

52.6 Resolving permissions for actions and services

The resolution of actions and services available to a given user follows the same process as the resolution of access rights.

When several action lists are defined on a data set for a given profile, the final actions list is dynamically generated after evaluating each individual action across all lists associated with the same user.

If some profile-action list associations are restrictive, an action is forbidden if at least one restrictive association forbids it. If there are no restrictive associations, an action is allowed if any association allows it.

Action and service permission resolution example

In this example, there are two users belonging to different roles and profiles:

User	Profiles
User 1	<ul style="list-style-type: none"> • user1 • role A • role B
User 2	<ul style="list-style-type: none"> • role C • role D

The permissions associated with the roles and profiles on a given table are as follows:

Profile	Create a record	Modify a record	Hide a record	Duplicate a record	Delete a record	Restriction policy
user1	Allowed	Forbidden	Allowed	Forbidden	Allowed	No
Role A	Allowed	Allowed	Forbidden	Allowed	Forbidden	Yes
Role B	Allowed	Forbidden	Allowed	Allowed	Forbidden	Yes
Role C	Allowed	Allowed	Forbidden	Forbidden	Forbidden	No
Role D	Allowed	Forbidden	Forbidden	Allowed	Forbidden	No

The actions available to each user after rights resolution are as follows:

Users	Available actions
User 1	Create a record
	Duplicate a record
User 2	Create a record
	Modify a record
	Duplicate a record

CHAPITRE 53

Criteria editor

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Conditional blocks](#)
3. [Atomic criteria](#)

53.1 Overview

The criteria editor is included in several different areas of the user interface. It allows defining table filters, as well as validation and computation rules on data. This editor is based on the XPath 1.0 Recommendation.

Two types of criteria exist: atomic criteria and conditional blocks.

Voir aussi [Supported XPath syntax](#) [p 267]

53.2 Conditional blocks

Conditional blocks are made up of atomic criteria and other conditional blocks. They express a condition based on the criteria. The following types of blocks exist:

- **No criteria match:** None of the criteria in the block match.
- **Not all criteria match:** At least one criterion in the block does no match.
- **All criteria match:** All criteria in the block match.
- **At least one criterion matches:** One or more of the criteria match.

53.3 Atomic criteria

An atomic predicate is defined by a field, an operator, and an expression (either a value or an XPath formula).

Field	Specifies the field of the table to which the criterion applies.
Operator	Specifies the operator used. Available operators depend on the data type of the field.
Value	Specifies the value or expression. See Expression [p 330] below.
Code only	If checked, specifies searching the underlying values for the field instead of labels, which are searched by default.

Expression

The expression can either be a fixed value or a formula. When creating a filter, only fixed values are authorized. During creation of a validation or computation rule, a formula can be created using the wizard.

Known limitation: The formula field does not validate input values, only the syntax and path are checked.

CHAPITRE 54

Performance guidelines

Ce chapitre contient les sections suivantes :

1. [Basic performance checklist](#)
2. [Checklist for data space usage](#)
3. [Memory management](#)
4. [Validation](#)
5. [Mass updates](#)
6. [Accessing tables](#)

54.1 Basic performance checklist

While EBX5 is designed to support large volumes of data, several common factors can lead to poor performance. Addressing the key points discussed in this section will solve the usual performance bottlenecks.

Expensive programmatic extensions

For reference, the table below details the programmatic extensions that can be implemented.

Use case	Programmatic extensions that can be involved
Validation	<ul style="list-style-type: none"> • programmatic constraints <code>Constraint^{API}</code> • computed values <code>ValueFunction^{API}</code>
Table access	<ul style="list-style-type: none"> • record-level permission rules <code>SchemaExtensionsContext.setAccessRuleOnOccurrence^{API}</code> • programmatic filters <code>AdaptationFilter^{API}</code>
EBX5 content display	<ul style="list-style-type: none"> • computed values <code>ValueFunction^{API}</code> • UI Components <code>UIBeanEditor^{API}</code> • node-level permission rules <code>SchemaExtensionsContext.setAccessRuleOnNode^{API}</code>
Data update	<ul style="list-style-type: none"> • triggers <code>Package com.orchestranetworks.schema.trigger^{API}</code>

For large volumes of data, cumbersome algorithms have a serious impact on performance. For example, a constraint algorithm's complexity is $O(n^2)$. If the data size is 100, the resulting cost is

proportional to 10 000 (this generally produces an immediate result). However, if the data size is 10 000, the resulting cost will be proportional to 10 000 000.

Another reason for slow performance is calling external resources. Local caching usually solves this type of problem.

If one of the use cases above displays poor performance, it is recommended to track the problem either through code analysis or using a Java profiling tool.

Directory integration

Authentication and permissions management involve the [user and roles directory](#) [p 407].

If a specific directory implementation is deployed and accesses an external directory, it can be useful to ensure that local caching is performed. In particular, one of the most frequently called methods is `Directory.isUserInRoleAPT`.

Aggregated lists

In a data model, when an element's cardinality constraint `maxOccurs` is greater than 1 and `osd:table` is declared on this element, it is implemented as a `Java List`. This type of element is called an [aggregated list](#) [p 463], as opposed to a table.

It is important to consider that there is no specific optimization when accessing aggregated lists in terms of iterations, user interface display, etc. Besides performance concerns, aggregated lists are limited with regard to many functionalities that are supported by tables. See [tables introduction](#) [p 467] for a list of these features.

Attention

For the reasons stated above, aggregated lists should be used only for small volumes of simple data (one or two dozen records), with no advanced requirements for their identification, lookups, permissions, etc. For larger volumes of data (or more advanced functionalities), it is recommended to use `osd:table` declarations.

54.2 Checklist for data space usage

[Data spaces](#) [p 78] available in semantic mode, are an invaluable tool for managing complex data life cycles. While this feature brings great flexibility, it also implies a certain overhead cost, which should be taken into consideration for optimizing usage patterns.

This section reviews the most common performance issues that can appear in case of an intensive use of many data spaces containing large tables, and how to avoid them.

Note

Sometimes, the use of data spaces is not strictly needed. As an extreme example, consider the case where every transaction triggers the following actions:

1. A data space is created.
2. The transaction modifies some data.
3. The data space is merged, closed, then deleted.

In this case, no future references to the data space are needed, so using it to make isolated data modifications is unnecessary. Thus, using Procedure^{API} already provides sufficient isolation to avoid conflicts from concurrent operations. It would then be more efficient to directly do the modifications in the target data space, and get rid of the steps which concern branching and merging.

For a developer-friendly analogy, this is like using a source-code management tool (CVS, SVN, etc.): when you need to perform a simple modification impacting only a few files, it is probably sufficient to do so directly on the main branch. In fact, it would be neither practical nor sustainable, with regard to file tagging/copying, if every file modification involved branching the whole project, modifying the files, then merging the dedicated branch.

Insufficient memory

When a table is in semantic mode (default), the EBX5 Java memory cache is used. It ensures a much more efficient access to data when this data is already loaded in the cache. However, if there is not enough space for working data, swaps between the Java heap space and the underlying database can heavily degrade overall performance.

This memory swap overhead can only occur for tables in a data space with an [on-demand loading strategy](#) [p 335].

Such an issue can be detected by looking at the [monitoring log file](#) [p 335]. If it occurs, various actions can be considered:

- reducing the number of child data spaces that contain large tables;
- reducing the number of indexes specifically defined for large tables (in semantic mode, the current limitation applies: for a given table, the content of its indexes is not shared among child data spaces);
- using relational mode instead of semantic mode;
- or (obviously) allocating more memory, or optimizing the memory used by applications for non-EBX5 objects.

Voir aussi

[Memory management](#) [p 334]

[Relational mode](#) [p 283]

Transaction cancels

In semantic mode, when a transaction has performed some updates in the current data space and then aborts, loaded indexes of the modified tables are reset. If updates on a large table are often cancelled

and, at the same time, this table is intensively accessed, then the work related to index rebuild will slow down the access to the table; moreover, the induced memory allocation and garbage collection can reduce the overall performance.

Voir aussi

Functional guard and exceptions `TableTrigger`^{API}

Procedure^{API}

Reorganization of database tables

As with any database, inserting and deleting large volumes of data may lead to fragmented data, which can deteriorate performance over time. To resolve the issue, reorganizing the impacted database tables is necessary. See [Monitoring and cleanup of relational database](#) [p 384].

A specificity of EBX5 is that the creation of data spaces and snapshots adds new entries to the table `{ebx.persistence.table.prefix}HTB`. It may therefore be necessary to schedule regular reorganizations of this table for large repositories in which many data spaces are created and deleted.

Voir aussi [Monitoring and cleanup of relational database](#) [p 384]

54.3 Memory management

Loading strategy

The administrator can specify the loading strategy of a data space or snapshot in its information. The default strategy is to load and unload the resources on demand. For resources that are heavily used, a *forced load* strategy is usually recommended.

The following table details the loading modes which are available in semantic mode. Note that the application server must be restarted so as to take into account any loading strategy change.

On-demand loading and unloading	<p>In this default mode, each resource in a data space is loaded or built only when it is needed. The resources of the data space are "soft"-referenced using the standard Java <code>SoftReference</code> class. This implies that each resource can be unloaded "at the discretion of the garbage collector in response to memory demand".</p> <p>The main advantage of this mode is the ability to free memory when needed. As a counterpart, this implies a load/build cost when an accessed resource has not yet been loaded since the server started up, or if it has been unloaded since.</p>
Forced loading	<p>If the forced loading strategy is enabled for a data space or snapshot, its resources are loaded asynchronously at server startup. Each resource of the data space is maintained in memory until the server is shut down or the data space is closed.</p> <p>This mode is particularly recommended for long-living data spaces and/or those that are used heavily, namely any data space that serves as a reference.</p>
Forced loading and prevalidation	<p>This strategy is similar to the forced loading strategy, except that the content of the loaded data space or snapshot will also be validated upon server startup.</p>

Monitoring

Indications of EBX5 load activity are provided by monitoring the underlying database, and also by the ['monitoring' logging category](#) [p 366].

If the numbers for *cleared* and *built* objects remain high for a long time, this is an indication that EBX5 is swapping.

Tuning memory

The maximum size of the memory allocation pool is usually specified using the Java command-line option `-Xmx`. As is the case for any intensive process, it is important that the size specified by this option does not exceed the available physical RAM, so that the Java process does not swap to disk at the operating-system level.

Tuning the garbage collector can also benefit overall performance. This tuning should be adapted to the use case and specific Java Runtime Environment used.

54.4 Validation

The internal incremental validation framework will optimize the work required when updates occur. The incremental validation process behaves as follows:

- The first call to a data set validation report performs a full validation of the data set. The [loading strategy](#) [p 334] can also specify a data space to be prevalidated at server startup.
- Data updates will transparently and asynchronously maintain the validation report, insofar as the updated nodes specify explicit dependencies. For example, standard and static facets, foreign key constraints, dynamics facets, selection nodes specify explicit dependencies.
- If a mass update is executed or if there are too many validation messages, the incremental validation process is stopped. The next call to the validation report will then trigger a full validation.
- If a transaction is cancelled, the validation state of the updated data set is reset. The next call to the validation report will trigger a full validation as well.

Certain nodes are systematically revalidated, however, even if no updates have occurred since the last validation. These are the nodes with *unknown dependencies*. A node has unknown dependencies if:

- It specifies a **programmatic constraint** `ConstraintAPI` in the default *unknown dependencies* mode,
- It declares a **computed value** `ValueFunctionAPI`, or
- It declares a dynamic facet that depends on a node that is itself a **computed value** `ValueFunctionAPI`.

Consequently, on large tables (beyond the order of 10^5), it is recommended to avoid nodes with unknown dependencies (or at least to minimize the number of such nodes). For programmatic constraints, the developer is able to specify two alternative modes that drastically reduce incremental validation cost: *local dependency* mode and *explicit dependencies*. For more information, see **Dependencies and validation** `DependenciesDefinitionContextAPI`.

Note

It is possible for an administrator user to manually reset the validation report of a data set. This option is available from the validation report section in EBX5.

54.5 Mass updates

Mass updates can involve several hundreds of thousands of insertions, modifications and deletions. These updates are usually infrequent (usually initial data imports), or are performed non-interactively (nightly batches). Thus, performance for these updates is less critical than for frequent or interactive operations. However, similar to classic batch processing, it has certain specific issues.

Transaction boundaries

It is generally not advised to use a single transaction when the number of atomic updates in the transaction is beyond the order of 10^4 . Large transactions require a lot of resources, in particular, memory, from EBX5 and from the underlying database.

To reduce transaction size, it is possible to:

- Specify the property [ebx.manager.import.commit.threshold](#) [p 371]. However, this property is only used for interactive archive imports performed from the EBX5 user interface.
- Explicitly specify a **commit threshold** `ProcedureContext.setCommitThresholdAPI` inside the batch procedure.
- Structurally limit the transaction scope by implementing `ProcedureAPI` for a part of the task and executing it as many times as necessary.

On the other hand, specifying a very small transaction size can also hinder performance due to the persistent tasks that need to be done for each commit.

Note

If intermediate commits are a problem because transactional atomicity is no longer guaranteed, it is recommended to execute the mass update inside a dedicated data space. This data space will be created just before the mass update. If the update does not complete successfully, the data space must be closed, and the update reattempted after correcting the reason for the initial failure. If it succeeds, the data space can be safely merged into the original data space.

Triggers

If required, triggers can be deactivated using the method `ProcedureContext.setTriggerActivationAPI`.

54.6 Accessing tables

Functionalities

Tables are commonly accessed through EBX5 and also through the `RequestAPI` API and data services. This access involves a unique set of functions, including a *dynamic resolution* process. This process behaves as follows:

- **Inheritance:** Inheritance in the data set tree takes into account records and values that are defined in the parent data set, using a recursive process. Also, in a root data set, a record can inherit some of its values from the data model default values, defined by the `xs:default` attribute.
- **Value computation:** A node declared as an `osd:function` is always computed on the fly when the value is accessed. See `ValueFunction.getValueAPI`.
- **Filtering:** An [XPath predicate](#) [p 267], a **programmatic filter** `AdaptationFilterAPI`, or a record-level **permission rule** `SchemaExtensionsContext.setAccessRuleOnOccurrenceAPI` requires a selection of records.
- **Sort:** A sort of the resulting records can be performed.

Accessing tables in semantic mode

Architecture and design

In order to improve the speed of operations on tables, indexes are managed by the EBX5 engine.

EBX5 advanced features, such as advanced life-cycle (snapshots and data spaces), data set inheritance, and flexible XML Schema modeling, have led to a specialized design for indexing mechanisms. This design can be summarized as follows:

- *Indexes* maintain an in-memory data structure on a full table.
- An index is not persisted, and building it requires loading all table blocks from the database. This tradeoff still results in a beneficial outcome, since the index can be retained in memory for longer than its corresponding table blocks.

Attention

Faster access to tables is ensured if indexes are ready and maintained in memory cache. As mentioned above, it is important for the Java Virtual Machine to have enough space allocated, so that it does not release indexes too quickly.

Performance considerations

The request optimizer favors the use of indexes when computing a request result.

Attention

- Only XPath filters are taken into account for index optimization.
- Non-primary-key indexes are not taken into account for child data sets.

Assuming the indexes are already built, the impacts on performance are as follows:

1. If the request does not involve filtering, programmatic rules, or sorting, accessing its first few rows (these fetched by a paged view) is almost instantaneous.
2. If the request can be resolved without an extra sort step (this is the case if it has no sort criteria, or if its sort criteria relate to those of the index used for computing the request), accessing the first few rows of a table should be fast. More precisely, it depends on the cost of the specific filtering algorithm that is executed when fetching at least 2000 records.
3. Both cases above guarantee an access time that is independent of the size of the table, and provides a view sorted by the index used. If an extra sort is required, the time taken by the first access depends on the table size according to an $N\log(N)$ function, where N is the number of records in the resolved view.

Note

The paginated requests automatically add the primary key to the end of the specified criterion, in order to ensure consistent ordering. Thus, the primary key fields should also be added to the end of any index intended to improve the performance of paginated requests. These include tabular and hierarchical views, and drop-down menus for table references.

If indexes are not yet built, or have been unloaded, additional time is required. The build time is $O(N\log(N))$.

Accessing the table data blocks is required when the request cannot be computed against a single index (whether for resolving a rule, filter or sort), as well as for building the index. If the table blocs are not present in memory, additional time is needed to fetch them from the database.

It is possible to get information through the [monitoring](#) [p 335] and request logging categories.

Other operations on tables

The new records creations or record insertions depend on the primary key index. Thus, a creation becomes almost immediate if this index is already loaded.

Accessing tables in relational mode

When computing a request result, the EBX5 engine delegates the following to the RDBMS:

- Handling of all request sort criteria, by translating them to an `ORDER BY` clause.
- Whenever possible, handling of the request filters, by translating them to a `WHERE` clause.

Attention

Only XPath filters are taken into account for index optimization. If the request includes non-optimizable filters, table rows will be fetched from the database, then filtered in Java memory by EBX5, until the requested page size is reached. This is not as efficient as filtering on the database side (especially regarding I/O).

Information on the transmitted SQL request is logged to the category *persistence*. See [Configuring the EBX5 logs](#) [p 366].

Indexing

In order to improve the speed of operations on tables, indexes may be declared on a table at the data model level. This will trigger the creation of an index of the corresponding table in the database.

When designing an index aimed at improving the performance of a given request, the same rules apply as for traditional database index design.

Setting a fetch size

In order to improve performance, a fetch size should be set according to the expected size of the result of the request on a table. If no fetch size is set, the default value will be used.

- In semantic mode, the default value is 2000.
- In mapped mode, the default value is assigned by the JDBC driver: 10 for Oracle and 0 for PostgreSQL.

Attention

On PostgreSQL, the default value of 0 instructs the JDBC driver to fetch the whole result set at once, which could lead to an `OutOfMemoryError` when retrieving large amounts of data. On the other hand, using `fetchSize` on PostgreSQL will invalidate server-side cursors at the end of the transaction. If, in the same thread, you first fetch a result set with a `fetchsize`, then execute a procedure that commits the transaction, then, accessing the next result will raise an exception.

Voir aussi

`Request.setFetchSizeAPI`

RequestResult^{API}

Guide d'administration (en anglais)

Installation & configuration

CHAPITRE 55

Supported environments

Ce chapitre contient les sections suivantes :

1. [Browsing environment](#)
2. [Supported application servers](#)
3. [Supported databases](#)

55.1 Browsing environment

Supported web browsers

The EBX5 web interface supports the following browsers:

Microsoft Internet Explorer 8, 9, 10, 11	Compatibility mode is not supported. Performance limitations: page loading in IE8 is generally 20 times slower than in other browsers; in IE9, IE10 and IE11, page loading is two times slower. This issue is observed when forms have many input components, and particularly many multi-occurred groups.
Mozilla Firefox 3.6 and latest version (see details)	As Mozilla Firefox is updated frequently, Orchestra Networks only fully supports version 3.6, while testing and making a best effort to support the latest version available.
Google Chrome	As Google Chrome is updated frequently and it is not possible to deactivate automatic updates, Orchestra Networks only tests and makes a best effort to support the latest version available.

Screen resolution

The minimum screen resolution for EBX5 is 1024x768, and only the default browser zoom (100%) is supported.

Refreshing pages

Browser page refreshes are not supported by EBX5. When a page refresh is performed, the last user action is re-executed, therefore could result in potential issues. It is thus imperative to use the action buttons and links offered by EBX5 instead of refreshing the page.

Browser configuration

The following features must be activated in your browser configuration for the user interface to work properly:

- JavaScript
- Ajax
- Pop-ups

Attention

Avoid using any browser extensions or plug-ins, as they could interfere with the proper functioning of EBX5.

55.2 Supported application servers

EBX5 supports the following configurations:

- Java Runtime Environment: JRE 1.5 or higher version, which obviously includes the limitations specified by the Java Virtual Machine implementation vendor. For example, for JRE and JDK 1.5, Oracle states that they are "not updated with the latest security patches and are not recommended for use in production". See [Oracle Java Archive site](#).
- Any Java application server that complies with Servlet 2.4 (or higher), for example Tomcat 5.5 or higher, WebSphere Application Server 6.1 or higher, WebLogic Server 9.2 or higher. See [Java EE deployment overview](#) [p 360].
- The application server must use UTF-8 encoding for HTTP query strings from EBX5. This can be set at the application server level.

For example, on Tomcat, you can set the server to always use the UTF-8 encoding, by setting `URIEncoding` to 'UTF-8' on the `<Connector>` in the `server.xml` configuration file. Alternatively, you can set the server to use the encoding of the request's body, by setting the parameter `useBodyEncodingForURI` to 'true' in `server.xml`.

Attention

Limitations apply regarding clustering and hot deployment/undeployment:

Clustering: EBX5 does not include a cache synchronization mechanism, thus it cannot be deployed into a cluster of active instances. See [Technical architecture](#) [p 380] for more information.

Hot deployment/undeployment: EBX5 does not support hot deployment/undeployment of web applications registered as EBX5 modules or of EBX5 built-in web applications.

55.3 Supported databases

The EBX5 repository supports the following Relational Database Management Systems with suitable JDBC drivers. It is important to follow database vendor recommendations and update policies regarding the database itself, as well as the JDBC driver:

IBM DB2 UDB v8.2 or higher.	Mapped table modes are not supported (History [p 289], Relational mode [p 283] and Replication [p 297]).
Oracle Database 10gR2 or higher.	<p>Relational mode [p 283] is only supported for Oracle 11g R2 or higher.</p> <p>Relational mode supports Unicode data. However, for Oracle, data loss may occur if the <i>database character set</i> is not 'UTF8' or 'AL32UTF8'. A workaround is to set the Java system property <code>oracle.jdbc.defaultNChar=true</code>.</p> <p>The distinction of null values encounters certain limitations. On simple <code>xs:string</code> elements, Oracle does not support the distinction between empty strings and null values. See Empty string management [p 498] for more information.</p> <p>The user with which EBX5 connects to the database requires the following privileges:</p> <ul style="list-style-type: none"> • CREATE SESSION, • CREATE TABLE, • ALTER SESSION, • CREATE SEQUENCE, • A non-null quota on its default tablespace.
PostgreSQL 8.4 or higher.	<p>When using PostgreSQL as the underlying database, a request fetch size must be set, otherwise the JDBC driver will fetch the whole result set at once. This could lead to an <code>OutOfMemoryError</code> when retrieving large amounts of data. See <code>Request.setFetchSize^{API}</code>.</p> <p>The user with which EBX5 connects to the database requires the <code>CONNECT</code> privilege on the database hosting the EBX repository. Beyond this, the default privileges on the public schema of this database are suitable.</p>
Microsoft SQL Server 2008 or higher.	<p>When used with Microsoft SQL Server, EBX5 uses the default database collation to compare and sort strings stored in the database. This applies to strings used in the data model definition, as well as data stored in relational and history tables. The default database collation can be specified when the database is created. Otherwise, the database engine</p>

server collation is used. To avoid naming conflicts or unexpected behavior, a case- and accent-sensitive collation as the default database collation must be used (the collation name is suffixed by "CS_AS" or the collation is binary).

The default setting to enforce transaction isolation on SQL Server follows a pessimistic model. Rows are locked to prevent any read/write concurrent accesses. This may cause liveliness issues for mapped tables (history or relational). To avoid such issues, it is recommended to activate [snapshot isolation](#) on your SQL Server database.

The user with which EBX5 connects to the database requires the following privileges:

- CONNECT, SELECT and CREATE TABLE on the database hosting the EBX repository,
- ALTER, CONTROL, UPDATE, INSERT, DELETE on its default schema.

H2 v1.3.170 or higher.

H2 is not supported for production environments.

The default H2 database settings do not allow consistent reads when records are modified. Relational tables are locked following a pessimistic model. To prevent concurrency issues, it is possible to activate the [MVCC feature](#). Note, however, that the H2 documentation states this feature is not yet fully tested.

For other relational databases, contact [Orchestra Networks technical support](#).

Attention

In order to guarantee the integrity of the EBX5 repository, it is strictly forbidden to perform direct modifications to the database (for example, using direct SQL writes), except in the specific use cases described in the section [SQL access to data in relational mode](#) [p 286].

Voir aussi

[Repository administration](#) [p 380]

[Data source of the EBX5 repository](#) [p 352]

[Configuring the EBX5 repository](#) [p 363]

CHAPITRE 56

Java EE deployment

Ce chapitre contient les sections suivantes :

1. [Introduction](#)
2. [Software components](#)
3. [Third-party libraries](#)
4. [Web applications](#)
5. [Deployment details](#)
6. [Java EE deployment examples](#)

56.1 Introduction

This chapter details deployment specifications for EBX5 on a Java application server. For specific information regarding supported application servers and inherent limitations, see [Supported environments](#). [p 343]

56.2 Software components

EBX5 uses the following components:

- Library `ebx.jar`
- [Third-party Java libraries](#) [p 347]
- [EBX5 built-in web applications](#) [p 350] and optional [custom web applications](#) [p 350]
- [EBX5 main configuration file](#) [p 361]
- [EBX5 repository](#) [p 380]
- [Default user and roles directory](#) [p 407], integrated within the EBX5 repository, or a third-party system (LDAP, RDBMS) for user authentication

Voir aussi [Supported environments](#) [p 343]

56.3 Third-party libraries

EBX5 requires several third-party Java libraries. These libraries must be deployed and be accessible from the class-loader of `ebx.jar`. Depending on the application server being used, these libraries may already be present or may need to be added manually.

Database drivers

The EBX5 repository requires a database. Generally, the required driver is configured along with a data source, if one is used. Depending on the database defined in the main configuration file, one of the following drivers is required. Keep in mind that, whichever database you use, the version of the JDBC client driver must be equal or higher to the version of the database server.

H2	Version 1.3.170 validated. Any 1.3.X version should be suitable. Note that H2 is not supported for production environments. http://www.h2database.com/
Oracle JDBC	Oracle database 10gR2, 11gR2 and 12cR1 are validated on their latest patchset update. Determine the driver that should be used according to the database server version and the Java runtime environment version. You can include <code>ojdbc6.jar</code> or <code>ojdbc7.jar</code> depending on the Java runtime environment version you use; it does not make any difference as EBX5 does not make use of any JDBC 4.1 specific feature. Oracle database JDBC drivers download.
DB2 JDBC	DB2 UDB V9.7 validated. JAR files to include: <code>db2jcc_license_cu.jar</code> and <code>db2jcc.jar</code>
SQL Server JDBC	SQL Server 2008, 2008R2 and 2012, with all corrective and maintenance patches applied, are validated. Remember to use an up-to-date JDBC driver, as some difficulties have been encountered with older versions. JAR file to include: <code>sqljdbc4.jar</code>
PostgreSQL	PostgreSQL 8.4, 9.0, 9.1, 9.2 and 9.3 validated Include the latest JDBC driver released for your database server and Java runtime environment. PostgreSQL JDBC drivers download.

Voir aussi

[Data source of the EBX5 repository](#) [p 352]

[Configuring the EBX5 repository](#) [p 363]

SMTP and emails

The libraries for JavaMail 1.2 email management and JavaBeans Activation Framework are required.

The following libraries are used by email features in EBX5. See [Activating and configuring SMTP and e-mails](#) [p 367] for the details of the configuration.

- mail.jar, version 1.2, from December 5, 2000
- smtp.jar, version 1.2, from December 5, 2000
- pop3.jar, version 1.2, from December 5, 2000
- activation.jar, version 1.0.1, from May 21, 1999, or the maintenance release version 1.0.2, from August 28, 2002

Voir aussi

[JavaMail](#)

[JavaBeans Activation Framework](#)

Secure Socket Layer (SSL)

These libraries are required if your web applications use SSL features.

- jsse.jar: <http://www.oracle.com/technetwork/java/download-141865.html>
- ibmjse.jar: <http://www.ibm.com/developerworks/java/jdk/security/>

Voir aussi [EBX5 main configuration file](#) [p 361]

Java Message Service (JMS)

When using JMS, version 1.1 or higher is required.

Depending on whether a Java EE application server or a Servlet/Java Server Pages (JSP) implementation is being used, the library required is as follows:

- For an application server based on Java EE (Java Platform Enterprise Edition), the required JMS provider library is available by default. See <http://www.oracle.com/technetwork/java/javaee/overview> for more information.
- For a Servlet/Java Server Pages (JSP) implementation using Java SE (Java Platform Standard Edition), for example Apache Tomcat, a JMS provider library such as [Apache ActiveMQ](#) may need to be added. See <http://www.oracle.com/technetwork/java/javase/overview> for more information.

Note

In EBX5, the supported JMS model is exclusively Point-to-Point (PTP). PTP systems allow working with queues of messages.

Voir aussi [EBX5 main configuration file](#) [p 361]

56.4 Web applications

EBX5 provides pre-packaged EARs that can be deployed directly if your enterprise has no custom EBX5 module web applications to add. If you are deploying custom web applications as EBX5 modules, it is recommended to rebuild an EAR containing your custom modules packaged at the same level as the built-in web applications.

For more information, see the note on [repackaging the EBX5 EAR](#) [p 360] at the end of this chapter.

EBX5 built-in web applications

EBX5 includes the following built-in web applications.

ebx	EBX5 entry point, which handles the initialization upon start up. See Deployment details [p 351] for more information.
ebx-root-1.0	EBX5 root web application. Any application that uses EBX5 requires the root web application to be deployed.
ebx-manager	EBX5 user interface web application.
ebx-dma	EBX5 data model assistant, which helps with the creation of data models through the user interface. Note: The data model assistant requires the ebx-manager user interface web application to be deployed.
ebx-dataservices	EBX5 data services web application. Data services allow external interactions with data spaces, data workflows, and the user and roles directory in the EBX5 repository using the SOAP and Web Services Description Language (WSDL) standards. Note: The EBX5 web service generator requires the deployment of the ebx-manager user interface web application.

Custom web applications

Every custom web application that is defined as an [EBX5 module](#) [p 453] must be registered using the method `ModulesRegister.registerWebApp()` in the Java API. Registration of modules is explained in the [EBX5 modules](#) [p 454] chapter.

Voir aussi

[Registration](#) [p 454]

[Module registration](#) [p 373]

56.5 Deployment details

Introduction

This section describes the various options that are offered for deploying the 'ebx' web application. These options are available in its deployment descriptor (WEB-INF/web.xml) and are complemented by the properties defined in the main configuration file.

Attention

For JBoss application servers, any unused resources must be removed from the WEB-INF/web.xml deployment descriptor.

Voir aussi

[EBX5 main configuration file](#) [p 361]

[Supported application servers](#) [p 344]

User interface and web access

The web application 'ebx' (packaged as ebx.war) contains the servlet FrontServlet, which handles initialization and serves as the sole user interface entry point for the EBX5 web tools.

Configuring the deployment descriptor for 'FrontServlet'

In the file WEB-INF/web.xml of the web application 'ebx', the following elements must be configured for FrontServlet:

/web-app/servlet/load-on-startup	To ensure that FrontServlet initializes upon EBX5 start up, the web.xml deployment descriptor must specify the element <code><load-on-startup>1</load-on-startup></code> .
/web-app/servlet-mapping/url-pattern	FrontServlet must be mapped to the path '/'.

Configuring the application server for 'FrontServlet'

- FrontServlet must be authorized to access other contexts, such as ServletContext.

For example, on Tomcat, this configuration is done using the attribute `crossContext` in the configuration file `server.xml`, as follows:

```
<Context path="/ebx" docBase="(...)" crossContext="true"/>
```

- When several EBX5 web components are to be displayed on the same HTML page, for instance using iFrames, it may be required to disable the management of cookies due to limitations present in certain Internet browsers.

For example, on Tomcat, this configuration is provided by the attribute `cookies` in the configuration file `server.xml`, as follows:

```
<Context path="/ebx" docBase="(...)" cookies="false"/>
```

Data source of the EBX5 repository

Note

If the EBX5 main configuration specifies the property `ebx.persistence.url`, then the environment entry below will be ignored by EBX5 runtime. See [Configuring the EBX5 repository](#) [p 363] for more information on this property.

The JDBC data source for EBX5 is specified in the deployment descriptor `WEB-INF/web.xml` of the 'ebx' web application as follows:

Reserved resource name	Default JNDI name	Description
<code>jdbc/EBX_REPOSITORY</code>	Weblogic: <code>EBX_REPOSITORY</code> JBoss: <code>java:/EBX_REPOSITORY</code>	JDBC data source for EBX5 Repository. Java type: <code>javax.sql.DataSource</code>

Voir aussi

[Configuring the EBX5 repository](#) [p 363]

[Rules for the access to the database and user privileges](#) [p 381]

[\[documentation HTML\]](#) Oracle data source configuration on WebSphere Application Server 6

[\[documentation HTML\]](#) SQL Server data source configuration on WebSphere Application Server 6

Mail sessions

Note

If the EBX5 main configuration does not set `ebx.mail.activate` to 'true', or if it specifies the property `ebx.mail.smtp.host`, then the environment entry below will be ignored by EBX5 runtime. See [SMTP](#) [p 367] in the EBX5 main configuration properties for more information on these properties.

