



Braintribe IT Technologies GmbH.

1. tribefire Documentation	4
1.1 tribefire cortex	7
1.1.1 Key Concepts	9
1.2 tribefire Control Center	12
1.2.1 Smart Information Models	16
1.2.2 Deployment Models	18
1.2.3 Installation	20
1.2.3.1 Installing tribefire	21
1.2.3.2 Starting tribefire	27
1.2.4 User Guides	32
1.2.4.1 Adding simple types to Object Properties	34
1.2.4.2 Assigning Roles and Groups to Users	37
1.2.4.3 Assigning Roles and Users to Groups	43
1.2.4.4 Attach Metadata from Extract	48
1.2.4.5 Auto Enrich	55
1.2.4.6 Configuring and setting up a connection to MsSQL	57
1.2.4.6.1 Downloading and Installing MsSQL	58
1.2.4.6.2 Configure MsSQL	64
1.2.4.6.3 Creating a MsSQL Connection	79
1.2.4.6.4 Creating a MsSQL access	85
1.2.4.6.5 Automatically create a new Model from your MsSQL database	88
1.2.4.6.6 Query your new MsSQL Model in tribefire GME	91
1.2.4.7 Configuring and setting up a connection to MySQL	97
1.2.4.7.1 Downloading and installing MySQL	98
1.2.4.7.2 Configure MySQL	105
1.2.4.7.3 Create a MySQL connection	114
1.2.4.7.4 Create a MySQL access	119
1.2.4.7.5 Automatically create a new Model from your MySQL database	122
1.2.4.7.6 Query your new MySQL Model in tribefire GME	126
1.2.4.8 Create a new Access	133
1.2.4.9 Create a Simple Query	139
1.2.4.10 Create a Template Query Action	150
1.2.4.11 Create Custom Workbench Access	157
1.2.4.12 Create Model From Zargo File	159
1.2.4.13 Create New Properties	165
1.2.4.14 Create Simple Instantiation Action	170
1.2.4.15 Creating User Roles and Groups	173
1.2.4.16 Deploy an Access	180
1.2.4.17 Extract Meta Data	183
1.2.4.18 Importing Data from Microsoft Excel	186
1.2.5 Control Center	207
1.2.5.1 Applications	213
1.2.5.2 Modeling	214
1.2.5.2.1 Custom Models / Distributed Models	215
1.2.5.2.2 Entity Types	220
1.2.5.3 Virtualization	223
1.2.5.3.1 Connections	224
1.2.5.3.2 Custom Accesses / System Accesses	237
1.2.5.3.3 Simulation Mode	242
1.2.5.3.4 Access Types	247
1.2.5.4 tribefire GME - System	255
1.2.5.4.1 Connection Tests	257
1.2.5.4.2 Deployments	262
1.2.5.4.3 Model Imports	266
1.2.5.4.4 Schema Updates	271
1.2.5.4.5 Files	276
1.2.5.4.6 Property Groups	285
1.2.5.4.7 Images	288
1.2.6 tribefire GME Control Center - Cortex Workbench	297
1.2.6.1 Hyperlink Action	301
1.2.6.2 Queries	302
1.2.6.2.1 Ordering	305
1.2.6.2.2 Restrictions	313
1.2.6.3 Import Automatic Folders	358
1.2.7 Standard Components	364
1.2.7.1 List View/Thumbnail View	365
1.2.7.2 Localization	368
1.2.7.3 Metadata	373
1.2.7.3.1 General Metadata Properties	374
1.2.7.3.2 Model Metadata	404
1.2.7.3.3 Entity Type Metadata	407
1.2.7.3.4 Property Metadata	438

1.2.7.3.5 Enum Type Metadata	481
1.2.7.4 Search Panel	482
1.2.7.5 tribefire Types	487
1.2.7.5.1 BindingArtifact	488
1.2.7.5.2 GmSimpleType	489
1.2.7.5.3 EntityType	490
1.2.7.5.4 EnumType	492
1.2.7.5.5 Model	494
1.2.7.5.6 Property	496
1.3 tribefire Cartridges	498
1.4 tribefire Core Models	499
1.4.1 Manipulation Model	500
1.4.2 Process Model	512
1.4.3 The MetaModel aka a model that models models	518
1.4.4 The Query Model	556
1.5 REST, Restlet and tribefire	573
1.5.1 Tribefire Restlet Calls	576
1.5.2 GmWebRpcServer - Example	586
1.5.3 tribefire Training Model	588
1.5.4 REST API Documentation	591
1.6 Standard Metadata	595
1.7 Modelling with UML	596
1.8 Dependency Management - Controlling the Uncontrollable	615
1.8.1 TFG-ACDC	618
1.8.2 TFG-BTAntTasks	647
1.8.3 TFG-Greyface	650
1.9 tribefire Development	666
1.9.1 Building CSP ServerDeps using the new platform-exclusion-based way	667
1.9.2 Transition from the ECM Service to the Access Service	668
1.9.3 Provider API	670
1.9.4 Codecs	677
1.10 tribefire System Requirements	680
1.11 tribefire Glossary	682

tribefire Documentation



tribefire Documentation

Welcome to the tribefire Technical Documentation Space. Its concept is to provide accurate introductions, detailed descriptions, technical information and real world examples with screen shots for all aspects of tribefire.

Regardless of whether you are a Developer, an Administrator or an End-User of tribefire, we are confident that you will find the right answers to whatever questions or queries you may have and that you will find the best solutions to whatever situation you may face in setting up and or utilizing tribefire.

We would very much appreciate your feedback and suggestions concerning the tribefire Documentation Space! Feel free to contact documentation@braintribe.com

Navigation

Use the following page tree to dive into the documentation's details:

Page Tree

- tribefire cortex
 - Key Concepts

- tribefire Control Center
 - Smart Information Models
 - Deployment Models
 - Installation
 - Installing tribefire
 - Starting tribefire
- User Guides
 - Adding simple types to Object Properties
 - Assigning Roles and Groups to Users
 - Assigning Roles and Users to Groups
 - Attach Metadata from Extract
 - Auto Enrich
 - Configuring and setting up a connection to MsSQL
 - Configuring and setting up a connection to MySQL
 - Create a new Access
 - Create a Simple Query
 - Create a Template Query Action
 - Create Custom Workbench Access
 - Create Model From Zargo File
 - Create New Properties
 - Create Simple Instantiation Action
 - Creating User Roles and Groups
 - Deploy an Access
 - Extract Meta Data
 - Importing Data from Microsoft Excel
- Control Center
 - Applications
 - Modeling
 - Virtualization
 - tribefire GME - System
- tribefire GME Control Center - Cortex Workbench
 - Hyperlink Action
 - Queries
 - Import Automatic Folders
- Standard Components
 - List View/Thumbnail View
 - Localization
 - Metadata
 - Search Panel
 - tribefire Types
- tribefire Cartridges
- tribefire Partner Portal
 - TF Quick Start Guide
 - Module: Setup & Installation
 - Module: Your first Model
 - Module: Edit your Model in the tribefire Control Center
 - Module: Workbench Configuration
 - Module: Access your Model data via REST
- tribefire Core Models
 - Manipulation Model
 - Process Model
 - The MetaModel aka a model that models models
 - The Query Model
- REST, Restlet and tribefire
 - Tribefire Restlet Calls
 - GmWebRpcServer - Example
 - tribefire Training Model
 - REST API Documentation
- Standard Metadata
- Modelling with UML
- Dependency Management - Controlling the Uncontrollable
 - TFG-ACDC
 - TFG-BTAntTasks
 - TFG-Greyface
- tribefire Development
 - Building CSP ServerDeps using the new platform-exclusion-based way
 - Transition from the ECM Service to the Access Service
 - Provider API
 - Codecs
- tribefire System Requirements
- tribefire Glossary

tribefire cortex

Table of Contents

▼ Expand / collapse

- tribefire Cortex
 - Information Processing
 - Technical Key Differentiators

tribefire Cortex

tribefire CORTEX is a Gödelized, Model-driven Expertsystem whose underlying paradigm is not foundin other model driven approaches (MDA Model Driven Architecture, ADM Architecture Driven Modernization) to software development.

The three aspects of cortex's paradigm are:

- **Expert System**
 - As an expert system, tribefire CORTEX separates the formal description of functionality from the functionality itself, making it available to decoupled processing and free recombination.
- **Model Driven**
 - The formal description of the functionality is given by means of a model definition, which serves as guarantor for type-safe relationships between the formal terms.
- **Gödelized**
 - The operands and operations of the Expert System are both models which normalizes the whole system, reminiscent of Kurt Gödel's mathematical methods. This normalization allows the description and implementation of reflectional operations of all kinds and therefore opens up a new power of expression and control.

The paradigm of tribefire CORTEX means building up all aspects of the software process from a modeling point of view. This does not only include the normal content but also the typical functionality which occurs when working with models. We revolutionize the usage of models in the following areas:

- definition
- reflection
- persistence
- validation
- querying
- transportatio
n
- manipulation
- reproduction
- relationship
- configuration
- security
- business content.

The functionality of these operational models is implemented by specific optimized experts. As the models and their described functionality are designed to work with other specific or arbitrary models, it is possible to make this functionality visible and configurable, whereas normal systems can only provide business data and static functionality. That means, we eliminate the separation of data and code as we subordinate everything to a paradigm. The main differentiator from the tribefire platform to classic software development approaches is the high reuse of code that by its nature is invariant for different specific models. In the tribefire platform there is an integrated expert system that automatically takes care of that without the need to write any code. If needed, of course, the system can be enhanced with your own code in several positions (validators, transformers/codecs, custom accesses, ...). In classic software development approaches big parts of the code needs to be rewritten again for new models despite its invariant nature

Information Processing

tribefire CORTEX's approach for information processing is characterized by the following points:

- **Classification**
 - Classification is achieved by the means of models that are there for describing certain domains of information systems. Models basically consist of a number of entities used for describing the peculiarities of an information object. Those entities then have properties that can be realized by simple types (Integer, String, Boolean, etc.) or complex types (referencing other entities).
- **Association**
 - Entities can be associated to each other by inheritance and relationships of different cardinality (single reference, set, list, map)
- **Dissociation**
 - This can be a matter of security, realized by the built-in Security Model, but dissociation can also be a matter of

performance. A system may consist of a huge network of information objects. Loading every single entity would harm the system performance. To avoid this, tribefire cortex uses Partial Representation, which allows the return of "absence information" as a first part of a query result. Further data is then loaded after a certain freely definable time period, e.g. 2 seconds or by directly accessing the entry (e.g. clicking on it)

- **Reflection**

- Reflection allows models to "talk" to each other. This is achieved by the means of the Metamodel. It provides the capability to:
 - Create/modify models and entity types at runtime
 - Create/modify entity instances and metadata at runtime
 - Let models read out values stored in other models
 - Let models know the state of other models

- **Reproduction**

- This aspect is covered by tribefire's access to the persistence layer by its different Integration Accesses, as well as by its built-in Manipulation Model. The Manipulation Model creates new Entity instances that track changes for any manipulation undertaken on a model's properties, as well as on the actual information (entity instances).

Technical Key Differentiators

- Invention of a new modeling and normalization paradigm, compared to existing approaches like MDA, ADM, incorporating OOP and AOP.
- Gödelized Expertsystem
- Combination of generic and expressive programming
- Persisting (storing) of complex models
- The possibility to execute complex queries against our object oriented database "SMOOD".
- A generic model editor for the definition (administration) of models, as well as for the viewing and editing of models (user interface)

Key Concepts

Table of Contents

Expand / Collapse

- What is tribefire?
- The 3-Way Approach - In a nutshell
 - Virtualization
 - Connectors
 - Models
 - Information Models
 - Prepackaged Information Models
 - Integration Models
 - System Models
 - Apps
 - Deployment
 - Smart Information Accesses
 - Integration Accesses
 - System Accesses
 - Smood

What is tribefire?

tribefire unlocks the value of your enterprise information. With tribefire it's simple to build Smart Enterprise Information Apps to serve real business needs while connecting to existing data sources. The tribefire platform offers 3 layers: a virtualization, a modeling and an application layer. Virtualization (the bottom layer) is responsible for the direct connection to the repositories, while the middle layer provides models to represent the data inside the application. Creating these models allows customers to generate completely new and valuable information out of different repositories. The top layer is the interface for developers, and offers a wide array of different ways to access the model.

The 3-Way Approach - In a nutshell

The 3-Way approach is based on the 3 layers of the tribefire platform and describes a unique way of building Smart Enterprise Information apps. Typical tribefire projects start by creating screen mockups of the desired Smart Enterprise Information App and will be created together with the end-user or management executives. Based on these screen mockups, the required information objects will be modeled in the modeling layer (middle layer) together with the experts from the business. After the model is deployed in the tribefire platform, Front-End Developers and Experience Designers can start developing their Smart Enterprise Information Apps by connecting to the tribefire API (application layer).

Virtualization

This layer is used to create direct connections to repositories. This is done through the use of a series of default connectors, including, but not limited to, database, sharepoint, documentum, twitter, facebook, etc. After creating a connection, they import the schema of such systems into tribefire as "Rough Types". These rough types, themselves, are models and are used by the integration accesses to create entities in an automatically generated system model. Each connector has a related integration access which allows this process to work.

Connectors

tribefire GME comes with a series of standard connectors which can be used to create connections to third party systems, for example, Jdbc databases, sharepoint, salesforce, documentum, twitter, etc. Each connector has its own properties that are used to configure each connector.

The connectors are the objects that make the actual connections to the underlying repositories. A connector will be configured and then deployed, allowing tribefire to import the structures of the data-source as "Rough Types" into the tribefire system. These Schema will then be used by integration accesses to create new models to represent the data in tribefire.

Models

The models layer is where the main power of tribefire GME is reveal. Everything in tribefire is a model. The models in this layer allow you to represent data inside the application. Using the integration accesses, you can automatically generate models which represent the structures of the data in the underlying repositories. As well as this, you can use tribefire GME to work in simulation mode. This allows you to configure and model in tribefire without a real connection between the system and repositories themselves.

There are also Information Models. Whereas System Models are used to map and virtualize data coming from a repository, Information models are based on real life use cases, which will be created by business experts. A information Model will then be mapped to a information access, which itself will have delegates. These delegates are the integration access and therefore, allows the integration model to use different repositories to get its information.



The tribefire system is built upon models, in that everything is a model. From a technical perspective there are no differences between the different categories of model (that is, Information Models, System Models, Integration Models and so on). We differentiate between

these different types of models to enable a clearer, more coherent understanding between the different roles our models are capable of undertaking.

Information Models

The information model is used to model common business scenarios and can contain basic business entities such as "Customer", "Company", "Opportunity" and so on. These models will be created by the business users (eventually by using tribefire's graphic modeler) with their method of choice (for example in Argo UML). Once the entity types have been created, their relationships between each other can be modeled. For example, the company entity type would have any number of customers, while both customer and company will have any number of people.