SMTP and email is declared in the deployment descriptor `WEB-INF/web.xml` of the 'ebx' web application as follows:

Reserved resource name	Default JNDI name	Description
<code>mail/EBX_MAIL_SESSION</code>	Weblogic: <code>EBX_MAIL_SESSION</code> JBoss: <code>java:/EBX_MAIL_SESSION</code>	Java Mail session used to send e-mails from EBX5. Java type: <code>javax.mail.Session</code>

JMS connection factory

Note

If the EBX5 main configuration does not activate JMS through the property `ebx.jms.activate`, the environment entry below will be ignored by the EBX5 runtime. See [JMS](#) [p 368] in the EBX5 main configuration properties for more information on this property.

The JMS connection factory is declared in the deployment descriptor `WEB-INF/web.xml` of the 'ebx' web application as follows:

Reserved resource name	Default JNDI name	Description	Required
jms/EBX_JMSConnectionFactory	Weblogic: EBX_JMSConnectionFactory JBoss: java:/ EBX_JMSConnectionFactory	JMS connection factory used by EBX5 to create connections with the JMS provider configured in the operational environment of the application server. Java type: javax.jms.ConnectionFactory	Yes

Note

For deployment on JBoss and WebLogic application servers with JNDI capabilities, you must update `EBX5.ear` or `EBX5ForWebLogic.ear` respectively for additional mappings of all required resource names to JNDI names.

JMS for data services

To configure data services to use JMS instead of the default HTTP, you must configure the [JMS connection factory](#) [p 353] and the following queues, declared in the `WEB-INF/web.xml` deployment descriptor of the 'ebx' web application. This is the only method for configuring JMS for data services.

When a SOAP request is received, the SOAP response is optionally returned if the header field `JMSReplyTo` is defined. If so, the fields `JMSCorrelationID` and `JMSType` are retained.

See [JMS](#) [p 368] for more information on the associated EBX5 main configuration properties.

Note

If the EBX5 main configuration does not activate JMS through the property `ebx.jms.activate`, then the environment entries below will be ignored by EBX5 runtime. See [JMS](#) [p 368] in the EBX5 main configuration properties for more information on this property.

Reserved resource name	Default JNDI name	Description	Required
<code>jms/EBX_QueueIn</code>	Weblogic: <code>EBX_QueueIn</code> JBoss: <code>java:/jms/EBX_QueueIn</code>	JMS queue for incoming SOAP requests sent to EBX5 by other applications. Java type: <code>javax.jms.Queue</code>	No
<code>jms/EBX_QueueFailure</code>	Weblogic: <code>EBX_QueueFailure</code> JBoss: <code>java:/jms/EBX_QueueFailure</code>	JMS queue for failures. It contains incoming SOAP requests for which an error has occurred. This allows replaying these messages if necessary. Java type: <code>javax.jms.Queue</code> Note: For this property to be read, the main configuration must also activate the queue for failures through the property <code>ebx.jms.activate.queueFailure</code> . See JMS [p 368] in the EBX5 main configuration properties for more information on these properties.	No

JMS for distributed data delivery (D3)

To configure D3 to use JMS instead of the default HTTP and TCP protocols, you must configure the [JMS connection factory](#) [p 353] and the following queues, declared in the `WEB-INF/web.xml` deployment descriptor of the 'ebx' web application.

Note

If the EBX5 main configuration does not activate JMS and D3 (slave, hub or master node) through the properties `ebx.d3.mode`, `ebx.jms.activate` and `ebx.jms.d3.activate`, then the environment entries below will be ignored by EBX5 runtime. See [JMS](#) [p 368] and [Distributed data delivery \(D3\)](#) [p 368] in the EBX5 main configuration properties for more information on these properties.

Common declarations on master and slave nodes (for shared queues)

Reserved resource name	Default JNDI name	Description
jms/EBX_D3MasterQueue	Weblogic: EBX_D3MasterQueue JBoss: java:/jms/EBX_D3MasterQueue	D3 master JMS queue (only for D3 mode 'slave' or 'hub'). It specifies the queue name used to send SOAP requests to the D3 master node. The message producer sets the master repository ID as a value of the header field JMSType. Java type: javax.jms.Queue
jms/EBX_D3ReplyQueue	Weblogic: EBX_D3ReplyQueue JBoss: java:/jms/EBX_D3ReplyQueue	D3 Reply JMS queue (for all D3 modes except 'single' mode). It specifies the name of the reply queue for receiving SOAP responses. The consumption is filtered using the header field JMSCorrelationID. Java type: javax.jms.Queue
jms/EBX_D3ArchiveQueue	Weblogic: EBX_D3ArchiveQueue JBoss: java:/jms/ EBX_D3ArchiveQueue	D3 JMS Archive queue (for all D3 modes except 'single' mode). It specifies the name of the transfer archive queue used by D3 node. The consumption is filtered using the header field JMSCorrelationID. If the archive weight is higher than the threshold specified in the property ebx.jms.d3.archiveMaxSizeInKB, the archive will be divided into several sequences. Therefore, the consumption is filtered using the header fields JMSXGroupID and JMSXGroupSeq instead. Java type: javax.jms.Queue
jms/EBX_D3CommunicationQueue	WebLogic: EBX_D3CommunicationQueue JBoss: java:/jms/ EBX_D3CommunicationQueue	D3 JMS Communication queue (for all D3 modes except 'single' mode). It specifies the name of the communication queue where the requests are received. The consumption is filtered using the header field JMSType which corresponds to the current repository ID. Java type: javax.jms.Queue

Note

These JNDI names are set by default, but can be modified inside the web application archive ebx.war, included in EBX5ForWebLogic.ear (if using Weblogic) or EBX5.ear (if using JBoss, Websphere or other application servers).

Optional declarations on master nodes (for slave-specific queues)

Note

Used for ascending compatibility prior to 5.5.0 or for mono-directional queues topology.

The deployment descriptor of the master node must be manually modified by declaring specific communication and archive queues for each slave node. It consists in adding resource names in 'web.xml' inside 'ebx.war'. The slave-specific queues can be used by one or several slaves.

The resource names can be freely named, but the physical names of their associated queue must correspond to the definition on slaves for resources `jms/EBX_D3ArchiveQueue` and `jms/EBX_D3CommunicationQueue`.

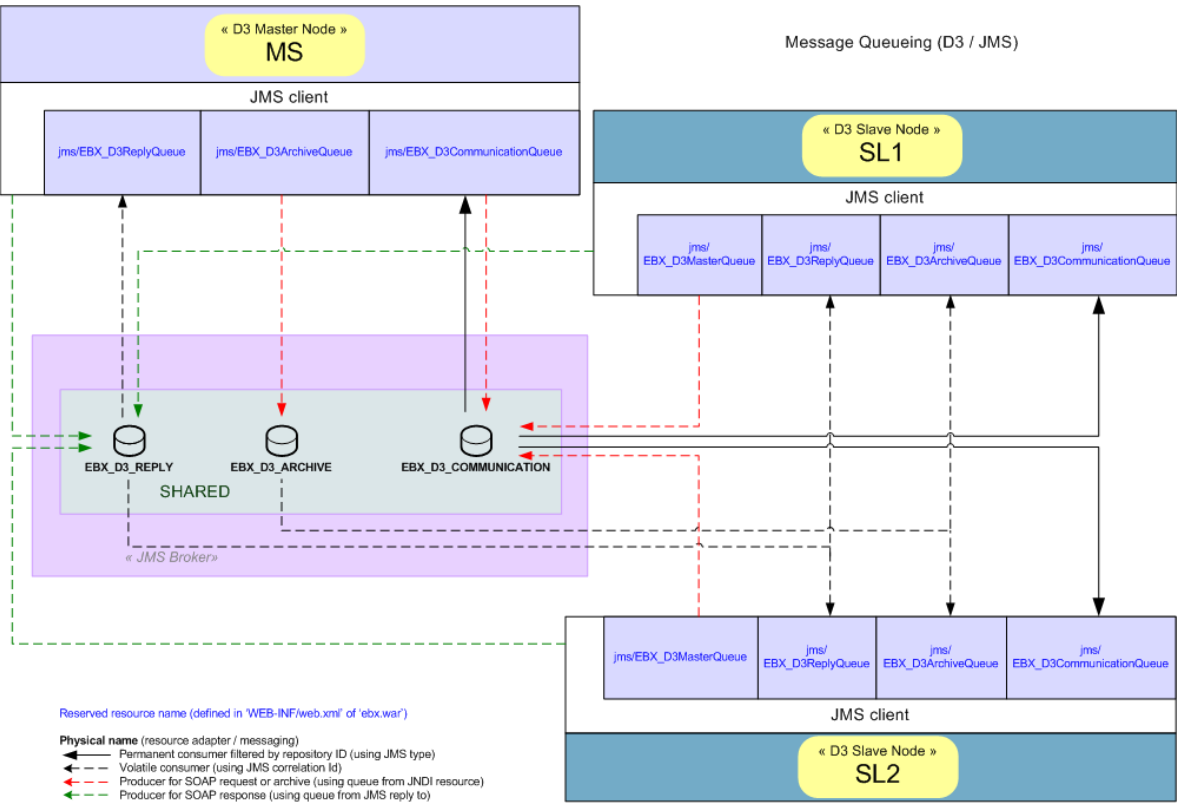
Note

Physical queue names matching: On registration, the slave node sends the communication and archive physical queue names. These queues are matched by physical queue name among all resources declared on the master node. If unmatched, the registration fails.

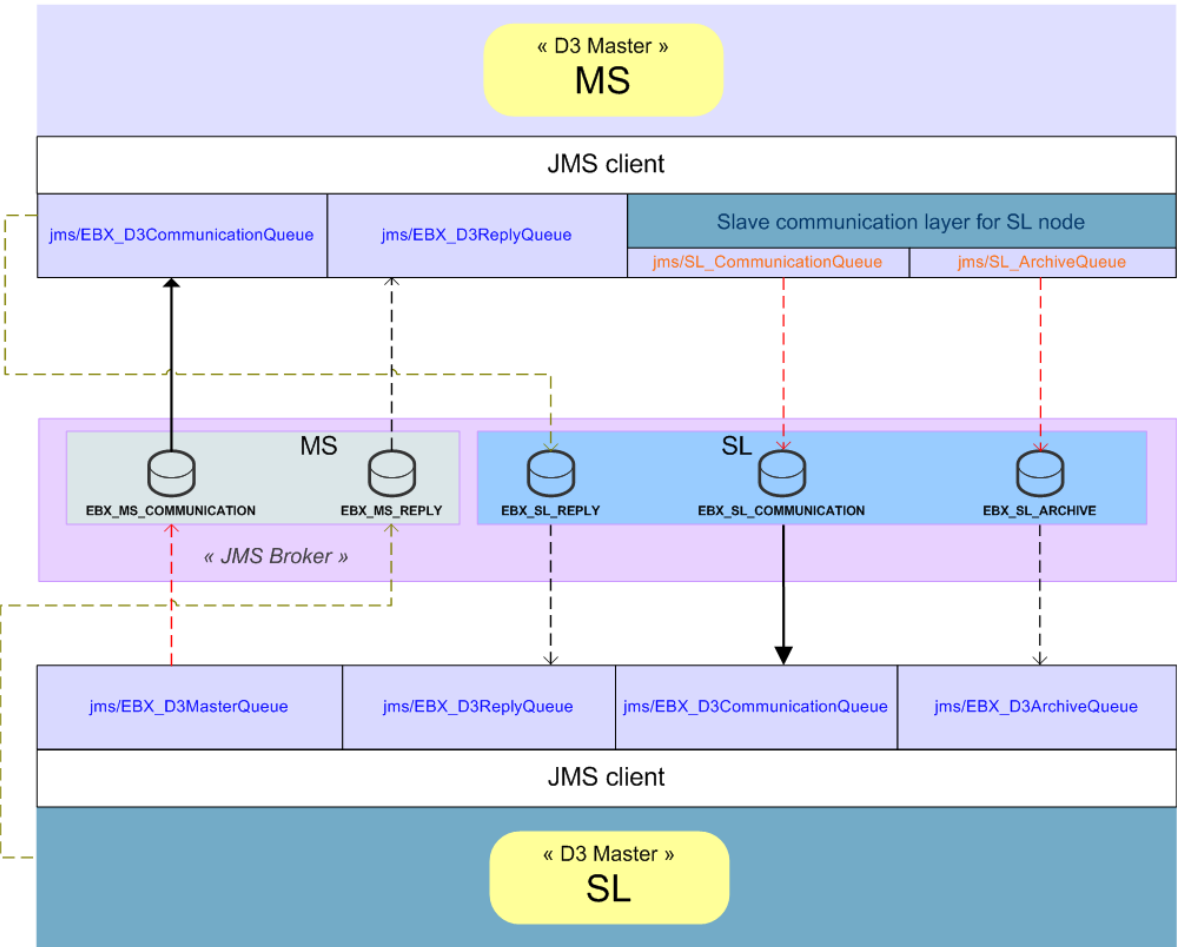
Examples of JMS configuration

	Shared queues	Specific queues
Master-Slave architecture	Between a master node and two slave nodes with shared queues [p 356]	Between a master node and a slave node with slave-specific queues [p 357]
Hub-Hub architecture	Between two hub nodes with shared queues [p 358]	Between two hub nodes with slave-specific queues [p 359]

Between a master node and two slave nodes with shared queues



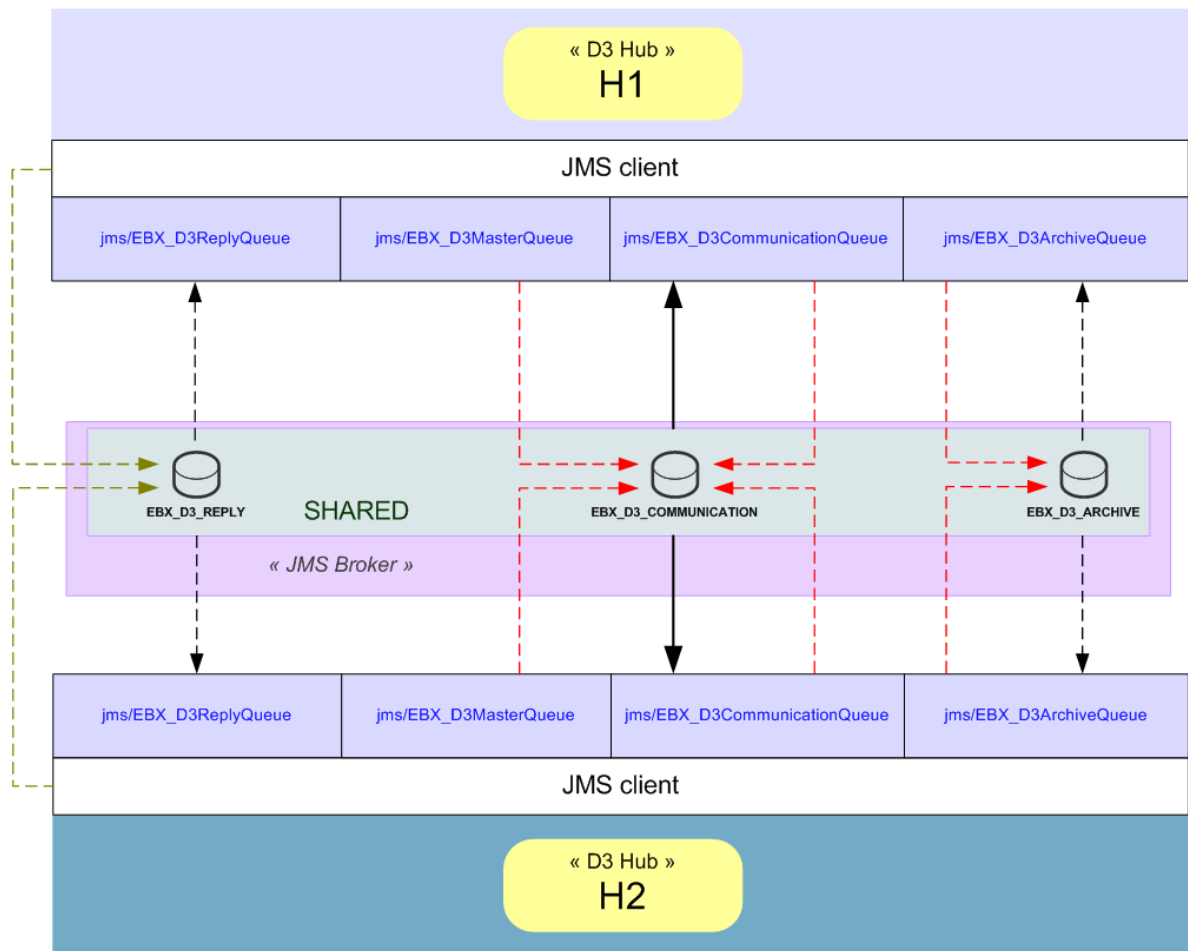
Between a master node and a slave node with slave-specific queues



Reserved or custom resource name (defined in 'WEB-INF/web.xml' of 'ebx.war')

- Physical name** (resource adapter / messaging)
- ← Permanent consumer filtered by repository ID (using JMS type)
 - ← - - - Volatile consumer (using JMS correlation Id)
 - ← - - - Producer for SOAP request or archive (using queue from JNDI resource)
 - ← - - - Producer for SOAP response (using queue from JMS reply to)

Between two hub nodes with shared queues

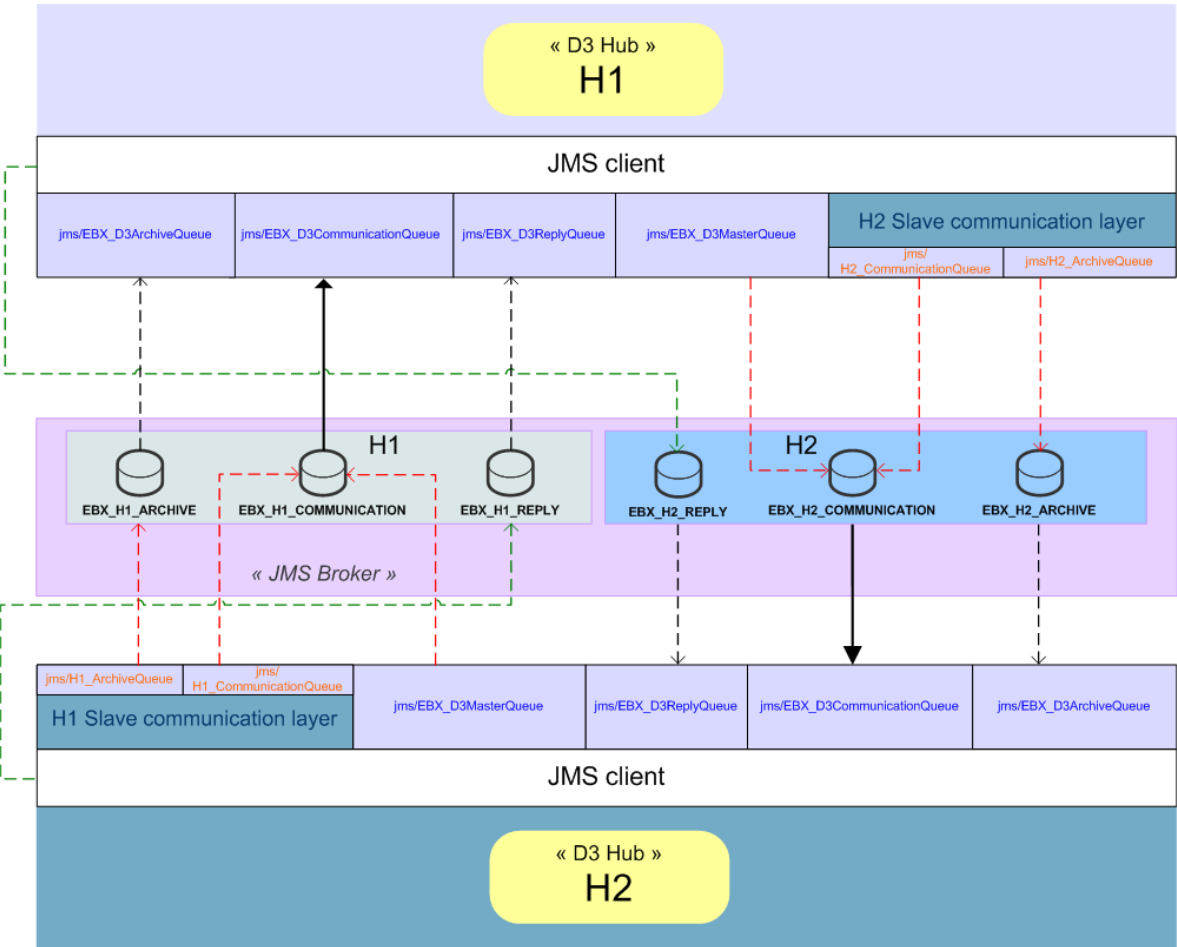


Reserved resource name (defined in 'WEB-INF/web.xml' of 'ebx.war')

Physical name (resource adapter / messaging)

- ← Permanent consumer filtered by repository ID (using JMS type)
- ← - - Volatile consumer (using JMS correlation Id)
- ← - - Producer for SOAP request or archive (using queue from JNDI resource)
- ← - - Producer for SOAP response (using queue from JMS reply to)

Between two hub nodes with slave-specific queues



56.6 Java EE deployment examples

EBX5 can be deployed on any Java EE application server that supports Servlet 2.4 or higher. The following documentation on Java EE deployment and installation notes are available:

- [\[documentation HTML\]](#) Installation on Tomcat 5.5
- [\[documentation HTML\]](#) Installation on WebSphere 6
- [\[documentation HTML\]](#) Installation on WebLogic 9.2

Attention

- The EBX5 installation notes on Java EE application servers do not replace the native documentation of each application server.
- They are *not* general installation recommendations, as the installation process is determined by architectural decisions, such as the technical environment, application mutualization, delivery process, and organizational decisions.
- In these examples, no additional EBX5 modules are deployed. To deploy additional modules, the best practice is to rebuild an EAR with the module as a web application at the same level as the other EBX5 modules. The web application must declare its class path dependency as specified by the Java™ 2 Platform Enterprise Edition Specification, v1.4:

J2EE.8.2 Optional Package Support

(...)

A JAR format file (such as a JAR file, WAR file, or RAR file) can reference a JAR file by naming the referenced JAR file in a Class-Path header in the Manifest file of the referencing JAR file. The referenced JAR file is named using a URL relative to the URL of the referencing JAR file. The Manifest file is named META-INF/MANIFEST.MF in the JAR file. The Class-Path entry in the Manifest file is of the form:

Class-Path: list-of-jar-files-separated-by-spaces

In an "industrialized" process, it is strongly recommended to develop a script that automatically builds the EAR, with the custom EBX5 modules, the EBX5 web applications, as well as all the required shared libraries.

- In order to avoid unpredictable behavior, the guideline to follow is to avoid any duplicates of `ebx.jar` or other libraries in the class-loading system.

CHAPITRE 57

EBX5 main configuration file

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Setting an EBX5 license key](#)
3. [Setting the EBX5 root directory](#)
4. [Configuring the EBX5 repository](#)
5. [Configuring the user and roles directory](#)
6. [Setting temporary files directories](#)
7. [Activating the XML audit trail](#)
8. [Configuring the EBX5 logs](#)
9. [Activating and configuring SMTP and e-mails](#)
10. [Configuring data services](#)
11. [Activating and configuring JMS](#)
12. [Configuring distributed data delivery \(D3\)](#)
13. [Configuring web access from end-user browsers](#)
14. [Configuring failover](#)
15. [Tuning the EBX5 repository](#)
16. [Miscellaneous](#)

57.1 Overview

The EBX5 main configuration file, by default named `ebx.properties`, contains most of the basic parameters for running EBX5. This is a Java properties file that uses the [standard simple line-oriented format](#).

The main configuration file complements the [Java EE deployment descriptor](#) [p 351]. Administrators can also perform further configuration through the user interface, which is then stored in the EBX5 repository.

Voir aussi

[Deployment details](#) [p 351]

[Front end administration](#) [p 391]

Location of the file

The access path to the main configuration file can be specified in several ways. In order of descending priority:

1. By defining the Java system property 'ebx.properties'. For example, this property can be set by adding the option `-Debx.properties=<filePath>` to the java command-line command. See [Java documentation](#).
2. By defining the servlet initialization parameter 'ebx.properties'.
This standard Java EE setting must be specified in the `web.xml` file of the web application 'ebx'. EBX5 accesses this parameter by calling the method `ServletConfig.getInitParameter("ebx.properties")` in the servlet `FrontServlet`.
See [getInitParameter](#) in the Oracle `ServletConfig` documentation.
3. By default, if nothing is specified, the main configuration file is located at `WEB-INF/ebx.properties` of the web application 'ebx'.

Note

In addition to specifying properties in the main configuration file, you can also set the values of properties directly in the system properties. For example, using the `-D` argument of the java command-line command.

Custom properties and variable substitution

The value of any property can include one or more variables that use the syntax `${propertyKey}`, where `propertyKey` is either a system property, or a property defined in the main configuration file.

For example, the default configuration file provided with EBX5 uses the custom property `ebx.home` to set a default common directory, which is then included in other properties.

57.2 Setting an EBX5 license key

Voir aussi [Installing a repository using the configuration assistant](#) [p 375]

```
#####
## EBX5 License number
## (as specified by your license agreement)
#####
ebx.license=paste_here_your_license_key
```

57.3 Setting the EBX5 root directory

The EBX5 root directory contains archives, the XML audit trail and, when the repository is persisted on H2 standalone mode, the H2 database files.

```
#####
## Path for EBX5 XML repository
#####
ebx.repository.directory=${ebx.home}/ebxRepository
```

57.4 Configuring the EBX5 repository

Before configuring the persistence properties of the EBX5 repository, carefully read the section [Technical architecture](#) [p 380] in the chapter 'Repository administration'.

The required library (driver) for each supported database is described in the chapter [Database drivers](#) [p 348].

Voir aussi

[Repository administration](#) [p 380]

[Rules for the access to the database and user privileges](#) [p 381]

[Supported databases](#) [p 345]

[Data source of the EBX5 repository](#) [p 352]

[Database drivers](#) [p 348]

```
#####
## The maximum time to set up the database connection,
## in milliseconds.
#####
ebx.persistence.timeout=10000
#####
## The prefix to add to all table names of persistence system.
## This may be useful for supporting multiple repositories in the relational database.
## Default value is 'EBX_'.
#####
ebx.persistence.table.prefix=

#####
## Case EBX5 persistence system is H2 'standalone'.
#####
ebx.persistence.factory=h2.standalone
ebx.persistence.user=sa
ebx.persistence.password=

#####
## Case EBX5 persistence system is H2 'server mode',
#####
#ebx.persistence.factory=h2.server

## Specific properties to be set only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:h2:tcp://127.0.0.1/ebxdb
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyy

#####
## Case EBX5 persistence system is Oracle database.
#####
#ebx.persistence.factory=oracle

## Specific properties to be set only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:oracle:thin:@127.0.0.1:1521:ebxDatabase
#ebx.persistence.driver=oracle.jdbc.OracleDriver
#ebx.persistence.user=xxxxxxxxx
#ebx.persistence.password=yyyyyyy

## Activate to use VARCHAR2 instead of NVARCHAR2 on Oracle (this should be required only in rare cases).
#ebx.persistence.oracle.useVARCHAR2=false

#####
## Case EBX5 persistence system is IBM DB2.
#####
#ebx.persistence.factory=db2

## Specific properties to be set only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
```

```
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:db2://127.0.0.1:50000/ebxDatabase
#ebx.persistence.driver=com.ibm.db2.jcc.DB2Driver
#ebx.persistence.user=xxxxxxx
#ebx.persistence.password=yyyyyyy

#####
## Case EBX5 persistence system is Microsoft SQL Server.
#####
#ebx.persistence.factory=sqlserver

## Specific properties to be set only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url= \
#jdbc:sqlserver://127.0.0.1:1036;datasenname=ebxDatabase
#ebx.persistence.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
#ebx.persistence.user=xxxxxxx
#ebx.persistence.password=yyyyyyy

#####
## Case EBX5 persistence system is PostgreSQL.
#####
#ebx.persistence.factory=postgresql

## Specific properties to be set only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.persistence.url=jdbc:postgresql://127.0.0.1:5432/ebxDatabase
#ebx.persistence.driver=org.postgresql.Driver
#ebx.persistence.user=xxxxxxx
#ebx.persistence.password=yyyyyyy
```

57.5 Configuring the user and roles directory

This parameter specifies the Java directory factory class name. It must only be defined if not using the default EBX5 directory.

Voir aussi

[Users and roles directory](#) [p 407]

DirectoryFactory^{API}

```
#####
## Specifies the Java directory factory class name.
## Value must be the fully qualified name of the Java class.
## The class must extend com.orchestranetworks.service.directory.DirectoryFactory.
#####
#ebx.directory.factory=xxx.yyy.DirectoryFactoryImpl
```

It is also possible to disable the built-in role "ADMINISTRATOR".

```
#####
## Specifies whether the built-in role ADMINISTRATOR is disabled.
## Default value is false.
#####
#ebx.directory.disableBuiltInAdministrator=true
```

57.6 Setting temporary files directories

Temporary files are stored as follows:

```
#####
## Directories for temporary resources.
#####
# The property ebx.temp.directory allows to specify a directory for temporary files.
# Default value is java.io.tmpdir
#
#ebx.temp.directory = ${java.io.tmpdir}
#ebx.temp.directory = /tmp/java

# The property ebx.temp.cache.directory allows to specify the directory containing temporary files for cache.
# Default value is ${ebx.temp.directory}/ebx.platform.
#ebx.temp.cache.directory = ${ebx.temp.directory}/ebx.platform
```

```
# The property ebx.temp.import.directory allows to specify the directory containing temporary files for import.  
# Default value is ${ebx.temp.directory}/ebx.platform.  
#ebx.temp.import.directory = ${ebx.temp.directory}/ebx.platform
```

57.7 Activating the XML audit trail

By default, the XML audit trail is activated. It can be deactivated using the following variable:

```
#####  
# The XML history has been replaced by an SQL history.  
# This old XML history can be deactivated using the following variable.  
# Default is true.  
#####  
ebx.history.xmlaudittrail.activated = true
```

Voir aussi [Audit trail](#) [p 411]

57.8 Configuring the EBX5 logs

The most important logging categories are:

ebx.log4j.category.log.kernel	Logs for EBX5 main features, processes, exceptions and compilation results of modules and data models.
ebx.log4j.category.log.workflow	Logs for main features, warnings and exceptions about workflow.
ebx.log4j.category.log.persistence	Logs related to communication with the underlying database.
ebx.log4j.category.log.setup	Logs for compilation results of all EBX5 objects, except for modules and data models.
ebx.log4j.category.log.validation	Logs for data sets validation results.
ebx.log4j.category.log.mail	<p>Logs for the activity related to the emails sent by the server (see Activating and configuring SMTP and e-mails [p 367]).</p> <p>Note: This category must not use the Custom SMTP appender [p 367] in order to prevent infinite loops.</p>
ebx.log4j.category.log.d3	Logs for D3 events on EBX5.
ebx.log4j.category.log.dataservices	Logs for data service events in EBX5.
ebx.log4j.category.log.monitoring	Raw logs for monitoring [p 335].
ebx.log4j.category.log.request	The optimization strategy for every Request ^{API} issued on a semantic table in the EBX5 repository.

Some of these categories can also be written to through custom code using the `LoggingCategory`^{API} interface.

```
#####
## Log4J properties:
##
## We have some specific syntax extensions:
## - Appender ebxFile:<aFileName>
## Defines a file appender with default settings (threshold=DEBUG)
##
## - property log.defaultConversionPattern is set by Java
#####
#ebx.log4j.debug=true
#ebx.log4j.disableOverride=
#ebx.log4j.disable=
```

```

ebx.log4j.rootCategory= INFO
ebx.log4j.category.log.kernel= INFO, Console, ebxFile:kernel, kernelMail
ebx.log4j.category.log.workflow= INFO, ebxFile:workflow
ebx.log4j.category.log.persistence= INFO, ebxFile:persistence
ebx.log4j.category.log.setup= INFO, Console, ebxFile:kernel
ebx.log4j.category.log.mail= INFO, Console, ebxFile:mail
ebx.log4j.category.log.frontEnd= INFO, Console, ebxFile:kernel
ebx.log4j.category.log.frontEnd.incomingRequest= INFO
ebx.log4j.category.log.frontEnd.requestHistory= INFO
ebx.log4j.category.log.frontEnd.UIComponentInput= INFO
ebx.log4j.category.log.fsm= INFO, Console, ebxFile:fsm
ebx.log4j.category.log.fsm.dispatch= INFO
ebx.log4j.category.log.fsm.pageHistory= INFO
ebx.log4j.category.log.wbp= FATAL, Console
#-----
ebx.log4j.appender.Console.Threshold = INFO
ebx.log4j.appender.Console=com.onwbp.org.apache.log4j.ConsoleAppender
ebx.log4j.appender.Console.layout=com.onwbp.org.apache.log4j.PatternLayout
ebx.log4j.appender.Console.layout.ConversionPattern=${log.defaultConversionPattern}
#-----
ebx.log4j.appender.kernelMail.Threshold = ERROR
ebx.log4j.appender.kernelMail = com.onwbp.org.apache.log4j.net.SMTPAppender
ebx.log4j.appender.kernelMail.To = admin@domain.com
ebx.log4j.appender.kernelMail.From = admin${ebx.site.name}
ebx.log4j.appender.kernelMail.Subject = EBX5 Error on Site ${ebx.site.name} (VM ${ebx.vm.id})
ebx.log4j.appender.kernelMail.layout.ConversionPattern=**Site ${ebx.site.name} (VM${ebx.vm.id})**%n
${log.defaultConversionPattern}
ebx.log4j.appender.kernelMail.layout = com.onwbp.org.apache.log4j.PatternLayout

#-----
ebx.log4j.category.log.monitoring= INFO, ebxFile:monitoring
ebx.log4j.category.log.dataServices = INFO, ebxFile:dataServices
ebx.log4j.category.log.d3= INFO, ebxFile:d3
ebx.log4j.category.log.request= INFO, ebxFile:request

```

Custom 'ebxFile' appender

The token `ebxFile:` can be used as a shortcut to define a daily rolling file appender with default settings. It must be followed by a file name. It then activates an appender that writes to a file located in the directory `ebx.logs.directory`, with a threshold set to `DEBUG`.

The property `ebx.log4j.appender.ebxFile.backup.Threshold` allows defining of the maximum number of backup files for daily rollover.

```

#####
## Directory of log files 'ebxFile:'
## This property is used by special appender prefixed
## by 'ebxFile:' (see log section below)
#####
ebx.logs.directory=${ebx.home}/ebxLog

#####
# Daily rollover threshold of log files 'ebxFile:'
# Specifies the maximum number of backup files for daily rollover of 'ebxFile:' appenders.
# When set to a negative value, backup log files are never purged.
# Default value is -1.
#####
ebx.log4j.appender.ebxFile.backup.Threshold=-1

```

Custom SMTP appender

The appender `com.onwbp.org.apache.log4j.net.SMTPAppender` provides an asynchronous email sender.

Voir aussi [Activating and configuring SMTP and e-mails](#) [p 367]

57.9 Activating and configuring SMTP and e-mails

The internal mail manager sends emails asynchronously. It is used by the workflow engine and by the custom SMTP appender `com.onwbp.org.apache.log4j.net.SMTPAppender`.

Voir aussi [Mail sessions](#) [p 352]

```
#####
## SMTP and e-mails
#####

## Activate e-mails (true or false, default is false).
## If activated, the deployer must ensure that the entry 'mail/EBX_MAIL_SESSION' is bound
## in the operational environment of the application server (except if a specific email
## configuration is used by setting the property ebx.mail.smtp.host below).
#ebx.mail.activate=false

## Polling interval is in seconds (default is 10).
#ebx.mail.polling.interval=10

## Specific properties to be set only if you want to ignore the standard
## deployment process of 'ebx' web application in the target operational environment
## (see the deployment descriptor 'web.xml' of 'ebx' web application).
#ebx.mail.smtp.host = smtp.domain.com
## SMTP port default is 25.
#ebx.mail.smtp.port= 25
#ebx.mail.smtp.login=
#ebx.mail.smtp.password=
## Activate SSL (true or false, default is false).
## If SSL is activated, a SSL factory and a SSL provider are required.
#ebx.mail.smtp.ssl.activate=true
#ebx.mail.smtp.ssl.provider=com.sun.net.ssl.internal.ssl.Provider
#ebx.mail.smtp.ssl.factory=javax.net.ssl.SSLSocketFactory
```

57.10 Configuring data services

```
#####
## Data services
#####

# Specifies the default value of the data services parameter 'disableRedirectionToLastBroadcast'.
# Default is false.
#ebx.dataservices.disableRedirectionToLastBroadcast.default=false

# Specifies the default value for deletion at the end of close and merge operations.
# If the parameter is set in the request operation, it overrides this default setting.
# If unspecified, default is false.
#ebx.dataservices.dataDeletionOnCloseOrMerge.default=false
#ebx.dataservices.historyDeletionOnCloseOrMerge.default=false

# Upon WSDL generation, specifies if the target namespace value corresponds to the content before 5.5.0 'ebx-
services'
# or 'urn:ebx:ebx-services' in conformity with the URI syntax.
# If the parameter is set to true, there is no check of the target namespace as URI at the WSDL generation.
# If unspecified, default is false.
#ebx.dataservices.wsdlTargetNamespace.disabledCheck=false
```

57.11 Activating and configuring JMS

Voir aussi [JMS for data services](#) [p 353]

```
#####
## JMS configuration for Data Services
#####

## Activates JMS (true or false, default is false).
## If activated, the deployer must ensure that the entry 'jms/EBX_JMSConnectionFactory'
## are bound in the operational environment of the application server.
## The entry 'jms/EBX_QueueIn' should also be bound to enable handling Data Services request
## using JMS.
#ebx.jms.activate=false

## Activates JMS queue for failures (true or false, default is false).
## If activated, the deployer must ensure that the entry 'jms/EBX_QueueFailure' is bound
## in the operational environment of the application server.
#ebx.jms.activate.queueFailure=false

## Number of concurrent listener(s)
```



```
## Default is 3.
## Property is used if ebx.jms.activate is set to true.
#ebx.jms.listeners.count=3
```

57.12 Configuring distributed data delivery (D3)

See [Configuring D3 nodes](#) [p 439] for the main configuration file properties pertaining to D3.

Voir aussi

[JMS for distributed data delivery \(D3\)](#) [p 354]

[Introduction to D3](#) [p 428]

57.13 Configuring web access from end-user browsers

URLs computing

By default, EBX5 runs in "standalone" mode, where external resources (images, JavaScript, etc.) are provided by the application server.

Also by default, URL-related parameters in the main configuration file do not have to be set.

In this case, the server name and the port are obtained from the initial request sent to EBX5.

```
#####
## EBX5 FrontServlet: default properties for computing servlet address
##
## {useLocalUrl}:
## If set to true, servlet address is a "local absolute" URL.
## (that is, a relative URL consisting of an absolute path: "/path")
## See RFC 2396, http://www.ietf.org/rfc/rfc2396.txt).
## This property is defined once for HTTP and HTTPS.
## Default value is false.
##
## {host}:
## If neither defined nor adapted, retrieves initial request host
## {port}:
## If neither defined nor adapted, retrieves initial request host
## {path}:
## Mandatory, may be empty
##
## Resulting address will be:
## protocol://{host}:{port}/{path}
##
## Each property for HTTP (except {port}) may be inherited from HTTPS property,
## and reciprocally.
#####

#ebx.servlet.useLocalUrl=true
#ebx.servlet.http.host=
#ebx.servlet.http.port=
#ebx.servlet.http.path=ebx/
#ebx.servlet.https.host=
#ebx.servlet.https.port=
#ebx.servlet.https.path=

#####
## External resources: default properties for computing external resources address
##
## The same rules apply as EBX5 FrontServlet properties (see comments).
##
## Each property may be inherited from EBX5 FrontServlet.
#####

#ebx.externalResources.useLocalUrl=true
#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
#ebx.externalResources.http.path=
#ebx.externalResources.https.host=
#ebx.externalResources.https.port=
#ebx.externalResources.https.path=
```

Proxy mode

Proxy mode allows using a front-end HTTP server to provide static resources (images, CSS, JavaScript, etc.). This architecture reduces the load on the application server for static HTTP requests. This configuration also allows using SSL security on the front-end server.

The web server sends requests to the application server according to a path in the URL. This `servletAlias` path is specified in the main configuration file.

The web server provides all external resources. These resources are stored in a dedicated directory, accessible using the path `resourcesAlias`.

EBX5 must also be able to access external resources from the file system. To do so, the property `ebx.webapps.directory.externalResources` must be specified.

The main configuration file is configured as follows:

```
#####
## Path for external resources if they are not
## delivered within web applications
## This field is mandatory if in proxy mode.
#####
ebx.webapps.directory.externalResources=
D:/http/resourcesFolder

#####

#ebx.servlet.useLocalUrl=true
#ebx.servlet.http.host=
#ebx.servlet.http.port=
ebx.servlet.http.path= servletAlias
#ebx.servlet.https.host=
#ebx.servlet.https.port=
ebx.servlet.https.path= servletAlias

#####

#ebx.externalResources.useLocalUrl=true
#ebx.externalResources.http.host=
#ebx.externalResources.http.port=
ebx.externalResources.http.path= resourcesAlias
#ebx.externalResources.https.host=
#ebx.externalResources.https.port=
ebx.externalResources.https.path= resourcesAlias
```

Reverse proxy mode

URLs generated by EBX5 for requests and external resources must contain a server name, a port number and a specific path prefix.

The properties are configured as follows:

```
#####
ebx.servlet.http.host= reverseDomain
#ebx.servlet.http.port=
ebx.servlet.http.path=ebx/
ebx.servlet.https.host= reverseDomain
#ebx.servlet.https.port=
ebx.servlet.https.path=ebx/
#####
## Web parameters (for external resources)
## if nothing is set, values are taken from servlet.
#####
ebx.externalResources.http.host= reverseDomain
#ebx.externalResources.http.port=
#ebx.externalResources.http.path=ebx/
ebx.externalResources.https.host= reverseDomain
#ebx.externalResources.https.port=
ebx.externalResources.https.path=ebx/
```

URL parameter encoding

URLs generated by EBX5 may contain sensitive characters according to some security policies. This can be avoided by enforcing strong URL encoding.

The properties are configured as follows:

```
#####
## URL parameter strong encoding
##
## If true, sensitive parameters are strongly encoded,
## by removing special characters (./'=", etc.) from URLs.
##
## Default value is false.
#####
ebx.urlParameters.strongEncoding=true
```

57.14 Configuring failover

These parameters are used to configure failover mode and activation key, as well as heartbeat logging in DEBUG mode.

Voir aussi [Failover with hot-standby](#) [p 381]

```
#####
## Mode used to qualify the way in which a server accesses the repository.
## Possible values are: unique, failovermain, failoverstandby.
## Default value is: unique.
#####
#ebx.repository.ownership.mode=unique

## Activation key used in case of failover. The backup server must include this
## key in the HTTP request used to transfer exclusive ownership of the repository.
## The activation key must be an alphanumeric ASCII string longer than 8 characters.
#ebx.repository.ownership.activationkey=

## Specifies whether to hide heartbeat logging in DEBUG mode.
## Default value is true.
#ebx.repository.ownership.hideHeartBeatLogForDebug=true
```

57.15 Tuning the EBX5 repository

Some options can be set so as to optimize memory usage.

The properties are configured as follows:

```
#####
## Technical parameters for memory and performance tuning
#####
# Import commit threshold allows to specify the commit threshold
# exclusively for the archive import launched directly from Manager.
#
# For more details about the commit threshold,
# see the JavaDoc ProcedureContext.setCommitThreshold().
# Default value is 0.
#
ebx.manager.import.commit.threshold=100
# Validation messages threshold allows to specify the maximum number of
# messages to consider when performing a validation.
# This threshold is considered for each severity in each validation report.
# When the threshold is reached:
# - for severities 'error' or 'fatal', the validation is stopped.
# - for severities 'info' or 'warning', the validation continues without
# registering messages beyond the threshold. However the number of messages
# is still counted and messages of other severities can still be added.
#
# When set to 0 or a negative value the threshold is not considered.
# Default value is 0.
#
```

```

ebx.validation.messages.threshold=100

# Specifies whether the validation report should be kept in memory,
# regardless of the loading strategy of the data space.
# Default value is true. However, it is recommended to deactivate it
# when the repository contains a large number of open data spaces and
# data sets.
#ebx.validation.report.keepInMemory=false

```

57.16 Miscellaneous

Activating data workflows

This parameter specifies whether data workflows are activated. This parameter is not taken into account on the fly. The server must be restarted whenever the value changes.

```

#####
## Workflow activation.
## Default is false.
#####
ebx.workflow.activation = true

```

Log procedure starts

This parameter specifies whether starts of procedure execution are logged.

```

#####
## Specifies whether transaction starts are logged. Default is false.
#####
ebx.logs.logTransactionStart = true

```

Log validation starts

This parameter specifies whether starts of data sets validation are logged.

```

#####
## Specifies whether validation starts are logged. Default is false.
#####
ebx.logs.logValidationStart = true

```

Deployment site identification

This parameter allows specifying the email address to which technical log emails are sent.

```

#####
## Unique Site Name
## --> used by monitoring emails and by the repository
#####
ebx.site.name= name@domain.com

```

Dynamically reloading the main configuration

Some parameters can be dynamically reloaded, without restarting EBX5. The parameter `thisfile.checks.intervalInSeconds` indicates how frequently the main configuration file is checked.

```

#####
### Checks if this file has been updated
### If value <= 0, no more checks will be done
#####
thisfile.checks.intervalInSeconds=1

```

In development mode, this parameter can be set as low as one second. On production systems, where changes are expected to be less frequent, the value can be greater, or set to '0' to disable hot reloading entirely.

This property is not always supported when the module is deployed as a WAR, as it would then depend on the application server.

Applications that use the EBX5 navigation engine

Application template reloading:

```
#####
## Reload templates when it is updated
## (default value for all EBX5 modules).
## (value can be overridden by each EBX5 module.
#####
templates.checksIfUpdated=true
```

Debug mode in the EBX5 module web pages (only used when developing):

```
#####
## End-User Debug Mode
## (default for all EBX5 modules ).
## Debug information appears on end-user web page.
#####
frontEnd.debugMode=false
```

Module registration

When the application server is started, all web applications declared as EBX5 modules have to be registered. Concurrently, depending on the loading strategy of data spaces, the deployment of the web application 'ebx' may require the compilation of data models and their modules. Since these modules may not yet be registered, as it depends on the order of web application deployment at server startup, a wait loop is implemented. This gives the module time to be registered.

Voir aussi [Packaging EBX5 modules](#) [p 453]

```
#####
## When the application server is started, all web applications declared as
## EBX5 modules have to register. This property
## specifies the estimated time in seconds taken by the application server
## to deploy all its web applications at startup. Beyond this time,
## if a required module has not yet registered, it is considered to be absent
## and an error is reported to 'kernel' log.
## Default is 30 seconds.
#####
#ebx.module.timeInSecondsForModuleRegistration=30
```

EBX5 run mode

This property defines how EBX5 runs. Three modes are available: *development*, *integration* and *production*.

When running in *development* mode, the [development tools](#) [p 523] are activated in EBX5 and more technical information is displayed.

Note

The administrator always has access to this information regardless of the mode used.

The technical information is as follows:

Documentation pane	In the case of a computed value, the Java class name is displayed. The button to get the path to a node is also displayed.
Data model assistant	Data model configuration and additional options, such as Services, Business Objects and Rules, Java Bindings, and some advanced properties.
Validation report	The Validation tab is displayed.
Log	The logs include additional technical information intended for the developer. For example, a warning is written to logs if a drop-down list is defined on a node which is not an enumeration in a UI Bean.

```
#####
## Server Mode
## Value must be one of: development, integration, production
## Default is production.
#####
backend.mode=integration
```

Note

There is no difference between the modes *integration* and *production*.

Resource filtering

This property allows the filtering of certain files and directories in the resource directory contents (resource type node, with an associated facet that indicates the directory that contains usable resources).

```
#####
## list (separated by comma) of regexps excluding resource
## the regexp must be of type "m:[pattern]:[options]".
## the list can be void
#####
ebx.resource.exclude=m:CVS/*:
```

Downgrading to IE8 display

With Internet Explorer, if EBX5 is included in a portal which is forced to "Compatibility Mode", the [IE compatibility mode](#) [p 399] of EBX5 is not applied by Internet Explorer. It is thus impossible to ensure a good display of graphics (such as images or beautiful title fonts). The display must be downgraded to IE8 mode for all Internet Explorer versions starting from 9.

```
#####
## Specifies whether the Internet Explorer display must be downgraded to version 8.
## Default value is false.
#####
#ebx.browser.ie.forceIE8=true
```

CHAPITRE 58

Installing a repository using the configuration assistant

The EBX5 Configuration Assistant helps with the initial configuration of the EBX5 repository. If EBX5 does not have a repository installed upon start up, the configuration assistant launches automatically.

Before starting the configuration of the repository, ensure that EBX5 is correctly deployed on the application server. See [Java EE deployment](#) [p 347].

Note

The EBX5 main configuration file must also be correctly configured. See [EBX5 main configuration file](#) [p 361].

Ce chapitre contient les sections suivantes :

1. [License key](#)
2. [Configuration steps](#)

58.1 License key

When you launch EBX5, the license key page displays automatically if no valid license key is available, that is, if there is no license key entered in the EBX5 main configuration file, or if the current license key has expired.

If you do not have a license key, you may obtain a trial license key at <http://www.orchestranetworks.com/support>.

58.2 Configuration steps

The EBX5 configuration assistant guides you through the following steps:

1. Validating the license agreement.
2. Configuring the repository.
3. Defining users in the default user and roles directory (if a custom directory is not defined).
4. Validating the information entered.
5. Installing the EBX5 repository.

Validating the license agreement

In order to proceed with the configuration, you must read and accept the product license agreement.

Configuring the repository

This page displays some of the properties defined in the EBX5 main configuration file. You also define several basic properties of the repository in this step.

Id of the repository (repositoryId)	Must uniquely identify the repository (in the scope of the enterprise). The identifier is 48 bits (6 bytes) long and is usually represented as 12 hexadecimal digits. This information is used for generating the Universally Unique Identifiers (UUIDs) of entities created in the repository, and also of transactions logged in the history. This identifier acts as the "UUID node", as specified by RFC 4122.
Repository label	Defines a user-friendly label that indicates the purpose and context of the repository.

Voir aussi [EBX5 main configuration file](#) [p 361]

Defining users in the default directory

If a custom user and roles directory is not defined in the EBX5 main configuration file, the configuration assistant allows you to define default users for the default user and roles directory.

An administrator user must be defined. You may optionally create a second user.

Voir aussi [Users and roles directory](#) [p 407]

Validating the information entered

Before proceeding with the installation of the repository, you can review the configuration of the repository and the information you have entered on the 'Configuration Summary' page. If you need to modify information, you can return to the previous pages using the configuration assistant's < **Back** buttons.

Once you have verified the configuration, click the button **Install the repository** > to proceed with the installation.

Installing the EBX5 repository

The repository installation is performed using the information that you provided. When the installation is complete, you are redirected to the repository login page.

CHAPITRE 59

Deploying and registering EBX5 add-ons

Note

Refer to the documentation of each add-on for additional installation and configuration information in conjunction with this documentation.

Ce chapitre contient les sections suivantes :

1. [Deploying an add-on module](#)
2. [Registering an add-on module](#)

59.1 Deploying an add-on module

Note

Each EBX5 add-on build is intended to run with a specific version of EBX5. Ensure that you have the correct build of the add-on for the version of EBX5 on which it will run.

The web application deployment descriptor for the add-on module must specify that class definitions and resources from the web application are to be loaded in preference to classes from the parent and server classloaders.

For example, on WebSphere Application Server, this can be done by setting `<context-priority-classloader>true</context-priority-classloader>` in the web-app element of the deployment descriptor.

On WebLogic, include `<prefer-web-inf-classes>True</prefer-web-inf-classes>` in `weblogic.xml`.

See the documentation on class loading of your application server for more information.

59.2 Registering an add-on module

To register a new EBX5 add-on in the repository:

1. Navigate to the 'Administration' area.
2. Click the down-arrow in the navigation pane and select the 'Add-ons' entry.
3. On the **Add-ons > Registered Add-ons** page, click the + button to create a new entry.
4. Select the add-on you are registering, and enter its license key.

Note

If the EBX5 repository is under a trial license, no license key is required for the add-on. The add-on will be subject to the same trial period as the EBX5 repository itself.

5. Click **Submit**.

Technical administration

CHAPITRE 60

Repository administration

Ce chapitre contient les sections suivantes :

1. [Technical architecture](#)
2. [Repository installation and upgrade](#)
3. [Repository backup](#)
4. [Archives directory](#)
5. [Repository attributes](#)
6. [Monitoring and cleanup of relational database](#)
7. [Monitoring and cleanup of file system](#)

60.1 Technical architecture

Overview

The main principles of the EBX5 technical architecture are as follows:

- A Java process (JVM) that runs EBX5 is limited to a single EBX5 repository. This repository is physically persisted in a [supported relational database instance](#) [p 345], accessed through a [configured data source](#) [p 363].
- A repository cannot be shared by multiple JVMs at any given time. However, a failover architecture may be used. These aspects are detailed in the sections [Single JVM per repository](#) [p 381] and [Failover with hot-standby](#) [p 381]. Furthermore, to achieve horizontal scalability, an alternative is to deploy a [distributed data delivery \(D3\)](#) [p 428] environment.
- A single relational database instance can support multiple EBX5 repositories (used by distinct JVMs). It is then required that they specify distinct table prefixes using the property `ebx.persistence.table.prefix`.

Voir aussi

[Configuring the EBX5 repository](#) [p 363]

[Supported databases](#) [p 345]

[Data source of the EBX5 repository](#) [p 352]

Rules for the access to the database and user privileges

Attention

In order to guarantee the integrity of persisted master data, **it is strictly forbidden to perform direct SQL writes to the database**, except in the specific use cases described in the section [SQL access to data in relational mode](#) [p 286].

It is recommended for the database user specified by the [configured data source](#) [p 363] to have 'create/alter' privileges on tables, indexes and sequences. This allows [automatic repository installation and upgrades](#) [p 383]. If this is not the case, it is still possible to perform [manual repository installation and upgrades](#) [p 383], however, [Relational mode](#) [p 283] and [History](#) [p 289] will be more inconvenient to maintain since modification of concerned models will require performing manual steps.

Voir aussi

[SQL access to history](#) [p 292]

[Accessing a replica table using SQL](#) [p 299]

[SQL access to data in relational mode](#) [p 286]

[Data source of the EBX5 repository](#) [p 352]

Single JVM per repository

A repository cannot be shared by multiple JVMs. If such a situation were to occur, it would lead to unpredictable behavior and potentially even corruption of data in the repository.

EBX5 performs checks to enforce this restriction. Before the repository becomes available, the repository must first acquire exclusive ownership on the relational database. After starting the repository, the JVM periodically checks that it still holds ownership of the repository.

These checks are performed by repeatedly tagging a technical table in the relational database. The shutdown command for the application server ensures that the tag on this technical table is removed. If the server shuts down unexpectedly, the tag may be left in the table. If this occurs, the server must wait several additional seconds upon restart to ensure that the table is not being updated by another live process.

Attention

To avoid an additional wait period upon the next start up, it is recommended to always cleanly shut down the application server.

Failover with hot-standby

The exclusion mechanism described above is compatible with failover architectures, where only one server is active at any given time in an active/passive cluster. To ensure this is the case, the main server must declare the property `ebx.repository.ownership.mode=failovermain`. The main server claims the repository database, as in the case of a single server.

A backup server can still start up, but it will not have access to the repository. It must declare the property `ebx.repository.ownership.mode=failoverstandby` to act as the backup server. Once started, the backup server is registered in the connection log. Its status can be retrieved using the Java

API or through an HTTP request, as described in the section [Repository status information and logs](#) [p 382] below.

In order to activate the backup server and transfer exclusive ownership of the repository to it, a specific request must be issued by an HTTP request or using the Java API:

- Using HTTP, the request must include the parameter `activationKeyFromStandbyMode`, and the value of this parameter must be equal to the value declared for the entry `ebx.repository.ownership.activationkey` in the EBX5 main configuration file. See [Configuring failover](#) [p 371].
- Using the Java API, call the method `RepositoryStatus.wakeFromStandbyAPI`.

The backup server then requests a clean shutdown for the main server, allowing any running transactions to finish. Only after the main server has yielded ownership can the backup server start using the repository.

Repository status information and logs

A log of all attempted Java process connections to the repository is available in the Administration area under 'History and logs' > 'Repository connection log'.

The status of the repository may be retrieved using the methods in the `RepositoryStatusAPI` API.

It is also possible to get repository status information using a HTTP request that includes the parameter `repositoryInformationRequest` with one of following values:

state	<p>The state of the repository in terms of ownership registration.</p> <ul style="list-style-type: none"> • D: Java process is stopped. • O: Java process has exclusive ownership of the database. • S: Java process is started in failover standby mode, but is not yet allowed to interact with the repository. • N: Java process has tried to take ownership of the database but failed because another process is holding it.
heart_beat_count	<p>The number of times that the repository has made contact since associating with the database.</p>
info	<p>Detailed information for the end user regarding the repository's registration status. The format of this information may be subject to modifications in future without explicit warning.</p>

60.2 Repository installation and upgrade

Automatic installation and upgrades

If the specified user for the EBX5 data source has permission to create and alter tables, indexes and sequences in the relational database, then the repository installation or upgrade is done automatically. Otherwise, an administrator must manually carry out the procedure detailed in the next section [Manual installation and upgrades](#) [p 383].

Note

Concerning relational mode and history, declaring a table in relational mode or history mode creates a dedicated table in the database. Similarly, in these modes, modifying the structure of an existing table alters its declaration in the database. Modifications may take place at any time for reasons other than repository creation. The EBX5 application database user must have 'create/alter' privileges on tables and indexes, since these actions must be carried out from EBX5.

Manual installation and upgrades

If the user specified for the EBX5 data source does *not* have permission to create or alter tables, indexes or sequences in the relational database, the administrator must carefully execute all the SQL scripts, located in the `ebx.software/files/ddl/` directory (select the folder corresponding to your RDBMS and then execute the scripts in ascending order).

If manually upgrading an existing repository, the administrator must execute the required subset of these SQL scripts; the file name of each script indicates the EBX5 version to which the patch applies.

Note

If a specific table prefix is specified by the property `ebx.persistence.table.prefix`, the default `EBX_` prefix must be modified accordingly in the provided scripts.

Voir aussi [Configuring the EBX5 repository](#) [p 363]

60.3 Repository backup

A global backup of the EBX5 repository must be delegated to the underlying RDBMS. The database administrator must use the standard backup procedures of the underlying database.

60.4 Archives directory

Archives are stored in a sub-directory named `archives` within `ebx.repository.directory` (see [configuration](#) [p 361]). This directory is automatically created during the first export from EBX5.

Attention

If manually creating this directory, ensure that the EBX5 process has read-write access to it. Furthermore, the administrator is responsible for cleaning this directory, as EBX5 does not maintain it.

Note

A file transferred between two EBX5 environments must be done outside of the product using tools such as FTP or simple file copies by network sharing.

60.5 Repository attributes

A repository has the following attributes:

repositoryId	Uniquely identifies a repository within the scope of the enterprise). It is 48 bits (6 bytes) and is usually represented as 12 hexadecimal digits. This information is used for generating UUIDs (Universally Unique Identifier) for entities created in the repository, as well as transactions logged in history tables or the XML audit trail. This identifier acts as the 'UUID node' part, as specified by <i>RFC 4122</i> .
repository label	Provides a user-friendly label that identifies the purpose and context of the repository. For example, "Production environment".
store format	Identifies the underlying persistence system, including the current version of its structure.

60.6 Monitoring and cleanup of relational database

Some entities accumulate during the execution of EBX5.

Attention

It is the *administrator's responsibility* to monitor and to clean up these entities.

Monitoring and reorganization

The persistence data source of the repository must be monitored through RDBMS monitoring.

The EBX5 tables specified in the default [semantic mode](#) [p 283] have their content persisted in a set of generic database tables. They are the following:

- The table `{ebx.persistence.table.prefix}HOM`, in which each record represents a data space or a snapshot (its name is `EBX_HOM` if the property `ebx.persistence.table.prefix` is unset).
- The table `{ebx.persistence.table.prefix}BRV`, where the data of EBX5 tables in semantic mode are segmented into blocks of at most 100 EBX5 records (its name is `EBX_BRV` if the property `ebx.persistence.table.prefix` is unset).
- The table `{ebx.persistence.table.prefix}HTB`, which defines which blocks belong to a given EBX5 table in a given data space or snapshot (its name is `EBX_HTB` if the property `ebx.persistence.table.prefix` is unset).

Note

For repositories with large volumes of data in semantic mode, and on which frequent or heavy updates occur, it may be necessary to schedule a regular reorganization of the above database tables suffixed by HTB and BRV, as well as their indexes. This is especially true for large repositories where many data spaces are created and deleted.

Database statistics

The performance of requests executed by EBX5 requires that the database has computed up-to-date statistics on its tables. Since database engines regularly schedule statistics updates, this should generally not be an issue. Yet, it could be needed to explicitly update the statistics in cases where tables are heavily modified over a short period of time (e.g. by an import which creates many records).

Impact on UI

Some UI component use these statistics to adapt their behavior, in order to prevent user from executing costly requests unwillingly.

For example, the combo box will not automatically search on user input if the table has a large volume of records. This behavior may also occur if the data base's statistics are not up to date, because a table may be considered as having a large volume of records even if it is not the case.

Cleaning up data spaces, snapshots, and history

A full cleanup of data spaces, snapshots, and history from the repository involves several stages:

1. Closing unused data spaces and snapshots to keep the cache to a minimal size.
2. Deleting data spaces, snapshots, and history.
3. Purging the remaining entities associated with the deleted data spaces, snapshots, and history from the repository.

Closing unused data spaces and snapshots

In order to keep the cache and the repository to a reasonable size, it is recommended to close any data spaces and snapshots that are no longer required. This can be done in the following ways:

- Through the user interface, in the Data Spaces area.
- From the 'Data spaces / Snapshots' table under 'Data spaces' in the Administration area, using the **Actions** menu in the workspace. The action can be used on a filtered view of the table.
- Through the Java API, using the method `Repository.closeHomeAPI`.
- Using the data service close data space and close snapshot operations. See [Closing a data space or snapshot](#) [p 249] for more information.

Once the data spaces and snapshots have been closed, the data and associated history can be cleaned from the repository.

Note

Closed data spaces and snapshots can be reopened in the Administration area, under 'Data spaces'.

Deleting data spaces, snapshots, and history

Data spaces, their associated history, and snapshots can be permanently deleted from the repository. After you have deleted a data space or snapshot, some entities will remain until a repository-wide purge of all obsolete data is performed. Thus, both stages, the deletion and the repository-wide purge, must be performed in order to completely remove the data and history. This process has been separated into two steps for performance considerations. As the total clean-up of the repository can be time-intensive, this allows the purge execution to be initiated during off-peak periods on the server.

Note

The process of deleting the history of a data space takes into account all history transactions recorded up until the point that the deletion is submitted. Any subsequent historized operations will not be included when you run the purge operation. If you want to delete these new transactions, you must delete the history of that data space again.

The deletion of data spaces, snapshots, and history can be performed in a number of different ways:

- Using the dedicated 'Create deletion requests' interface, accessible from the Administration **Actions** menu in the Administration area. This interface presents all items available for deletion using hierarchical views, categorized into the following entries in the navigation panel:

Closed data spaces	Displays all data spaces and snapshots that are currently closed, and thus can be deleted. You can choose to delete the history associated with data spaces at the same time.
Open data spaces	Displays all data spaces that are currently open, for which you can delete the associated history.
Deleted data spaces	Displays all data spaces that have already been deleted, but have associated history remaining in the repository. You may delete this remaining history.

- From the 'Data spaces / Snapshots' table under 'Data spaces' in the Administration area, using the **Actions** menu in the workspace. The action can be used on a filtered view of the table.
- Using the Java API, specifically, the methods `Repository.deleteHomeAPI` and `RepositoryPurge.markHomeForHistoryPurgeAPI`.
- At the end of data service close data space operation, using the parameters `deleteDataOnClose` and `deleteHistoryOnClose`, or at the end of the merge data space operation, using the parameters `deleteDataOnMerge` and `deleteHistoryOnMerge`.

Purging remaining entities after data space, snapshot, and history deletion

Once items have been deleted, a purge can be executed to clean up the remaining data from *all* deletions performed up until that point. A purge can be initiated in the following ways:

- Through the user interface, by selecting Administration **Actions** > **Execute purge** in the navigation pane.
- Using the Java API, specifically the method `RepositoryPurge.purgeAllAPI`.
- Using the task scheduler. See [Task scheduler](#) [p 421] for more information.

The purge process is logged into the directory `${ebx.repository.directory}/db.purge/`.

Cleaning up tables having unreadable records

Some data model model evolutions can lead to unreadable data:

- A column containing null values is added to the primary key of a table.
- The type of a column has changed to a different type with no possible conversion.

In these situations, records that do not match the new table definition are no longer visible, but remain persisted in the table. (This allows retrieving the records if switching back to the previous table definition). When such records are encountered, an informative error is recorded in EBX5 logs.

EBX5 provides the option to clean the records that no longer conform to the model, once the new version of the data model is stabilized. This allows recovering space in the database and getting rid

of error messages. Please proceed carefully, as this operation permanently removes all unreadable records from the selected table, and cannot be undone.

Cleaning up unreadable records is done by selecting Administration **Actions** > **Clean up unreadable records** in the navigation pane.

Cleaning up other repository entities

It is the *administrator's responsibility* to monitor and to regularly cleanup the following entities.

Cleaning up database resources of mapped tables

EBX5 gives the ability to purge mapped tables wherever it detects a mapped table in the database that is no longer used. This means that mapped mode must be deactivated before the mapped table database resources can be purged.

To deactivate mapped mode for a table, follow the procedure below.