It is possible to reuse preexisting models, so that you don't have to model the same common business scenarios over and over.

The Information Model can be deployed straight away by creating an information access and can be accessed through one of tribefire's API (Java, REST, Web Services). The information Model would then be used in simulation mode, which allows for rapid evaluation of the model.

Prepackaged Information Models

As well as being able to design and build custom information models, tribefire also comes installed with a number of prepackaged information models. These models are used to carry out various different operations, such as processes, document and resource handling.

Integration Models

An Integration Model is used to represent data-structures that are imported through the connection to a data source. Created automatically during the configuration of an Integration Access, these models are the building blocks for information models. This means that when creating a Information Model, the entities will be linked to various numbers of integration accesses (and therefore, Integration Models) which will then be used to provide the real data, coming from a repository, in the information model. As the name would suggest, they integrate the different sources of data that will be used in the custom-made Information Model.

System Models

The system models are used, as the name suggests, for system operations. They come in two different flavours: Core Models and Platform Models. The core models are used to query and manipulate data within the tribefire system, as well as to deploy and handle metamodels. Platform Models deal with platform specific operations, such as security and users.

Apps

The apps are the top layer of the tribefire system. They are used to visualize the information, to give a greater mobility to users. Because of the decouple nature of tribefire, developers do not need to worry about where they get their data from. They can simply design their app and then plugin in later. Again, making use of tribefire's simulation mode, this app can be tested without a connection to any real system or data.

Deployment

A model on its own is not enough when we can to query or manipulate it. For that we must create an access which is then deployed to the Execution layer. There are three main types of accesses: Smart Information Accesses, Integration Accesses and Smood Accesses. Accesses are the means to retrieve and store generic entities. As the name suggests, accesses allow tribefire to alter, configure and manipulate models. They provide accesses to the various models in tribefire and therefore there are different types of accesses which deal with the different types of models.

 As with models, there is no technical difference between the accesses, other than the configuration parameters they require. We have call these different accesses by different names to enable a clearer understanding of what each type of access does and it roles within tribefire.

Smart Information Accesses

The Smart Information Access is related to the Smart Information Model, that is the custom model designed based upon a business case. The Smart Information Access combines several integration accesses(it's delegates) into a single model that best fits the needed perspective of the data.

Integration Accesses

Integration accesses are used in conjunction with connectors to physically talk to underlying repositories. They use the schema imported by the connector to automatically generate a model and its entity types. These accesses will be used by a smart information access to combine the different integration accesses needed to fully describe and fill the custom business model with data. However, once deployed they can also be accessed directly to query and manipulate data. It is also possible to configure these accesses and use them in simulation mode, so that no actual connection to an underlying repository is needed and allows you to test the configuration of your models without the worry about the data-source behind it.

System Accesses

System accesses provide accesses to specific system models. Each model requires an access so that it can be queries, manipulated and changed, and system models are no different. An example of a system access would be for workbenches. A workbench allows you to create queries for your integration or information accesses (and thus, your integration and information models). There is a standard workbench model which comes packaged with tribefire. Each time a new workbench access is created by using the Create Workbench Access button, you create a new instance of this model and then an System Access which accesses this System Model.

Smood

Smood stands for Smart Memory Object Oriented Database and is Braintribe's own implementation for model persistence, with extended query capabilities. It is used for internal storage for all meta models (that is, the description and configuration of all models and accesses). When working in simulation mode as smood access is also used to store test data on the system. This take the place of whichever access that you have placed in simulation mode.

tribefire Control Center



TRIBEFIRE
Control Center



Work in progress

▼ tribefire GME table of contents...

- Smart Information Models
- Deployment Models
- Installation
 - Installing tribefire
 - Starting tribefire
- User Guides
 - Adding simple types to Object Properties
 - Assigning Roles and Groups to Users
 - Assigning Roles and Users to Groups
 - Attach Metadata from Extract
 - Auto Enrich
 - Configuring and setting up a connection to MsSQL
 - Downloading and Installing MsSQL
 - Configure MsSQL
 - Creating a MsSQL Connection
 - Creating a MsSQL access
 - Automatically create a new Model from your MsSQL database
 - Query your new MsSQL Model in tribefire GME
 - Configuring and setting up a connection to MySQL
 - Downloading and installing MySQL
 - Configure MySQL
 - Create a MySQL connection
 - Create a MySQL access
 - Automatically create a new Model from your MySQL database
 - Query your new MySQL Model in tribefire GME
 - Create a new Access
 - Create a Simple Query
 - Create a Template Query Action
 - Create Custom Workbench Access
 - Create Model From Zargo File
 - Create New Properties
 - Create Simple Instantiation Action
 - Creating User Roles and Groups
 - Deploy an Access
 - Extract Meta Data
 - Importing Data from Microsoft Excel

- Control Center
 - Applications
 - Modeling
 - Custom Models / Distributed Models
 - Entity Types
 - Virtualization
 - Connections
 - Connectors
 - Generic Jdbc Connection
 - Jdbc URLs
 - MsSQL Connector
 - Custom Accesses / System Accesses
 - Simulation Mode
 - Access Types
 - Hibernate Access
 - Smart Enterprise Information Access
 - Smood Access
- tribefire GME - System
 - Connection Tests
 - Deployments
 - Model Imports
 - Schema Updates
 - Files
 - Uploading Files
 - Downloading Files
 - Property Groups
 - Images
 - Uploading a New Image
 - Viewing an Image
- tribefire GME Control Center - Cortex Workbench
 - Hyperlink Action
 - Queries
 - Ordering
 - Restrictions
 - Paging
 - Conditions
 - Abstract Junction
 - Conjunction
 - Disjunction
 - Negation
 - Value Comparison
 - PropertyOperand
 - Full Text Comparison
 - Import Automatic Folders

- Standard Components
 - List View/Thumbnail View
 - Localization
 - Metadata
 - General Metadata Properties
 - Conflict Priority
 - Important
 - Inheritance
 - Selectors
 - Conjunction Selector
 - Disjunction Selector
 - Negation Selector
 - Role Selector
 - Use Case Selector
 - Model Metadata
 - Model Display Info
 - Model Icon
 - Entity Type Metadata
 - Database Mapping
 - Default Sort
 - Entity Compound Viewing 2
 - Entity Condensation
 - Entity Emphasis
 - Entity Icon
 - Entity Instantiation Disabled
 - Entity Priority
 - Entity Simplification
 - Entity Type Display Info
 - Entity Visibility
 - Entry Deletion
 - Group Priority
 - Querying Allowed
 - Selective Information
 - Property Metadata
 - Bidirectional Property
 - Collection Element Count Constraint
 - Excel Column Mapping
 - Excel Identity Management Property
 - Excel Reference Column Mapping
 - Group Assignment
 - Mandatory Property
 - On Change
 - Password Property
 - Property Bulleting
 - Property Display Info
 - Property Editable
 - Property Empty String
 - Property Icon
 - Property Mapping
 - Property Priority
 - Property Shares Entities
 - Property Simplification
 - Property Visibility
 - Range Boundary
 - String Regexp Constraint
 - Unique Key Constraint
 - Enum Type Metadata
 - Search Panel
 - tribefire Types
 - BindingArtifact
 - GmSimpleType
 - EntityType
 - EnumType
 - Model
 - Property

Introduction

tribefire Control Center allows you to administer tribefire. It provides a wide range of functionality, including the ability to create, import and

configure models, view important log information and update database schema. In addition to this, Control Center gives you full control over your accesses and connections, vital when connecting to underlying repositories. You can create, view and deploy your accesses and connections, as well as being able to configure the simulation mode.



Due to the powerful nature of tribefire cortex, and its flexible properties, the following screenshot is only an example of what is possible with tribefire Control Center. Using the Workbench, you can configure Expert Mode to suit your own requirements. However, the following example highlights the key concepts of tribefire cortex and how it is then represented in the tribefire Control Center.

The screenshot shows the tribefire Control Center interface. The top navigation bar includes a logo, a search bar, and links for 'Welcome' and 'C. Cortex'. The main dashboard features several large orange icons representing different system components: 'Images' (document), 'Integration Accesses' (person), 'System Models' (gear), 'Connections' (lightbulb), 'Information Models' (cubes), 'Information Accesses' (info circle), and 'Information Processes' (graph). To the left, a sidebar titled 'Quick Access' lists several categories with their respective sub-items:

- Smart Enterprise Information**
 - Applications
 - Care Connect
 - Executive 360
 - Finance Connect
 - Member App
- Modeling**
 - Information Models
 - Information Processes
 - System Models
- Virtualization**
 - Connections
 - Information Accesses
 - Integration Accesses
 - System Accesses
- System**
 - Protocol
 - Connection Tests
 - Deployments
 - Model Imports
 - Schema Updates
- Resources**
 - Files
 - Groups
 - Images
 - Types

Smart Information Models

Table of Contents

▼ Expand / collapse

- Smart Information Models
 - Smart Information Model
 - Smart Access
 - Workbench and Workbench's Workbench

Smart Information Models

Smart Information Model

Although in tribefire all models are the same from a technical standpoint, to give a clearer understanding of each model's functionality within the system, we have separated them into logical types. Whereas the system models are used to govern and control the behavior of tribefire itself, smart information models are used to describe business use cases. This could be the modeling of a business process, a doctors office, or to describe sales information. However, regardless of what type of information you wish to model, you can use tribefire Control Center to do so. There are also many other options available to create your business model, such as using a Java IDE or UML, for example.

Smart Access

Each model that is created or used in tribefire must have a corresponding access. The access, as its name suggests, allows you to access the model's data for configuration. Therefore, after creating a model in, or uploading one to, tribefire, you must provide it with an access. In this instance, we provide the Smart Information Model with a Smart Access. This access can then combine various integration accesses (known to the Smart Access as delegates) which will provide your model with real data from various underlying repositories. The last step would be to deploy this access to the execution layer. Until you have successfully deployed an access, it can't be truly said to belong to the tribefire system and therefore, no real operations can be carried out on it.

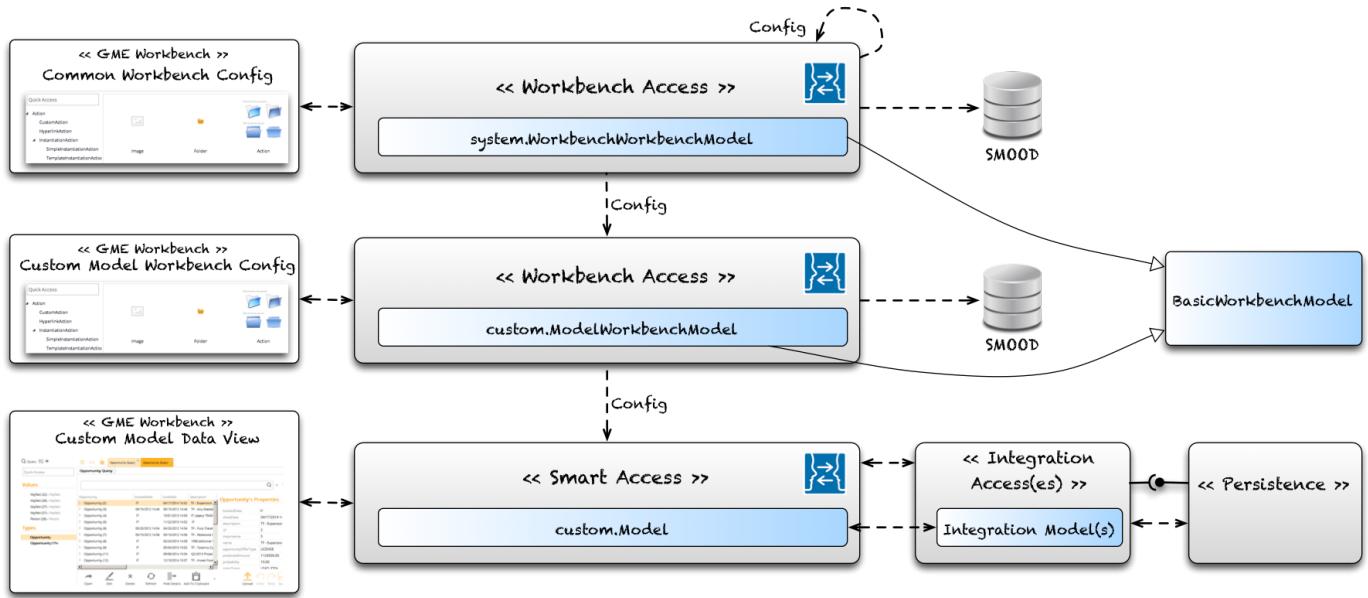
Although the main aim of creating a Smart Information Model and connecting it to repositories is to allow designers to create apps that can display and manipulate this data, you can also use the access itself to view and query the same data. This can be done by using the tribefire Explorer and it is useful for testing the configuration of your setup. You can either use Explorer to add, edit, delete or simple query real data, or by placing your access in simulation mode you can add and edit your configuration with mock data. This allows you to test the system without affecting any real data located in your repositories.

Workbench and Workbench's Workbench

It is also possible to configure the Data View, allowing you to display custom actions and queries in its menu panel. This is done through the use of a Workbench. The Workbench is an access based on a model and can be quickly created through the use of the [Create Workbench Access](#) button in Control Center.

In addition to this you can also create a Workbench for your Workbench. This allows you to customize your configuration tool. This is done through the use of an access based on another Workbench.

The diagram below shows how the different components are related and interact with each other.



By using the Workbench, you can create queries and custom actions on your model.