For history mode:

- Deactivate historization of the table in the data model, or
- Remove the table from the data model

For relational mode:

- Remove the table from the data model, or
- Deactivate the relational mode on the data model. As data models in semantic mode cannot be used for relational data spaces, it is thus necessary to create a data set on a semantic data space using this modified data model. EBX5 will then detect that relational mode was previously used, and will therefore propose the relational table database resources for purge.

Once mapped mode has been deactivated, you can perform a clean-up procedure similar to the process described in [Deleting data spaces, snapshots, and history](#) [p 386]. To select the tables to clean up, open the 'Create deletion requests' interface, accessible from the Administration **Actions** menu in the Administration area. Select 'Database tables' in the navigation pane.

A purge can then be executed to clean up the remaining data from *all* deletions, that is, deleted data spaces, snapshots, history, and database resources, performed up until that point. A purge can be initiated by selecting Administration **Actions** > **Execute purge** in the navigation pane.

Task scheduler execution reports

Task scheduler execution reports are persisted in the 'executions report' table, in the 'Task scheduler' section of the Administration area. Scheduled tasks constantly add to this table as they are executed. Even when an execution terminates normally, the records are not automatically deleted. It is thus recommended to delete old records regularly.

User interactions

User interactions are used by the EBX5 component as a reliable means for an application to initiate and get the result of a service execution. They are persisted in the *ebx-interactions* administration section. It is recommended to regularly monitor the user interactions table, as well as to clean it, if needed.

Workflow history

The workflow events are persisted in the workflow history table, in the 'Workflow' section of the Administration area. Data workflows constantly add to this table as they are executed. Even when an

execution terminates normally, the records are not automatically deleted. It is thus recommended to delete old records regularly.

60.7 Monitoring and cleanup of file system

Attention

In order to guarantee the correct operation of EBX5, the disk usage and disk availability of the following directories must be supervised by the administrator, as EBX5 does not perform any cleanup:

- **XML audit trail:** `${ebx.repository.directory}/History/`
- **Archives:** `${ebx.repository.directory}/archives/`
- **Logs:** [ebx.logs.directory](#) [p 366]
- **Temporary directory:** [ebx.temp.directory](#)

Attention

For **XML audit trail**, if large transactions are executed with full update details activated (the default setting), the required disk space can be quite high.

Attention

For pagination in the data services `getChanges` operation, a persistent store is used in the **Temporary directory**. Large changes may require a large amount of disk space.

Voir aussi

[XML audit Trail](#) [p 412]

[Tuning the EBX5 repository](#) [p 371]

CHAPITRE 61

Front end administration

Several administrative tasks can be performed from the Administration area of the EBX5 user interface.

Ce chapitre contient les sections suivantes :

1. [Administration tools](#)
2. [Data spaces](#)
3. [User directory](#)
4. [Global permissions](#)
5. [Administrative delegation](#)
6. [Perspectives configuration](#)
7. [User interface configuration](#)
8. [Lineage](#)
9. [Event broker](#)
10. [Other technical tables](#)

61.1 Administration tools

By clicking on 'Actions' under the 'Administration' area, basic administration tools are available.

System information

This page lists information related to the repository, the operating system and EBX5 configuration.

Modules and data models

This tool lists all registered modules as well as existing data models.

User sessions

This tool lists all user sessions and allows terminating active sessions.

61.2 Data spaces

Some data space administrative tasks can be performed from the Administration area of EBX5 by selecting 'Data spaces'.

Data spaces/snapshots

This table lists all the existing data spaces and snapshots in the repository, whether open or closed. You can view and modify the information of data spaces included in this table.

Voir aussi [Data space information](#) [p 84]

From this section, it is also possible to close open data spaces, reopen previously closed data spaces, as well as delete and purge open or closed data spaces, associated history, and snapshots.

Voir aussi [Cleaning up data spaces, snapshots, and history](#) [p 385]

Data space permissions

This table lists all the existing permission rules defined on all the data spaces in the repository. You can view the permission rules and modify their information.

Voir aussi [Permissions sur un espace de données](#) [p 86]

Repository history

The table 'Deleted data spaces/snapshots' lists all the data spaces that have already been purged from the repository.

61.3 User directory

If the default directory provided and integrated into EBX5 is used, the 'Directory' administration section allows defining which users can connect and what their roles are.

Voir aussi [Users and roles directory](#) [p 407]

Policy

These properties configure the policies of the user and roles directory, for example, whether or not users can edit their own profiles.

Users table

This table lists all the users defined in the internal directory. New users can be added from there.

Roles table

This table lists all the user defined in the internal directory. New roles can be created in this table.

61.4 Global permissions

Global permission rules can be created in EBX5.

The 'Display area' property allows restricting access to areas of the software. To define the access rules, select 'Global permissions' in the 'Administration' area.

Profil	Indicates on which profile the rule will be applied.
Restriction d'accès	Indique si les permissions définies ici restreignent celles définies pour d'autres profils. See the Restriction policy concept [p 323] for more information.
Espaces de données	Définit les permissions pour la section Espaces de données.
Modèles de données	Définit les permissions pour la section Modèles de données.
Modèles de workflow	Définit les permissions pour la section Modèles de workflow.
Workflows de données	Définit les permissions pour les Workflows de données.
Services de données	Définit les permissions pour la section Services de données. And Définit les permissions d'accès à la servlet http(s) connecteur WSDL . [p 220].
Administration	Définit les permissions pour la section Administration.

Note

Permissions can be defined by administrators and by the data space or data set owner.

61.5 Administrative delegation

An administrator can delegate administrative rights to a non-administrator user, either for specific actions or for all activities.

The administrative delegation is defined under 'Administration' in the global permissions profile.

If all necessary administrative rights have been delegated to non-administrator users, it becomes possible to disable the built-in 'ADMINISTRATOR' role.

Voir aussi [Configuring the user and roles directory](#) [p 364]

61.6 Perspectives configuration

An unlimited number of perspectives can be configured. To each perspective corresponds a data set. Data set inheritance is available and allows sharing easily items between perspectives.

Voir aussi [Inheritance](#) [p 23]

A perspective menu is displayed on the left side vertical navigation bar and consists of menu items of different types.

Menu item types:

Section Menu Item	This is a top level menu item. It contains other menu items.
Group Menu Item	This is a container for other menu items.
Action Menu Item	This menu item displays a UI service in the workspace area.

Perspectives available by default

Root perspective

The root perspective can define menu items that are inherited by all other perspectives. It can be made available to end users but this is not recommended.

Advanced perspective

This perspective menus cannot be customized. The advanced perspective is available by default to all end-users but access can be restricted.

Note: Administrators can always access this perspective even when it is deactivated.

Perspective creation

To create a perspective, select the parent (or root) perspective and click on the + sign to create its corresponding child data set.

Voir aussi [Creating an inheriting child data set](#) [p 104]

Perspective properties

The available perspective properties are:

Activé	Indique que la perspective est visible des utilisateurs autorisés.
Profils Autorisés	La liste des profils utilisateur autorisés pour la perspective.
Sélection par défaut	L'élément de menu qui est sélectionné par défaut. This property is not available for the advanced perspective.

Menu view

This view displays the perspective menu. It is a hierarchical table view.

From this view, a user can create, delete or reorder menu item records.

Voir aussi [Hierarchical table view](#) [p 23]

Menu item properties

The available perspective properties are:

Type	<p>Le type de l'élément de menu</p> <p>Voir aussi Menu item types [p 394]</p>
Parent	<p>Le parent de l'élément de menu.</p> <p>This property is not available for section menu items.</p>
Libellé	<p>Le libellé de l'élément de menu.</p> <p>The label is optional for action menu items. If not specified, the label will be dynamically generated by EBX when the menu item is displayed.</p>
Icône	<p>L'icône pour l'élément de menu.</p> <p>Icon can be either "standard" (provided by EBX5) or an image, specified by an URL, that can be hosted on any web server.</p> <p>This property is not available for section menu items.</p>
Séparateur haut	<p>Indique que la section élément de menu a un séparateur haut.</p> <p>This property is only available for section menu items.</p>
Action	<p>The UI service to execute when the user clicks on the menu item.</p> <p>Voir aussi User interface services [p 189]</p> <p>If an end user is authorized to view the perspective but not to execute the UI service, an access denied message will be displayed when the user clicks on the menu item.</p> <p>This property is only available for action menu items.</p>
Sélection après fermeture	<p>L'élément de menu qui sera sélectionné quand le service se terminera.</p> <p>Built-in services use this property when the user clicks on the 'Close' button.</p> <p>This property is only available for action menu items.</p>

61.7 User interface configuration

Some options are available in the Administration area for configuring the web interface, in the 'User interface' section.

Attention

Be careful when configuring the 'URL Policy'. If the web interface configuration is invalid, it can lead to the EBX5 application being unusable. If this occurs, use the "rescue mode" by setting `frontEnd.rescueMode.enable=true` in [EBX5 main configuration file](#) [p 361], and accessing the following URL in your browser as a built-in administrator user: `http://.../ebx/?onwbpID=iebx-manager-rescue`.

Session configuration

These parameters configure the user session options:

User session default locale	Default session locale.
Session time-out (in seconds)	Maximum duration of user inactivity before the session is considered to be inactive and is terminated. A negative value indicates that the session should never timeout.

Interface configuration

Entry policy

Describes the URL to access the application.

Login URL	If the user is not authenticated, the session is forwarded to this URL.
------------------	-------------------------------------------------------------------------

The entry policy defines an EBX5 login page, replacing the default one.

If defined,

- it replaces an authentication URL that may have been defined using a specific user Directory^{API},
- it is used to build the permalinks in the user interface,
- if the URL is full, that is, starting with `http://` or `https://`, it replaces the URL of the workflow email configuration.

URL policy

Describes the URL and proxy policy. Both dynamic (servlet) and static (resources) URLs can be configured.

HTTP servlet policy	Header content of the servlet HTTP request: <ul style="list-style-type: none"> • if a field is not set, the default value in the environment configuration is used, • if a default value is not set, the value in the initial request is used.
HTTPS servlet policy	Header content of the servlet HTTPS request: <ul style="list-style-type: none"> • if a field is not set, the default value is chosen (in environment configuration), • if a default value is not set, the value in the initial request is used.
HTTP external resource policy	Header content of the external resource URL in HTTP: <ul style="list-style-type: none"> • if a field is not set, the default value in the environment configuration is used, • if a default value is not set, the value in the initial request is used.
HTTPS external resource policy	Header content of the external resource URL in HTTPS: <ul style="list-style-type: none"> • if a field is not set, the default value in the environment configuration is used, • if a default value is not set, the value in the initial request is used.

Exit policy

Describes how the application is exited.

Normal redirection	Specifies the redirection URL used when exiting the session normally.
Error redirection	Specifies the redirection URL used when exiting the session due to an error.

Graphical interface configuration

Application locking

EBX5 availability status:

Disponibilité	Pour maintenance, l'application peut être fermée au public (mais toujours accessible aux administrateurs). Les paramètres définis sont appliqués immédiatement.
Message d'indisponibilité	Message affiché aux utilisateurs quand l'accès est restreint aux administrateurs.


Security policy

EBX5 access security policy. These parameters only apply on new HTTP sessions.

Restriction d'accès IP	Restreindre l'accès aux adresses IP déclarées (voir ci-dessous).
Description de restriction IP	Regular expression representation of IP addresses authorized to access EBX5. For example, <code>((127\.\0\.\0\.\1) (192\.\168\.*\.*))</code> grants access to the local machine and the network IP range 192.168.*.*.
Unicité de session	Specifies whether EBX5 should control the uniqueness of user sessions. When set to 'Yes', if a user does not log out before closing the browser, it will not be possible for that user to log in again until the previous session has timed out.

Ergonomics and layout

EBX5 ergonomics parameters:

Nombre maximal de colonnes d'une table	En fonction des performances du réseau et du navigateur, ajustez le nombre maximal de colonnes d'une table à afficher (dans le contenu d'un jeu de données). Cette propriété n'est pas prise en compte lorsqu'une vue est appliquée sur une table.
Largeur automatique maximale des colonnes de tables	Cette valeur définit une largeur automatique maximale pour chaque colonne lors de l'initialisation de la table. Ceci permet d'éviter que les colonnes avec un contenu très long (tel qu'une URL) ne prennent trop de largeur. La largeur des colonnes reste modifiable avec la souris au delà de cette valeur.
Nombre maximal d'éléments développés d'une hiérarchie	Définit, pour les hiérarchies, la limite du nombre d'éléments qui peuvent être développés par l'action "Développer tout". Une valeur inférieur ou égale à 0 désactive ce paramètre.
Filtre de table sélectionné par défaut	Définit le filtre de table sélectionné par défaut dans la liste des filtres affichés avec la vue tabulaire. En cas de modification, les utilisateurs doivent se déconnecter et se reconnecter afin d'utiliser la nouvelle valeur.
Afficher la boîte de message automatiquement	Defines the message severity threshold for displaying the messages pop-up.
Mode de compatibilité IE	<p>Defines whether or not to compensate for Internet Explorer 8+ displaying EBX5 in compatibility mode.</p> <p>In order to prevent Internet Explorer browsers from using compatibility mode when displaying the repository user interface, the meta-tag <code>http-equiv="X-UA-Compatible" content="IE=EmulateIE8"</code> is added to the header of pages. However, in some local environments, this setting may conflict with existing policies, in which case this header must be omitted by setting the parameter to 'No'. The default value is 'Yes'.</p> <p>See Specifying Document Compatibility Modes  for more information.</p>
Formulaires : largeur des libellés	La largeur des libellés des champs dans les formulaires.

Formulaires : largeur des champs	La largeur des champs de saisie dans les formulaires.
Formulaires : hauteur des champs texte	La hauteur des champs de saisie de type text dans les formulaires.
Formulaires : édition de liste	Nombre de lignes cachées à générer dans l'interface utilisateur, disponibles pour créer de nouvelles occurrences dans la liste.
Formulaires : largeur de l'éditeur HTML	La largeur de l'éditeur HTML dans les formulaires.
Formulaires : hauteur de l'éditeur HTML	La hauteur de l'éditeur HTML dans les formulaires.
Sélection en liste : taille de page	Nombre de lignes affichées dans une page du composant de sélection dans une liste (utilisé pour sélectionner les clé étrangères, les énumérations, etc.).
Formulaire d'enregistrement : mode de présentation des noeuds	En fonction des performances du réseau et du navigateur, ajustez la manière d'afficher chaque noeud non terminal du formulaire d'enregistrement. En ce qui concerne le poids de la page téléchargée, le mode lien est léger, les modes développé et réduit sont plus lourds. En cas de modification de cette propriété, les utilisateurs doivent se déconnecter et se reconnecter afin que la nouvelle valeur soit prise en compte.

Default option values

Defines default values for options in the user interface.

Import/Export

CSV : jeu de caractères	Définit le jeu de caractères utilisé par défaut pour les imports et les exports CSV.
CSV : séparateur de champ	Définit le caractère séparateur utilisé par défaut pour les imports et les exports CSV.
CSV : séparateur de liste	Définit le caractère séparateur de liste utilisé par défaut pour les imports et les exports CSV.
Mode d'import	Spécifie le mode utilisé par défaut pour les imports.
Valeurs XML manquantes à nul	Si 'Oui', quand un nœud est manquant ou vide dans le fichier importé, la valeur est considérée comme 'nulle' lors de la mise à jour des enregistrements existants. Si 'Non', la valeur n'est pas modifiée.

Colors and themes

Customizes EBX5 colors and themes.

URL de l'icône du site (favicon)	Le format recommandé est ICO car il est compatible avec Internet Explorer.
URL du logo (SVG)	Laissez le champ vide pour utiliser l'image PNG. L'image SVG est utilisée sur les navigateurs compatibles. Le système utilisera le logo PNG si le navigateur n'est pas compatible. Si l'image PNG n'est pas renseignée, l'image GIF/JPG sera utilisée. Le logo doit avoir une hauteur maximale de 40px. Si la hauteur est supérieure à 40px, elle sera rognée à 40px de hauteur. La largeur du logo détermine la position des boutons du bandeau.
URL du logo (PNG)	L'image PNG est utilisée sur les navigateurs compatibles, sinon le système utilisera l'image GIF/JPG. Laissez ce champ et le champ du logo SVG vide pour utiliser l'image GIF/JPG. Le logo doit avoir une hauteur maximale de 40px. Si la hauteur est supérieure à 40px, elle sera rognée à 40px de hauteur. La largeur du logo détermine la position des boutons du bandeau.
URL du logo (GIF/JPG)	L'image GIF/JPG est utilisée quand les images PNG et SVG ne sont pas renseignées, ou sur Internet Explorer 6. Les formats recommandés sont GIF et JPG. Le logo doit avoir une hauteur maximale de 40px. Si la hauteur est supérieure à 40px, elle sera rognée à 40px de hauteur. La largeur du logo détermine la position des boutons du bandeau.
Principale	Couleur principale de l'interface utilisateur, utilisée pour les sélections et les surbrillances.
Secondaire	Couleur secondaire de l'interface utilisateur, utilisée pour certains textes.
Bandeau	Couleur de fond du bandeau de l'interface utilisateur. Par défaut, définie à la même valeur que la couleur principale.
Surbrillance du bandeau	Couleur de la surbrillance du bandeau. Par défaut, définie à la même valeur que la couleur secondaire.
Pastille du workflow	Couleurs de fond et de texte/bordure de la pastille du workflow (compteur de workflows utilisateur).

Page de login (fond)	Couleur de fond de la page de login. Par défaut, définie à une version plus claire de la couleur principale.
Boutons	Couleurs des boutons et luminosité du texte et de l'icone des boutons.
Surbrillance des boutons	Couleur des boutons en surbrillance et sélectionnés par défaut. Par défaut, définie à la même valeur que la couleur principale.
Onglets	Couleur des onglets des formulaires.
Surbrillance des onglets	Couleur des onglets sélectionnées. Par défaut, définie à la même valeur que la couleur principale.
Panneau de navigation	Couleur de fond du panneau de navigation.
Barre du bas du formulaire	Couleur de fond de la barre du bas du formulaire.
Vue historique de table : données techniques	Couleur de fond des cellules de données techniques dans la vue historique de table.
Vue historique de table : création	Couleur de fond des cellules ayant l'état 'création' dans la vue historique de table.
Vue historique de table : suppression	Couleur de fond des cellules ayant l'état 'suppression' dans la vue historique de table.
Vue historique de table : mise à jour	Couleur de fond des cellules ayant l'état 'mise à jour' dans la vue historique de table.

61.8 Lineage

To administrate lineage, three tables are accessible:

- **Authorized profiles:** Profiles must be added to this table to be used for data lineage WSDL generation.
- **History:** Lists the general data lineage WSDLs and their configurations.
- **JMS location:** Lists the JMS URL locations.

61.9 Event broker

Overview

EBX5 offers the ability to receive notifications and information related to specific events using the event broker feature. This feature consists in sending notifications related to EBX5 core events to the subscriber according to their chosen topic.

Terminology

Event broker	Notification component for loosely-coupled event handling. Consists of dispatching fired events from EBX5 core to concerned subscribers. The event broker is mainly used for monitoring and statistical purposes.
Topic	Corresponds to the EBX5 event type that contains messages. The number of subscribers registered to a topic is unlimited.
Subscriber	Client implementation in the modules that receive the events related to the subscribed topic(s).

Topics

Repository	Corresponds to operations in the repository, such as: start-up and purge.
Data space and snapshot	Corresponds to operations in the data space and in the snapshot, such as: create, close, reopen, delete, archive export and archive import (only for data space merge).
User session	Corresponds to the operations related to user authentication, such as: login and logout.

Administration

The management console is located under 'Event broker' in the 'Administration' area. It contains three tables: 'Topics', 'Subscribers' and 'Subscriptions'.

All content is read-only, except for the following operations:

- Topic and subscriber can be manually activated or deactivated using dedicated services.
- Subscribers that are no longer registered to the broker can be deleted.

61.10 Other technical tables

Several technical tables can be accessed in the 'Administration' area of the EBX5 user interface. These tables are for internal use only and their content should not be edited manually, unless removing obsolete or erroneous data.

Auto-increments	Lists all auto-increment fields in the repository.
Interactions	Lists all interactions in the repository.
Workflows	Lists all data workflows and their tokens in the repository, and the information associated with each, including history.
User preferences	Lists the user preferences of users of the repository.
Views configuration	Allows accessing the Views, Groups of views, User filters and Default views lists defined by users of the repository.

CHAPITRE 62

Users and roles directory

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Concepts](#)
3. [Default directory](#)
4. [Custom directory](#)

62.1 Overview

EBX5 uses a directory for user authentication and user role definition.

A default directory is provided and integrated into EBX5 repository. It is also possible to integrate another type of enterprise directory.

Voir aussi

[*Configuring the user and roles directory*](#) [p 364]

[*Custom directory*](#) [p 409]

62.2 Concepts

In EBX5, a user can be a member of several roles, and a role can be shared by several users. Moreover, a role can be included in another role. The generic term *profile* is used to describe either a user or a role.

In addition to the directory-defined roles, EBX5 provides the following *built-in roles*:

Role	Definition
Profile.ADMINISTRATOR	Built-in Administrator role. Allows performing general administrative tasks.
Profile.READ_ONLY	Built-in read-only role. A user associated with the read-only role can only view the EBX5 repository, with no rights for performing modifications in the repository.
Profile.OWNER	Dynamic built-in owner role. This role is checked dynamically depending on the current element. It is only activated if the user belongs to the profile defined as owner of the current element.
Profile.EVERYONE	All users belong to this role.

Information related to profiles is primarily defined in the directory.

Attention

Associations between users and the built-in roles *OWNER* and *EVERYONE* are managed automatically by EBX5, and thus must not be modified by the directory.

User permissions are managed separately from the directory. See [Permissions](#) [p 317].

Voir aussi

[profil](#) [p 19]

[rôle](#) [p 20]

[utilisateur](#) [p 19]

[administrateur](#) [p 20]

[annuaire des utilisateurs et des rôles](#) [p 20]

62.3 Default directory

Directory content

The default directory is represented by the data set 'Directory', in the Administration area.

This data set contains tables for users and roles.

Depending on the policies defined, users can modify information related to their own accounts.

Note

It is not possible to delete or duplicate the default directory.

Password recovery procedure

In the default directory, passwords are encrypted (by default with a SHA256-like algorithm), and stored in this state. Consequently, it is impossible to retrieve lost passwords. A new password must be generated and sent to the user.

There are two options for this procedure:

1. A notification email is sent to the administrator, the administrator manually changes the password and sends the new password to the user.
2. A procedure automatically generates a new password and sends it to the user.

By default, the first option is used. To activate the second option, specify the property `ebx.password.remind.auto=true` in the [EBX5 main configuration file](#) [p 361].

Note

For security reasons, the password recovery procedure is not available for administrator profiles. If required, use the administrator recovery procedure instead.

Administrator recovery procedure

If all the 'login/password' credentials of the administrators are lost, a special procedure must be followed. A specific directory class redefines an administrator user with login 'admin' and password 'admin'.

To activate this procedure:

- Specify the following property in the [EBX5 main configuration file](#) [p 361]:
`ebx.directory.factory=
com.orchestranetworks.service.directory.DirectoryDefaultRecoverFactory`
- Start EBX5 and wait until the procedure completes.
- Reset the 'ebx.directory.factory' property.
- Restart EBX5 and connect using the 'admin' account.

Note

While the 'ebx.directory.factory' property is set for recovery procedure, authentication of users will be denied.

62.4 Custom directory

As an alternative to the default directory, it is possible to integrate a specific enterprise directory. For example, an LDAP instance, a relational database or a specific directory model instantiated into EBX5.

Voir aussi *DirectoryFactory*^{API}

CHAPITRE 63

Audit trail

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Update details and disk management](#)
3. [File organization](#)

63.1 Overview

XML audit trail is a feature that allows logging updates into XML files. An alternative history feature is also available to record table updates in the relational database; see [History](#) [p 289].

Any persistent updates performed in the EBX5 repository is logged to an audit trail XML file. Procedure executions are also logged, even if they do not perform any updates, as procedures are always considered to be transactions. The following information is logged:

- Transaction type, such as data set creation, record modification, record deletion, specific procedure, etc.
- Data space or snapshot on which the transaction is executed.
- Transaction source. If the action was initiated by EBX5, this source is described by the user identity, HTTP session identifier and client IP address. If the action was initiated programmatically, only the user's identity is logged.
- Optional "trackingInfo" value regarding the session
- Transaction date and time (in milliseconds);
- Transaction UUID (conform to the Leach-Salz variant, version 1);
- Error information; if the transaction has failed.
- Details of the updates done. If there are updates and history detail is activated, see next section.

63.2 Update details and disk management

The audit trail is able to describe all updates made into the EBX5 repository, at the finest level. Thus, the XML files can be quite large and the audit trail directory must be carefully supervised. The following should be taken into account:

1. If an archive import is executed in non-interactive mode (without a change set), the audit trail does not detail the updates; it only specifies the archive that has been imported. In this case, if it is important to keep a fine trace of the import-replace, the archive itself must be preserved.
2. If an archive import is executed in interactive mode (with a change set), or if a data space is merged to its parent, the resulting log size will nearly triple the unzipped size of the archive. Furthermore, for consistency concerns, each transaction is logged into a temporary file (in the audit trail directory) before being moved into the main file. Therefore, EBX5 requires *at least six times the unzipped size of the largest archive that may be imported*.
3. In the context of a custom procedure that performs many updates not requiring auditing, it is possible for the developer to disable detailed history using the method `ProcedureContext.setHistoryActivationAPI`.

Voir aussi [EBX5 monitoring](#) [p 389]

63.3 File organization

All audit trail files are stored in the directory `${ebx.repository.directory}/History`.

"Closed" audit files

Each file is named as follows:

```
<yyyy-mm-dd>-part<nn>.xml
```

where `<yyyy-mm-dd>` is the file date and `<nn>` is the file index for the current day.

Writing to current audit files

When an audit file is being written, the XML structure implies working in an "open mode". The XML elements of the modifications are added to a text file named:

```
<yyyy-mm-dd>-part<nn>Content.txt
```

The standard XML format is still available in an XML file that references the text file. This file is named:

```
<yyyy-mm-dd>-part<nn>Ref.xml
```

These two files are then re-aggregated in a "closed" XML file when the repository has been cleanly shut down, or if EBX5 is restarted.

Example of an audit directory

```
2004-04-05-part00.xml
2004-04-05-part01.xml
2004-04-06-part00.xml
2004-04-06-part01.xml
2004-04-06-part02.xml
2004-04-06-part03.xml
2004-04-07-part00.xml
2004-04-10-part00.xml
2004-04-11-part00Content.txt
2004-04-11-part00Ref.xml
```


CHAPITRE 64

Data model administration

Ce chapitre contient les sections suivantes :

1. [Administrating publications and versions](#)
2. [Migration of previous data models in repository](#)

64.1 Administrating publications and versions

Technical data about data model publications and versions can be accessed in the *Administration* section by an administrator.

Data Modeling contains the two following tables:

- *Publications*. Stores the publications available in the repository.
- *Versions*. Stores the versions of the data models available in the repository.

These tables are in read-only but it is however possible to delete manually a publication or a version.

Important: If a publication or a version is deleted then the content of associated data sets will become unavailable. So these technical data must be deleted with caution.

It is possible to spread these technical data to other EBX5 repositories exporting an archive from a EBX5 repository and importing it to another one. It may be useful for propagating the evolutions of data models to other repositories.

64.2 Migration of previous data models in repository

In versions before 5.2.0, published data models not depending on module were generated in the file system directory `${ebx.repository.directory}/schemas/`, with the name of the data model (*product.xsd* for example if the data model is named *Product*). Since the 5.2.0, this kind of data model is now fully managed inside EBX5 through *Publications*. That is, republishing an existing data model migrates it as a *Publication* and redirects linked data sets to the new embedded data model. The previous XML Schema Document located in `${ebx.repository.directory}/schemas/` is renamed and suffixed with *toDelete* meaning that the document is no more used and can be safely deleted.

CHAPITRE 65

Data workflow administration

To define general parameters for data workflow execution, manage workflow publications, and oversee data workflows in progress, navigate to the Administration area. Click the down arrow in the navigation pane and select the entry 'Workflow'.

Ce chapitre contient les sections suivantes :

1. [Execution of workflows and history](#)
2. [Interactions](#)
3. [Workflow publications](#)
4. [Configuration](#)

65.1 Execution of workflows and history

Several tables facilitate the management of the data workflows currently in progress. These tables are accessible by navigating to **root > Execution of workflows and history** in the navigation pane.

Workflows table

The 'Workflows' table contains instances of all data workflows in the repository, including those invoked as sub-workflows. A data workflow is a particular execution instance of a workflow model publication. This table provides access to the data context variables for all data workflows. It can be used to access the status of advancement of the data workflow in terms of current variable values, and in case of data workflow suspension, to modify the variable values.

Tokens table

The 'Tokens' table allows managing the progress of data workflows. Each token marks the current step being executed in a running data workflow, as well as the current state of the data workflow.

Voir aussi [token](#) [p 27]

Work items table

The 'Work items' table contains all the work items associated with user tasks that currently exist. If necessary, you can manually allocate a work item to a user from this table in the case of a blockage in a data workflow through this table. It is preferable, however, to use the buttons in the workspace of the 'Data Workflows' area whenever possible to allocate, reallocate, and deallocate work items.

Voir aussi [work item](#) [p 27]

'Waiting workflows' table

The 'Waiting workflows' table contains all the workflows waiting for an event. If needed, a service is available to clean this table: this service deletes all lines associated with a deleted workflow.

Voir aussi [tâche d'attente](#) [p 26]

History table

The 'History' table contains all actions that have been performed during the execution of workflows. The table is especially useful to view the states of various objects related to workflows at a given moment. This includes actions on work items, variables in the data context, as well as tokens. In case of error, a technical log is available in addition to the contents of the history table.

From the 'Services' menu in the workspace, you can clear the history of completed data workflows or data workflows older than a certain date.

Note

In cases where unexpected inconsistencies have arisen between the workflow execution technical tables, data workflows may encounter errors. It may then be necessary to run the operation 'Clean up inconsistencies in workflow execution tables' from the **Services** menu in the navigation panel under Administration > Workflows.

65.2 Interactions

An *interaction* is generated automatically for every work item that is created. It is a local data context of a work item and is accessible from an EBX5 session. When a work item is executed, the user performs the assigned actions based upon its interaction, independent of the workflow engine. User tasks define mappings for their input and output parameters to link interactions with the overall data contexts of data workflows.

Interactions can be useful for monitoring the current parameters of work items. For example, an interaction can be updated manually by a trigger or a UI service.

65.3 Workflow publications

The 'Workflow publications' table is a technical table that contains all the workflow model publications in the repository. This table associates published workflow models with their snapshots. It is not recommended to directly modify this table, but rather to use the actions available in the 'Workflow Modeling' area to make changes to publications.

65.4 Configuration

Email configuration

In order for email notifications to be sent during data workflow execution, the following settings must be configured under 'Email configuration':

- 'Activate email notification' must be set to 'Yes'.
- The 'From email' field must be filled in with the email address from to mark as the sender of notification emails.

Priorities configuration

The property 'Default priority' defines how data workflows and their work items across the repository appear if they have not set a priority level. For example, if this property is set to the value 'Normal', any workflow and work item with no priority will appear to have the priority 'Normal'.

The table 'priorities' defines all priority levels available to data workflows in the repository. You may add as many integer priority levels as required, along with their labels, which will appear when users hover over the priority icon in work item tables. You may also select the icons that correspond to each priority level, either from the set provided by EBX5, or by specifying a URL to an icon image file.

Temporal tasks

Under 'Temporal tasks', you can set the polling interval for time-dependent tasks in the workflow, such as deadlines and reminders. If no interval value is set, the steps in progress are checked every hour.

CHAPITRE 66

Task scheduler

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [Configuration from EBX5](#)
3. [Cron expression](#)
4. [Task definition](#)
5. [Task configuration](#)

66.1 Overview

EBX5 offers the ability to schedule programmatic tasks.

Note

In order to avoid conflicts and dead-locks, tasks are scheduled in a single queue.

66.2 Configuration from EBX5

The declaration of schedules and tasks is done by selecting 'Task scheduler' in the Administration area.

- **Schedules:** defines scheduling using "cron expressions".
- **Tasks:** configures tasks, including parametrizing task instances and user profiles for their execution.
- **Scheduled tasks:** current schedule, including task scheduling activation/deactivation.
- **Executions report:** reports of each scheduled task run that appear immediately after the task is triggered. The reports include actions to interrupt, pause, or resume running tasks, when made available by the task definition.

66.3 Cron expression

(An extract of the [Quartz Scheduler](#) documentation)

The task scheduler uses "cron expressions", which are able to create firing schedules such as: "At 8:00am every Monday through Friday" or "At 1:30am every last Friday of the month".

Format

A cron expression is a string comprised of 6 or 7 fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field. The fields are as follows:

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	Yes	0-59	, - * /
Minutes	Yes	0-59	, - * /
Hours	Yes	0-23	, - * /
Day of month	Yes	0-31	, - * ? / L W
Month	Yes	1-12 or JAN-DEC	, - * /
Day of week	Yes	1-7 or SUN-SAT	, - * ? / L #
Year	No	0-59	empty, 1970-2099

A cron expression can be as simple as this: "*** * * * ? ***",

or more complex, like this: "**0/5 14,18,3-39,52 * ? JAN,MAR,SEP MON-FRI 2002-2010**".