Deployment Models

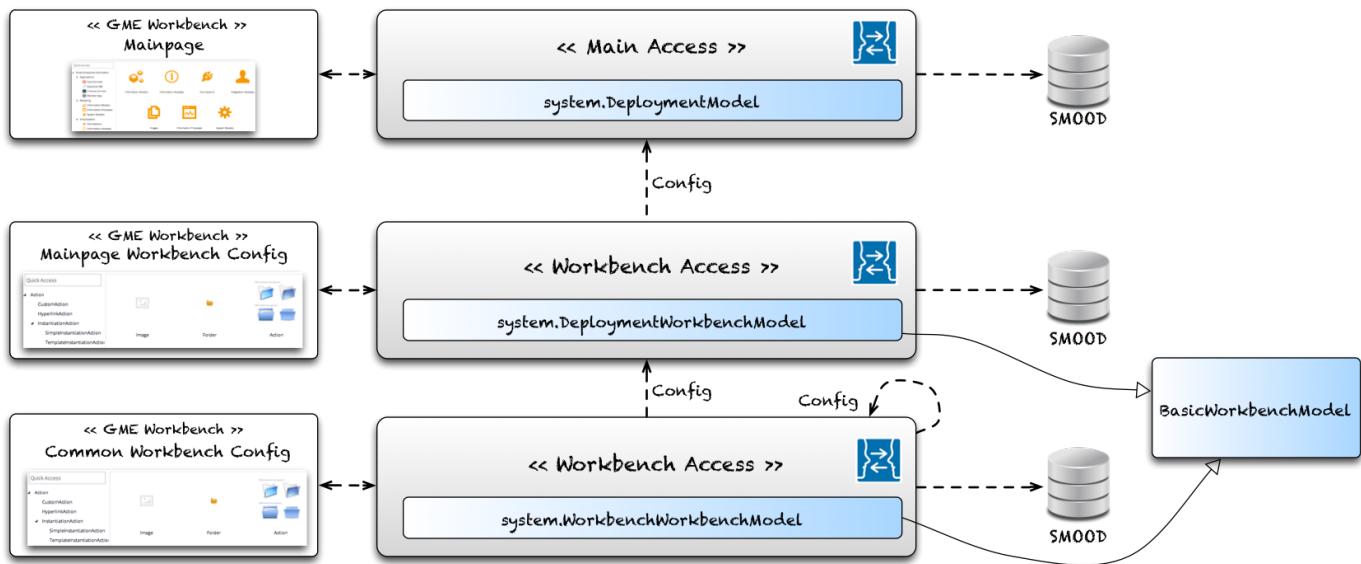
Table of Contents

Expand / collapse

- Deployment Models
 - Cortex
 - Cortex Workbench and Workbench
 - Authentication and Authorization

Deployment Models

The tribefire Control Center is a powerful tool which gives you control over the operation and functionality of your tribefire installation. You can use the Control Center to create, design and configure Models, build accesses and connections and even query and create data using tribefire Explorer. This is where you can make your models come alive, and actually consists of three parts: the Control Center itself (cortex), Cortex Workbench and Workbench. Each separate part are related to each other in a hierarchical structure. That is, the Workbench is used to configure the Cortex Workbench and then the Cortex Workbench is used to configure the Control Center. The diagram below shows how the three parts are related to each other and how they interact.



Cortex

tribefire cortex is a model-driven expert system. This means that everything found in tribefire is based on a model, and tribefire Control Center is no different. Because of the nature of tribefire each access, or connection, needs to be deployed to the execution layer, before any changes can be made on them. Therefore, the Control Center, which is used primarily for these functions, is based on the system.DeploymentModel. The prefix system indicates clearly that it is a system model. Although all models are technically the same in tribefire cortex, we partition them into logical divisions for a coherent, clear understanding.

If you want to learn more about the technical nature of tribefire cortex, please [click here](#).

In addition to this, each model, regardless of what kind, needs an access that allows it to be used, changed and manipulated. That means that what is actually used when you work with Control Center is an access based on the DeploymentModel.

Cortex Workbench and Workbench

If we wish to configure the Control Center, that means we use the Cortex Workbench. Likewise, the Cortex Workbench, which is an access based on the system.DeploymentWorkbenchModel, can be used to create and edit links, actions and queries found in the Control Center. In short, it allows you to opportunity to fully customize tribefire.

Finally, to configure the Cortex Workbench, you need to use the Workbench. As before, the Workbench is an access based on the system.WorkbenchWorkbenchModel. This allows you to fully customize your Workbench.

Authentication and Authorization

In addition to the three standard deployment models (Cortex, Cortex Workbench and Workbench), there is Authentication and Authorization access, based on the SecurityServiceApiModel. As the name suggests this is the access that we use when dealing with administrative permissions and roles within the tribefire system. With this access you can create new users, groups and roles.

Installation



Work in progress!

Introduction

This section of the documentation will show you how to install your instance of tribefire GME and how to login.

- [Installing tribefire GME](#)
- [Starting tribefire GME](#)

Installing tribefire

Table of Contents

▼ Expand / collapse

- Introduction
- Prerequisites
- Installation

Introduction

tribefire GME can be installed using the tribefire-installer. This will install the components that you need to get started and create direct links on your desktop, these links allow you to access the tribefire Control Center, as well as being able to start and stop the tribefire host.

It is recommended that you check tribefire's [system requirements](#) before installation.

The tribefire Nucleus package includes, tribefire Control Center and Explorer, tribefire Host Manager(Tomcat) and tribefire Host.

- Control Center - The Control Center allows you to control, create and edit connections, models and access. It is the main administration tool for tribefire.
- Explorer - The Explorer allows you to manipulate and query access and data associated with it.
- Host Manager - Provided by tomcat, the host manager allows you to control and manage the servlets running on your system.
- tribefire Host - Powered by Cortex, tribefire Host is a thin server which controls communication between the Control Center and tribefire's underlying system.

Prerequisites

- Java Development Kit (JDK) version 7 or higher



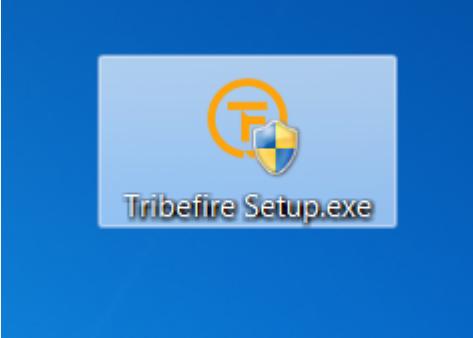
After installing JDK, you must set the environmental variables so that your system, and the tribefire-installer, knows where to find it. The environmental variables include:

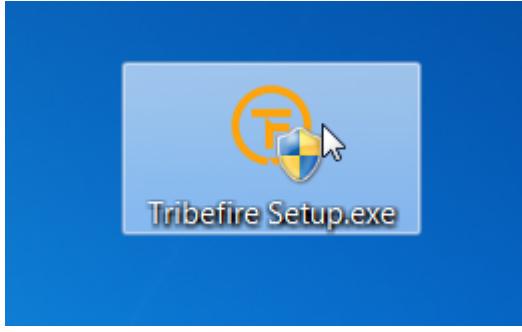
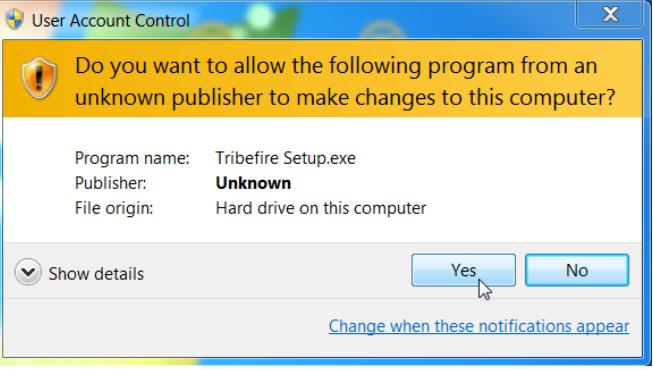
- JAVA_HOME
- JRE_HOME

The value of both should be their respective installation directories. After adding these patches, you must add the following to the Path variable:

- %JAVA_HOME%\bin

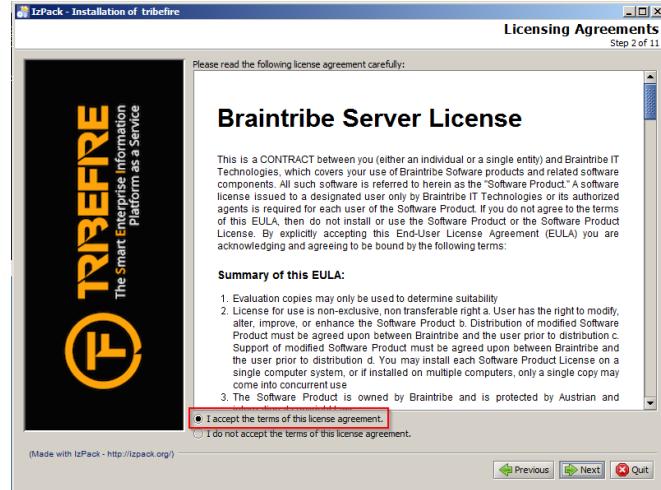
Installation

Step #	Task
1	<p>The installer comes in the form of an executable file.</p> 

2	<p>Double click Tribefire Setup.exe to begin the installation process.</p> 
3	<p>Click Yes.</p> 
4	<p>You may receive a warning stating that no JDK has been found. If you are sure that you have JDK installed on your system, click yes. Otherwise, click no and install JDK.</p> 
5	<p>The installation progress will begin. Click Next to continue.</p> 

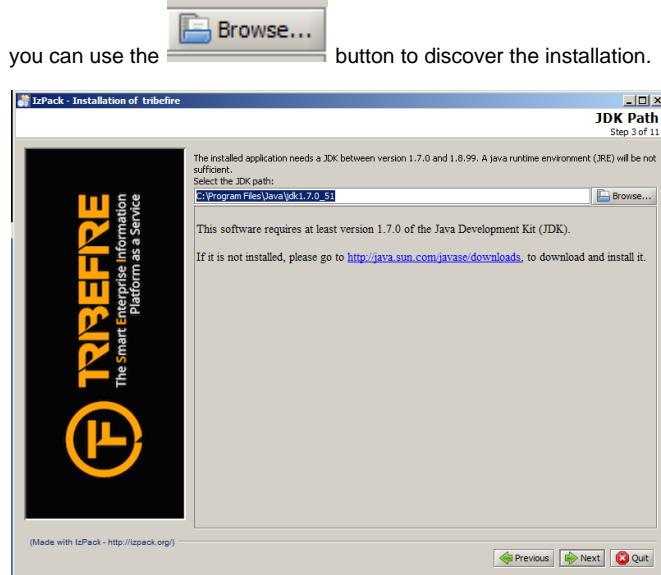
6

Click **I accept the terms of this license agreement.**, and click **Next** to continue.



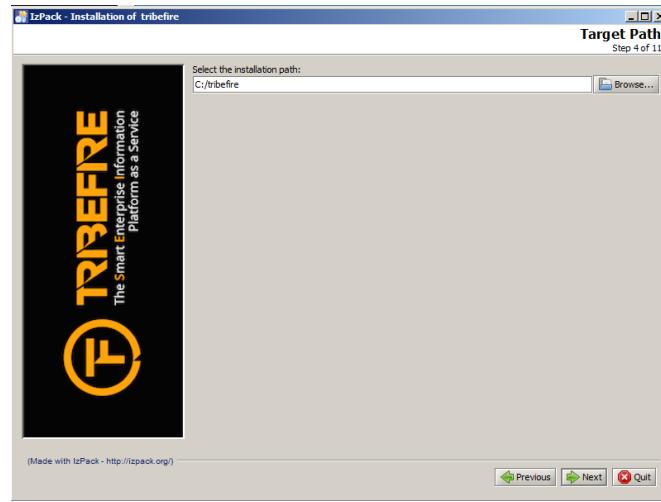
7

The tribefire-installer should automatically discover the location of your JDK installation. If this is the case, click **Next** to continue. However, if the tribefire-installer can not find the JDK installation path,



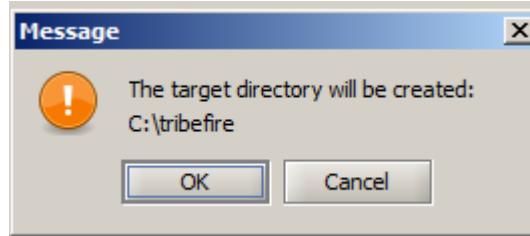
8

Set the installation path and then click **Next** to continue.



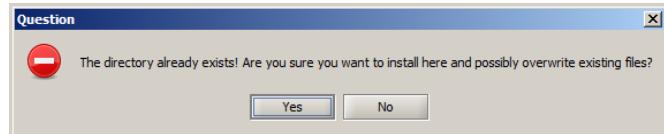
9

If the directory does not exist, you will receive a message confirming its creation. Click **OK** to continue.



However, if the directory already exists, you will be prompted as to whether it is okay to install to this directory. Click **OK** if you wish to continue, otherwise, select a different directory.

! Overwriting a directory may delete or overwrite files which already exist in this directory. If you are not sure whether this is okay, it is recommended you install to a new directory.



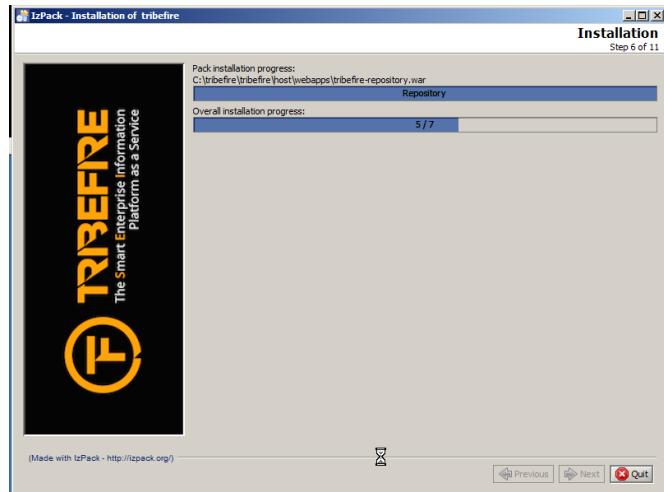
10

Select the components you wish to be installed. The Base, Host and Services options are required.



11

tribefire will now be installed.



12

Select whether you would like to start tribefire as a service or not, and then click **Next** to continue.

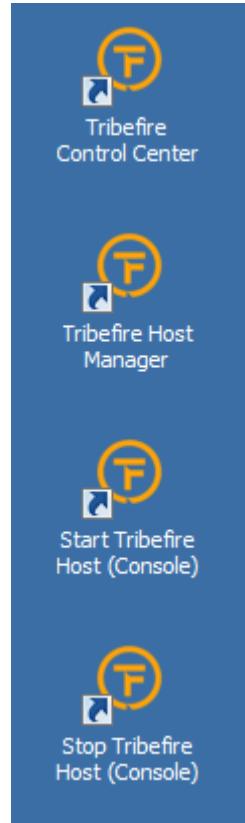


13

Select whether you would like the installer to create shortcuts on the desktop and the start-menu.



Your installation is now complete. If selected to create shortcuts, you will find them on the desktop.



Starting tribefire

Table of Contents

▼ Expand / collapse

- Introduction
- Starting tribefire host
 - Shortcut
 - Services
- Accessing tribefire Control Center
 - tribefire Control Center URL syntax
 - Logging into to a specific access

Introduction

This page will show you how to start your tribefire host and access the tribefire Control Center, including the correct syntax of the URL.

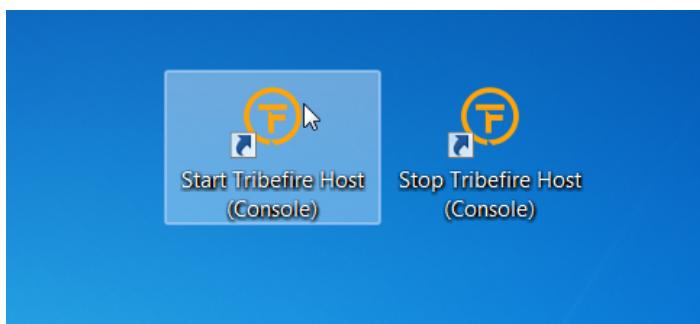
For details on how to install tribefire.

Starting tribefire host

Before you can access the tribefire control center, you must ensure that the tribefire host is running. There are two ways of doing this, either with the shortcut provided during the installation process, or by using services.

Shortcut

During installation, a series of shortcuts will be created on your desktop. Click the shortcut named **Start Tribefire Host (Console)**.



A command window will be opened and the tribefire host will start. You will receive the notification *Info: Server startup in XXXX ms*. This means that the tribefire host has started and you can now access the Control Center.

```

Tomcat
Apr 10, 2014 10:08:13 AM com.braintribe.web.servlet.IocWebapp
INFO: Done initializing context javax.servlet.ServletContextEvent[source=org.apache.catalina.core.ApplicationContextFacade@39a12a0e]
Apr 10, 2014 10:08:13 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\tribefire\tribefire\host\webapps\docs
Apr 10, 2014 10:08:14 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\tribefire\tribefire\host\webapps\examples
Apr 10, 2014 10:08:14 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\tribefire\tribefire\host\webapps\host-manager
Apr 10, 2014 10:08:14 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\tribefire\tribefire\host\webapps\manager
Apr 10, 2014 10:08:14 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\tribefire\tribefire\host\webapps\ROOT
Apr 10, 2014 10:08:14 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-apr-8080"]
Apr 10, 2014 10:08:14 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-apr-8009"]
Apr 10, 2014 10:08:14 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 19654 ms

```

Services

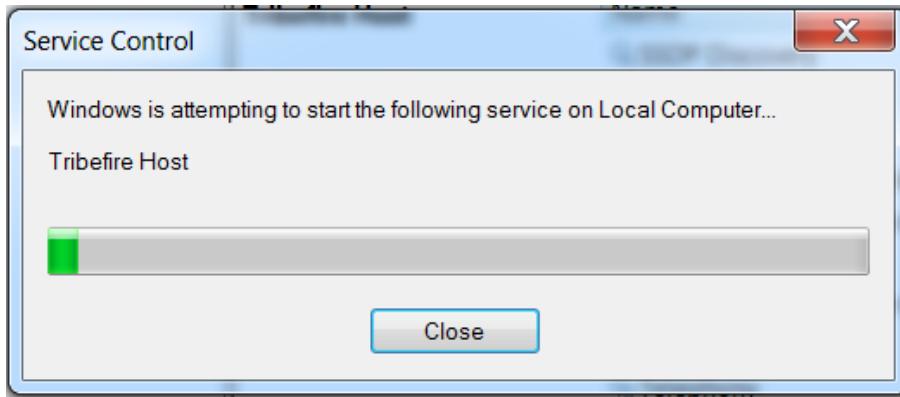
From the start menu, type **Services** into the search field and press Enter.



Scroll down until you find the Tribefire Host entry.

Services (Local)					
	Name	Description	Status	Startup Type	Log On As
Start the service	SSDP Discovery	Discovers n...	Started	Manual	Local Servi...
	Storage Service	Enforces gr...	Started	Manual	Local Syst...
	Superfetch	Maintains a...	Started	Manual	Local Syst...
	System Event Notification Service	Monitors sy...	Started	Automatic	Local Syst...
	Tablet PC Input Service	Enables Tab...	Started	Manual	Local Syst...
	Task Scheduler	Enables a u...	Started	Automatic	Local Syst...
	TCP/IP NetBIOS Helper	Provides su...	Started	Automatic	Local Servi...
	TdmService	Manages se...	Started	Automatic	Local Syst...
	Telephony	Provides Tel...	Started	Manual	Network S...
	Themes	Provides us...	Started	Automatic	Local Syst...
	Thread Ordering Server	Provides or...	Started	Manual	Local Servi...
	TPM Base Services	Enables acc...	Started	Manual	Local Servi...
	Tribefire Host	Braintribe T...	Automatic	Local Syst...	
	UPnP Device Host	Allows UPn...	Started	Manual	Local Servi...
	User Profile Service	This service ...	Started	Automatic	Local Syst...
	Virtual Disk	Provides m...	Started	Manual	Local Syst...
	VMware Authorization Service	Authorizati...	Started	Automatic	Local Syst...
	VMware DHCP Service	DHCP servic...	Started	Automatic	Local Syst...
	VMware NAT Service	Network ad...	Started	Automatic	Local Syst...
	VMware USB Arbitration Service	Arbitration ...	Started	Automatic	Local Syst...
	Volume Shadow Copy	Manages an...	Started	Manual	Local Syst...
	Wave Authentication Manager ...	Manages se...	Started	Automatic	Local Syst...

Select it and click **Start** to begin the tribefire host. The Tribefire Host service will start.



Once it the service has started, you can now access the tribefire Control Center.

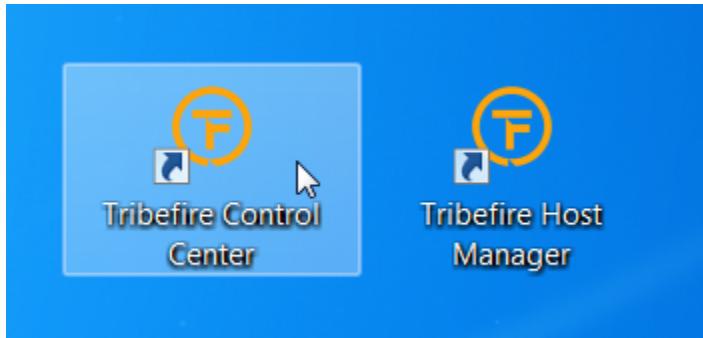
Accessing tribefire Control Center

Once the Tribefire Host is running, you can now access the tribefire Control Center by using a compatible browser.

tribefire Control Center is currently supported by the following browsers:

- Internet Explorer 10 or higher
- Google Chrome 30 or higher
- Safari 6.1.1 or higher
- Firefox 25 or higher

If you have a shortcut on your desktop, you can simply click on it...



and the tribefire Control Center will open in your default browser.



You can login by entering your valid credentials and then clicking the **Sign In** button.

tribefire Control Center URL syntax

You can also manually enter the internet address by using the following syntax:

```
IPADDRESS_OF_SERVER:OPEN_PORT/NAME_OF_CONTROL_CENTER
```

Value	Description
IP Address of Server	This is the IP address of the server that has tribefire Control Center installed
Open Port	This is the open port value of the server that is running Control Center. Generally, the open port is 8080. However, if you are unsure of which port to use, ask your system administrator
Name of Control Center	This is the name of the Control Center installation running on the server.

! The name of the Control Center is case sensitive and so must correspond exactly with the name of the Control Center installation file as found in the webapps folder of your J2EE container.

! If you are accessing the Control Center on a the machine on which tribefire has been installed, you can use the value */localhost* instead of the IP address.

Logging into to a specific access

When you enter the standard address (as shown above) then Control Center will automatically load with cortex, the main administrative tool used to created connections, accesses and models. Using the cog located at the top-right of the screen, you are able to choose between the different accesses that are currently instantiated on your tribefire installation.

However, if you wish to log directly into an access, without going through the development model, then you can do so by using the following syntax:

```
IPADDRESS_OF_SERVER:OPEN_PORT_NUMBER/NAME_OF_CONTROL_CENTER/?accessId=NAME_OF_ACCESS
```

Where the NAME_OF_ACCESS corresponds to a valid access configured in tribefire Control Center

The browser will then loading the login screen and, after you have entered your valid credentials, will then display the access that you entered in the address bar of your browser.

User Guides

Table of contents...

- Introduction
- Control Center
 - Modeling
 - Accesses
 - Connections
 - Databases
- Workbench
- Security
- Miscellaneous

Child pages

Expand / collapse

- Adding simple types to Object Properties
- Assigning Roles and Groups to Users
- Assigning Roles and Users to Groups
- Attach Metadata from Extract
- Auto Enrich
- Configuring and setting up a connection to MsSQL
- Configuring and setting up a connection to MySQL
- Create a new Access
- Create a Simple Query
- Create a Template Query Action
- Create Custom Workbench Access
- Create Model From Zargo File
- Create New Properties
- Create Simple Instantiation Action
- Creating User Roles and Groups
- Deploy an Access
- Extract Meta Data
- Importing Data from Microsoft Excel

Introduction

This section of the documentation contains a selection of tutorials which will guide you through some of the most important functionality in tribefire GME. They will use practical examples to not only give you an idea of usage but also to show you how to correctly configure various components.

Control Center

Modeling

[Attach Metadata From Extract](#)

[Create Model From Zargo File](#)

[Create New Properties](#)

[Extract Metadata](#)

[Importing data from Excel spreadsheet](#)

Accesses

[Create a new Access](#)

[Deploy a new Access](#)

Connections

Databases

[Configuring and setting up a connection to MsSQL](#)

[Configuring and setting up a connection to MySQL](#)

Workbench

This section contains guides relating to the functionality of the Workbench

[Create a custom Workbench](#)

[Create Simple Instantiation Action](#)

[Create a Simple Query](#)

[Create a Template Query Action](#)

Security

[Assigning Roles and Groups to Users](#)

[Assigning Roles and Users to Groups](#)

[Creating User Roles and Groups](#)

Miscellaneous

[Adding simple types to Object Properties](#)

Adding simple types to Object Properties

Introduction

This page will show you how to add a simple type to an object's property.

Navigate to the object you wish to add a simple property type to.

SimpleOrdering's Properties

<i>Entity Id</i>	21
Direction	ascending
Order By	Ø

In the main details panel, using either the simple or detailed view, use the expand icon to expand the objects properties for editing.

The screenshot shows the navigation path: Query Query > EntityQuery (10) > Ordering: SimpleOrdering (21). Below the path, there is a tree view with 'SimpleOrdering' expanded, showing 'SimpleOrdering (21)' and 'Order By: Ø'. To the right, there are two columns: 'Direction' set to 'Ascending' and 'Entity Id' set to '21'. A red box highlights the 'Order By: Ø' field.

Click the Add button...



...and the Selection Constellation will be displayed.

Values

- boolean
- date
- decimal
- double
- float
- integer
- long
- string

```

ComparisonOperator.equal - com.braintribe.model.generic.pr.criteria.ComparisonOperator
ComparisonOperator.greater - com.braintribe.model.generic.pr.criteria.ComparisonOperator
ComparisonOperator.greaterOrEqual - com.braintribe.model.generic.pr.criteria.ComparisonOperator
ComparisonOperator.less - com.braintribe.model.generic.pr.criteria.ComparisonOperator
ComparisonOperator.lessOrEqual - com.braintribe.model.generic.pr.criteria.ComparisonOperator
ComparisonOperator.notEqual - com.braintribe.model.generic.pr.criteria.ComparisonOperator
DeleteMode.dropReferences - com.braintribe.model.generic.manipulation.DeleteMode
DeleteMode.failIfReferenced - com.braintribe.model.generic.manipulation.DeletedMode
DeleteMode.ignoreReferences - com.braintribe.model.generic.manipulation.DeleteMode
EntityTypeStrategy.assignable - com.braintribe.model.generic.typecondition.EntityTypeStrategy
EntityTypeStrategy.assignable - com.braintribe.model.generic.pr.criteria.typematch.EntityTypeStrategy
EntityTypeStrategy.equals - com.braintribe.model.generic.typecondition.EntityTypeStrategy
EntityTypeStrategy.equals - com.braintribe.model.generic.pr.criteria.typematch.EntityTypeStrategy
ExecutionMode.asynchron - com.braintribe.model.action.ExecutionMode
ExecutionMode.synchron - com.braintribe.model.action.ExecutionMode
Operator.contains - com.braintribe.model.query.Operator
Operator.equals - com.braintribe.model.query.Operator

```

Show Details Finish Cancel

Enter the value of your simple type into the text field at the top of the add window.

! You can enter any value of the simple type you like and tribefire GME will automatically recognize it depending on the context. For example if you type "True" it will offer you the option of choosing between string and boolean.

Values

(1) **string - a string**

(3)

Values

- 1 decimal - 0.00
- 2 double - 0.00
- 3 float - 0.00
- 4 integer - 0
- 5 long - 0
- 6 string - 0

true

Values

- (1) boolean - true
- (3) string - true
- (4)

Select the type you would like and then click finish at the bottom of the add window.

Your new simple type will be added to the objects property.

Query Query > EntityQuery (10) > Ordering: SimpleOrdering (21)

SimpleOrdering

- ▲ SimpleOrdering (21)
 - Order By: astring

Assigning Roles and Groups to Users

Table of Contents

Expand / collapse

- Assigning Roles and Groups to Users
 - Introduction
 - Assign a Role to a User
 - Assign a Group to a User

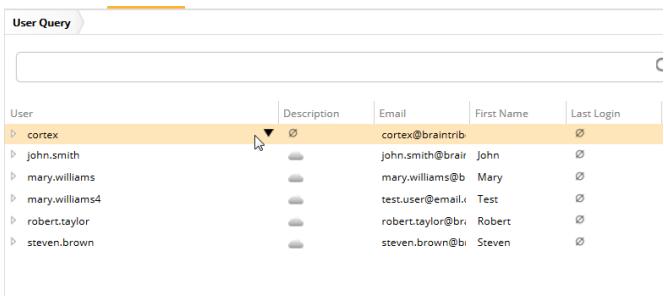
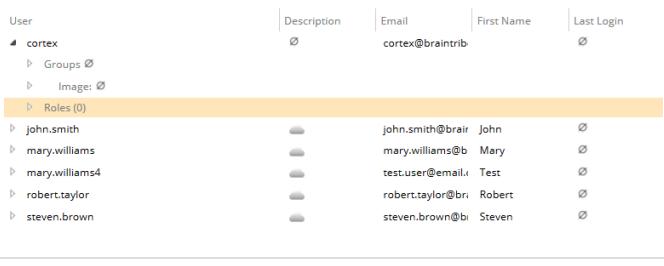
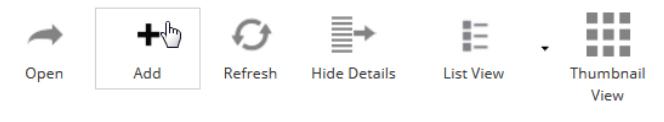
Assigning Roles and Groups to Users

Introduction

Each user should have rights and permissions that constrain their access and the scope of tasks they can undertake. A role is created that defines what these tasks and actions are and are then assigned to a user. A collection of roles can be gathered together into a group, based on logical operations - Admin groups, for example, have wider rights and permissions than guests. A user can then be assigned to a group, and will then inherit the roles defined.