Note

The legal characters and the names of months and days of the week are not case sensitive. MON is the same as mon.

Special characters

A cron expression is a string comprised of 6 or 7 fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field. The fields are as follows:

- ***** ("all values") - used to select all values within a field. For example, "*" in the minute field means "every minute".
- **?** ("no specific value") - useful when you need to specify something in one of the two fields in which the character is allowed, but not the other. For example, if I want my trigger to fire on a particular day of the month (say, the 10th), but don't care what day of the week that happens to be, I would put "10" in the day-of-month field, and "?" in the day-of-week field. See the examples below for clarification.
- **-** - used to specify ranges. For example, "10-12" in the hour field means "the hours 10, 11 and 12".
- **,** - used to specify additional values. For example, "MON,WED,FRI" in the day-of-week field means "the days Monday, Wednesday, and Friday".
- **/** - used to specify increments. For example, "0/15" in the seconds field means "the seconds 0, 15, 30, and 45". And "5/15" in the seconds field means "the seconds 5, 20, 35, and 50". You can also specify '/' after the " **character - in this case** " is equivalent to having '0' before the '/'. '1/3' in the day-of-month field means "fire every 3 days starting on the first day of the month".
- **L** ("last") - has different meaning in each of the two fields in which it is allowed. For example, the value "L" in the day-of-month field means "the last day of the month" - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means "7" or "SAT". But if used in the day-of-week field after another value, it means "the last xxx day of the month" - for example "6L" means "the last friday of the month". When using the 'L' option, it is important not to specify lists, or ranges of values, as you'll get confusing results.
- **W** ("weekday") - used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: "the nearest weekday to the 15th of the month". So if the 15th is a Saturday, the trigger will fire on Friday the 14th. If the 15th is a Sunday, the trigger will fire on Monday the 16th. If the 15th is a Tuesday, then it will fire on Tuesday the 15th. However if you specify "1W" as the value for day-of-month, and the 1st is a Saturday, the trigger will fire on Monday the 3rd, as it will not 'jump' over the boundary of a month's days. The 'W' character can only be specified when the day-of-month is a single day, not a range or list of days.

Note

The 'L' and 'W' characters can also be combined in the day-of-month field to yield 'LW', which translates to "last weekday of the month".

- **#** - used to specify "the nth" day-of-week day of the month. For example, the value of "6#3" in the day-of-week field means "the third Friday of the month" (day 6 = Friday and "#3" = the 3rd one in the month). Other examples: "2#1" = the first Monday of the month and "4#5" = the fifth Wednesday of the month. Note that if you specify "#5" and there is not 5 of the given day-of-week in the month, then no firing will occur that month.

Examples

Expression	Meaning
0 0 12 * * ?	Fire at 12pm (noon) every day.
0 15 10 ? * *	Fire at 10:15am every day.
0 15 10 * * ?	Fire at 10:15am every day.
0 15 10 * * ? *	Fire at 10:15am every day.
0 15 10 * * ? 2005	Fire at 10:15am every day during the year 2005.
0 * 14 * * ?	Fire every minute starting at 2pm and ending at 2:59pm, every day.
0 0/5 14 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, every day.
0 0/5 14,18 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day.
0 0-5 14 * * ?	Fire every minute starting at 2pm and ending at 2:05pm, every day.
0 10,44 14 ? 3 WED	Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.
0 15 10 ? * MON-FRI	Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday.
0 15 10 15 * ?	Fire at 10:15am on the 15th day of every month.
0 15 10 L * ?	Fire at 10:15am on the last day of every month.
0 15 10 ? * 6L	Fire at 10:15am on the last Friday of every month.
0 15 10 ? * 6L 2002-2005	Fire at 10:15am on every last friday of every month during the years 2002, 2003, 2004 and 2005.
0 15 10 ? * 6#3	Fire at 10:15am on the third Friday of every month.
0 0 12 1/5 * ?	Fire at 12pm (noon) every 5 days every month, starting on the first day of the month.
0 11 11 11 11 ?	Fire every November 11th at 11:11am.

Note

Pay attention to the effects of '?' and '*' in the day-of-week and day-of-month fields!

Support for specifying both a day-of-week and a day-of-month value is not complete (you must currently use the '?' character in one of these fields).

Be careful when setting fire times between the hours of the morning when "daylight savings" changes occur in your locale (for US locales, this would typically be the hour before and after 2:00 AM - because the time shift can cause a skip or a repeat depending on whether the time moves back or jumps forward).

66.4 Task definition

EBX5 scheduler comes with some predefined tasks.

Custom scheduled tasks can be added by the means of **scheduler** Package `com.orchestranetworks.schedulerAPI` Java API.

The declaration of schedules and tasks is done by selecting 'Task scheduler' in the Administration area.

66.5 Task configuration

A user must be associated with a task definition; this user will be used to generate the **session** `SessionAPI` that will run the task.

Note

The user will not be authenticated, and no password is required. As a consequence, a user with no password set in the directory can only be used to run scheduled tasks.

A custom task can be parametrized by means of JavaBean specification (getter and setter).

Supported parameter types are:

- `java.lang.boolean`
- `java.lang.int`
- `java.lang.Boolean`
- `java.lang.Integer`
- `java.math.BigDecimal`
- `java.lang.String`
- `java.lang.Date`
- `java.net.URI`
- `java.net.URL`

Parameter values are set in XML format.

Distributed Data Delivery (D3)

CHAPITRE 67

Introduction to D3

Ce chapitre contient les sections suivantes :

1. [Overview](#)
2. [D3 terminology](#)
3. [Known limitations](#)

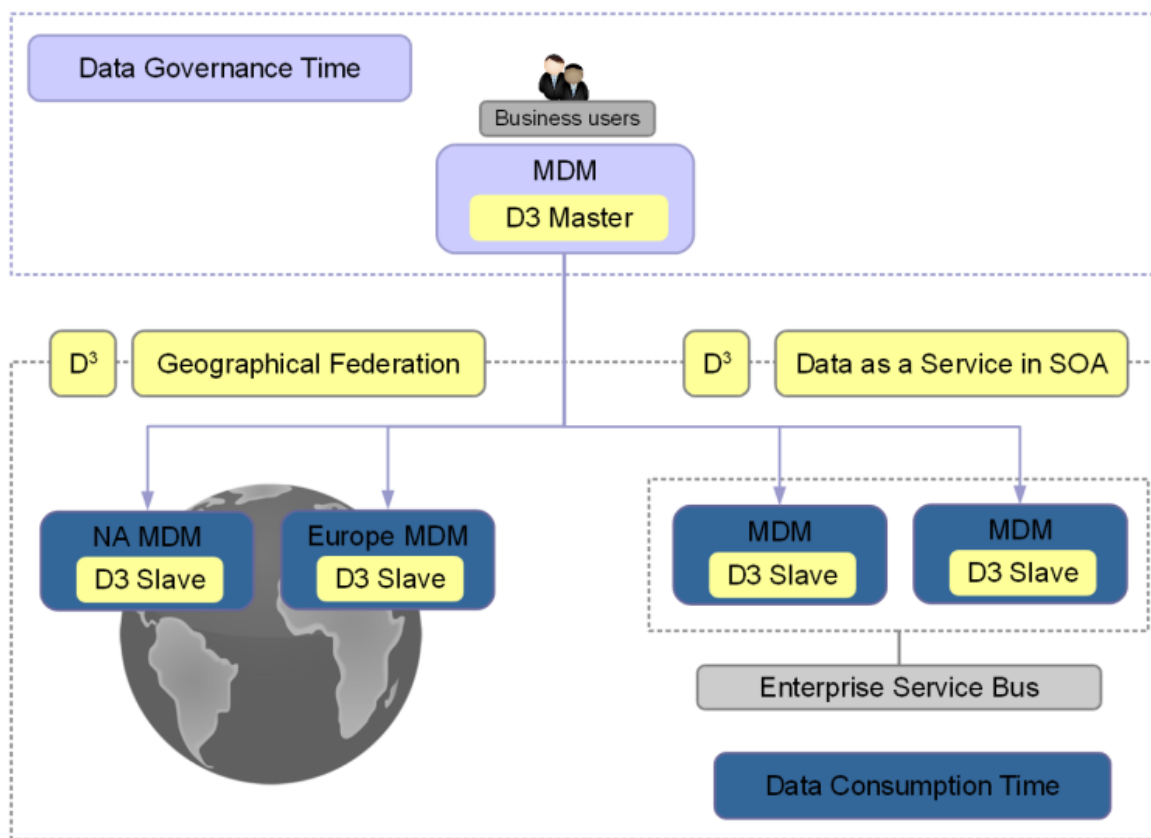
67.1 Overview

EBX5 offers the ability to send data from one EBX5 instance to other instances. Using a broadcast action, it also provides an additional layer of security and control to the other features of EBX5. It is particularly suitable for situations where data governance requires the highest levels of data consistency, approvals and the ability to rollback.

D3 architecture

A typical D3 installation consists of one master instance and multiple slave instances. In the master, a Data Steward declares which data spaces must be broadcast, as well as which user profile is allowed to broadcast them to the slaves. The Data Steward also defines delivery profiles, which are groups of one or more data spaces.

Each slave must define from which delivery profiles it receives broadcasts.



Involving third-party systems

The features of D3 also allow third-party systems to access data managed in EBX5 through data services. Essentially, when a system consumes the data of a delivery data space, the data is transparently redirected to the last broadcast snapshot. This ensures a more controlled and reliable view of the managed data.

Third-party systems can either access data directly through the master node or through a slave node. Thus, a physical architecture consisting of a master node and no slave nodes is possible.

Protocols

If JMS is activated, conversation between a master node and a slave node is based on SOAP over JMS, while archive transfer is based on JMS binary messages.

If JMS is not activated, conversation between a master node and a slave node is based on SOAP over HTTP(S), while binary archive transfer is based on TCP sockets. If HTTPS is used, make sure that the target node connector is correctly configured by enabling SSL with a trusted certificate.

Voir aussi [JMS for distributed data delivery \(D3\)](#) [p 354]

67.2 D3 terminology

broadcast	Send a publication of an official snapshot of data from a master node to slave nodes. The broadcast transparently and transactionally ensures that the data is transferred to the slave nodes.
delivery data space	A delivery data space is a data space that can be broadcast to authenticated and authorized users using a dedicated action. By default, when a data service accesses a delivery data space on any node, it is redirected to the last snapshot that was broadcast. See Data services [p 435].
delivery profile	A delivery profile is a logical name that groups one or more delivery data spaces. Slave nodes subscribe to one or more delivery profiles.
cluster delivery mode	Synchronization with subscribed slave nodes is performed in a two-phase commit transactional process. This delivery mode is designed to respond to a high volume of queries using load balancing and/or fault tolerance. It ensures the consistency of data in the cluster between slave nodes and their master delivery data spaces. Master and slave nodes use the same last broadcast snapshots.
federation delivery mode	Synchronization is performed in a single phase, and with each registered slave node independently. This delivery mode is designed to be used with geographically distributed and/or heterogeneous architectures where response time and network availability cannot be guaranteed. At any one time, slaves can be at different last broadcast snapshots. The synchronization processes are thus independent of one another and replay individual slave nodes are performed for certain broadcast failures.
master node	An instance of EBX5 that can define one or more delivery data spaces, and to which slave nodes can subscribe. A master node can also act as a regular EBX5 server.
slave node	An instance of EBX5 attached to a master node, in order to receive delivery data space broadcasts. Besides update restrictions on delivery data spaces, the slave node acts as a regular EBX5 server.

hub node

An instance of EBX5 acting as both a master node and a slave node. Master delivery data spaces and slave delivery data spaces **must** be disjoint.

67.3 Known limitations

General limitations

- Each slave node must have only one master node.
- Embedded data models cannot be used in D3 data spaces. Therefore, it is not possible to create a data set based on a publication in a D3 data space.
- The compatibility is not assured if at least one slave product version is different from the master.

Broadcast and delivery data space limitations

- Access rights on data spaces are not broadcast, whereas access rights on data sets are.
- Data space information is not broadcast.
- Data spaces defined in relational mode cannot be broadcast.
- If a data space and its parent are broadcast, their parent-child relationship will be lost in the slave nodes.
- Once a snapshot has been broadcast to a slave, subsequent broadcasts of *any* snapshot with the same name will result in restoring the originally broadcast version of that same name on the slave. That is, if the original snapshot on the master is purged and a new one is created with the same name and subsequently broadcast, then the content of the slave will be restored to that of the previously broadcast snapshot, and not the latest one of the same name.
- To guarantee data space consistency between D3 nodes, the data model (embedded or packaged in a module) on which the broadcast contents are based, must be the same between the master node and its slave nodes.
- On a slave delivery data space, if several slave nodes are registered, and if replication is enabled in data models, it will be effective for all nodes. No setting is available to activate/deactivate replication according to D3 nodes.
- Replication on slave nodes does not take part in the distributed transaction: it is automatically triggered after commit.

Administration limitations

Technical data spaces cannot be broadcast, thus the EBX5 default user directory cannot be synchronized using D3.

CHAPITRE 68

D3 broadcasts and delivery data spaces

Ce chapitre contient les sections suivantes :

1. [Broadcast](#)
2. [Slave registration](#)
3. [Accessing delivery data spaces](#)

68.1 Broadcast

Scope and contents of a broadcast

A D3 broadcast occurs at the data space or snapshot level. For data space broadcasts, D3 first creates a snapshot to capture the current state, then broadcasts this newly created snapshot.

A broadcast performs one of the following procedures depending on the situation:

- An update of the differences computed between the new broadcast snapshot and the current 'commit' one on the slave.
- A full synchronization containing all data sets, tables, records, and permissions. This is done on the first broadcast to a given slave node, if the previous slave commit is not known to the master node, or on demand using the UI service in '[D3] Master Configuration'.

Voir aussi [Services on master nodes](#) [p 442]

Performing a broadcast

The broadcast can be performed:

- By the end user, using the action **Broadcast** available in the data space or snapshot (this action is available only if the data space is registered as a delivery data space)
- Using custom Java code that uses `D3NodeAsMasterAPI`.

Conditions

To be able to broadcast, the following conditions must be fulfilled:

- The authenticated user profile has permission to broadcast.
- The data space or snapshot to be broadcasted has no validation errors.

Note: Although it is not recommended, it is possible to force a broadcast of a delivery data space that contains validation errors. In order to do this, set the maximum severity threshold allowed in a delivery data space validation report under '[D3] Configuration of master' in the Administration area.

- The D3 master configuration has no validation errors on the following scope: the technical record of the concerned delivery data space and all its dependencies (dependent delivery mappings, delivery profiles and registered slaves).
- The data space or snapshot does not contain any tables in relational mode.
- There is an associated delivery profile.
- If broadcasting a data space, the data space is not locked.
- If broadcasting a snapshot, the snapshot belongs to a data space declared as delivery data space and is not already the current broadcast snapshot (though a rollback to a previously broadcast snapshot is possible).
- The data space or snapshot contains differences compared to the last broadcast snapshot.

Persistence

When a master node shuts down, all waiting or in progress broadcast requests abort, then they will be persisted on a temporary file. On startup, all aborted broadcasts are restarted.

Voir aussi [Temporary files](#) [p 444]

Note

Broadcasts are performed asynchronously. Therefore, no information is displayed in the user interface about the success or failure of a broadcast. Nevertheless, it is possible to monitor the broadcast operations inside '[D3] Master configuration'. See [Supervision](#) [p 443].

68.2 Slave registration

Scope and contents

An initialization occurs at the slave level according to the delivery profiles registered in the EBX5 main configuration file of the slave. When the master receives that initialization request, it creates or updates the slave entry, then sends the last broadcast snapshot of all registered delivery data spaces.

Note

If the registered slave repository ID or communication layer already exists, the slave entry in 'Registered slaves' technical table is updated, otherwise a new entry is created.

Performing an initialization

The initialization can be done:

- Automatically on slave node server startup.
- Manually when calling the slave service 'Register slave'.

Conditions

To be able to register, the following conditions must be fulfilled:

- The D3 mode must be 'hub' or 'slave'.
- The master and slave authentication parameters must correspond to the master administrator and slave administrator defined in their respective directories.
- The delivery profiles defined on the slave node must exist in the master configuration.
- All data models contained in the registered data spaces must exist in the slave node. If embedded, the data model names must be the same. If packaged, they must be located at the same module name and the schema path in module must be the same in both the master and slave nodes.
- The D3 master configuration has no validation error on the following scope: the technical record of the registered slave and all its dependencies (dependent delivery profiles, delivery mappings and delivery data spaces).

Note

To set the parameters, see the slave or hub EBX5 properties in [Configuring master, hub and slave nodes](#) [p 439].

68.3 Accessing delivery data spaces

Data services

By default, when a data service accesses a delivery data space, it is redirected to the current snapshot, which is the last one broadcast. However, this default behavior can be modified either at the request level or in the global configuration.

Voir aussi [Common parameter 'disableRedirectionToLastBroadcast'](#) [p 229]

Access restrictions

On the master node, a delivery data space can neither be merged nor closed. Other operations are available depending on permissions, for example, modifying a delivery data space directly, creating a snapshot independent from a broadcast, or creating and merging a child data space.

On the slave node, aside from the broadcast process, no modifications of any kind can be made to a delivery data space, whether by the end user, data services, or a Java program. Furthermore, any data space-related operations, such as merge, close, etc., are forbidden on the slave node.

D3 broadcast Java API

The last broadcast snapshot may change between two calls if a broadcast has taken place in the meantime. If a fully stable view is required for several successive calls, these calls need to specifically refer to the same snapshot.

To get the last broadcast snapshot, see `D3Node.getBroadcastVersionAPI`.

CHAPITRE 69

D3 administration

Ce chapitre contient les sections suivantes :

1. [Quick start](#)
2. [Configuring D3 nodes](#)
3. [Supervision](#)

69.1 Quick start

This section introduces the configuration of a basic D3 architecture with two EBX5 instances. Before starting, please check that each instance can work fine with its own repository.

Note

Deploy EBX5 on two different web application containers. If both instances are running on the same host, ensure that all communication TCP ports are distinct.

Declare an existing data space on the master node

The objective is to configure and broadcast an existing data space from a *master* node.

This configuration is performed on the entire D3 infrastructure ([master](#) [p 430] and [slave](#) [p 430] nodes included)

Update `ebx.properties` *master* node configuration file with:

1. Define D3 mode as master in key `ebx.d3.mode`.

Note

The *master* node can be started after configuration.

After authenticating as a built-in administrator, navigate inside the administration tab:

1. Prerequisite: Check that node is configured as *master* node (in 'Actions' menu use 'System information' and check 'D3 mode').
2. Open '[D3] Master configuration' administration feature.
3. Add data space to be broadcasted into 'Delivery data spaces' table, and declare the allowed profile.
4. Add [delivery profile](#) [p 430] into the 'Delivery profiles' table (it must correspond to a logical name) and declare the delivery mode. Possible values are: [cluster mode](#) [p 430] or [federation mode](#) [p 430].
5. Map the delivery data space with the delivery profile into 'Delivery mapping' table.

Note

The *master* node is now ready for slave(s) registration on delivery profile.
Check that D3 broadcast menu appears in 'Actions' of the data space or one of its snapshots.

Configure slave node for registration

The objective is to configure and register the *slave* node based on a delivery profile and communications settings.

Update `ebx.properties` slave node configuration file with:

1. Define D3 mode as *slave* in key `ebx.d3.mode`.
2. Define the [delivery profile](#) [p 430] set on *master* node in key `ebx.d3.delivery.profiles` (delivery profiles must be separated by a comma and a space).
3. Define *master* node user authentication (must have the built-in administrator profile) for node communications in `ebx.d3.master.username` and `ebx.d3.master.password`.
4. Define [HTTP/TCP protocols](#) [p 441] for *master* node communication, by setting a value for the property key `ebx.d3.master.url`
(for example `http://localhost:8080/ebx-dataservices/connector`).
5. Define *slave* node user authentication (must have the built-in administrator profile) for node communications in `ebx.d3.slave.username` and `ebx.d3.slave.password`.
6. Define [HTTP/TCP protocols](#) [p 441] for *slave* node communication, by setting a value for the property key `ebx.d3.slave.url`
(for example `http://localhost:8090/ebx-dataservices/connector`).

Note

The *slave* node can be started after configuration.

After authenticating as a built-in administrator, navigate inside the administration tab:

1. Prerequisite: Check that node is configured as *slave* node (in 'Actions' menu use 'System information' and check 'D3 mode').
2. Open '[D3] Slave configuration' administration feature.
3. Check the information on screen 'Master information': No field should have value 'N/A'.

Note

Please check that the model is available before broadcasting (from data model assistant, it must be published).

Then *slave* node is now ready for broadcast.

69.2 Configuring D3 nodes

Runtime configuration of master and hub nodes through the user interface

The declaration of delivery data spaces and delivery profiles is done by selecting the '[D3] Master configuration' feature from the Administration area, where you will find the following tables:

Delivery data spaces	Declarations of the data spaces that can be broadcasted.
Delivery profiles	Profiles to which slaves nodes can subscribe. The delivery mode must be defined for each delivery profile.
Delivery mapping	The association between delivery data spaces and delivery profiles.

Note

The tables above are read-only while some broadcasts are pending or in progress.

Configuring master, hub and slave nodes

This section details how to configure a node in its EBX5 main configuration file.

Voir aussi [Overview](#) [p 361]

Master node

In order to act as a *master* node, an instance of EBX5 must declare the following property in its main configuration file.

Sample configuration for `ebx.d3.mode=master` node:

```
#####
## D3 configuration
#####
# Configuration for master, hub and slave
#####
# Optional property.
# Possibles values are single, master, hub, slave
```

```
# Default is single meaning the server will be a standalone instance.
ebx.d3.mode=master
```

Voir aussi [master node](#) [p 430]

Hub node

In order to act as a *hub* node (combination of master and slave node configurations), an instance of EBX5 must declare the following property in its main configuration file.

Sample configuration for `ebx.d3.mode=hub` node:

```
#####
## D3 configuration
#####
# Configuration for master, hub and slave
#####
# Optional property.
# Possibles values are single, master, hub, slave
# Default is single meaning the server will be a standalone instance.
ebx.d3.mode=hub

#####
# Configuration dedicated to hub or slave
#####
# Profiles to subscribe to
# Mandatory property if ebx.d3.mode=hub or ebx.d3.mode=slave
ebx.d3.delivery.profiles=

# User and password to be used to communicate with the master.
# Mandatory properties if ebx.d3.mode=hub or ebx.d3.mode=slave
ebx.d3.master.username=
ebx.d3.master.password=

# User and password to be used by the master to communicate with the hub or slave.
# Mandatory property if ebx.d3.mode=hub or ebx.d3.mode=slave
ebx.d3.slave.username=
ebx.d3.slave.password=
```

Voir aussi [hub node](#) [p 431]

Slave node

In order to act as a *slave* node, an instance of EBX5 must declare the following property in its main configuration file.

Sample configuration for `ebx.d3.mode=slave` node:

```
#####
## D3 configuration
#####
# Configuration for master, hub and slave
#####
# Optional property.
# Possibles values are single, master, hub, slave
# Default is single meaning the server will be a standalone instance.
ebx.d3.mode=slave

#####
# Configuration dedicated to hub or slave
#####
# Profiles to subscribe to
# Mandatory property if ebx.d3.mode=hub or ebx.d3.mode=slave
ebx.d3.delivery.profiles=

# User and password to be used to communicate with the master.
# Mandatory properties if ebx.d3.mode=hub or ebx.d3.mode=slave
ebx.d3.master.username=
ebx.d3.master.password=

# User and password to be used by the master to communicate with the hub or slave.
# Mandatory property if ebx.d3.mode=hub or ebx.d3.mode=slave
ebx.d3.slave.username=
ebx.d3.slave.password=
```


Voir aussi [slave node](#) [p 430]

Configuring the network protocol of a node

This section details how to configure the network protocol of a node in its EBX5 main configuration file.

Voir aussi [Overview](#) [p 361]

HTTP(S) and socket TCP protocols

Sample configuration for `ebx.d3.mode=hub` or `ebx.d3.mode=slave` node with HTTP(S) network protocol:

```
#####
# HTTP(S) and TCP socket configuration for D3 hub and slave
#####
# URL to access the data services connector of the master
# Mandatory property if ebx.d3.mode=hub or ebx.d3.mode=slave and JMS for D3 is not activated.
# This property will be ignored if JMS for D3 is activated.
# The URL must follow this pattern: [protocol]://[master_host]:[master_port]/ebx-dataservices/connector
# Where the possible values of 'protocol' are 'http' or 'https'.
ebx.d3.master.url=

# URL to access the data services connector of the slave
# Mandatory property if ebx.d3.mode=hub or ebx.d3.mode=slave and JMS for D3 is not activated.
# This property will be ignored if JMS for D3 is activated.
# The URL must follow this pattern: [protocol]://[slave_host]:[slave_port]/ebx-dataservices/connector
# Where the possible values of 'protocol' are 'http' or 'https'.
ebx.d3.slave.url=

# Minimum port to use to transfer archives on TCP mode.
# Must be a positive integer above zero and below 65535.
# If not set, a random port will be used.
#ebx.d3.slave.socket.range.min=

# Max port to use on TCP mode to transfer archives.
# Must be a positive integer above ebx.d3.slave.socket.range.min and below 65535.
# Mandatory if ebx.d3.slave.socket.range.min is set.
#ebx.d3.slave.socket.range.max=
```

JMS protocol

If JMS is activated, the following properties can be defined in order to enable JMS functionalities for a D3 node.

Sample configuration for all D3 nodes with JMS network protocol:

```
#####
## JMS configuration for D3
#####
# Taken into account only if Data Services JMS is configured properly
#####
# Configuration for master, hub and slave
#####
# Default is false, activate JMS for D3
## If activated, the deployer must ensure that the entries
## 'jms/EBX_D3ReplyQueue', 'jms/EBX_D3ArchiveQueue' and 'jms/EBX_D3CommunicationQueue'
## are bound in the operational environment of the application server.
## On slave or hub mode, the entry 'jms/EBX_D3MasterQueue' must also be bound.
ebx.jms.d3.activate=false

# Change default timeout when use reply queue, default is 10000 milliseconds
#ebx.jms.d3.reply.timeout=10000

# Archive maximum size in KB for the JMS body message. If exceeds, the message
# is transferred into several sequences messages in a same group, where each one does
# not exceed the maximum size defined.
# Must be a positive integer equals to 0 or above 100.
# Default is 0 that corresponds to unbounded.
#ebx.jms.d3.archiveMaxSizeInKB=

#####
# Configuration dedicated to hub or slave
```

```
#####
# Master repository ID, used to set a message filter for the concerned master when sending JMS message
# Mandatory property if ebx.jms.d3.activate=true and if ebx.d3.mode=hub or ebx.d3.mode=slave
#ebx.jms.d3.master.repositoryId=
```

Voir aussi [JMS for distributed data delivery \(D3\)](#) [p 354]

Services on master nodes

Services to manage a master node are available in the Administration area of the slave node under '[D3] Master configuration' and also on the tables 'Delivery data spaces' and 'Registered slaves'. The services are:

Relaunch replays	Immediately relaunch all replays for waiting federation deliveries.
Delete slave delivery data space...	Delete the delivery data space on chosen slave nodes and/or unregister it from the configuration of the D3 master node. To access the service, select a delivery data space from the 'Delivery data spaces' table on the master node, then launched the wizard.
Fully resynchronize	Broadcast the full content of the last broadcast snapshot to the registered slaves.
Subscribe a slave node	Subscribe a set of selected slave nodes.
Deactivate slaves	Remove the selected slaves from the broadcast scope and switch their states to 'Unavailable'.
Unregister slaves	Completely remove the selected slaves from the master node.

Note

The master services above are hidden while some broadcasts are pending or in progress.

Services on slave nodes

Services are available in the Administration area under *[D3] Configuration of slave* to manage its subscription to the master node and perform other actions:

Register slave	Re-subscribes the slave node to the master node if it has been unregistered.
Unregister slave	Disconnects the slave node from the master node.
Close and delete snapshots	<p>Clean up a slave delivery data space.</p> <p>To access the service, select a delivery data space from the 'Delivery data spaces' table on the slave node, then follow the wizard to close and delete snapshots based on their creation dates.</p> <p>Note: The last broadcast snapshot is automatically excluded from the selection.</p>

69.3 Supervision

The last broadcast snapshot is highlighted in the snapshot table of the data space, it is represented by an icon displayed in the first column.

Master node management console

Several tables compose the management console for the master node, located in the Administration area of the master node, under '[D3] Master configuration'. They are as follows:

Registered slaves	Slaves registered with the master node. From this table, several services are available on each record.
Broadcast history	History of broadcast operations that have taken place.
Slave registration log	History of initialization operations that have taken place.
Detailed history	History of archive deliveries that have taken place. The list of associated delivery archives can be accessed from the tables 'Broadcast history' and 'Initialization history' using selection nodes.

Master node supervision services

Available in the administration area of the master node under '[D3] Master configuration'. The services are as follows:

Check slave node information	Lists the slaves and related information, such as the slave's state, associated delivery profiles, and delivered snapshots.
Clear history content	Deletes all records in all history tables, such as 'Broadcast history', 'Slave registration log' and 'Detailed history'.

Slave node monitoring through the Java API

A slave monitoring class can be created to implement actions that are triggered when the slave's status switches to either 'Available' or 'Unavailable'. To do so, it must implement the interface `NodeMonitoring`. This class must be outside of any EBX5 module and accessible from the class-loader of 'ebx.jar' and its full class name must be specified under '[D3] Slave configuration'.

Voir aussi `NodeMonitoring`^{API}

Log supervision

The technical supervision can be done through the log category 'ebx.d3', declared in the EBX5 main configuration file. For example:

```
ebx.log4j.category.log.d3= INFO, Console, ebxFile:d3
```

Voir aussi [Configuring the EBX5 logs](#) [p 366]

Temporary files

Some temporary files, such as exchanged archives, SOAP messages, broadcast queue, (...), are created and written to the EBX5 temporary directory. This location is defined in EBX5 main configuration file:

```
#####
## Directories for temporary resources.
#####
# When set, allows specifying a directory for temporary files different from java.io.tmpdir.
# Default value is java.io.tmpdir
ebx.temp.directory = ${java.io.tmpdir}

# Allows specifying the directory containing temporary files for cache.
# If unset, the used directory is ${ebx.temp.directory}/ebx.platform.
#ebx.temp.cache.directory = ${ebx.temp.directory}/ebx.platform

# When set, allows specifying the directory containing temporary files for import.
# If unset, the used directory is ${ebx.temp.directory}/ebx.platform.
#ebx.temp.import.directory = ${ebx.temp.directory}/ebx.platform
```

Guide du développeur (en anglais)

CHAPITRE 70

Notes to developers

Ce chapitre contient les sections suivantes :

1. [Terminology in version 5](#)

70.1 Terminology in version 5

In version 5, EBX5 introduced a new vocabulary for users. These terminology changes, however, do not impact the API; Java classes, data services WSDLs and EBX5 components still use version 4 terminology.

Voir aussi [Glossaire](#) [p 19]

The following table summarizes the mappings between version 5 terminology and previous terminology:

Term in EBX5	Term prior to version 5
Data set	Adaptation instance
Child data set	Child adaptation instance
Data model	Data model
Data space	Branch
Snapshot	Version
Data space or snapshot	Home
Data Workflow	Workflow instance
Workflow model	Workflow definition
Workflow publication	Workflow
Data services	Data services
Field	Attribute
Inherited field	Inherited attribute
Record	Record/occurrence
Validation rule	Constraint
Simple/advanced control	Simple/advanced constraint

Model design

CHAPITRE 71

Introduction

A data model is a structural definition of the data to be managed in the EBX5 repository. Data models contribute to EBX5's ability to guarantee the highest level of data consistency and facilitate data management.

Specifically, the data model is a document that conforms to the XML Schema standard (W3C recommendation). Its main features are as follows:

- A rich library of well-defined [simple data types](#) [p 457], such as integer, boolean, decimal, date, time;
- The ability to define additional [simple types](#) [p 459] and [complex types](#) [p 459];
- The ability to define simple lists of items, called [aggregated lists](#) [p 463];
- [Validation constraints](#) [p 485] (facets), for example, enumerations, uniqueness constraints, minimum/maximum boundaries.

EBX5 also uses the extensibility features of XML Schema for other useful information, such as:

- [Predefined types](#) [p 459], for example, locale, resource, html;
- Definition of [tables](#) [p 467] and [foreign key constraints](#) [p 471];
- Mapping data in EBX5 to Java beans;
- [Advanced validation constraints](#) [p 485] (extended facets), such as dynamic enumerations;
- Extensive [presentation information](#) [p 503], such as labels, descriptions, and error messages.

Note

EBX5 supports a subset of the W3C recommendation, as some features are not relevant to Master Data Management.

Ce chapitre contient les sections suivantes :

1. [Model Editor](#)
2. [References](#)
3. [Relationship between data sets and data models](#)
4. [Pre-requisite for XML Schemas](#)
5. [Conventions](#)
6. [Schemas with reserved names](#)

71.1 Model Editor

The data model can be defined using an XML Schema editor or through the data model assistant. The data model assistant has the advantage of being integrated into the EBX5 user interface, abstracting the verbose underlying XML.

71.2 References

For an introduction to XML Schema, see the W3Schools [XML Schema Tutorial](#).

Voir aussi

[XML Schema Part 0: Primer](#)

[XML Schema Part 1: Structures](#)

[XML Schema Part 2: Datatypes](#)

71.3 Relationship between data sets and data models

Each root data set is associated with a single data model. At data space creation, an associated data model is selected, on which to base the data set.

Voir aussi [Création du jeu de données](#) [p 103]

71.4 Pre-requisite for XML Schemas

In order for an XML Schema to be accepted by EBX5, it must include a global element declaration that includes the attribute `osd:access="--"`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-->
<!-- {ebx.copyright.text} -->
<!-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:osd="urn:ebx-schemas:common_1.0" xmlns:fmt="urn:ebx-schemas:format_1.0">
  <xs:import namespace="urn:ebx-schemas:common_1.0"
    schemaLocation="http://schema.orchestranetworks.com/common_1.0.xsd"/>
  <xs:element name="root" osd:access="--">
    ...
  </xs:element>
</xs:schema>
```

71.5 Conventions

By convention, namespaces are always defined as follows:

Prefix	Namespace
xs:	http://www.w3.org/2001/XMLSchema
osd:	urn:ebx-schemas:common_1.0
fmt:	urn:ebx-schemas:format_1.0

71.6 Schemas with reserved names

Several data models in EBX5 have reserved names.

All references to other data models (using the attribute `schemaLocation` for an import, include or redefine) that end with one of the following strings are reserved:

- `common_1.0.xsd`
- `org_1.0.xsd`
- `coreModel_1.0.xsd`
- `session_1.0.xsd`

These XSD files correspond to the schemas provided for the module `ebx-root-1.0`, at the path `/WEB-INF/ebx/schemas`. The attribute `schemaLocation` can reference the files at this location or a copy, if the file names are the same. This is useful if you want to avoid a module dependency on `ebx-root-1.0`.

For security reasons, EBX5 uses an internal definition for these schemas to prevent any modification.

CHAPITRE 72

Packaging EBX5 modules

An EBX5 module allows packaging a data model along with its resources, such as included XML Schema Documents and Java classes.

On a Java EE application server, a module in the EBX5 repository is equivalent to a standard Java EE web application. This provides features such as class-loading isolation, WAR or EAR packaging, web resources exposure, hot-redeployment. In addition, if your user application is a web application, it is possible to merge the EBX5 module with your application, in order to simplify deployment.

Ce chapitre contient les sections suivantes :

1. [Structure](#)
2. [Declaration](#)
3. [Registration](#)

72.1 Structure

An EBX5 module contains the following files:

1. /WEB-INF/web.xml:

This is the standard Java EE deployment descriptor. It must ensure that the EBX5 module is registered when the application server is launched. See [Registration](#) [p 454].

2. /WEB-INF/ebx/module.xml:

This mandatory document defines the main properties and services of the module.

3. /www/:

This optional directory contains all external resources, which are accessible by public URL. This directory is localized and structured by resource type (HTML, images, JavaScript files, stylesheets). External resources in this directory can be referenced by data models using the type `osd:resource`.

72.2 Declaration

A module is declared using the document /WEB-INF/ebx/module.xml. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:ebx-schemas:module_2.3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ebx-schemas:module_2.3 http://schema.orchestranetworks.com/module_2.3.xsd">
  <name>moduleTest</name>
  <locales>
    <locale isDefault="true">it</locale>
```

```
<locale>en-US</locale>
</locales>
</module>
```

See the [associated schema](#) for documentation about each property. The main properties are as follows:

Element	Description	Required
name	Defines the unique identifier of the module in the server instance. The module name usually corresponds to the name of the web application (the name of its directory).	Yes.
publicPath	Defines a path other than the module's name identifying the web application in public URLs. This path is added to the URL of external resources of the module, when computing absolute URLs. If this field is not defined, the public path is the module's name, defined above.	No.
locales	Defines the locales supported by the data models in the module. This list must contain all the locales that are included in the data models within the module, and that are exposed to the end user (EBX5 will not be able to display labels and messages in a language that is not declared in this list). If the element is not present, the module supports the locales of EBX5.	No.
services	Declares UI services. See Declaration and configuration [p 192] of UI services.	No.
beans	Declares reusable Java bean components. See the workflow package <code>Package com.orchestranetworks.workflow^{API}</code> in the Java API.	No.
ajaxComponents	Declares Ajax components. See Declaring an Ajax component in a module <code>UIAjaxComponent^{API}</code> in the Java API.	No.

72.3 Registration

In order to be identifiable to EBX5, a module must be registered at runtime when the application server is launched. For a web application, every EBX5 module must:

1. Contain a Servlet whose standard method `init` invokes `ModulesRegister.registerWebApp(...)`. See the code example below.
2. Make a standard declaration of this servlet in the deployment descriptor `/WEB-INF/web.xml`.
3. Ensure that this servlet will be launched at server startup by adding the following standard element to the deployment descriptor: `<load-on-startup>1</load-on-startup>`.

Deployment descriptor example

The follow is an example of a Java EE deployment descriptor (`/WEB-INF/web.xml`):

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

<web-app>
  <servlet>
    <servlet-name>InitEbxServlet</servlet-name>
    <servlet-class>com.foo.RegisterServlet</servlet-class>
```

```
<load-on-startup>1</load-on-startup>
</servlet>
</web-app>
```

Note

- If Java classes are located in the web application (in /WEB-INF/classes or /WEB-INF/lib) and these classes are used as data model resources in the module, the specific servlet class must also be located in the web application. This is because the servlet class is internally used as a hook to the application's class-loader.
- It is recommended for the servlet to also implement the method `destroy()`, otherwise stopping the web application will not work.

Once the method `ModulesRegister.unregisterWebApp()` has been executed, the data models and associated data sets become unavailable.

EBX5 supports the hot-deployment and hot-redeployment operations implemented by Java EE application servers: deployment-start, stop-restart, stop-redeployment-restart, stop of a web application. However, these operations remain sensitive, particularly concerning class-loading and potential complex dependencies, consequently these operations must be carefully monitored.

- All module registrations and unregistrations are logged to the `log.kernel` category.
- If an exception occurs while loading a module, the cause is written to the application server log.

Registration example

Java code example of a servlet that registers/unregisters the module in EBX5:

```
package com.foo;
import javax.servlet.*;
import javax.servlet.http.*;
import com.onwbp.base.repository.*;
/**
 */
public class RegisterServlet extends HttpServlet
{
    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);
        ModulesRegister.registerWebApp(this, config);
    }
    public void destroy()
    {
        ModulesRegister.unregisterWebApp(this, this.getServletConfig());
    }
}
```

Voir aussi [Module registration](#) [p 373]

CHAPITRE 73

Data types

This chapter details the data types supported by EBX5.

Voir aussi [*Tables and relationships*](#) [p 467]

Ce chapitre contient les sections suivantes :

1. [XML Schema built-in simple types](#)
2. [XML Schema named simple types](#)
3. [XML Schema complex types](#)
4. [Extended simple types defined by EBX5](#)
5. [Complex types defined by EBX5](#)
6. [Aggregated lists](#)
7. [Including external data models](#)

73.1 XML Schema built-in simple types

The table below lists all the simple types defined in XML Schema that are supported by EBX5, along with their corresponding Java types.

XML Schema type	Java class	Notes
xs:string	<code>java.lang.String</code>	
xs:boolean	<code>java.lang.Boolean</code>	
xs:decimal	<code>java.math.BigDecimal</code>	Converted to a double when displayed in the user interface. Loss of precision and/or rounding problems can occur if the <code>BigDecimal</code> exceeds 15 digits. Moreover, the inaccurately displayed value will replace the correct one if the form with the displayed value is submitted.
xs:dateTime	<code>java.util.Date</code>	
xs:time	<code>java.util.Date</code>	The date portion of the returned Date is always set to '1970/01/01'.
xs:date	<code>java.util.Date</code>	The time portion of the returned Date is always the beginning of the day, that is, '00:00:00'.
xs:anyURI	<code>java.net.URI</code>	
xs:Name (xs:string restriction)	<code>java.lang.String</code>	
xs:int (xs:decimal restriction)	<code>java.lang.Integer</code>	
xs:integer (xs:decimal restriction)	<code>java.lang.Integer</code>	This mapping does not comply with the XML Schema recommendation. Although the XML Schema specification states that <code>xs:integer</code> has no value space limitations, this value space is, in fact, restricted by the Java specifications of the <code>java.lang.Integer</code> object.

The mapping between XML Schema types and Java types are detailed in the section [Mapping of data types](#) [p 519].

73.2 XML Schema named simple types

Named simple types can be defined when designing a data model for redefining an existing built-in simple type. A *named simple type* can be reused in the data model.

Restrictions:

- In the data model, only the element restriction is allowed in a named simple type, and even then only derivation by restriction is supported. Notably, the elements `list` and `union` are not supported.
- Facet definition is not cumulative. That is, if an element and its named type both define the same kind of facet then the facet defined in the type is overridden by the local facet definition. However, this restriction does not apply to programmatic facets defined by the element `osd:constraint`. For `osd:constraint`, if an element and its named type both define a programmatic facet with different Java classes, the definition of these facets will be cumulative. Contrary to the XML Schema Specification, EBX5 is not strict regarding to the definition of a facet of the same kind in an element and its named type. That is, the value of a same kind of facet defined in an element is not checked according to the one defined in the named type. However, in the case of static enumerations defined both in an element and its type the local enumeration will be replaced by the intersection between these enumerations.
- It is not possible to define different types of enumerations on both an element and its named type. For instance, you cannot specify a static enumeration in an element and a dynamic enumeration in its named type.
- It is not possible to simultaneously define a pattern facet in both an element and its named type.

73.3 XML Schema complex types

Complex types can be defined when designing a data model. A *named complex type* can be reused in the data model.

Restrictions:

- In the data model, only the element sequence is allowed. Notably, attribute definition is not supported.
- Type extensions are not supported in the current version of EBX5.

73.4 Extended simple types defined by EBX5

EBX5 provides pre-defined simple data types:

XML Schema type	Java class
<code>osd:text</code> (xs:string restriction)	<code>java.lang.String</code>
<code>osd:html</code> (xs:string restriction)	<code>java.lang.String</code>
<code>osd:email</code> (xs:string restriction)	<code>java.lang.String</code>
<code>osd:password</code> (xs:string restriction)	<code>java.lang.String</code>
<code>osd:resource</code> (xs:anyURI restriction)	internal class
<code>osd:locale</code> (xs:string restriction)	<code>java.util.Locale</code>

The above types are defined by the internal schema `common-1.0.xsd`. They are defined as follows:

osd:text	<p>This type represents textual information. For a basic <code>xs:string</code>, its default user interface in EBX5 consists of a dedicated editor with several lines for input and display.</p> <pre><xs:simpleType name="text"> <xs:restriction base="xs:string" /> </xs:simpleType></pre>
osd:html	<p>This represents a character string with HTML formatting. A WYSIWYG editor is provided in EBX5.</p> <pre><xs:simpleType name="html"> <xs:restriction base="xs:string" /> </xs:simpleType></pre>
osd:email	<p>This represents an email address as specified by the RFC822 standard.</p> <pre><xs:simpleType name="email"> <xs:restriction base="xs:string" /> </xs:simpleType></pre>
osd:password	<p>This represents an encrypted password. A specific editor is provided in EBX5.</p> <pre><xs:element name="password" type="osd:password" /></pre> <p>The default editor performs an encryption using the SHA-256 algorithm. This encryption function is also available from a Java client using the method <code>DirectoryDefault.encryptString^{API}</code>.</p> <p>It is also possible for the default editor to use a different encryption mechanism by specifying a class that implements the interface <code>Encryption^{API}</code>.</p> <pre><xs:element name="password" type="osd:password"> <xs:annotation> <xs:appinfo> <osd:uiBean class="com.orchestranetworks.ui.UIPassword"> <encryptionClass>package.EncryptionClassName</encryptionClass> </osd:uiBean> </xs:appinfo> </xs:annotation> </xs:element></pre>
osd:resource	<p>This represents a resource in a module for EBX5. A resource is a file of type image, HTML, CSS or JavaScript, stored in a module within EBX5. It requires the definition of the facet FacetOResource [p 490].</p> <pre><xs:simpleType name="resource"> <xs:restriction base="xs:anyURI" /> </xs:simpleType></pre>

osd:locale

This represents a geographical, political or cultural location. The locale type is translated into Java by the class `java.util.Locale`.

```
<xs:simpleType name="locale">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ar" osd:label="Arabic" />
    <xs:enumeration value="ar_AE" osd:label="Arabic (United Arab
Emirates)" />
    <xs:enumeration value="ar_BH" osd:label="Arabic (Bahrain)" />
    <xs:enumeration value="ar_DZ" osd:label="Arabic (Algeria)" />
    <xs:enumeration value="ar_EG" osd:label="Arabic (Egypt)" />
    <xs:enumeration value="ar_IQ" osd:label="Arabic (Iraq)" />
    ...
    <xs:enumeration value="vi_VN" osd:label="Vietnamese (Vietnam)" /
  >
  <xs:enumeration value="zh" osd:label="Chinese" />
  <xs:enumeration value="zh_CN" osd:label="Chinese (China)" />
  <xs:enumeration value="zh_HK" osd:label="Chinese (Hong Kong)" />
  <xs:enumeration value="zh_TW" osd:label="Chinese (Taiwan)" />
</xs:restriction>
</xs:simpleType>
```

73.5 Complex types defined by EBX5

EBX5 provides pre-defined complex data types:

XML Schema type	Description
osd:UDA	User Defined Attribute: This type allows any user, according to their access rights, to define a value associated with an attribute defined in a dictionary called a UDA Catalog.
osd:UDACatalog	Catalog of User Defined Attributes: This type consists of a table in which attributes can be specified. This catalog is used by all osd:UDA elements declared in the same data model.

osd:UDA

A User Defined Attribute (UDA) supports both the `minOccurs` and `maxOccurs` attributes, as well as the attribute `osd:UDACatalogPath`, which specifies the path of the corresponding catalog.

```
<xs:element name="firstUDA" type="osd:UDA" minOccurs="0"
maxOccurs="unbounded" osd:UDACatalogPath="//insuranceCatalog" />
<xs:element name="secondUDA" type="osd:UDA" minOccurs="1"
maxOccurs="1"
osd:UDACatalogPath="/root/userCatalog" />
<xs:element name="thirdUDA" type="osd:UDA" minOccurs="0"
maxOccurs="1"
osd:UDACatalogPath="//userCatalog" />
```

In the manager, when working with a UDA, the editor will adapt itself to the type of the selected attribute.

osd:UDACatalog

Internally, a catalog is represented as a table. The parameters `minOccurs` and `maxOccurs` must be specified.

Several catalogs can be defined in the same data model.

```
<xs:element name="insuranceCatalog" type="osd:UDACatalog"
minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation xml:lang="en-US">Insurance Catalog.</
xs:documentation>
    <xs:documentation xml:lang="fr-FR">Catalog assurance.</
xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="userCatalog" type="osd:UDACatalog" minOccurs="0"
maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation xml:lang="en-US">User catalog.</
xs:documentation>
    <xs:documentation xml:lang="fr-FR">Catalogue utilisateur.</
xs:documentation>
  </xs:annotation>
</xs:element>
```

Only the following types are available for creating new attributes:

- xs:string
- xs:boolean
- xs:decimal

- xs:dateTime
- xs:time
- xs:date
- xs:anyURI
- xs:Name
- xs:int
- osd:html
- osd:email
- osd:password
- osd:locale
- osd:text

Restrictions on User Defined Attributes and Catalogs

The following features are unsupported on UDA elements:

- Facets
- Functions using the `osd:function` property
- UI bean editors using the `osd:uiBean` property
- The `osd:checkNullInput` property
- History features
- Replication
- Inheritance features, using the `osd:inheritance` property

As UDA catalogs are internally considered to be tables, the restrictions that apply to tables also exist for UDACatalog elements.

73.6 Aggregated lists

In XML Schema, the maximum number of times an element can occur is determined by the value of the `maxOccurs` attribute in its declaration. If this value is strictly greater than 1 or is unbounded, the data can have multiple occurrences. If no `osd:table` declaration is included, this element is called an *aggregated list*. In Java, it is then represented as an instance of class `java.util.List`.

The following is an example of an aggregated list that defines the pricing of a loan product, depending on the amount borrowed.

```
<xs:element name="pricing" minOccurs="0" maxOccurs="unbounded"
  osd:access="RW">
  <xs:annotation>
    <xs:documentation>
      <osd:label>Pricing</osd:label>
      <osd:description>Pricing grid </osd:description>
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="amount" type="xs:int">
        <xs:annotation>
          <xs:documentation>
            <osd:label>Amount borrowed</osd:label>
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```

</xs:annotation>
</xs:element>
<xs:element name="monthly" type="xs:int">
  <xs:annotation>
    <xs:documentation>
      <osd:label>Monthly payment </osd:label>
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="cost" type="xs:int">
  <xs:annotation>
    <xs:documentation>
      <osd:label>Cost</osd:label>
    </xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

Aggregated lists have a dedicated editor in EBX5. This editor allows you to add occurrences or to delete occurrences.

Attention

The addition of an `osd:table` declaration to an element with `maxOccurs > 1` is a very important consideration that must be taken into account during the design process. An aggregated list is severely limited with respect to the many features that are supported by tables. Some features unsupported on aggregated lists that are supported on tables are:

- Performance and memory optimization;
- Lookups, filters and searches;
- Sorting, view and display in hierarchies;
- Identity constraints (primary keys and uniqueness constraints);
- Detailed permissions for creation, modification, delete and particular permissions at record level;
- Detailed comparison and merge.

Thus, *aggregated lists should be used only for small volumes of simple data (one or two dozen occurrences), with no advanced requirements*. For larger volumes of data or more advanced functionalities, it is strongly advised to use an `osd:table` declarations.

For more information on table declarations, see [Tables and relationships](#) [p 467].

73.7 Including external data models

Including another data model in your current model allows you to use the reusable types that are defined in that data model. You can thus use the inclusion of external data models to share data types between multiple XML Schema Documents.

To include another XML Schema Document in your model, thereby including the data types that it defines, specify the element `xs:include` as follows:

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:osd="urn:ebx-schemas:common_1.0" xmlns:fmt="urn:ebx-schemas:format_1.0">
  <xs:include schemaLocation="./schemaToInclude.xsd"/>
  ...
</xs:schema>

```

The attribute `schemaLocation` is mandatory and must specify either an absolute or relative path to the XML Schema Document to include.

The inclusion of XML Schema Documents is not namespace aware, thus all included data types must belong to the same namespace. As a consequence, including XML Schema Documents that define data types of the same name is not supported.

EBX5 includes extensions with specific URNs for including embedded data models and data models packaged in modules.

To include an embedded data model in a model, specify the URN defined by EBX5. For example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:osd="urn:ebx-schemas:common_1.0" xmlns:fmt="urn:ebx-schemas:format_1.0">
  <xs:include schemaLocation="urn:ebx:publication:myPublication"/>
  ...
</xs:schema>
```

To include a data model packaged in a module, specify the specific URN defined by EBX5. For example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:osd="urn:ebx-schemas:common_1.0" xmlns:fmt="urn:ebx-schemas:format_1.0">
  <xs:include schemaLocation="urn:ebx:module:aModuleName:/WEB-INF/ebx/schema/myDataModel.xsd"/>
  ...
</xs:schema>
```

See `SchemaLocation`^{API} for more information about specific URNs supported by EBX5.

Note

If the packaged data model uses Java resources, the class loader of the module containing the data model will be used at runtime for resolving these resources.

CHAPITRE 74

Tables and relationships

Ce chapitre contient les sections suivantes :

1. [Tables](#)
2. [Foreign keys](#)
3. [Associations](#)
4. [Selection nodes](#)

74.1 Tables

Overview

EBX5 supports the features of relational database tables, including the handling of large volumes of records, and identification by primary key.

Tables provide many benefits that are not offered by [aggregated lists](#) [p 463]. Beyond relational capabilities, some features that tables provide are:

- filters and searches;
- sorting, views and hierarchies;
- identity constraints: primary keys, [foreign keys](#) [p 471] and [uniqueness constraints](#) [p 487];
- specific permissions for creation, modification, and deletion;
- dynamic and contextual permissions at the individual record level;
- detailed comparison and merge;
- ability to have inheritance at the record level (see [data set inheritance](#) [p 312]);
- performance and memory optimization.

Voir aussi

[Foreign keys](#) [p 471]

[Associations](#) [p 474]

[Actions sur les jeux de données existants](#) [p 117]

[Vue tabulaire simple](#) [p 108]

[Vues hiérarchiques](#) [p 108]

[History](#) [p 289]

Declaration

A table element, which is an element with *maxOccurs* > 1, is declared by adding the following annotation:

```
<xs:annotation>
  <xs:appinfo>
    <osd:table>
      <primaryKeys>/pathToField1 /pathToField...n</primaryKeys>
    </osd:table>
  </xs:appinfo>
</xs:annotation>
```

Common properties

Element	Description	Required
primaryKeys	<p>Specifies the primary key fields of the table.</p> <p>Each field of the primary key must be denoted by its absolute XPath notation that starts just under the root element of the table. If there are multiple fields in the primary key, the list is delimited by whitespace.</p> <p>Note: Whitespaces in primary keys of type <code>xs:string</code> are handled differently. See Whitespace handling for primary keys of type string [p 497].</p>	Yes.
defaultLabel	<p>Defines the end-user display of records. Multiple variants can be specified:</p> <ul style="list-style-type: none"> A static non-localized expression is defined using the <code>defaultLabel</code> element, for example: <pre><defaultLabel>Product: \${./productCode}</defaultLabel></pre> Static localized expressions are specified using the <code>defaultLabel</code> element with the attribute <code>xml:lang</code>, for example: <pre><defaultLabel xml:lang="fr-FR">Produit : \${./productCode}</defaultLabel> <defaultLabel xml:lang="en-US">Product: \${./productCode}</defaultLabel></pre> A JavaBean that implements the interface <code>UILabelRenderer^{API}</code> and/or the interface <code>UILabelRendererForHierarchy^{API}</code>. The JavaBean is specified by means of the attribute <code>osd:class</code>, for example: <pre><defaultLabel osd:class="com.wombat.MyLabel"></defaultLabel></pre> <p>Note: The priority of the tags when displaying the user interface is the following:</p> <ol style="list-style-type: none"> <code>defaultLabel</code> tags with a JavaBean (but it is not allowed to define several renderers of the same type); <code>defaultLabel</code> tags with a static localized expression using the <code>xml:lang</code> attribute; <code>defaultLabel</code> tag with a static non-localized expression. <p>Attention: Les droits d'accès définis dans les jeux de données associés ne sont pas appliqués lors de l'affichage des libellés. Les champs habituellement cachés à cause de droits d'accès seront affichés dans les libellés des enregistrements.</p>	No.
index	<p>Specifies an index for speeding up requests that match this index (see performances [p 331]).</p> <p>The attribute name is mandatory. Each field of the index must be denoted by its absolute XPath notation, which starts just under the root element of the table. If there are multiple fields in the index, the list is delimited by whitespace.</p> <p>Note:</p> <ul style="list-style-type: none"> Indexing only concerns semantic and relational tables. History and replica tables are not affected. It is possible to define multiple indexes on a table. It is not possible to define two indexes with the same name. 	No.

Element	Description	Required
	<ul style="list-style-type: none"> It is not possible to declare two indexes containing the same exact fields. An indexed field must be terminal. An indexed field cannot be a list or be under a list. A field declared as an <i>inherited field</i> cannot be indexed. A field declared as a function cannot be indexed. <p>For performance purposes, the following nodes are automatically indexed:</p> <ul style="list-style-type: none"> Primary keys nodes; See primary keys [p 467]. Nodes defining a foreign key constraint. See foreign key constraint [p 471]. Nodes declared as being unique. See uniqueness constraint [p 487]. Auto-incremented nodes. See auto-incremented values [p 500]. 	

Example

Below is an example of a product catalog:

```
<xs:element name="Products" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>
      <osd:label>Product Table </osd:label>
      <osd:description>List of products in Catalog </osd:description>
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:annotation>
      <xs:appinfo>
        <osd:table>
          <primaryKeys>./productRange /productCode</primaryKeys>
          <index name="indexProductCode">/productCode</index>
        </osd:table>
      </xs:appinfo>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="productRange" type="xs:string"/><!-- key -->
      <xs:element name="productCode" type="xs:string"/><!-- key -->
      <xs:element name="productLabel" type="xs:string"/>
      <xs:element name="productDescription" type="xs:string"/>
      <xs:element name="productWeight" type="xs:int"/>
      <xs:element name="productType" type="xs:string"/>
      <xs:element name="productCreationDate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Catalogs" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>
      <osd:label>Catalog Table</osd:label>
      <osd:description>List of catalogs</osd:description>
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:annotation>
      <xs:appinfo>
        <osd:table>
          <primaryKeys>/catalogId</primaryKeys>
        </osd:table>
      </xs:appinfo>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="catalogId" type="xs:string"/><!-- key -->
      <xs:element name="catalogLabel" type="xs:string"/>
      <xs:element name="catalogDescription" type="xs:string"/>
      <xs:element name="catalogType" type="xs:string"/>
      <xs:element name="catalogPublicationDate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
</xs:complexType>
</xs:element>
```

Properties related to data set inheritance

The following properties are only valid in the context of data set inheritance:

Element	Description	Required
onDelete- deleteOccultingChildren	Indique si lors de la suppression d'un enregistrement les enregistrements occultants doivent aussi être supprimés dans les jeux de données enfants. Valid values are: never or always.	No, default is never.
mayCreateRoot	Specifies whether root record creation is allowed. The expression must follow the syntax below. See definition modes [p 312].	No, default is always.
mayCreateOverwriting	Indique si un enregistrement peut être surchargé dans des jeux de données enfants. The expression must follow the syntax below. See definition modes [p 312].	No, default is "always".
mayCreateOcculting	Indique si un enregistrement occultant peut être créé dans des jeux de données enfants. The expression must follow the syntax below. See definition modes [p 312].	No, default is "always".
mayDuplicate	Indique si un enregistrement peut être dupliqué. The expression must follow the syntax below.	No, default is "always".
mayDelete	Indique si un enregistrement peut être supprimé. The expression must follow the syntax below.	No, default is "always".

The may . . . expressions specify when the action is possible, though the ultimate availability of the action also depends user access rights. The expressions have the following syntax:

expression ::= always | never | <condition>*

condition ::= [root:yes | root:no]

"always": the operation is "always" possible (but user rights may restrict this).

"never": the operation is never possible.

"root:yes": the operation is possible if the record is in a root instance.

"root:no": the operation is not possible if the record is in a root instance.

If the record does not define any specific conditions, the default is used.

Voir aussi [Data set inheritance](#) [p 311]

74.2 Foreign keys

Declaration

A reference to a [table](#) [p 467] is defined using the extended facet `osd:tableRef`.

The node holding the `osd:tableRef` declaration must be of type `xs:string`. At instantiation, any value of the node identifies a record in the target table using its **primary key syntax** `PrimaryKeyAPI`.

This extended facet is also interpreted as an enumeration whose values refer to the records in the target table.

Element	Description	Required
tablePath	XPath expression that specifies the target table.	Yes.
container	Reference of the data set that contains the target table.	Only if element 'branch' (data space) is defined. Otherwise, default is current data set.
branch	Reference of the data space that contains the container data set.	No, default is current data space or snapshot.
display	<p>Custom display for presenting the selected foreign key in the current record and the sorted list of possible keys. Two variants can be specified, either pattern-based expressions, or a JavaBean if the needs are very specific:</p> <ul style="list-style-type: none"> Static expressions are specified using the <code>display</code> and <code>pattern</code> elements. These static expressions can be localized using the additional attribute <code>xml:lang</code> on the <code>pattern</code> element, for example: <pre><display> <pattern>Product : \${../productCode}</pattern> <pattern xml:lang="fr-FR">Produit : \${../productCode}</pattern> <pattern xml:lang="en-US">Product: \${../productCode}</pattern> </display></pre> A JavaBean that implements the interface <code>TableRefDisplay</code>^{op1}. It is specified using the attribute <code>osd:class</code>. For example: <pre><display osd:class="com.wombat.MyLabel"></display></pre> <p>It is not possible to define both variants on the same foreign key element.</p> <p>Attention: Les droits d'accès définis dans les jeux de données associés ne sont pas appliqués lors de l'affichage des libellés. Les champs habituellement cachés à cause de droits d'accès seront affichés dans les libellés des enregistrements.</p>	No, if the <code>display</code> property is not specified, the table's record rendering [p 469] is used.
filter	<p>Specifies an additional constraint that filters the records of the target table. Two types of filters are available:</p> <ul style="list-style-type: none"> An XPath filter is an XPath predicate in the target table context. It is specified using the <code>predicate</code> element. For example: <pre><filter><predicate>type = \${../refType}</predicate></filter></pre> <p>A localized validation message can be specified using the element <code>validationMessage</code>, which will be displayed to the end-user at validation time if a record is not accepted by the filter.</p> <p>A specific severity level can be defined in a nested <code>severity</code> element. The default severity is 'error'.</p> <p>Each localized message variant is defined in a nested <code>message</code> element with its locale in an <code>xml:lang</code> attribute.</p>	No.

Element	Description	Required
	<p>To specify a default message for unsupported locales, define a message element with no <code>xml:lang</code> attribute.</p> <ul style="list-style-type: none"> A programmatic filter is a JavaBean that implements the interface <code>TableRefFilter^{AP1}</code>. It is specified using the attribute <code>osd:class</code>. For example: <pre><filter osd:class="com.wombat.MyFilter"></filter></pre> <p>Additional validation messages can be specified during the setup of the programmatic filter using the dedicated methods of the interface <code>TableRefFilterContext^{AP1}</code>.</p> <p>Note:</p> <p>The attributes <code>osd:class</code> and the property predicate cannot be set simultaneously.</p>	
validation	<p>Specifies localized validation messages for the <code>osd:tableRef</code> and error management policy.</p> <p>A specific severity level can be defined in a nested <code>severity</code> element. The default severity is 'error'.</p> <p>An error management policy can be defined in a nested <code>blocksCommit</code> element. The error management policy that blocks all operations does not apply to filters. That is, a foreign key constraint is not blocking if a referenced record exists but does not satisfy a foreign key filter. In this case, updates are not rejected, and a validation error will be reported.</p> <p>Each localized message variant is defined in a nested <code>message</code> element with its locale in an <code>xml:lang</code> attribute. To specify a default message for unsupported locales, define a message element with no <code>xml:lang</code> attribute.</p>	No.

Attention

You can create a data set which has a foreign key to a container that does not exist in the repository. However, the content of this data set will not be available until the container is created. After the creation of the container, a data model refresh is required to make the data set available. When creating a data set that refers to a container that does not yet exist, the following limitations apply:

- Triggers defined at the data set level are not executed.
- Default values for fields that are not contained in tables are not initialized.
- During an archive import, it is not possible to create a data set that refers to a container that does not exist.

Example

The example below specifies a foreign key in the 'Products' table to a record of the 'Catalogs' table.

```
<xs:element name="catalog_ref" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:tableRef>
          <tablePath>/root/Catalogs</tablePath>
          <display>
            <pattern xml:lang="en-US">Catalog: ${./catalogId}</pattern>
            <pattern xml:lang="fr-FR">Catalogue : ${./catalogId}</pattern>
          </display>
          <validation>
            <severity>error</severity>
            <blocksCommit>onInsertUpdateOrDelete</blocksCommit>
            <message>A default error message</message>
          </validation>
        </osd:tableRef>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

```

<message xml:lang="en-US">A localized error message</message>
<message xml:lang="fr-FR">Un message d'erreur localisé</message>
</validation>
</osd:tableRef>
</osd:otherFacets>
</xs:appinfo>
</xs:annotation>
</xs:element>

```

Voir aussi

[Table definition](#) [p 467]

Primary key syntax *PrimaryKey*^{API}

[Extraction of foreign keys \(XPath predicate syntax\)](#) [p 271]

[Associations](#) [p 474]

[View for advanced selection](#) [p 512]

SchemaNode.getFacetOnTableReference^{API}

SchemaFacetTableRef^{API}

74.3 Associations

Overview

An association provides an abstraction over an existing relationship in the data model, and allows an easy model-driven integration of *associated objects* in the user interface and in data services.

Two main types of associations are supported:

- 'By foreign key' specifies the inverse relationship of an existing [foreign key field](#) [p 471].
- 'Over a link table' specifies a relationship based on an intermediate link table (such tables are often called "join tables"). This link table has to define two foreign keys, one referring to the 'source' table (the table holding the association element) and another one referring to the 'target' table.