In tribefire, security is handled by the system.securityService model. After creating a users, there are various properties which can be defined. Two important properties are **Roles** and **Groups**. Here you can define which roles the user has and to which group they belong.

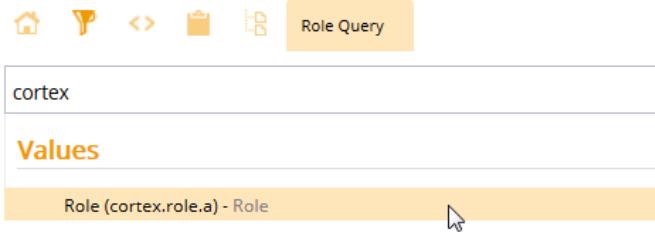
Assign a Role to a User

Step #	Task																																																		
1	<p>Select the user whom you wish to assign a role to.</p>  <table border="1"> <thead> <tr> <th>User</th> <th>Description</th> <th>Email</th> <th>First Name</th> <th>Last Login</th> </tr> </thead> <tbody> <tr> <td>cortex</td> <td>∅</td> <td>cortex@braintribe.com</td> <td>John</td> <td>∅</td> </tr> <tr> <td>john.smith</td> <td>∅</td> <td>john.smith@braintribe.com</td> <td>John</td> <td>∅</td> </tr> <tr> <td>mary.williams</td> <td>∅</td> <td>mary.williams@braintribe.com</td> <td>Mary</td> <td>∅</td> </tr> <tr> <td>mary.williams4</td> <td>∅</td> <td>test.user@email.it</td> <td>Test</td> <td>∅</td> </tr> <tr> <td>robert.taylor</td> <td>∅</td> <td>robert.taylor@braintribe.com</td> <td>Robert</td> <td>∅</td> </tr> <tr> <td>steven.brown</td> <td>∅</td> <td>steven.brown@braintribe.com</td> <td>Steven</td> <td>∅</td> </tr> </tbody> </table>	User	Description	Email	First Name	Last Login	cortex	∅	cortex@braintribe.com	John	∅	john.smith	∅	john.smith@braintribe.com	John	∅	mary.williams	∅	mary.williams@braintribe.com	Mary	∅	mary.williams4	∅	test.user@email.it	Test	∅	robert.taylor	∅	robert.taylor@braintribe.com	Robert	∅	steven.brown	∅	steven.brown@braintribe.com	Steven	∅															
User	Description	Email	First Name	Last Login																																															
cortex	∅	cortex@braintribe.com	John	∅																																															
john.smith	∅	john.smith@braintribe.com	John	∅																																															
mary.williams	∅	mary.williams@braintribe.com	Mary	∅																																															
mary.williams4	∅	test.user@email.it	Test	∅																																															
robert.taylor	∅	robert.taylor@braintribe.com	Robert	∅																																															
steven.brown	∅	steven.brown@braintribe.com	Steven	∅																																															
2	<p>Click the  icon to collapse the user and select the property <i>Role S</i>.</p>  <table border="1"> <thead> <tr> <th>User</th> <th>Description</th> <th>Email</th> <th>First Name</th> <th>Last Login</th> </tr> </thead> <tbody> <tr> <td>cortex</td> <td>∅</td> <td>cortex@braintribe.com</td> <td>John</td> <td>∅</td> </tr> <tr> <td>Groups</td> <td>∅</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Image:</td> <td>∅</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Roles (0)</td> <td>∅</td> <td></td> <td></td> <td></td> </tr> <tr> <td>john.smith</td> <td>∅</td> <td>john.smith@braintribe.com</td> <td>John</td> <td>∅</td> </tr> <tr> <td>mary.williams</td> <td>∅</td> <td>mary.williams@braintribe.com</td> <td>Mary</td> <td>∅</td> </tr> <tr> <td>mary.williams4</td> <td>∅</td> <td>test.user@email.it</td> <td>Test</td> <td>∅</td> </tr> <tr> <td>robert.taylor</td> <td>∅</td> <td>robert.taylor@braintribe.com</td> <td>Robert</td> <td>∅</td> </tr> <tr> <td>steven.brown</td> <td>∅</td> <td>steven.brown@braintribe.com</td> <td>Steven</td> <td>∅</td> </tr> </tbody> </table>	User	Description	Email	First Name	Last Login	cortex	∅	cortex@braintribe.com	John	∅	Groups	∅				Image:	∅				Roles (0)	∅				john.smith	∅	john.smith@braintribe.com	John	∅	mary.williams	∅	mary.williams@braintribe.com	Mary	∅	mary.williams4	∅	test.user@email.it	Test	∅	robert.taylor	∅	robert.taylor@braintribe.com	Robert	∅	steven.brown	∅	steven.brown@braintribe.com	Steven	∅
User	Description	Email	First Name	Last Login																																															
cortex	∅	cortex@braintribe.com	John	∅																																															
Groups	∅																																																		
Image:	∅																																																		
Roles (0)	∅																																																		
john.smith	∅	john.smith@braintribe.com	John	∅																																															
mary.williams	∅	mary.williams@braintribe.com	Mary	∅																																															
mary.williams4	∅	test.user@email.it	Test	∅																																															
robert.taylor	∅	robert.taylor@braintribe.com	Robert	∅																																															
steven.brown	∅	steven.brown@braintribe.com	Steven	∅																																															
3	<p>Click the Add button.</p> 																																																		

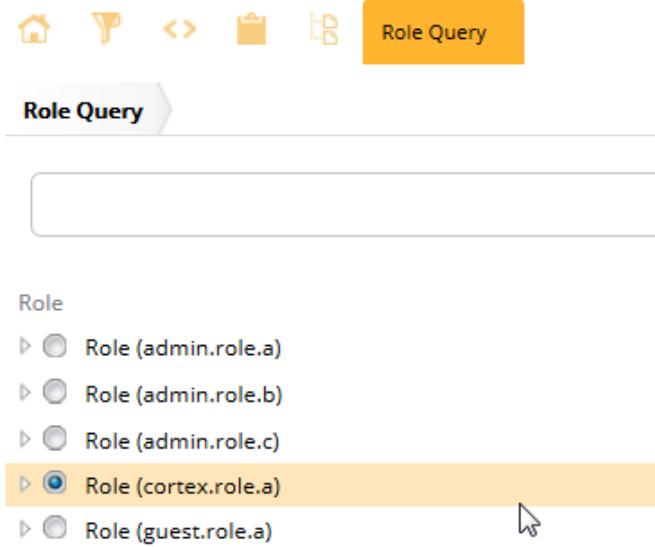
4

The Selection Constellation will be displayed.

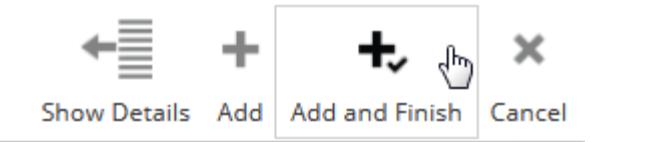
Select the role that you wish by searching for it using the search field...



or click the Role Query and selecting it from the list.



You can add as many roles as you wish by using the **Add** and then **Add and Finish** when selecting the last role you wish to select.



5	<p>The role(s) selected will be assigned to the user.</p> <h3>User's Properties</h3> <table border="1" style="width: 100%; border-collapse: collapse;"> <tbody> <tr><td><i>Id</i></td><td>cortex</td></tr> <tr><td>Description</td><td>Ø</td></tr> <tr><td>Email</td><td>cortex@braintribe.com</td></tr> <tr><td>First Name</td><td></td></tr> <tr><td>Last Login</td><td>Ø</td></tr> <tr><td>Last Name</td><td>Cortex</td></tr> <tr><td>Name</td><td>Ø</td></tr> <tr><td>Password</td><td>cortex</td></tr> <tr><td colspan="2">– Groups</td></tr> <tr><td colspan="2">Ø</td></tr> <tr><td colspan="2">– Image</td></tr> <tr><td colspan="2">Ø</td></tr> <tr><td colspan="2">– Roles</td></tr> <tr><td colspan="2">• Role (cortex.role.a)</td></tr> </tbody> </table>	<i>Id</i>	cortex	Description	Ø	Email	cortex@braintribe.com	First Name		Last Login	Ø	Last Name	Cortex	Name	Ø	Password	cortex	– Groups		Ø		– Image		Ø		– Roles		• Role (cortex.role.a)	
<i>Id</i>	cortex																												
Description	Ø																												
Email	cortex@braintribe.com																												
First Name																													
Last Login	Ø																												
Last Name	Cortex																												
Name	Ø																												
Password	cortex																												
– Groups																													
Ø																													
– Image																													
Ø																													
– Roles																													
• Role (cortex.role.a)																													
6	<p>Click Save to persist your changes.</p> <div style="text-align: center; margin-top: 10px;"> <p>Upload Undo Redo Save</p> </div>																												

Assign a Group to a User

Step #	Task																																			
1	<p>Select the user whom you wish to assign a group to.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> User Query <input style="width: 100%;" type="text"/> </div> <div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>User</th> <th>Description</th> <th>Email</th> <th>First Name</th> <th>Last Login</th> </tr> </thead> <tbody> <tr><td>cortex</td><td>Ø</td><td>cortex@braintribe.com</td><td>Ø</td><td></td></tr> <tr><td>john.smith</td><td></td><td>john.smith@braintribe.com</td><td>John</td><td>Ø</td></tr> <tr><td>mary.williams</td><td></td><td>mary.williams@braintribe.com</td><td>Mary</td><td>Ø</td></tr> <tr><td>test.user@email.it</td><td></td><td>test.user@email.it</td><td>Test</td><td>Ø</td></tr> <tr><td>robert.taylor</td><td></td><td>robert.taylor@braintribe.com</td><td>Robert</td><td>Ø</td></tr> <tr><td>steven.brown</td><td></td><td>steven.brown@braintribe.com</td><td>Steven</td><td>Ø</td></tr> </tbody> </table> </div>	User	Description	Email	First Name	Last Login	cortex	Ø	cortex@braintribe.com	Ø		john.smith		john.smith@braintribe.com	John	Ø	mary.williams		mary.williams@braintribe.com	Mary	Ø	test.user@email.it		test.user@email.it	Test	Ø	robert.taylor		robert.taylor@braintribe.com	Robert	Ø	steven.brown		steven.brown@braintribe.com	Steven	Ø
User	Description	Email	First Name	Last Login																																
cortex	Ø	cortex@braintribe.com	Ø																																	
john.smith		john.smith@braintribe.com	John	Ø																																
mary.williams		mary.williams@braintribe.com	Mary	Ø																																
test.user@email.it		test.user@email.it	Test	Ø																																
robert.taylor		robert.taylor@braintribe.com	Robert	Ø																																
steven.brown		steven.brown@braintribe.com	Steven	Ø																																

2

Click the  icon to collapse the user and select the property Groups.

User	Description	Email	First Name
▲ cortex	∅	cortex@braintri...	
▷ Groups ∅			
▷ Image: ∅			
▷ Roles (1)			
▷ john.smith		john.smith@brai...	John
▷ mary.williams		mary.williams@b...	Mary
▷ mary.williams4		test.user@email.c...	Test
▷ robert.taylor		robert.taylor@bra...	Robert
▷ steven.brown		steven.brown@bi...	Steven

3

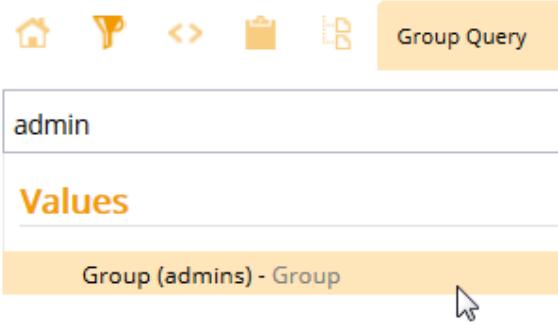
Click the **Add** button.



4

The Selection Constellation will be displayed.

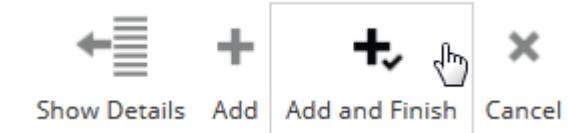
Select the group that you wish by searching for it using the search field...



or click the Group Query and selecting it from the list.

Group	Conflict Prior...	Description
Group (admins)	0.00	
Group (Developers)	0.00	
Group (guests)	0.00	
Group (mary.williams5)	0.00	
Group (operators)	0.00	

You can add as many groups as you wish by using the **Add** and then **Add and Finish** when selecting the last group you wish to select.



5	<p>The group(s) selected will be assigned to the user.</p> <p>User's Properties</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tbody> <tr> <td style="width: 15%;">Id</td><td>cortex</td></tr> <tr> <td>Description</td><td>Ø</td></tr> <tr> <td>Email</td><td>cortex@braintribe.com</td></tr> <tr> <td>First Name</td><td></td></tr> <tr> <td>Last Login</td><td>Ø</td></tr> <tr> <td>Last Name</td><td>Cortex</td></tr> <tr> <td>Name</td><td>Ø</td></tr> <tr> <td>Password</td><td>cortex</td></tr> </tbody> </table> <p>– Groups</p> <ul style="list-style-type: none"> • Group (admins) <p>– Image</p>  <p>– Roles</p> <ul style="list-style-type: none"> • Role (cortex.role.a) 	Id	cortex	Description	Ø	Email	cortex@braintribe.com	First Name		Last Login	Ø	Last Name	Cortex	Name	Ø	Password	cortex
Id	cortex																
Description	Ø																
Email	cortex@braintribe.com																
First Name																	
Last Login	Ø																
Last Name	Cortex																
Name	Ø																
Password	cortex																
6	<p>Click Save to persist your changes.</p> <div style="text-align: center; margin-top: 10px;">  Upload  Undo  Redo  Save </div>																

Assigning Roles and Users to Groups

Table of Contents

Expand / collapse

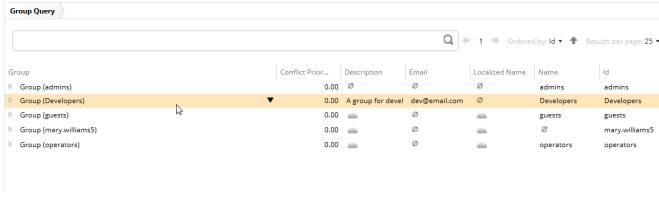
- Assigning Roles and Users to Groups
 - Introduction
 - Assign Roles to a Group
 - Assign Users to a Group

Assigning Roles and Users to Groups

Introduction

Groups are used to gather together roles and users that should have the same permissions and rights; examples of groups include admin and guests. In tribefire you can [create groups](#) using the system.securityService model and then use these new instances to add roles and users which belong to this group.

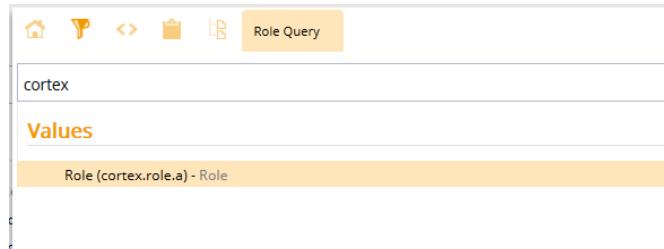
Assign Roles to a Group

Step #	Task
1	Select the group to which you would like to assign a role. 
2	Click the  icon to collapse the group and select the property <i>Roles</i> . 
3	Click the Add button. 

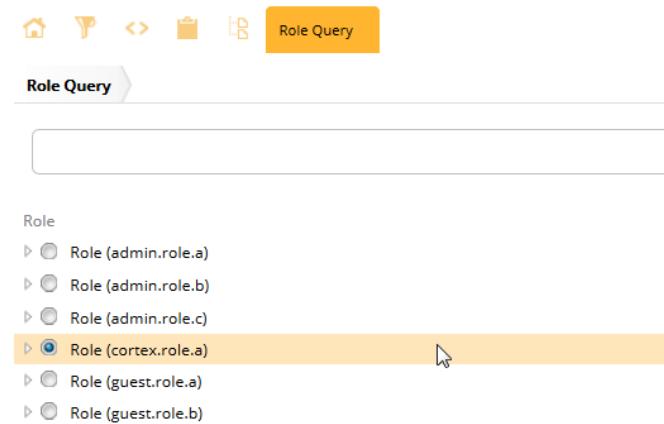
4

The Selection Constellation will be displayed.

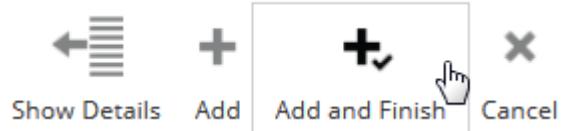
Select the role that you wish by searching for it using the search field...



or click the Role Query and selecting it from the list.



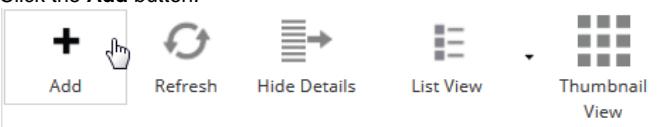
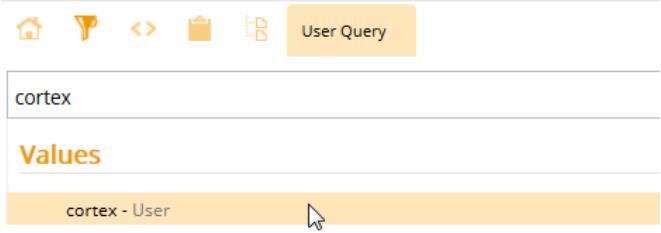
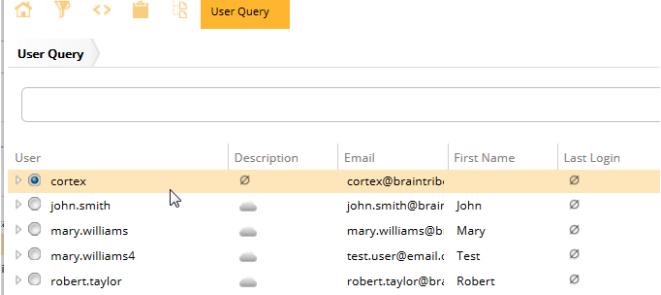
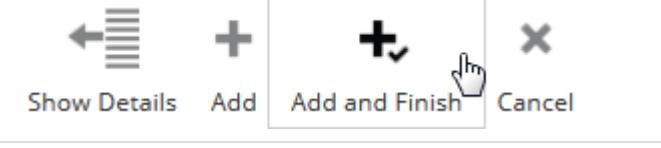
You can add as many roles as you wish by using the **Add** and then **Add and Finish** when selecting the last role you wish to select.



5	<p>The role(s) selected will be assigned to the group.</p> <h3>Group's Properties</h3> <table border="1" style="width: 100%; border-collapse: collapse;"> <tbody> <tr><td style="width: 20%;">Id</td><td>Developers</td></tr> <tr><td>Conflict Priority</td><td>0.00</td></tr> <tr><td>Description</td><td>A group for developers</td></tr> <tr><td>Email</td><td>dev@email.com</td></tr> <tr><td>Localized Name</td><td>Ø</td></tr> <tr><td>Name</td><td>Developers</td></tr> <tr><td colspan="2">– Image</td></tr> <tr><td colspan="2">Ø</td></tr> <tr><td colspan="2">– Roles</td></tr> <tr><td colspan="2">• Role (cortex.role.a)</td></tr> <tr><td colspan="2">– Users</td></tr> </tbody> </table>	Id	Developers	Conflict Priority	0.00	Description	A group for developers	Email	dev@email.com	Localized Name	Ø	Name	Developers	– Image		Ø		– Roles		• Role (cortex.role.a)		– Users	
Id	Developers																						
Conflict Priority	0.00																						
Description	A group for developers																						
Email	dev@email.com																						
Localized Name	Ø																						
Name	Developers																						
– Image																							
Ø																							
– Roles																							
• Role (cortex.role.a)																							
– Users																							
6	<p>Click Save to persist your changes.</p> <div style="text-align: center; margin-top: 20px;">  Upload Undo Redo Save </div>																						

Assign Users to a Group

Step #	Task																																																																					
1	<p>Select the group to which you would like to assign a role.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <p>Group Query <input type="text"/></p> <p>Ordered by: Id ▾ Results per page: 25 ▾</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Group</th> <th>Conflict Prior...</th> <th>Description</th> <th>Email</th> <th>Localized Name</th> <th>Name</th> <th>Id</th> </tr> </thead> <tbody> <tr><td>Group (admins)</td><td>0.00</td><td>Ø</td><td>Ø</td><td>Ø</td><td>admins</td><td>admins</td></tr> <tr><td>Group (Developers)</td><td>0.00</td><td>A group for devel</td><td>dev@email.com</td><td>Ø</td><td>Developers</td><td>Developers</td></tr> <tr><td>Group (guests)</td><td>0.00</td><td>Ø</td><td>Ø</td><td>Ø</td><td>guests</td><td>guests</td></tr> <tr><td>Group (mary.williams5)</td><td>0.00</td><td>Ø</td><td>Ø</td><td>Ø</td><td>mary.williams5</td><td>mary.williams5</td></tr> <tr><td>Group (operators)</td><td>0.00</td><td>Ø</td><td>Ø</td><td>Ø</td><td>operators</td><td>operators</td></tr> </tbody> </table> </div> <p>Click the  icon to collapse the group and select the property <i>Users</i>.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Group</th> <th>Conflict Prior...</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>Group (admins)</td><td>0.00</td><td>Ø</td></tr> <tr><td>Group (Developers)</td><td>0.00</td><td>A group for devel</td></tr> <tr><td> Image: Ø</td><td></td><td></td></tr> <tr><td> Roles (1)</td><td></td><td></td></tr> <tr><td> Users (0)</td><td></td><td></td></tr> <tr><td> Group (guests)</td><td>0.00</td><td>Ø</td></tr> <tr><td> Group (mary.williams5)</td><td>0.00</td><td>Ø</td></tr> <tr><td> Group (operators)</td><td>0.00</td><td>Ø</td></tr> </tbody> </table> </div>	Group	Conflict Prior...	Description	Email	Localized Name	Name	Id	Group (admins)	0.00	Ø	Ø	Ø	admins	admins	Group (Developers)	0.00	A group for devel	dev@email.com	Ø	Developers	Developers	Group (guests)	0.00	Ø	Ø	Ø	guests	guests	Group (mary.williams5)	0.00	Ø	Ø	Ø	mary.williams5	mary.williams5	Group (operators)	0.00	Ø	Ø	Ø	operators	operators	Group	Conflict Prior...	Description	Group (admins)	0.00	Ø	Group (Developers)	0.00	A group for devel	Image: Ø			Roles (1)			Users (0)			Group (guests)	0.00	Ø	Group (mary.williams5)	0.00	Ø	Group (operators)	0.00	Ø
Group	Conflict Prior...	Description	Email	Localized Name	Name	Id																																																																
Group (admins)	0.00	Ø	Ø	Ø	admins	admins																																																																
Group (Developers)	0.00	A group for devel	dev@email.com	Ø	Developers	Developers																																																																
Group (guests)	0.00	Ø	Ø	Ø	guests	guests																																																																
Group (mary.williams5)	0.00	Ø	Ø	Ø	mary.williams5	mary.williams5																																																																
Group (operators)	0.00	Ø	Ø	Ø	operators	operators																																																																
Group	Conflict Prior...	Description																																																																				
Group (admins)	0.00	Ø																																																																				
Group (Developers)	0.00	A group for devel																																																																				
Image: Ø																																																																						
Roles (1)																																																																						
Users (0)																																																																						
Group (guests)	0.00	Ø																																																																				
Group (mary.williams5)	0.00	Ø																																																																				
Group (operators)	0.00	Ø																																																																				
2																																																																						

3	<p>Click the Add button.</p> 
4	<p>The Selection Constellation will be displayed. Select the user that you wish by searching for it using the search field...</p> <p></p> <p>or selecting the User Query and selecting it from the list.</p> <p></p> <p>You can add as many users as you wish by using the Add and then Add and Finish when selecting the last user you wish to select.</p> <p></p>

5	<p>The user(s) selected will be assigned to the group.</p> <h3>Group's Properties</h3> <table border="1" style="width: 100%; border-collapse: collapse;"> <tbody> <tr> <td style="width: 15%;">Id</td><td>Developers</td></tr> <tr> <td>Conflict Priority</td><td>0.00</td></tr> <tr> <td>Description</td><td>A group for developers</td></tr> <tr> <td>Email</td><td>dev@email.com</td></tr> <tr> <td>Localized Name</td><td>Ø</td></tr> <tr> <td>Name</td><td>Developers</td></tr> <tr> <td colspan="2">– Image</td></tr> <tr> <td colspan="2">Ø</td></tr> <tr> <td colspan="2">– Roles</td></tr> <tr> <td colspan="2">• Role (cortex.role.a)</td></tr> <tr> <td colspan="2">– Users</td></tr> <tr> <td colspan="2">• cortex</td></tr> </tbody> </table>	Id	Developers	Conflict Priority	0.00	Description	A group for developers	Email	dev@email.com	Localized Name	Ø	Name	Developers	– Image		Ø		– Roles		• Role (cortex.role.a)		– Users		• cortex	
Id	Developers																								
Conflict Priority	0.00																								
Description	A group for developers																								
Email	dev@email.com																								
Localized Name	Ø																								
Name	Developers																								
– Image																									
Ø																									
– Roles																									
• Role (cortex.role.a)																									
– Users																									
• cortex																									
6	<p>Click Save to persist your changes.</p> <div style="text-align: center; margin-top: 10px;">  Upload  Undo  Redo  Save </div>																								

Attach Metadata from Extract

Attach Metadata from Extract

If you have model that is similar to another model, for example, if it is an older version or a version from a different installation instance of tribefire, it is possible to attach metadata which has previously been extracted from this model. After using the [Extract Meta Data](#) button, you will have an XML file containing of all the selected models metadata. You can use the Attach MetaData from extract button to use this metadata XML file to another model. Doing so will automatically add the saved metadata properties to another model.



Usage Example

Following the process from the [Extract Meta Data](#) documentation, you will have a metadata XML file which contains all the metadata from a model called personModel. There also exists a newer version of this model, which does not have any metadata to begin with. We will attach this metadata to the newer version of the model using the metadata XML file extracted from the older version of this model.

personModel

- baseType: Ø
- Entity Types (2)
 - GenericEntity
 - Person
 - Artifact Binding: Ø
 - Meta Data (0)
- Properties (6)
 - ID
 - firstName
 - Entity Type: Ø
 - Meta Data (0)
 - Type: Ø
 - secondName
 - dateOfBirth
 - birthplace
 - nationality
- Super Types (1)
- Enum Types Ø
- Meta Data Ø
- simpleTypes Ø

Uploading Metadata to tribefire

i After extracting the metadata, the resource file will automatically be available in the File link. You can download this resource to use in another instance of tribefire. The following steps are only necessary if you plan to do this. Otherwise, if you are attaching the extracted metadata on the **same** instance, you can ignore the following steps.

The first step to attaching metadata is to upload this XML file to your tribefire system. To do so, click on the files link found in the left-hand menu panel of tribefire Control Center. This will open the Files Query and show you all the files associated with this instance of tribefire.

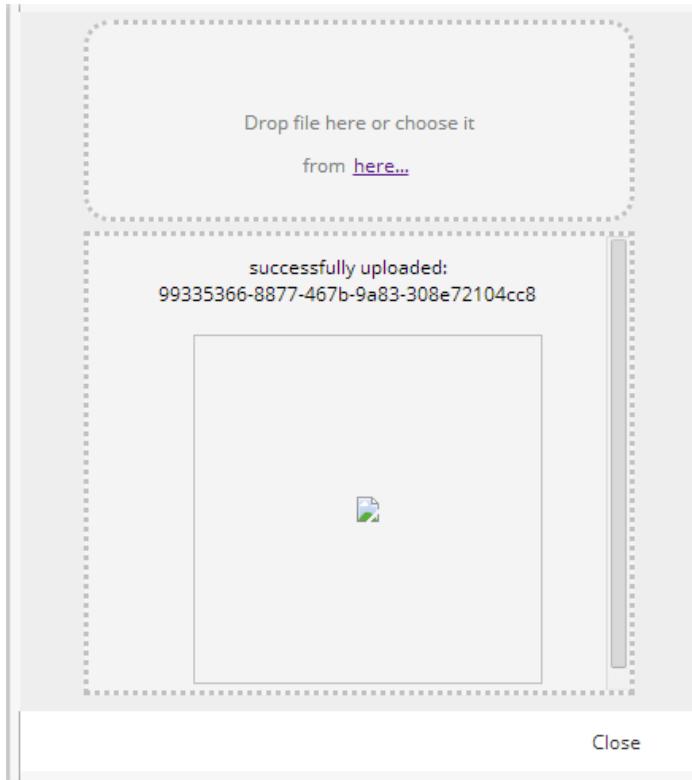
The screenshot shows the tribefire Control Center. At the top, there's a navigation bar with icons for Home, Back, Forward, and a search bar labeled 'Files Query'. Below the navigation bar is a sidebar with a 'Quick Access' section and a main menu. The main menu is organized into several categories: Smart Enterprise Information (Applications, Modeling, Virtualization), System (Protocol), and Resources (Files, Groups, Images, Types). The 'Files' option under 'Resources' is highlighted with a red box. The central area of the screen is titled 'Files Query' and contains a large empty text input field, also highlighted with a red box.

i To upload a file to tribefire, you can use any details panel that is display. Every instance of the details panel has the Upload option which is used to upload files and images. We use the Files Query for this example for the ease of understanding.