For an association it is also possible to:

- Filter associated objects by specifying an additional XPath filter.
- Configure a tabular view to define the fields that must be displayed in the associated table.
- Define how associated objects are to be rendered in forms.
- Hide/show associated objects in data service 'select' operation. See [Hiding a field in Data Services](#) [p 512].
- Specify the minimum and maximum numbers of associated objects that are required.
- Add validation constraints using XPath predicates for restricting associated objects.

Voir aussi

SchemaNode.getAssociationLink^{API}

SchemaNode.isAssociationNode^{API}

AssociationLink^{API}

Declaration

Associations are defined in the data model using the XML Schema element `osd:association` under `xs:annotation/appInfo`.

Restrictions:

- An association must be a simple element of type `xs:string`.
- An association can only be defined inside a table.

Note

The "official" cardinality constraints (`minOccurs="0"` `maxOccurs="0"`) are required because, from an instance of XML Schema, the corresponding node is absent. In other words, an association has no value and is considered as a "virtual" element as far as XML and XML Schema is concerned.

The table below presents the elements that can be defined under `xs:annotation/appInfo/osd:association`.

Element	Description	Required
<code>tableRefInverse</code>	<p>Defines the properties of an association that is the inverse relationship of a <i>foreign key</i>.</p> <p>Element <code>fieldToSource</code> defines the foreign key that refers to the source table of the association. Element <code>fieldToSource</code> is mandatory and must specify a foreign key field that refers to the table containing the association.</p>	Yes if the association is the inverse relationship of a <i>foreign key</i> , otherwise no.
<code>linkTable</code>	<p>Defines the properties of an association over a link table.</p> <p>Element <code>table</code> specifies the link table used by the association. Element <code>table</code> is mandatory and must refer to an existing table.</p> <p>Important: In order to be used by an association, a link table must define a primary key that is composed of auto-incremented fields or/and the foreign key to the source or target table of the association.</p> <p>Element <code>fieldToSource</code> defines the foreign key that refers to the source table of the association. Element <code>fieldToSource</code> is mandatory and must specify a foreign key field that refers to the table containing the association.</p> <p>Element <code>fieldToTarget</code> defines the foreign key that refers to the target table of the association. Element <code>fieldToTarget</code> is mandatory and must specify a foreign key field.</p>	Yes if the association is over a link table, otherwise no.
<code>xpathFilter</code>	<p>Defines an XPath predicate to filter associated objects. It is only possible to use fields from the source and the target table when defining an XPath filter. That is, if it is an association over a link table, it is not possible to use fields of the link table in the XPath filter.</p>	No.

It is possible to refer to another data set. For that, the following properties must be defined either under element `tableRefInverse` or `linkTable` depending on the type of the association:

Element	Description	Required
<code>schemaLocation</code>	Defines the data model containing the fields used by the association. The data model is defined using a specific URN that allows referring to embedded data models and data models packaged in modules. See <code>SchemaLocation^{API}</code> for more information about specific URNs supported by EBX5.	Yes.
<code>dataSet</code>	Defines the data set used by the association. This data set must use the data model specified by the element <code>schemaLocation</code> .	Yes.
<code>dataSpace</code>	Defines the data space containing the data set used by the association.	No.

Important: When creating a data set, you can create a data set that defines an association to a container that does not yet exist in the repository. However, the content of this data set will not be available immediately upon creation. After the absent container is created, a data model refresh is required in order to make the data set available. When creating a data set that refers to a container that does not yet exist, the following limitations apply:

- Triggers defined at the data set level are not executed.
- Default values on fields outside tables are not initialized.
- During an archive import, it is not possible to create a data set that refers to a container that does not exist.

User interface integration

It is possible to define how associated objects are to be rendered in forms, using the element `osd:defaultView/displayMode` under `xs:annotation/appinfo`.

Possible values are:

- `inline`, specifies that associated records are to be rendered in the form at the same position of the association in the data model.
- `tab`, specifies that associated records are to be rendered in a specific tab.

By default, associated records are rendered `inline` if this property is not defined.

The following example specifies that associated objects are to be rendered `inline` in the form:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
      </osd:association>
      <osd:defaultView>
        <displayMode>inline</displayMode>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

The following example specifies that associated objects are to be rendered in a specific tab:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
      </osd:association>
      <osd:defaultView>
        <displayMode>tab</displayMode>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

Customize view of associated objects

A specific tabular view can be specified to define the fields that must be displayed in the target table. If a tabular view is not defined, all columns that a user is allowed to view, according to the granted access rights, are displayed. A tabular view is defined using the element `osd:defaultView/tabularView` under `xs:annotation/appinfo`.

The table below shows the elements that can be defined under `osd:defaultView/tabularView`.

Element	Description	Required
column	Define a field of the target table to display. The specified path must be absolute from the target table and must refer to an existing field. Several column elements can be defined to specify the fields that are to be displayed.	No
sort	Define a field that can be used to sort associated objects. Several sort elements can be defined to specify the fields that can be used to sort associated objects. Element <code>nodePath</code> defines the path of the field that can be used to sort associated objects. Element <code>isAscending</code> specifies whether the sort order is ascending (true) or descending (false).	No.

The following example shows how to define a tabular view from the previous association between a catalog and its products:

```
<xs:element name="Products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
      </osd:association>
      <osd:defaultView>
        <tabularView>
          <column>/productRange</column>
          <column>/productCode</column>
          <column>/productLabel</column>
          <column>/productDescription</column>
          <sort>
            <nodePath>/productLabel</nodePath>
            <isAscending>true</isAscending>
          </sort>
        </tabularView>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
```

```
</xs:element>
```

Actions in the user interface

In the user interface, it is possible to perform the following actions:

- **Create:** it allows directly creating an object in the target table of the association. When a new object is created, it is automatically associated with the current record.
- **Associate:** associates an existing object with the current record. In the case of an association over a link table, a record in the link table is automatically created to materialize the link between the current record and the existing object.
- **Delete:** deletes selected associated objects in the target table of the association.
- **Detach:** breaks the semantic link between the current record and the selected associated objects. In the case of an association over a link table, the records in the link table are automatically deleted, to break the links between the current record and associated objects.

Validation

Some controls can be defined on associations, in order to restrict associated objects. These controls are defined under the element `osd:association`.

The table below presents the controls that can be defined under `xs:annotation/appInfo/osd:association`.

Element	Description	Required
<code>minOccurs</code>	Specifies the minimum number of associated objects that are required for this association. This minimum number is defined using the element value and must be a positive integer.	No, by default the minimum is not restricted.
<code>maxOccurs</code>	Specifies the maximum number of associated objects that are allowed for this association. This maximum number is defined by the element value and must be either a positive integer or the raw string unbounded which indicates that this maximum is not restricted. The maximum number of associated objects must be greater than the minimum number of associated objects.	No, by default the maximum is not restricted.
<code>constraint</code>	<p>Defines an XPath predicate for restricting associated records. It is only possible to use fields from the source and the target table when defining an XPath predicate. That is, if it is an association over a link table, it is not possible to use fields of the link table in the XPath predicate.</p> <p>In associated data sets, a validation message of the specified severity is added and displayed to the end user at validation time when an associated record does not comply with the specified constraint.</p>	No.
<code>validation</code>	<p>A validation message can be defined under the elements <code>minOccurs</code>, <code>maxOccurs</code> and <code>constraint</code>, using the element <code>validation</code>. The severity of the validation message is specified using the element <code>severity</code>. Possible severities are: <code>error</code>, <code>warning</code> and <code>info</code>.</p> <p>If the severity is not specified, then by default, the severity <code>error</code> is used.</p> <p>A localized validation message can be specified using the element <code>message</code>, which will be displayed to the end-user at validation time if an association does not comply with this constraint. Each localized message variant is defined in a nested message element with its locale in an <code>xml:lang</code> attribute. To specify a default message for unsupported locales, define a message element with no <code>xml:lang</code> attribute.</p>	No.

Data services integration

It is possible to define whether associated objects must be hidden in the Data Service select operation. For this, the property `osd:defaultView/hiddenInDataServices` under `xs:annotation/xs:appinfo` can be set on the association. Setting the property to 'true' will hide associated objects in the Data Service select operation. If this property is not defined, then by default, associated objects will be shown in the Data Service select operation.

Voir aussi

[Hiding a field in Data Services](#) [p 512]

[Association field](#) [p 228]

Examples

For example, the product catalog data model defined [previously](#) [p 470] specifies that a product belongs to a catalog (explicitly defined by a foreign key in the 'Products' table). The reverse relationship (that a catalog has certain products) is not easily represented in XML Schema, unless the 'Catalogs' table includes the following association that is the inverse of a foreign key:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
      </osd:association>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

For an association over a link table, we can consider the previous example and bring some updates. For instance, the foreign key in the 'Products' table is deleted and the relation between a product and a catalog is redefined by a link table (named 'Catalogs_Products') that has a primary key made of two foreign keys: one that refers to the 'Products' table (named 'productRef') and another to the 'Catalogs' table (named 'catalogRef'). The following example shows how to define an association over a link table from this new relationship:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <linkTable>
          <table>/root/Catalogs_Products</table>
          <fieldToSource>./catalogRef</fieldToSource>
          <fieldToTarget>./productRef</fieldToTarget>
        </linkTable>
      </osd:association>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

The following example shows an association that refers to a foreign key in another data set. In this example the 'Products' and 'Catalogs' tables are not in the same data set:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <schemaLocation>urn:ebx:module:aModuleName:/WEB-INF/ebx/schema/products.xsd</schemaLocation>
          <dataSet>Products</dataSet>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
      </osd:association>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

The following example defines an XPath filter to associate only products of type 'Technology':

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
        <xpathFilter>./productType = 'Technology'</xpathFilter>
      </osd:association>
    </xs:appinfo>
  </xs:annotation>
```



```
</xs:element>
```

The following example specifies the minimum number of products that are required for a catalog:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
        <minOccurs>
          <value>1</value>
        <validation>
          <severity>warning</severity>
          <message xml:lang="en-US">One product should at least be associated to this catalog.</message>
          <message xml:lang="fr-FR">Un produit doit au moins être associé à ce catalogue.</message>
        </validation>
        </minOccurs>
      </osd:association>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

The following example specifies that an catalog must contain at most ten products:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:association>
        <tableRefInverse>
          <fieldToSource>/root/Products/catalog_ref</fieldToSource>
        </tableRefInverse>
        <maxOccurs>
          <value>10</value>
        <validation>
          <severity>warning</severity>
          <message xml:lang="en-US">Too much products for this catalog.</message>
          <message xml:lang="fr-FR">Ce catalogue a trop de produits.</message>
        </validation>
        </maxOccurs>
      </osd:association>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

74.4 Selection nodes

Attention

From version EBX5 5.5.0, it is recommended to use associations instead of selection nodes. Associations provide more features than selection nodes, and no further evolutions will be made on selection nodes. The only characteristic on selection nodes that is not supported by associations is the way to display the selection of records as a link.

See [Associations](#) [p 474] for more information.

An element declaration can define a dynamic contextual XPath selection. In this case, the user interface provides a link so that the user can navigate to the pre-filtered table corresponding to the selection.

A selection node is useful for creating an association between two entities in the user interface, as well as for validation purposes.

Voir aussi [Foreign keys](#) [p 471]

For example, the product catalog data model defined [previously](#) [p 470] specifies that a product belongs to a catalog (explicitly defined by a foreign key in the 'Products' table). The reverse

relationship (that a catalog has certain products) is not easily represented in XML Schema, unless the 'Catalogs' table includes the following selection node:

```
<xs:element name="products" minOccurs="0" maxOccurs="0" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:select>
        <xpath>/root/Products[catalog_ref =${../catalogId}]</xpath>
        <minOccurs>1</minOccurs>
        <minOccursValidationMessage>
          <severity>error</severity>
          <message>A default validation message.</message>
          <message xml:lang="en-US">A validation message in English.</message>
          <message xml:lang="fr-FR">Un message de validation en français.</message>
        </minOccursValidationMessage>
        <maxOccurs>10</maxOccurs>
        <maxOccursValidationMessage>
          <severity>error</severity>
          <message>A default validation message.</message>
          <message xml:lang="en-US">A validation message in English.</message>
          <message xml:lang="fr-FR">Un message de validation en français.</message>
        </maxOccursValidationMessage>
      </osd:select>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

The element `minOccurs` specifies that, to be valid, an catalog must be associated with at least one product.

The element `maxOccurs` specifies that, to be valid, an catalog must be associated with at most ten products.

Note

The "official" cardinality constraints (`minOccurs="0"` `maxOccurs="0"`) are required because, from an instance of XML Schema, the corresponding node is absent. In other words, a selection node is a "virtual" element as far as XML and XML Schema is concerned.

It is also possible to specify additional constraints on the relationship. In the following example, each relationship between an catalog and a product is valid if the date of creation of the product is prior to the date of publication of the catalog:

```
<osd:select>
  <xpath>/root/Products[catalog_ref =${../catalogId}]</xpath>
  <constraintPredicate>date-less-than(productCreationDate, ${../catalogPublicationDate})</constraintPredicate>
  <constraintPredicateValidationMessage>
    <severity>error</severity>
    <message>A default validation message.</message>
    <message xml:lang="en-US">A validation message in English.</message>
    <message xml:lang="fr-FR">Un message de validation en français.</message>
  </constraintPredicateValidationMessage>
```

</osd:select>

Element	Description	Required
xpath	<p>Specifies the selection to be performed, relative to the current node.</p> <p>Examples: /root/Products[catalog_ref =\${../catalogId}] or //Products[catalog_ref =\${../catalogId}] or ../Products[catalog_ref =\${../catalogId}] .</p> <p>The path up to the predicate, for example ../Products, specifies the target table to be filtered. This part of the path is resolved relative to the current table root.</p> <p>If the selection depends on the local state, the XPath expression predicate must include references to the node on which it depends using the notation \${<relative-path>} where relative-path is a path that identifies the element relative to the node that holds the selection link.</p> <p>See EBX5 XPath supported syntax (p 267).</p>	Yes.
container	Reference of the data set that contains the target table.	No, default is current data set.
minOccurs	Specifies an additional validation constraint: the selection must be at least the specified size.	No, default is 0.
minOccursValidationMessage	<p>Specifies additional localized messages that will be displayed if the selection does not comply with the minOccurs constraint.</p> <p>A specific severity level can be defined in a nested severity element. The default severity is 'error'.</p> <p>Each localized message variant is defined in a nested message element with its locale in an xml:lang attribute. To specify a default, non-localized message, define a message element with no xml:lang attribute.</p>	No.
maxOccurs	Specifies an additional validation constraint: the selection must be at most the specified size.	No, by default the maximum is not restricted.
maxOccursValidationMessage	<p>Specifies an additional localized message that will be displayed if the selection does not comply with the maxOccurs constraint.</p> <p>A specific severity level can be defined in a nested severity element. The default severity is 'error'.</p> <p>Each localized message variant is defined in a nested message element with its locale in an xml:lang attribute. To specify a default, non-localized message, define a message element with no xml:lang attribute.</p>	No.
constraintPredicate	Specifies an additional validation constraint: each relation of the selection must satisfy the specified	No.

Element	Description	Required
	XPath predicate. The notation \${<relative-path>} has the same meaning as for the xpath element (see above).	
constraintPredicateValidationMessage	<p>Specifies a localized message to display when the constraint predicate is not satisfied.</p> <p>A specific severity level can be defined in a nested severity element. The default severity is 'error'.</p> <p>Each localized message variant is defined in a nested message element with its locale in an xml:lang attribute. To specify a default, non-localized message, define a message element with no xml:lang attribute.</p>	No.

Important: When creating a data set, you can create a data set that defines a selection node to a container that does not yet exist in the repository. However, the content of this data set will not be available immediately upon creation. After the absent container is created, a data model refresh is required to make the data set available. When creating a data set that refers to a container that does not yet exist, the following limitations apply:

- Triggers defined at the data set level are not executed.
- Default values on fields outside tables are not initialized.
- During an archive import, it is not possible to create a data set that refers to a container that does not exist.

CHAPITRE **75**

Constraints, triggers and functions

Facets allow you to define data constraints in your data models. EBX5 supports XML Schema facets and provides extended and programmatic facets for advanced data controls.

Ce chapitre contient les sections suivantes :

1. [XML Schema supported facets](#)
2. [Extended facets](#)
3. [Programmatic facets](#)
4. [Control policy](#)
5. [Triggers and functions](#)

75.1 XML Schema supported facets

The tables below show the facets that are supported by different data types.

Key:

- **X** - Supported
- **1** - The whiteSpace facet can be defined, but is not interpreted by EBX5
- **2** - In XML Schema, boundary facets are not allowed on the type string. Nevertheless, EBX5 allows such facets as extensions.
- **3** - The `osd:resource` type only supports the facet `Facet0Resource`, which is required. See [Extended Facets](#) [p 488].

	length	minLength	max Length	pattern	enumeration	white Space
xs:string	X	X	X	X	X	1
xs:boolean				X		1
xs:decimal				X	X	1
xs:dateTime				X	X	1
xs:time				X	X	1
xs:date				X	X	1
xs:anyURI	X	X	X	X	X	1
xs:Name	X	X	X	X	X	1
xs:integer				X	X	1
osd:resource [p 459] ³						

	fraction Digits	total Digits	max Inclusive	max Exclusive	min Inclusive	min Exclusive
xs:string			2	2	2	2
xs:boolean						
xs:decimal	X	X	X	X	X	X
xs:dateTime			X	X	X	X
xs:time			X	X	X	X
xs:date			X	X	X	X

	fraction Digits	total Digits	max Inclusive	max Exclusive	min Inclusive	min Exclusive
xs:anyURI						
xs:Name			2	2	2	2
xs:integer	X	X	X	X	X	X
osd:resource [p 459] ³						

Example:

```
<xs:element name="loanRate">
  <xs:simpleType>
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="4.5" />
      <xs:maxExclusive value="17.5" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Uniqueness constraint

It is possible to define a uniqueness constraint, using the standard XML Schema element [xs:unique](#). This constraint indicates that a value or a set of values has to be unique inside a table.

Example:

In the example below, a uniqueness constraint is defined on the 'publisher' table, for the target field 'name'. This means that no two records in the 'publisher' table can have the same name.

```
<xs:element name="publisher">
  ...
  <xs:complexType>
    <xs:sequence>
      ...
      <xs:element name="name" type="xs:string" />
      ...
    </xs:sequence>
  </xs:complexType>
  <xs:unique name="uniqueName">
    <xs:annotation>
      <xs:appinfo>
        <osd:validation>
          <severity>error</severity>
          <message>Name must be unique in table.</message>
          <message xml:lang="en-US">Name must be unique in table.</message>
          <message xml:lang="fr-FR">Le nom doit être unique dans la table.</message>
        </osd:validation>
      </xs:appinfo>
    </xs:annotation>
    <xs:selector xpath="." />
    <xs:field xpath="name" />
  </xs:unique>
</xs:element>
```

A uniqueness constraint has to be defined within a table and has the following properties:

Property	Description	Mandatory
name attribute	Identifies the constraint in the data model.	Yes
xs:selector element	Indicates the table to which the uniqueness constraint applies using a restricted XPath expression ('..' is forbidden). It can also indicate an element within the table (without changing the meaning of the constraint).	Yes
xs:field element	Indicates the field in the context whose values must be unique, using a restricted XPath expression. It is possible to indicate that a set of values must be unique by defining multiple xs:field elements.	Yes

Note

Undefined values (null values) are ignored on uniqueness constraints applied to single fields. On multiple fields, undefined values are taken into account. That is, sets of values are considered as being duplicated if they have the same defined and undefined values.

Additional localized validation messages can be defined using the element `osd:validation` under the elements `annotation/appinfo`. If no custom validation messages are defined, a built-in validation message will be used.

Limitations:

1. The target of the `xs:field` element must be in a table.
2. The uniqueness constraint does not apply to fields inside an aggregated list.
3. The uniqueness constraint does not apply to computed fields.

75.2 Extended facets

EBX5 provides additional constraints that are not specified in XML Schema, but that are useful for managing master data.

In order to guarantee XML Schema conformance, these extended facets are defined under the element `annotation/appinfo/otherFacets`.

Foreign keys

EBX5 allows to create a reference to an existing table by the meaning of a specific facet. See [Foreign keys](#) [p 471] for more information.

Dynamic constraints

Dynamic constraint facets retain the semantics of XML Schema, but the value attribute is replaced with a path attribute that allows fetching the value from another element. The available dynamic constraints are:

- length
- minLength
- maxLength
- maxInclusive
- maxExclusive
- minInclusive
- minExclusive

Using these facets, the data model can be modified dynamically.

Example:

```
<xs:element name="amount">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:minInclusive path="/domain/Loan/Pricing/AmountMini/amount" />
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:element>
```

In this example, the boundary of the facet `minInclusive` is not statically defined. The value of the boundary comes from the node `/domain/Loan/Pricing/AmountMini/amount`

FacetOResource constraint

This facet must be defined for each resource type. It has the following attributes:

moduleName	Indicates, using an alias, the EBX5 module that contains the resource. If the resource is contained in the current module, the alias must be preceded by "wbp". Otherwise, the alias must be one of the values defined in the element <dependencies> in the file module.xml.
resourceType	Represents the resource type using one of the following values: "ext-images", "ext-jscripts", "ext-stylesheets", "ext-html".
relativePath	Represents the local directory where the resource is located, just under the directory resourceType. In the example below, the resource is in the directory www/common/images/. Thus, the resource is located at www/common/images/promotion/ where www/ is the directory at the same level as the WEB-INF/ directory. Furthermore, if a resource is defined in a localized directory, for example www/fr/, it will only be taken into account if another resource with the same name is defined in the directory www/common/.

This facet has the same behavior as an enumeration facet: the values are collected by recursively listing all the files in the local path in the specified resource type directory in the specified module.

Example:

```
<xs:element name="promotion" type="osd:resource">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:FacetOResource osd:moduleName="wbp"
          osd:resourceType="ext-images" osd:relativePath="promotion/" />
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

For an overview of the standard directory structure of an EBX5 module (Java EE web application), see [Structure](#) [p 453].

excludeValue constraint

This facet verifies that a value is not the same as the specified excluded value.

In this example, the empty string is excluded from the allowed values.

Example:

```
<xs:element name="roleName">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:excludeValue value="">
          <osd:validation>
            <severity>error</severity>
          </osd:validation>
        </osd:excludeValue>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

```

    <message>Please select address role(s).</message>
  </osd:validation>
</osd:excludeValue>
</osd:otherFacets>
</xs:appinfo>
</xs:annotation>
<xs:simpleType type="xs:string" />
</xs:element>

```

excludeSegment constraint

This facet verifies that a value is not included in a range of values. Boundaries are excluded.

Example:

In this example, values between 20000 and 20999 are not allowed.

```

<xs:element name="zipCode">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:excludeSegment minValue="20000" maxValue="20999">
          <osd:validation>
            <severity>error</severity>
            <message>Postal code not valid.</message>
          </osd:validation>
        </osd:excludeSegment>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
  <xs:simpleType type="xs:string" />
</xs:element>

```

Enumeration facet defined using another node

By default, an enumeration facet is described statically in XML Schema.

The content of an enumeration facet can also be provided dynamically by a list of simple elements in the data model.

Example:

In this example, the content of an enumeration facet is sourced from the node CountryList.

```

<xs:annotation>
  <xs:appinfo>
    <osd:otherFacets>
      <osd:enumeration osd:path="../../CountryList" />
    </osd:otherFacets>
  </xs:appinfo>
</xs:annotation>

```

The node CountryList:

- Must be an aggregated list, that is, `maxOccurs > 1`.
- Must be a list of elements of the same type as the node with the enumeration facet.

Must be a node outside a table if the node with the enumeration facet is not inside a table.

Must be a node outside a table or in the same table as the node with the enumeration facet if the node with this enumeration is inside a table.

Example:

```

<xs:element name="FacetEnumBasedOnList">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CountryList" maxOccurs="unbounded">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="DE" osd:label="Germany" />
            <xs:enumeration value="AT" osd:label="Austria" />
            <xs:enumeration value="BE" osd:label="Belgium" />
            <xs:enumeration value="JP" osd:label="Japan" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

    <xs:enumeration value="KR" osd:label="Korea" />
    <xs:enumeration value="CN" osd:label="China" />
  </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CountryChoice" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:enumeration osd:path="../../CountryList" />
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

75.3 Programmatic facets

A programmatic constraint can be added to any XML element declaration for a simple type.

In order to guarantee XML Schema conformance, programmatic constraints are specified under the element annotation/appinfo/otherFacets.

Programmatic constraints

A programmatic constraint is defined by a Java class that implements the interface `Constraint`^{API}.

As additional parameters can be defined, the implemented Java class must conform to the JavaBean protocol.

Example:

In the example below, the Java class must define the methods: `getParam1()`, `setParam1(String)`, `getParamX()`, `setParamX(String)`, etc.

```

<xs:element name="amount">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:constraint class="com.foo.CheckAmount">
          <param1>...</param1>
          <param...n>...</param...n>
        </osd:constraint>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:element>

```

Voir aussi *JavaBean specifications* Package `com.orchestranetworks.schema`^{API}

Programmatic enumeration constraints

An enumeration constraint adds an ordered list of values to a basic programmatic constraint. This facet allows selecting a value from a list. It is defined by a Java class that implements the interface `ConstraintEnumeration`^{API}.

Example:

```

<xs:element name="amount">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:constraintEnumeration class="com.foo.CheckAmountInEnumeration">
          <param1>...</param1>
          <param...n>...</param...n>
        </osd:constraintEnumeration>
      </osd:otherFacets>
    </xs:appinfo>
  ...
</xs:element>

```

```

</xs:annotation>
...
</xs:element>

```

Constraint on 'null' values

In some cases, a value is only mandatory if some conditions are satisfied, for example, if another field has a given value. In this case, the standard XML Schema attribute `minOccurs` is insufficient because it is static.

In order to check if a value is mandatory according to its context, the following requirements must be satisfied:

1. A programmatic constraint must be defined by a Java class (see above).
2. This class must implement the interface `ConstraintOnNull`^{API}.
3. The XML Schema cardinality attributes must specify that the element is optional (`minOccurs="0"` and `maxOccurs="1"`).

Note

By default, constraint on 'null' values is not checked upon user input. In order to enable checking at input, the '[checkNullInput](#)' property [p 496] must be set. Also, if the element is terminal, the data set must also be activated.

Example:

```

<xs:element name="amount" minOccurs="0" maxOccurs="1">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:constraint class="com.foo.CheckIfNull">
          <param1>...</param1>
          <param...n>...</param...n>
        </osd:constraint>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:element>

```

Constraints on table

A constraint on table is defined by a Java class that implements the interface `ConstraintOnTable`^{API}. It can only be defined on table nodes.

As additional parameters can be defined, the implemented Java class must conform to the JavaBean protocol.

Example:

In the example below, the Java class must define the methods: `getParam1()`, `setParam1(String)`, `getParamX()`, `setParamX(String)`, etc.

```

<xs:element name="myTable" type="MyTableType" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <osd:table>
        <primaryKeys>/key</primaryKeys>
      </osd:table>
      <osd:otherFacets>
        <osd:constraint class="com.foo.checkTable">
          <param1>...</param1>
          <param...n>...</param...n>
        </osd:constraint>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>

```

</xs:element>

Attention

For performance reasons, constraints on tables are only checked when getting the validation report of a data set or table. This means that these constraints are not checked when updates, such as record insertions, deletions or modifications, occur on tables. However, the internal incremental validation framework will optimize the validation cost of these constraints if dependencies are defined. For more information, see [Validation](#) [p 336].

Voir aussi *JavaBeans Specifications Package* `com.orchestranetworks.schemaAPI`

75.4 Control policy

Blocking and non-blocking constraints

When an update in the repository is performed, and this update adds a validation error according to a given constraint, it is possible to specify whether the new error blocks the update (and cancels the transaction) or it is considered as non-blocking (so that the update can be committed and the error can be corrected later). The element `blocksCommit` within the element `osd:validation` allows this specification, with the following supported values:

onInsertUpdateOrDelete	<p>Specifies that the constraint must always remain valid after an operation (data set update, data set deletion, record creation, update or deletion). In this case, any operation that would violate the constraint is rejected and the values remain unchanged.</p> <p>This is the default and mandatory policy for primary key constraints, data type conversion constraints (an integer or a date must be well-written) and also structural and foreign key constraints in relational data models and mapped tables.</p>
onUserSubmit-checkModifiedValues	<p>Specifies that the constraint must remain valid whenever a user modifies the associated value and submits a form. In this case, any form input that would violate the constraint is rejected and the values remain unchanged.</p> <p>This is the default policy for all blocking constraints mentioned in the previous case. For example, a foreign key constraint is by default not blocking (a record referred by other records can be deleted, etc.), except in the context of a form submit.</p>
never	<p>Specifies that the constraint must never block operations. In this case, any operation that would violate the constraint is allowed. In the context of the user interface this constraint does not block form submission if the user sets a value that violates this constraint.</p>

On foreign key constraints, the control policy that blocks all operations does not apply to filtered records. That is, a foreign key constraint is not blocking if a referenced record exists but does not satisfy a foreign key filter. In this case, updates are not rejected and a validation error occurs.

It is not possible to specify a control policy on structural constraints that are defined on relational data models or in mapped tables. That is, this property is not available for fixed length, maximum length, maximum number of digits, and decimal place constraints due to the validation policy of the underlying RDBMS blocking constraints.

This property does not apply to archive imports. That is, all blocking constraints, except structural constraints, are always disabled when importing archives.

Voir aussi

[*Facet validation message with severity*](#) [p 506]

[*Foreign keys*](#) [p 471]

[*Relational mode*](#) [p 283]

XML Schema facet

The control policy is described by the element `osd:validation` in `annotation/appinfo` under the definition of the facet.

Example:

```
<xs:element name="zipCode">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minInclusive value="1000">
        <xs:annotation>
          <xs:appinfo>
            <osd:validation>
              <blocksCommit>onInsertUpdateOrDelete</blocksCommit>
            </osd:validation>
          </xs:appinfo>
        </xs:annotation>
      </xs:minInclusive>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML Schema enumeration facet

The control policy is described by the element `osd:enumerationValidation` in `annotation/appinfo` under the definition of the field.

Example:

```
<xs:element name="Gender">
  <xs:annotation>
    <xs:appinfo>
      <osd:enumerationValidation>
        <blocksCommit>onInsertUpdateOrDelete</blocksCommit>
      </osd:enumerationValidation>
    </xs:appinfo>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="0" osd:label="male" />
      <xs:enumeration value="1" osd:label="female" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

EBX5 facet

The control policy is described by the element `osd:validation` under the definition of the facet (which is defined in `annotation/appinfo/otherFacets`).

The control policy with values `onInsertUpdateOrDelete` and `onUserSubmit-checkModifiedValues` is only available on `osd:excludeSegment`, `osd:excludeValue` and `osd:tableRef` EBX5 facets.

The control policy with value `never` can be defined on all EBX5 facets. On programmatic constraints, the control policy with value `never` can only be set directly during the setup of the corresponding constraint. See `ConstraintContext.setBlocksCommitToNeverAPI` and `ConstraintContextOnTable.setBlocksCommitToNeverAPI` in the Java API for more information.

Example:

```
<xs:element name="price" type="xs:decimal">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:minInclusive path="../priceMin">
          <osd:validation>
            <blocksCommit>onInsertUpdateOrDelete</blocksCommit>
          </osd:validation>
        </osd:minInclusive>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

Check 'null' input

According to the EBX5 default validation policy, in order to allow temporarily incomplete input, a mandatory element is not checked for completion upon user input. Rather, it is verified at data set validation only. If completion must be checked immediately upon user input, the element must additionally specify the attribute `osd:checkNullInput="true"`.

Note

A value is mandatory if the data model specifies a mandatory element, either statically, using `minOccurs="1"`, or dynamically, using a constraint on 'null'. For terminal elements, mandatory values are only checked for an activated data set. For non-terminal elements, the data set does not need to be activated.

Example:

```
<xs:element name="amount" osd:checkNullInput="true" minOccurs="1">
  ...
</xs:element>
```

Voir aussi

[Constraint on 'null'](#) [p 493]

[Whitespace management](#) [p 497]

[Empty string management](#) [p 498]

EBX5 whitespace management for data types

According to XML Schema (see <http://www.w3.org/TR/xmlschema-2/#rf-whiteSpace>), whitespace handling must follow one of the procedures *preserve*, *replace* or *collapse*:

preserve	No normalization is performed, the value is unchanged.
replace	All occurrences of #x9 (tab), #xA (line feed) and #xD (carriage return) are replaced with #x20 (space).
collapse	After the processing according to the replace procedure, contiguous sequences of #x20 are then collapsed to a single #x20, and any leading or trailing #x20s are removed.

EBX5 general whitespace handling

EBX5 complies with the XML Schema recommendation:

- For nodes of type `xs:string`, whether a primary key element or not, whitespaces are always preserved and an empty string is never converted to `null`.
- For other nodes (non-`xs:string` type), whitespaces are always collapsed and empty strings are converted to `null`.

Attention

Exceptions:

- For nodes of type `osd:html` or `osd:password`, whitespaces are always preserved and empty strings are converted to `null`.
- For nodes of type `xs:string` that define the property `osd:checkNullInput="true"`, an empty string is interpreted as `null` at user input by EBX5.

Whitespace handling for primary keys of type string

For primary key columns of type `xs:string`, a default EBX5 constraint is defined. This constraint forbids empty strings and non-collapsed whitespace values at validation.

However, if the primary key node specifies its own `xs:pattern` facet, this facet overrides the default EBX5 constraint. For example, the specific pattern `".*"` would accept any strings, although this is not recommended.

The default constraint allows handling certain ambiguities. For example, it would be difficult for a user to distinguish between the following strings: `"12 34"` and `"12 34"`. For generic values, this would not create conflicts, however, errors would occur for primary keys.

Voir aussi [Tables and relationships](#) [p 467]

Empty string management

Default conversion

For nodes of type `xs:string`, no distinction is made at user input between an empty string and a `null` value. That is, an empty string value is automatically converted to `null` at user input.

Distinction between empty strings and 'null' value

There are certain cases where the distinction is made between an empty string and the `null` value, such as when:

- A primary key defines a pattern that allows empty strings.
- An element defines a foreign key constraint and the target table has a single primary key defining a pattern that allows empty strings.
- An element defines a static enumeration that contains an empty string.
- An element defines a dynamic enumeration to another element with one of the aforementioned cases.

If the distinction is made between an empty string and a `null` value, this implies the following behaviors:

- An empty string will not be converted to `null` at user input,
- Input fields for nodes of type `xs:string` will display an additional button for setting the value of the node to `null`,
- At validation time an empty string will be considered to be a compliant value with regard to the `minOccurs="1"` property.

Attention

In relational mode, the Oracle database does not support the distinction between empty strings and `null` values, and these specific cases are not supported.

Voir aussi [Relational mode](#) [p 283]

75.5 Triggers and functions

Computed values

By default, data is read and persisted in the XML repository. Nevertheless, data may be the result of a computation and/or external database access, for example, an RDBMS or a central system.

EBX5 allows taking into account other data in the current data set context.

This is made possible by defining *functions*.

A function is specified in the data model using the `osd:function` element (see example below).

- The value of the `class` attribute must be the qualified name of a Java class that implements the Java interface `ValueFunction`^{API}.
- Additional parameters may be specified at the data model level, in which case the JavaBean convention is applied.

Example:

```
<xs:element name="computedValue">
  <xs:annotation>
    <xs:appinfo>
      <osd:function class="com.foo.ComputeValue">
        <param1>...</param1>
        <param...n>...</param...n>
      </osd:function>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:element>
```

Triggers

Data sets or table records can be associated with methods that are automatically executed when some operations are performed, such as creations, updates, or deletes.

In the data model, these triggers must be declared under the `annotation/appinfo` element using the `osd:trigger` element.

For data set triggers, a Java class that extends the abstract class `InstanceTrigger`^{API} must be declared inside the element `osd:trigger`.

In the case of data set triggers, it is advised to define `annotation/appinfo/osd:trigger` tags just under the root element of the data model.

Example:

```
<xs:element name="root" osd:access="--">
  ...
  <xs:annotation>
    <xs:appinfo>
      <osd:trigger class="com.foo.MyInstanceTrigger">
        <param1>...</param1>
        <param...n>...</param...n>
      </osd:trigger>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:element>
```

For the definition of table record triggers, a Java class that extends the abstract class `TableTrigger`^{API} must be defined inside the `osd:trigger` element. It is advised to define the `annotation/appinfo/osd:trigger` elements just under the element describing the associated table or table type.

Examples:

On a table element:

```
<xs:element name="myTable" type="MyTableType" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <osd:table>
        <primaryKeys>/key</primaryKeys>
      </osd:table>
      <osd:trigger class="com.foo.MyTableTrigger" />
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

On a table type element:

```
<xs:complexType name="MyTableType">
  ...
  <xs:annotation>
    <xs:appinfo>
      <osd:trigger class="com.foo.MyTableTrigger">
        <param1>...</param1>
        <param...n>...</param...n>
      </osd:trigger>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:complexType>
```

As additional parameters can be defined, the implemented Java class must conform to the JavaBean protocol. In the example above, the Java class must define the methods: `getParam1()`, `setParam1(String)`, `getParamX()`, `setParamX(String)`, etc.

Auto-incremented values

It is possible to define auto-incremented values. Auto-incremented values are only allowed inside tables, and they must be of type `xs:int` or `xs:integer`.

An auto-increment is specified in the data model using the element `osd:autoIncrement` under the element `annotation/appinfo`.

Example:

```
<xs:element name="autoIncrementedValue" type="xs:int">
  <xs:annotation>
    <xs:appinfo>
      <osd:autoIncrement />
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

Also, there are two optional elements, `start` and `step`:

- The `start` attribute specifies the first value for this auto-increment. If this attribute is not specified, then the value 1 is set by default.
- The `step` attribute specifies the step for the next value to be generated by the auto-increment. If this attribute is not specified, then the value 1 is set by default.

Example:

```
<xs:element name="autoIncrementedValue" type="xs:int">
  <xs:annotation>
    <xs:appinfo>
      <osd:autoIncrement>
        <start>100</start>
        <step>5</step>
      </osd:autoIncrement>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

A field specifying an `osd:autoIncrement` has the following behavior:

- The computation and allocation of the field value are performed whenever a new record is inserted and the field value is undefined.
- No allocation is performed if a programmatic insertion already specifies a non-null value. For example, if an archive import or an XML import specifies the value, that value is preserved. Consequently, the allocation is not performed for a record insertion in occulting or overwriting modes.
- A newly allocated value is, whenever possible, unique in the scope of the repository. More precisely, the uniqueness of the allocation spans over all the data sets of the data model, and it also spans over all the data spaces. The latter case allows the merging of a data space to its parent with a reasonable guarantee that there will not be a conflict if the `osd:autoIncrement` is part of the records' primary key.

This principle has a very specific limitation: when a mass update transaction that specifies values is performed at the same time as a transaction that allocates a value on the same field, it is possible that the latter transaction will allocate a value that will be set by the first transaction (there is no locking between different data spaces).

Internally, the auto-increment value is stored in the 'Auto-increments' table of the repository. In the user interface, it can be accessed by administrators in the Administration area. This field is automatically updated so that it defines the greatest value ever set on the associated `osd:autoIncrement` field, in any instance or data space in the repository. This value is computed, taking into account the max value found in the table being updated.

In certain cases, for example when multiple environments have to be managed (development, test, production), each with different auto-increment ranges, it may be required to avoid this "max value" check. This particular behavior can be achieved using the `disableMaxTableCheck` property. It is generally not recommended to enable this property unless it is absolutely necessary, as this could generate conflicts in the auto-increment values. However, this property can be set in the following ways:

- Locally, by setting a parameter element in auto-increment declaration:
`<disableMaxTableCheck>true</disableMaxTableCheck>`,
- For the whole data model, by setting `<osd:disableMaxTableCheck="true">` in the element `xs:appinfo` of the data model declaration, or
- Globally, by setting the property `ebx.autoIncrement.disableMaxTableCheck=true` in the EBX5 main configuration file.

See [EBX5 main configuration file](#) [p 361].

Note

When this option is enabled globally, it becomes possible to create records in the table of auto-increments, for example, by importing from XML or CSV. If this option is not selected, creating records in the table of auto-increments is prohibited to ensure the integrity of the repository.

CHAPITRE 76

Labels and messages

EBX5 allows you to provide labels and error messages for your data models to be displayed in the interface.

Ce chapitre contient les sections suivantes :

- 1. [Label and description](#)
- 2. [Enumeration labels](#)
- 3. [Mandatory error message \(osd:mandatoryErrorMessage\)](#)
- 4. [Conversion error message](#)
- 5. [Facet validation message with severity](#)

76.1 Label and description

A label and a description can be added to each node in an adaptation model.

In EBX5, each adaptation node is displayed with its label. If no label is defined, the name of the element is used.

Two different notations can be used:

Full	The label and description are defined by the elements <code><osd:label></code> and <code><osd:description></code> respectively.
Simple	The label is extracted from the text content, ending at the first period ('.'), with a maximum of 60 characters. The description uses the remainder of the text.

The description may also have a hyperlink, either a standard HTML href to an external document, or a link to another node of the adaptation within EBX5.

- When using the href notation or any other HTML, it must be properly escaped.
- EBX5 link notation is not escaped and must specify the path of the target, for example:
`<osd:link path="../../../misc1">Link to another node in the adaptation</osd:link>`

Example:

```
<xs:element name="misc1" type="xs:string">
  <xs:annotation>
    <xs:documentation>
```

```

Miscellaneous 1. This is the description of miscellaneous element #1.
Click <a href="http://www.orchestranetworks.com" target="_blank">here</a>
to learn more.
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="misc2" type="xs:string">
  <xs:annotation>
    <xs:documentation>
      <osd:label>
        Miscellaneous 2
      </osd:label>
      <osd:description>
        This is the miscellaneous element #2 and here is a
        <osd:link path="../misc1"> link to another node in the
        adaptation</osd:link>.
      </osd:description>
    </xs:documentation>
  </xs:annotation>
</xs:element>

```

If a node points to a named type, then the label of the node replaces the label of the named type. The same mechanism applies to the description of the node (element `osd:description`).

Note

Regarding whitespace management, the label of a node is always *collapsed* when displayed. That is, contiguous sequences of blanks are collapsed to a single blank, and leading and trailing blanks are removed. In descriptions, however, whitespaces are always *preserved*.

Dynamic labels and descriptions

As an alternative to statically defining the localized labels and descriptions for each node, it is possible to specify a Java class that programmatically determines the labels and descriptions for the nodes of the data model. To define the class, include the element `osd:documentation`, with the attribute `class` in the data model. It is possible to pass JavaBean properties using nested parameter elements.

Example:

```

<xs:schema ...>
  <xs:annotation>
    <xs:appinfo>
      <osd:documentation class="com.foo.MySchemaDocumentation">
        <param1>...</param1>
        <param2>...</param2>
      </osd:documentation>
    </xs:appinfo>
  </xs:annotation>
  ...
</xs:schema ...>

```

The labels and descriptions that are provided programmatically take precedence over the ones defined locally on individual nodes.

Voir aussi *SchemaDocumentation*^{APT}

76.2 Enumeration labels

In an enumeration, a simple, non-localized label can be added to each enumeration element, using the attribute `osd:label`.

Attention

Labels defined for an enumeration element are always collapsed when displayed.

Example:

```
<xs:element name="Service" maxOccurs="unbounded">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="1" osd:label="Blue" />
      <xs:enumeration value="2" osd:label="Red" />
      <xs:enumeration value="3" osd:label="White" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

It is also possible to fully localize the labels using the standard `xs:documentation` element. If both non-localized and localized labels are added to an enumeration element, the non-localized label will be displayed in any locale that does not have a label defined.

Example:

```
<xs:element name="access" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="readOnly">
        <xs:annotation>
          <xs:documentation xml:lang="en-US">
            read only
          </xs:documentation>
          <xs:documentation xml:lang="fr-FR">
            lecture seule
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="readWrite">
        <xs:annotation>
          <xs:documentation xml:lang="en-US">
            read/write
          </xs:documentation>
          <xs:documentation xml:lang="fr-FR">
            lecture écriture
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="hidden">
        <xs:annotation>
          <xs:documentation xml:lang="en-US">
            hidden
          </xs:documentation>
          <xs:documentation xml:lang="fr-FR">
            masqué
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

76.3 Mandatory error message (`osd:mandatoryErrorMessage`)

If the node specifies the attribute `minOccurs="1"` (default behavior), then an error message, which must be provided, is displayed if the user does not complete the field. This error message can be defined specifically for each node using the element `osd:mandatoryErrorMessage`.

Example:

```
<xs:element name="birthDate" type="xs:date">
  <xs:annotation>
    <xs:documentation>
      <osd:mandatoryErrorMessage>
        Please give your birth date.
      </osd:mandatoryErrorMessage>
    </xs:documentation>
  </xs:annotation>
</xs:element>
```

The mandatory error message can be localized:

```
<xs:documentation>
  <osd:mandatoryErrorMessage xml:lang="en-US">
    Name is mandatory
  </osd:mandatoryErrorMessage>
  <osd:mandatoryErrorMessage xml:lang="fr-FR">
    Nom est obligatoire
  </osd:mandatoryErrorMessage>
</xs:documentation>
```

Note

Regarding whitespace management, the enumeration labels are always *collapsed* when displayed.

76.4 Conversion error message

For each predefined XML Schema element, it is possible to define a specific error message if the user entry has an incorrect format.

Example:

```
<xs:element name="email" type="xs:string">
  <xs:annotation>
    <xs:documentation>
      <osd:ConversionErrorMessage xml:lang="en-US">
        Please enter a valid e-mail address.
      </osd:ConversionErrorMessage>
      <osd:ConversionErrorMessage xml:lang="fr-FR">
        Saisissez un e-mail valide.
      </osd:ConversionErrorMessage>
    </xs:documentation>
  </xs:annotation>
</xs:element>
```

76.5 Facet validation message with severity

The validation message that is displayed when the value of a field does not comply with a constraint can define a custom severity, a default non-localized message, and localized message variants. If no severity is specified, the default level is error. Blocking constraints *must* have the severity error.

XML Schema facet (osd:validation)

The validation message is described by the element `osd:validation` in `annotation/appinfo` under the definition of the facet.

Example:

```
<xs:element name="zipCode">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <!--facet is not localized, but validation message is localized-->
      <xs:minInclusive value="01000">
        <xs:annotation>
          <xs:appinfo>
            <osd:validation>
              <severity>error</severity>
              <message>Non-localized message.</message>
              <message xml:lang="en-US">English error message.</message>
              <message xml:lang="fr-FR">Message d'erreur en français.</message>
            </osd:validation>
          </xs:appinfo>
        </xs:annotation>
      </xs:minInclusive>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XML Schema enumeration facet (osd:enumerationValidation)

The validation message is described by the element `osd:enumerationValidation` in `annotation/appinfo` under the definition of the field.

Example:

```
<xs:element name="Gender">
  <xs:annotation>
    <xs:appinfo>
      <osd:enumerationValidation>
        <severity>error</severity>
        <message>Non-localized message.</message>
        <message xml:lang="en-US">English error message.</message>
        <message xml:lang="fr-FR">Message d'erreur en français.</message>
      </osd:enumerationValidation>
    </xs:appinfo>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="0" osd:label="male" />
      <xs:enumeration value="1" osd:label="female" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

EBX5 facet (osd:validation)

The validation message is described by the element `osd:validation` under the definition of the facet (which is defined in `annotation/appinfo/otherFacets`).

Example:

```
<xs:element name="price" type="xs:decimal">
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:minInclusive path="../priceMin">
          <osd:validation>
            <severity>error</severity>
            <message>Non-localized message.</message>
            <message xml:lang="en-US">English error message.</message>
            <message xml:lang="fr-FR">Message d'erreur en français.</message>
          </osd:validation>
        </osd:minInclusive>
      </osd:otherFacets>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```


CHAPITRE 77

Additional properties

Ce chapitre contient les sections suivantes :

1. [Default values](#)
2. [Access properties](#)
3. [Information](#)
4. [Default view](#)
5. [Comparison mode](#)
6. [Apply last modifications policy](#)
7. [Categories](#)

77.1 Default values

In a data model, it is possible to specify a default value for a field using the attribute `default`. This property is used to assign a default value if no value is defined for a field.

The default value is displayed in the user input field at creation time. That is, the default value will be displayed when creating a new record or adding a new occurrence to an aggregated list.

Example:

In this example, the element specifies a default string value.

```
<xs:element name="fieldwithDefaultValue" type="xs:string" default="aDefaultValue" />
```

77.2 Access properties

The attribute `osd:access` defines the access mode, that is, whether the data of a particular data model node can be read and/or written. This attribute must have one of the following values: RW, R-, CC or - -.

For each XML Schema node, three types of adaptation are possible:

1. *Adaptation terminal node*

This node is displayed with an associated value in EBX5. When accessed using the method `Adaptation.get()`, it uses the adaptation search algorithm.

2. *Adaptation non-terminal node*

This node is a complex type that is only displayed in EBX5 if it has one child node that is also an adaptation terminal node. It has no value of its own. When accessed using the method `Adaptation.get()`, it returns `null`.

3. *Non-adaptable node*

This node is not an adaptation terminal node and has no child adaptation terminal nodes. This node is never displayed in EBX5. When access using the method `Adaptation.get()`, it returns the node default value if one is defined, otherwise returns `null`.

Voir aussi *Adaptation*^{API}

Access mode	Behavior
RW	<i>Adaptation terminal node</i> : value can be read and written in EBX5.
R -	<i>Adaptation terminal node</i> : value can only be read in EBX5.
CC	<i>Cut</i> : This is not an adaptation terminal node and none of its children are adaptation terminal nodes. This "instruction" has priority over any child node regardless of the value of their <i>access</i> attribute.
--	If the node is a simple type, it is not adaptable. If the node is a complex type, it is not an adaptation terminal node and does not define any child nodes. The root node of a data model must specify this access mode.
Default	If the <i>access</i> attribute is not defined: <ul style="list-style-type: none"> • If the node is a computed value, it is considered to be R - • If the node is a simple type and its value is not computed, it is considered to be RW • If the node is an aggregated list, it is then a terminal value and is considered to be RW • Otherwise, it is not an adaptation terminal node and it does not define anything about its child nodes.

Example:

In this example, the element is adaptable because it is an adaptation terminal node.

```
<xs:element name="proxyIpAddress" type="xs:string" osd:access="RW"/>
```

77.3 Information

The element `osd:information` allows specifying additional information. This information can then be used by the integration code, for any purpose, by calling the method `SchemaNode.getInformationAPI`.

Example:

```
<xs:element name="misc" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <osd:information>
        This is the text information of miscellaneous element.
      </osd:information>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

77.4 Default view

Hiding a field in the default view

It is possible for a field to be hidden by default in EBX5 by using the element `osd:defaultView/hidden`. This property is used to hide elements from the default view of a data set without defining specific access permissions. That is, elements hidden by default will not be visible in any default forms and views, whether tabular or hierarchical, generated from the structure of the associated data model.

Attention

- If a field is configured to be hidden in the default view of a data set, then the access permissions associated with this field will not be evaluated.
- It is possible to display a field that is hidden in the default view of a data set by defining a view. Only in this case will the access permissions associated with this field will be evaluated to determine whether the field will be displayed or not.

Example:

In this example, the element is hidden in the default view of a data set.

```
<xs:element name="hiddenField" type="xs:string" minOccurs="0"/>
  <xs:annotation>
    <xs:appinfo>
      <osd:defaultView>
        <hidden>true</hidden>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

Hiding a field in search tools

To specify whether or not to hide an element in search tools, use the element `osd:defaultView/hiddenInSearch="true|false|textSearchOnly"`.

If this property is set to `true` then the field will not be selectable in the text and typed search tools of a data set.

If this property is set to `textSearchOnly` then the field will not be selectable only in the text search of a data set but it will be selectable in the typed search.

Note

If a group is configured as hidden in search tools or only in the text search, then all the fields nested under this group will not be displayed respectively in the search tools or only in the text search.

Example:

```
<xs:element name="hiddenFieldInSearch" type="xs:string" minOccurs="0"/>
  <xs:annotation>
    <xs:appinfo>
      <osd:defaultView>
        <hiddenInSearch>true</hiddenInSearch>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

In this example, the element is hidden in the text and typed search tools of a data set.

```
<xs:element name="hiddenFieldOnlyInTextSearch" type="xs:string" minOccurs="0"/>
  <xs:annotation>
    <xs:appinfo>
      <osd:defaultView>
        <hiddenInSearch>textSearchOnly</hiddenInSearch>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

In this example, the element is hidden only in the text search tool of a data set.

Hiding a field in Data Services

To specify whether or not to hide an element in data services, use the element `osd:defaultView/hiddenInDataServices`. For more information, see [Disabling fields from data model](#) [p 227].

Note

- If a group is configured as being hidden, then all the fields nested under this group will be considered as hidden by data services.

Example:

```
<xs:element name="hiddenFieldInDataService" type="xs:string" minOccurs="0"/>
  <xs:annotation>
    <xs:appinfo>
      <osd:defaultView>
        <hiddenInDataServices>true</hiddenInDataServices>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

In this example, the element is hidden in the Data Service select operation.

Defining a view for the combo box selector of a foreign key

It is possible to specify a published view that will be used to display the target table or the hierarchical view of a foreign key for a smoother selection. If a view has been defined, the selector will be displayed

in the user interface in the combo box of this foreign key. The definition of a view can be done by using the XML Schema element `osd:defaultView/widget/viewForAdvancedSelection`.

Note

- This property can only be defined on foreign key fields.
- The published view must be associated with the target table of the foreign key.
- If the published view does not exist then the advanced selection is not available in the foreign key field.

Example:

In this example, the name of a published view is defined to display the target table of a foreign key in the advanced selection.

```
<xs:element name="catalog_ref" type="xs:string" minOccurs="0"/>
  <xs:annotation>
    <xs:appinfo>
      <osd:otherFacets>
        <osd:tableRef>
          <tablePath>/root/Catalogs</tablePath>
        </osd:tableRef>
      </osd:otherFacets>
      <osd:defaultView>
        <widget>
          <viewForAdvancedSelection>catalogView</viewForAdvancedSelection>
        </widget>
      </osd:defaultView>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

See [Combo-box selector](#) [p 51] for more information.

77.5 Comparison mode

The attribute `osd:comparison` can be included on a terminal node element in order to set its comparison mode. This mode controls how differences are detected for the element during comparisons. The possible values for the attribute are:

default	Element is visible during comparisons of its data.
ignored	<p>No changes will be detected when comparing two versions of the content in records or data sets.</p> <p>During a merge, the data values of an ignored element are not merged when contents are updated, even if the values are different. For new content, the values of ignored elements are merged.</p> <p>During an archive import, values of ignored elements are not imported when contents are updated. For new content, the values of ignored elements are imported.</p>

Note

- If a group is configured as being ignored during comparisons, then all the fields nested under this group will also be ignored.
- If a terminal field does not include the attribute `osd:comparison`, then it will be included by default during comparisons.

Restrictions:

- This property cannot be defined on non-terminal fields.
- Primary key fields cannot be ignored during comparison.

Example:

In this example, the first element is explicitly ignored during comparison, the second element is explicitly included.

```
<xs:element name="fieldExplicitlyIgnoredInComparison"
  type="xs:string" minOccurs="0" osd:comparison="ignored"/>
<xs:element name="fieldExplicitlyNotIgnoredInComparison"
  type="xs:string" minOccurs="0" osd:comparison="default"/>
```

77.6 Apply last modifications policy

The attribute `osd:applyLastModification` can be defined on a terminal node element in order to specify if this element must be included or not in the apply last modifications service that can be executed in a table of a data set.

The possible values for the attribute are:

default	Last modifications can be applied to this element.
ignored	This element is ignored from the apply last modifications service. That is, the last modification that has been performed on this element cannot be applied to other records.

Note

- If a group is configured as being ignored by the apply last modifications service, then all fields nested under this group will also be ignored.
- If a terminal field does not include the attribute `osd:applyLastModification`, then it will be included by default in the apply last modifications service.

Restriction:

- This property cannot be defined on non-terminal fields.

Example:

In this example, the first element is explicitly ignored in the apply last modifications service, the second element is explicitly included.

```
<xs:element name="fieldExplicitlyIgnoredInApplyLastModification"
  type="xs:string" minOccurs="0" osd:applyLastModification="ignored"/>
<xs:element name="fieldExplicitlyNotIgnoredApplyLastModification"
  type="xs:string" minOccurs="0" osd:applyLastModification="default"/>
```

77.7 Categories

Categories can be used for "filtering", by restricting the display of any data model elements that are located in tables.

To create a category, add the attribute `osd:category` to a table node in the data model XSD.

Filters on data

In the example below, the attribute `osd:category` is added to the node in order to create a category named *mycategory*.

```
<xs:element name="rebate" osd:category="mycategory">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="label" type="xs:string"/>
      <xs:element name="beginDate" type="xs:date"/>
      <xs:element name="endDate" type="xs:date"/>
      <xs:element name="rate" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

To activate a defined category filter on a data set in the user interface, select **Actions > Categories** > *<category name>* from the navigation pane.

Predefined categories

Two categories with localized labels are predefined:

- Hidden

An instance node, including a table node itself, is hidden in the default view, but can be revealed by selecting **Actions > Categories > [hidden nodes]** from the navigation pane.

A table record node is always hidden.

- Constraint (deprecated)

Restriction

Categories do not apply to table record nodes, except the category 'Hidden'.

Java reference

CHAPITRE 78

Mapping to Java

Ce chapitre contient les sections suivantes :

1. [How to access data from Java?](#)
2. [Concurrency and isolation levels](#)
3. [Mapping of data types](#)
4. [Java bindings](#)

78.1 How to access data from Java?

Read access

Data can be read from various generic Java classes, mainly `AdaptationAPI` and `ValueContextAPI`. The getter methods for these classes return objects that are typed according to the mapping rules described in the section [Mapping of data types](#) [p 519].

Write access

Updates of data must be performed in a well-managed context:

- In the context of a procedure execution, by calling the methods `setValue...` of the interface `ValueContextForUpdateAPI`, or
- During user input validation, by calling the method `setNewValue` of the class `ValueContextForInputValidationAPI`.

Modification of mutable objects

According to the mapping that is described in the section [Mapping of data types](#) [p 519], some accessed Java objects are mutable objects. These are instances of `List`, `Date` or any `JavaBean`. Consequently, these objects can be locally modified by their own methods. However, such modifications will remain local to the returned object unless one of the above setters is invoked and the current transaction is successfully committed.

78.2 Concurrency and isolation levels

Highest isolation level

The highest isolation level in ANSI/ISO SQL is `SERIALIZABLE`. Three execution methods guarantee the `SERIALIZABLE` isolation level within the scope of a data space:

- If the client code is run inside a `ProcedureAPI` container. This is the case for every update, for exports to XML, CSV or archive, and for data services.
- If the client code accesses a data space that has been explicitly locked. See `LockSpecAPI`.
- If the client code accesses data in a snapshot.

Note

For custom read-only transactions that run on a data space, it is recommended to use `ReadOnlyProcedureAPI`.

Default isolation level

If the client code is run outside the contexts that enable `SERIALIZABLE`, its isolation level depends on the persistence mode:

- In semantic mode, the default isolation level is `READ UNCOMMITTED`.
- In relational mode, the default isolation level is the database default isolation level.

Voir aussi [Overview of modes](#) [p 283]

Java access specificities

In a Java application, a record is represented by an instance of the class `Adaptation`. This object is initially linked to the corresponding persisted record, however, unless the client code is executed in a context that enables the [SERIALIZABLE](#) [p 519] isolation level, the object can become "disconnected" from the persisted record. If this occurs and concurrent updates have been performed, they will not be reflected in the `Adaptation` object.

Therefore, it is important for the client code to either be in a `SERIALIZABLE` context, or to regularly look up or refresh the `Adaptation` object.

Voir aussi

`AdaptationHome.findAdaptationOrNullAPI`

`AdaptationTable.lookupAdaptationByPrimaryKeyAPI`

`Adaptation.getUpToDateInstanceAPI`

78.3 Mapping of data types

This section describes how XML Schema type definitions and element declarations are mapped to Java types.

Simple data types

Basic rules for simple data types

Each XML Schema simple type corresponds to a Java class, the mapping is documented in the table [XML Schema built-in simple types](#) [p 458].

Voir aussi `SchemaNode.createNewOccurrence`^{API}

Multiple cardinality on a simple element

If the attribute `maxOccurs` is greater than 1, the element is an aggregated list and the corresponding instance in Java is an instance of `java.util.List`.

Elements of the list are instances of the Java class that is determined from the mapping of the simple type (see previous section).

Complex data types

Complex type definitions without a class declaration

By default (no attribute `osd:class`), a terminal node of a complex type is instantiated using an internal class. This class provides a generic JavaBean implementation. However, if custom client Java code has to access these values, it is recommended to use a custom JavaBean. To do so, use the `osd:class` declaration described in the next section.

It is also possible to transparently instantiate, read and modify the mapped Java object, with or without the attribute `osd:class`, by invoking the methods `SchemaNode.createNewOccurrence`^{API}, `SchemaNode.executeRead`^{API} and `SchemaNode.executeWrite`^{API}.

Mapping of complex types to custom JavaBeans

It is possible to map an XML Schema complex type to a custom Java class. This is done by adding the attribute `osd:class` to the complex node definition. Unless the element has `xs:maxOccurs > 1`, you must also specify the attribute `osd:access` for the node to be considered a *terminal* node. If the element has `xs:maxOccurs > 1`, it is automatically considered to be terminal.

The custom Java class must conform to the JavaBean protocol. This means that each child of the complex type must correspond to a JavaBean property of the class. Additionally, each JavaBean property must be a read-write property, and its implementation must ensure that the value set by the setter method is returned, as-is, by the getter method. Contextual computations are not allowed in these methods.

Example

In this example, the class Java `com.carRental.Customer` must define the methods `getFirstName()` and `setFirstName(String)`.

A JavaBean can have a custom user interface within EBX5, by using a `UIBeanEditor`^{API}.

```
<xs:element name="customer" osd:access="RW">
  <xs:complexType name="subscriber" osd:class="com.carRental.Customer">
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```


Multiple cardinality on a complex element

If the attribute `maxOccurs` is greater than 1, then the corresponding instance in Java is:

- An instance of `java.util.List` for an aggregated list, where every element in the list is an instance of the Java class determined by the [mapping of simple types](#) [p 458], or
- An instance of `AdaptationTableAPI`, if the property `osd:table` is specified.

78.4 Java bindings

Java bindings allow generating Java types that reflect the structure of the data model. The Java code generation can be done in the user interface. See [Generating Java bindings](#) [p 523].

Benefits

Ensuring the link between XML Schema structure and Java code provides a number of benefits:

- **Development assistance:** Auto-completion when typing an access path to parameters, if supported by your IDE.
- **Access code verification:** All accesses to parameters are verified at code compilation.
- **Impact verification:** Each modification of the data model impacts the code compilation state.
- **Cross-referencing:** By using the reference tools of your IDE, it is easy to verify where a parameter is used.

Consequently, it is strongly recommended to use Java bindings.

XML declaration

The specification of the Java types to be generated from the data model is included in the main schema.

Each binding element defines a generation target. It must be located at, in XPath notation, `xs:schema/xs:annotation/xs:appinfo/ebxbnd:binding`, where the prefix `ebxbnd` is a reference to the namespace identified by the URI `urn:ebx-schemas:binding_1.0`. Several binding elements can be defined if you have different generation targets.

The attribute `targetDirectory` of the element `ebxbnd:binding` defines the root directory used for Java type generation. Generally, it is the directory containing the project source code, `src`. A relative path is interpreted based on the current runtime directory of the VM, as opposed to the XML schema.

See [bindings XML Schema](#).

Bindings XML example

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ebxbnd="urn:ebx-schemas:binding_1.0">
  <xs:annotation>
    <xs:appinfo>
      <!-- The bindings define how this schema will be represented in Java.
      Several <binding> elements may be defined, one for each target. -->
      <ebxbnd:binding
        targetDirectory="._/ebx-demos/src-creditOnlineStruts-1.0/">
        <jpathConstants typeName="com.creditonline.RulesPaths">
          <nodes root="/rules" prefix="" />
        </jpathConstants>
        <jpathConstants typeName="com.creditonline.StylesheetConstants">
          <nodes root="/stylesheet" prefix="" />
        </jpathConstants>
      </ebxbnd:binding>
    </xs:appinfo>
  </xs:annotation>
  ...
```

```
</xs:schema>
```

Java constants can be defined for XML schema paths. To do so, generate one or more interfaces from a schema node, including the root node /. The example generates two Java path constant interfaces, one from the node /rules and the other from the node /stylesheet in the schema. Interface names are described by the element `javaPathConstants` with the attribute `typeName`. The associated node is described by the element nodes with the attribute `root`

CHAPITRE 79

Tools for Java developers

EBX5 provides Java developers with tools to facilitate use of the EBX5 API, as well as integration with development environments.

Ce chapitre contient les sections suivantes :

1. [Activating the development tools](#)
2. [Data model refresh tool](#)
3. [Generating Java bindings](#)
4. [Path to a node](#)

79.1 Activating the development tools

To activate the development tools, run EBX5 in *development mode*. This is specified in the EBX5 main configuration file [EBX5 run mode](#) [p 373] using the property `backend.mode=development`.

79.2 Data model refresh tool

If you edit your data model directly as an XML Schema document without using the data-modeling tool provided by EBX5, you can refresh it without restarting the application server.

In the Administration area, select **Actions > Refresh updated data models** (or **Actions > Refresh all data models**).

Attention

Since the operation is critical regarding data consistency, refreshing the data models acquires a global exclusive lock on the repository. This means that most of the other operations (data access and update, validation, etc.) will wait until completion of the data model refresh.

79.3 Generating Java bindings

The Java types specified by Java bindings can be generated from a data set or data model, by selecting **Actions > Generate Java** in the navigation pane.

Voir aussi [Java bindings](#) [p 521]

79.4 Path to a node

The field 'Data path' is displayed in the documentation pane of a node. This field indicates the path to the node, which can be useful when writing XPath formulas.

Note

This field is always available to administrators.