At the bottom right-hand corner of tribefire Control Center there are a series of buttons, including Upload. Click the Upload button to import a file to tribefire.



You can either drag the file over the import window or click the from here... link to import the file. Once imported you will receive a success message. Click Close to close the window.



Click the magnifying glass icon to refresh your search. Your newly uploaded file should be displayed.

Files Query

RawResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resolver URI	Width In Cm	Id	Created
▷ MetaDataExtract-personModel	14368	0.00	8693d36048f3024	application/xml	0	∅	0.00	RN_104	03/25/2014 13:02

Attaching Metadata

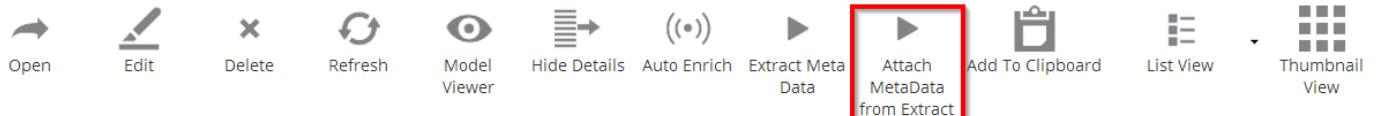
Select the model that you wish to attach metadata to...

System Models Query

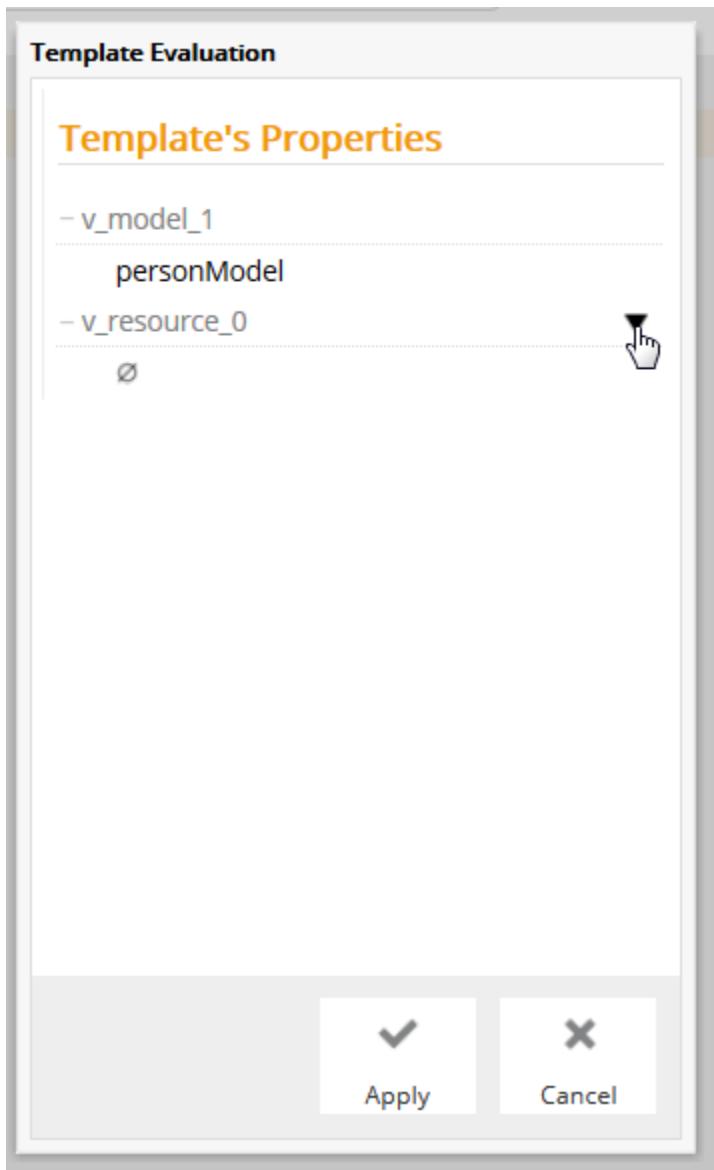
Model
▷ BasicDeploymentWorkbenchModel#1.0
▷ personModel
▷ BasicDeploymentModel#1.0
▷ WorkbenchModel#2.0

A mouse cursor is hovering over the 'personModel' entry. A tooltip labeled 'Model' is visible near the cursor.

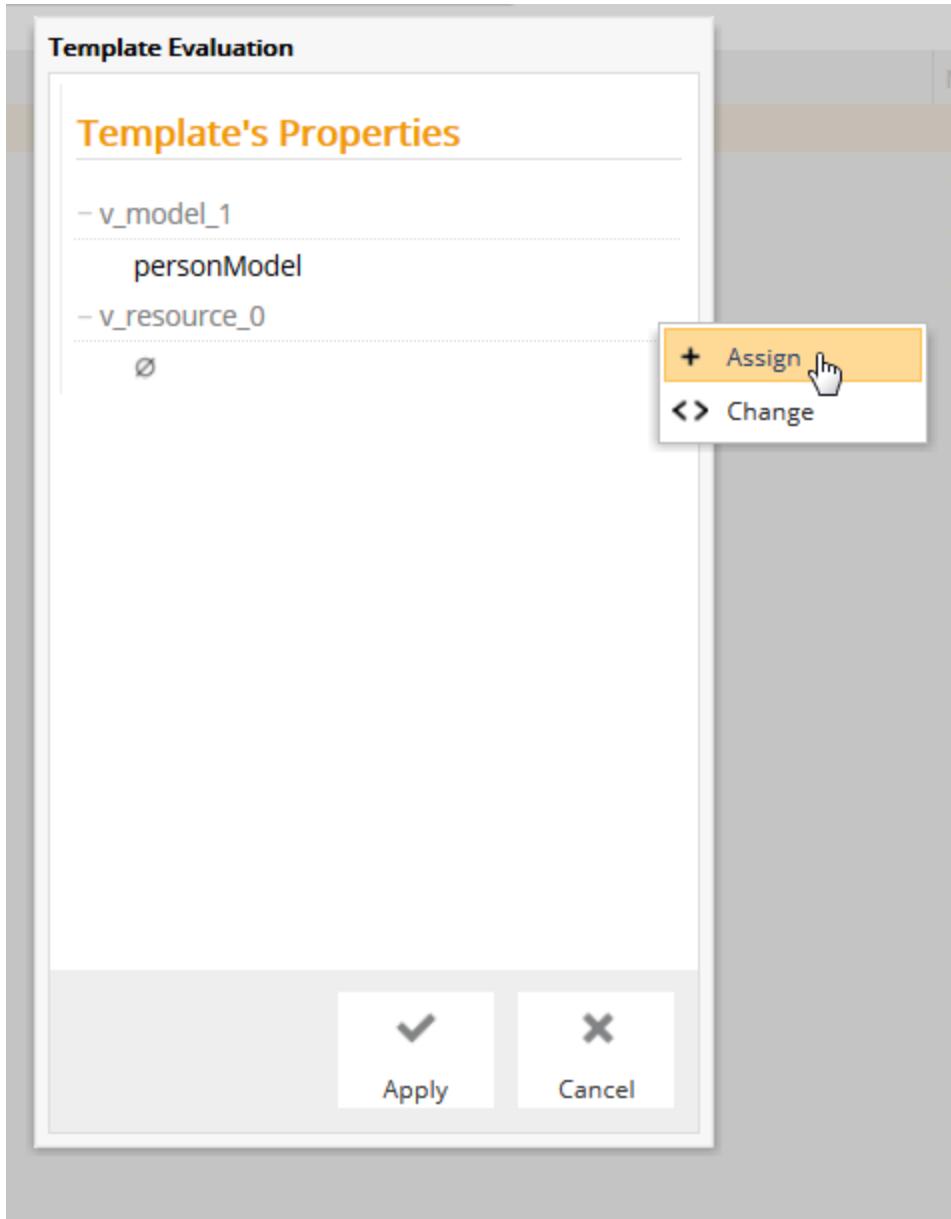
...and click select the Attach MetaData from Extract button



Mouse hover over the resource field and then click the context-menu icon.



Select Assign.

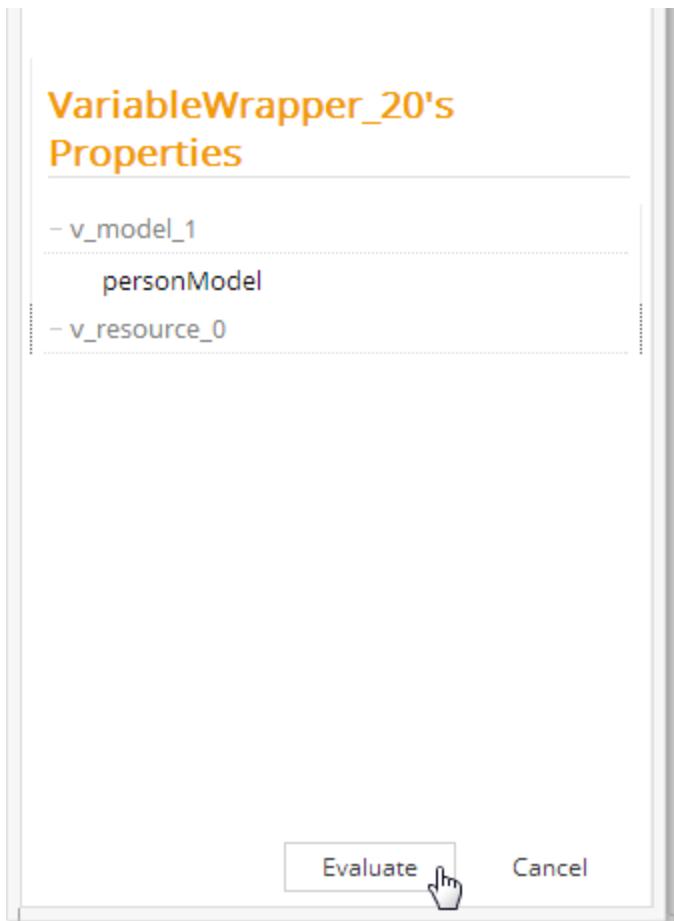


The Selection Constellation will appear. You can either search for the file we uploaded or you can use the Resource Query to select it from a list. Once you have selected the resource that you need, click Finish to complete the selection.

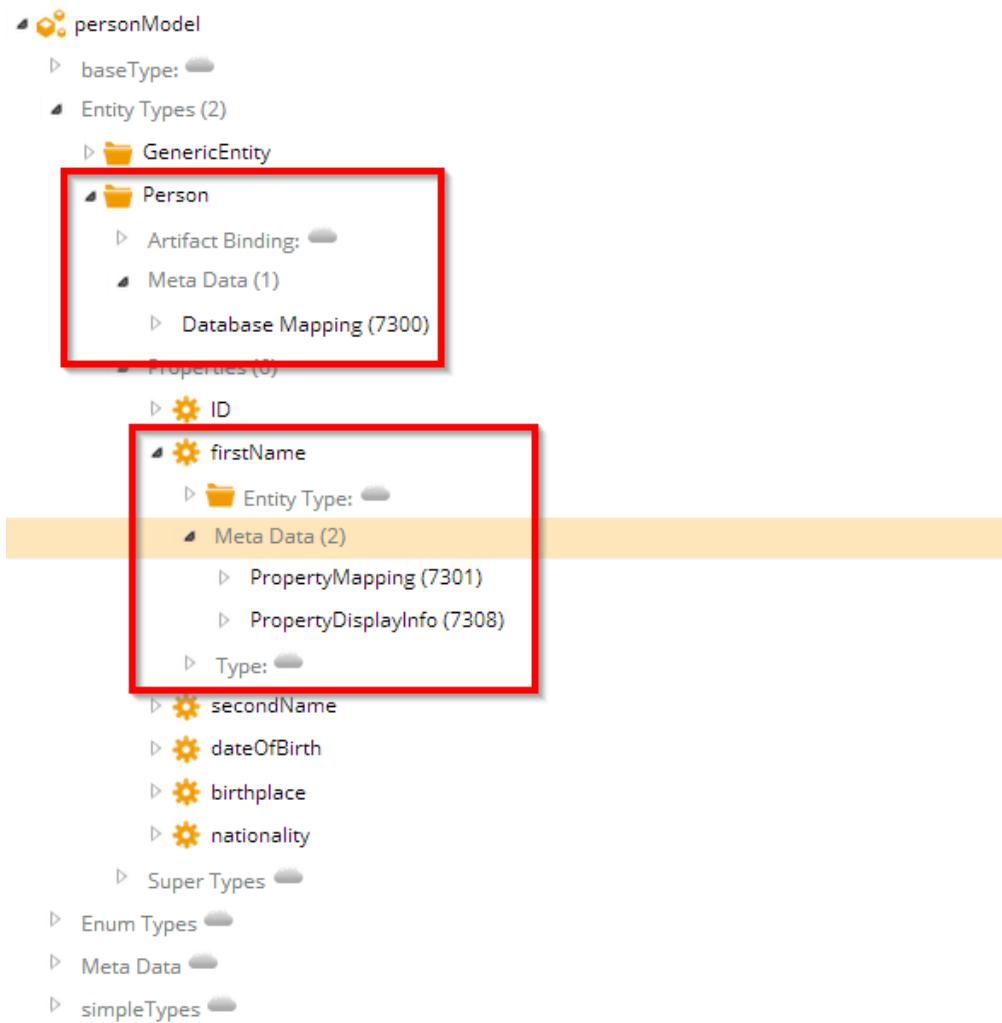
Resource Query

Resource	File Size	heightInCm	Md5	Mime Type	pageCount	resolverURI	widthInCm	Id	Created	Creator	info	Name
infoOrange16.png	1547	0.00	715eb1a7f179d86	image/png	1	∅	0.00	RN_103	10/29/2013 18:15	cortex		infoOrange16.png
MetaDataExtract-personModel@13	14368	0.00	8693d36049f3024	application/xml	0	∅	0.00	RN_104	03/25/2014 13:02	cortex		MetaDataExtract-per
fileStackOrangeSmall.png	1151	0.00	1a65a322706863	image/png	1	∅	0.00	RN_34	10/23/2013 14:13	csp		fileStackOrangeSmal
folderOrange.png	1118	0.00	b89383fcab34d23	image/png	1	∅	0.00	RN_35	10/23/2013 14:16	csp		folderOrange.png
settingsOrange.png	1293	0.00	adcb334e0eb2e51	image/png	1	∅	0.00	RN_36	10/23/2013 14:17	csp		settingsOrange.png
usersOrangeSmall.png	1384	0.00	4acfe57f3c267601	image/png	1	∅	0.00	RN_37	10/23/2013 15:50	csp		usersOrangeSmall.png

Finally select the Evaluate button to attach the metadata to your model



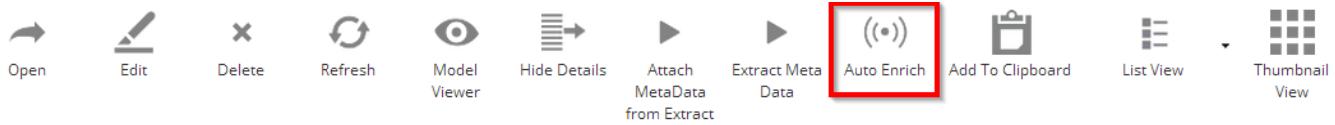
If successful you will receive a message confirming the action's success. Your metadata will now have been attached to your model.



Auto Enrich

Auto Enrich

Auto Enriching is the operation of automatically adding metadata to a model. This allows for a quick and simple way of enriching your model, and its types and properties, with information that better help describe the model. You can Enrich your models by selecting the model that you would like to enrich and clicking the Auto Enrich button in tribefire Expert Mode. It is possible to use the Auto Enrich feature on all models, regardless of their type.



Usage

Step #	Task
1	<p>In the following example, we have an Information Model called Sales Model. This model has many different types, each with their own properties. In this screenshot we can see that currently there are no metadata configured for this model.</p>
2	<p>Select the Model...</p> <p>...and then click Auto Enrich to add metadata to this model.</p>

3

If the action is completed successfully, you will receive a message confirming the action's success. You will see that metadata has automatically been added to your model.

4

This particular metadata property, regulates how the column should be shown in Explorer. Instead of using the properties name (firstName) it will now display the value of this metadata property (First Name)

PropertyDisplayInfo's Properties

Font Color Ø

– origin

Ø

General

Id 7136

Conflict Priority 0.00

Inheritance enabled

Name First Name

– Description

Ø

– Selector

Ø

Configuring and setting up a connection to MsSQL

Table of Contents

▼ Expand / collapse

- Introduction

Child pages

▼ Expand / collapse

- Downloading and Installing MsSQL
- Configure MsSQL
- Creating a MsSQL Connection
- Creating a MsSQL access
- Automatically create a new Model from your MsSQL database
- Query your new MsSQL Model in tribefire GME

Introduction

Using tribefire GME you can create a connection to a MsSQL database, allowing you to manipulate the data stored there. Once you have created a connection, you can import the database schema and automatically generate models from these tables, which gives you the opportunity to create, edit, delete entries in the database, as well as being able to execute queries from within tribefire GME. In addition to this, you can combine different accesses together to create a new access from disparate repositories, which will display a fully 360 degree picture of your data. This is done through the use of a Smart Enterprise Information Access.

 To make sure that you can properly connect to your MsSQL server, you must make sure that you have to correct drivers in your tribefire host's library directory.

It is recommended that you use the [MicrosoftJdbc4Driver](#) option when creating a connection. Therefore, you should place the [sqljdbc4.jar](#) driver in the directory:

```
tribefire Installation Directory/host/lib
```

Once you have added your driver, make sure that you restart your tribefire host.

There are several steps required to configure tribefire GME so that it can be connected to a MsSQL.

- Downloading and Installing MsSQL
- Configure MsSQL
- Create a MsSQL connection
- Create a MsSQL access
- Automatically create a new Model from your MsSQL database
- Query your new MsSQL Model in tribefire GME

Downloading and Installing MsSQL

Table of Contents

- ▼ Expand / Collapse
 - Introduction
 - Task

Introduction

Before we can create a connection to a MsSQL server, we must download and install an instance of it first.

If you have already have a version of MsSQL installed then you can skip this stage.

Task

Step #	Task
--------	------

1

Download MsSQL from Microsoft's website



Select Language: English

Download

Microsoft® SQL Server® 2012 Express is a powerful and reliable free data management system that delivers a rich and reliable data store for lightweight Web Sites and desktop applications.

Select the appropriate version (either 32- or 64-bit) and language.
Next, click Download to download the file.

64-Bit

Choose the download you want

File Name	Size
ENU\sql\EXPRESS_x64_ENU.exe	132.3 MB
<input checked="" type="checkbox"/> ENU\sql\EXPRESSADV_x64_ENU.exe	1.3 GB
ENU\sql\EXPRESSWT_x64_ENU.exe	669.9 MB
ENU\sql\LocalDB.MSI	33.0 MB
ENU\sql\ManagementStudio_x64_ENU.exe	600.2 MB
ENU\sql\EXPRESS_x86_ENU.exe	116.7 MB

Download Summary:

1. ENU\sql\EXPRESSADV_x64_ENU.exe

Total Size: 1.3 GB

Next**32-Bit**

Choose the download you want

File Name	Size
ENU\sql\EXPRESS_x86_ENU.exe	116.7 MB
ENU\sql\EXPRESSPR2_x86_ENU.exe	101.5 MB
<input checked="" type="checkbox"/> ENU\sql\EXPRESSADV_x86_ENU.exe	1.3 GB
ENU\sql\EXPRESSWT_x86_ENU.exe	706.1 MB
ENU\sql\LocalDB.MSI	27.8 MB
ENU\sql\ManagementStudio_x86_ENU.exe	614.9 MB

Download Summary:

1. ENU\sql\EXPRESSADV_x86_ENU.exe

Total Size: 1.3 GB

NextClick **Save File** to download the MsSQL database.

Opening SQLEXPRADV_x86_ENU.exe

You have chosen to open:

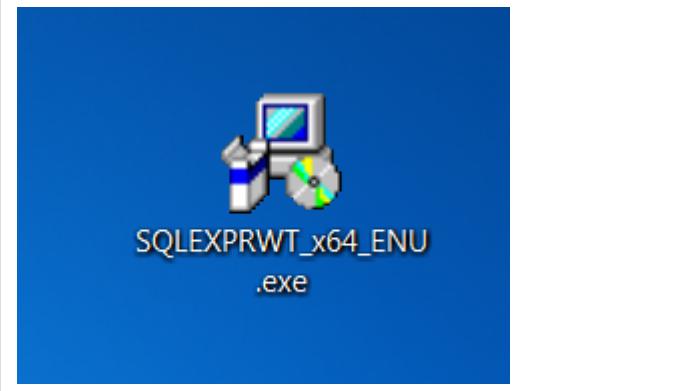
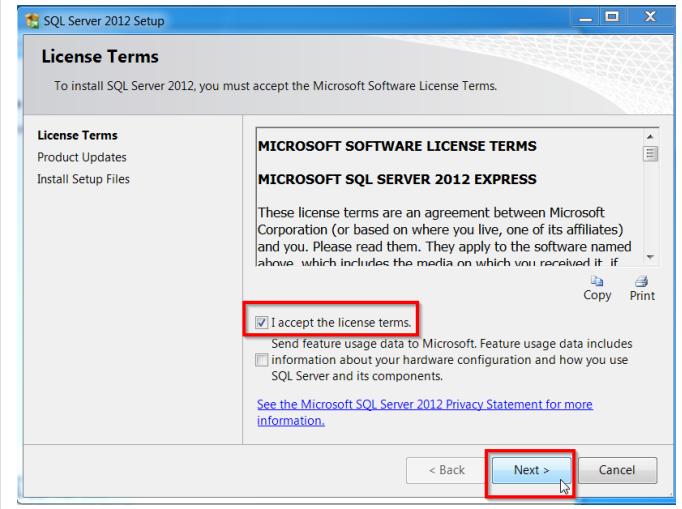
SQLEXPRADV_x86_ENU.exe

which is: Binary File (1,3 GB)

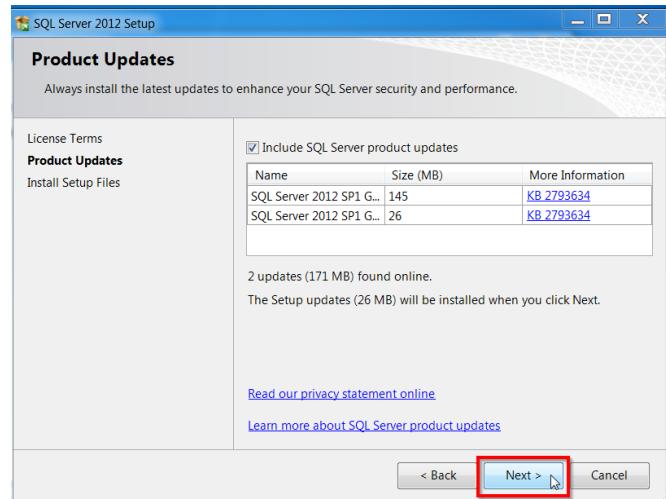
from: <http://download.microsoft.com>

Would you like to save this file?

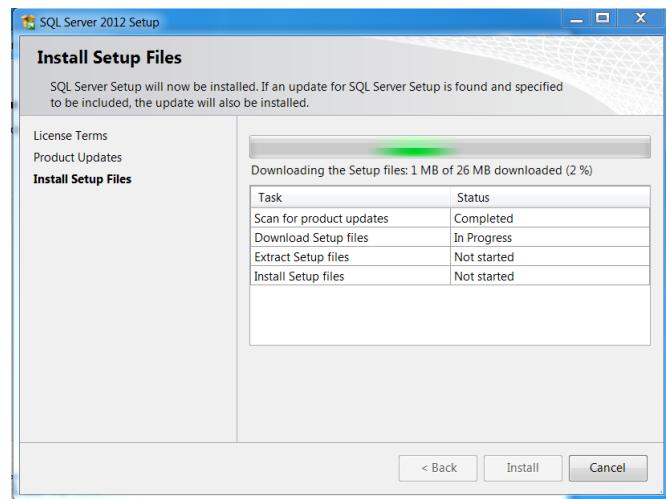
Save File**Cancel**

2	<p>Double Click the downloaded file to begin installation.</p> 
3	<p>After Installation begins, select the option: <i>New SQL Server stand-alone installation or add features to an existing installation.</i></p> 
4	<p>Check the <i>I accept the license terms</i> box and then click Next.</p> 

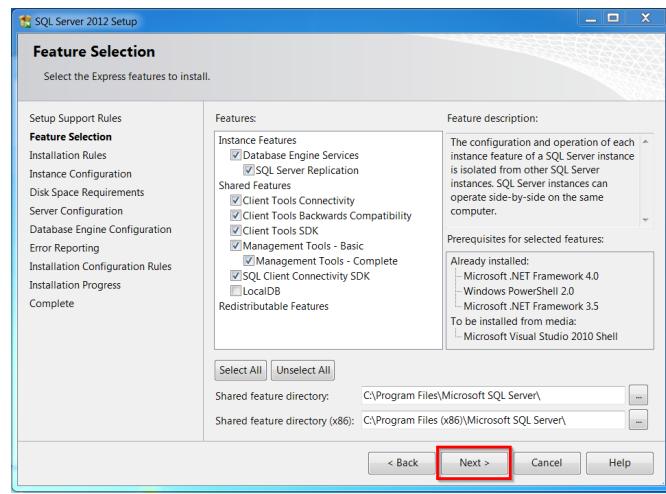
5

Click **Next** to install any updates.

The updates will then be installed.

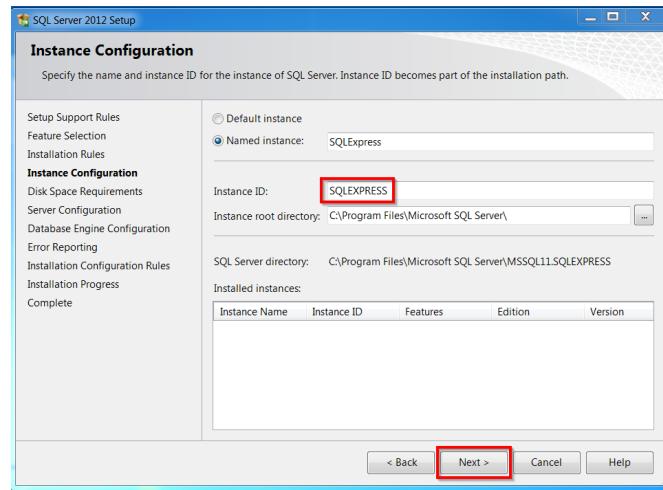


6

Click **Next** to select the features which should be installed. You can also decide the location to where MySQL should be installed.

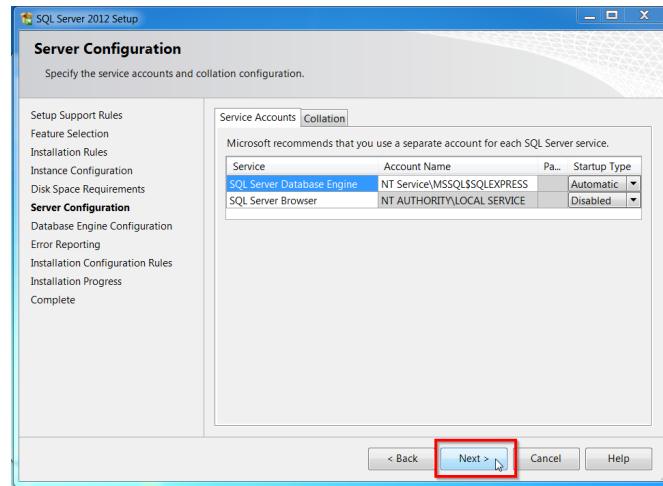
7

Before clicking **Next**, make a note of the Instance ID; this will be needed when configuring your MsSQL connection. Once you have done so, click **Next** to continue.



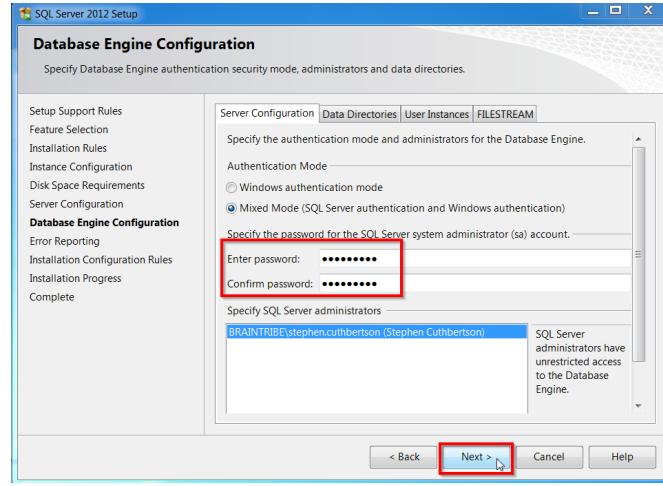
8

Click **Next** to continue.



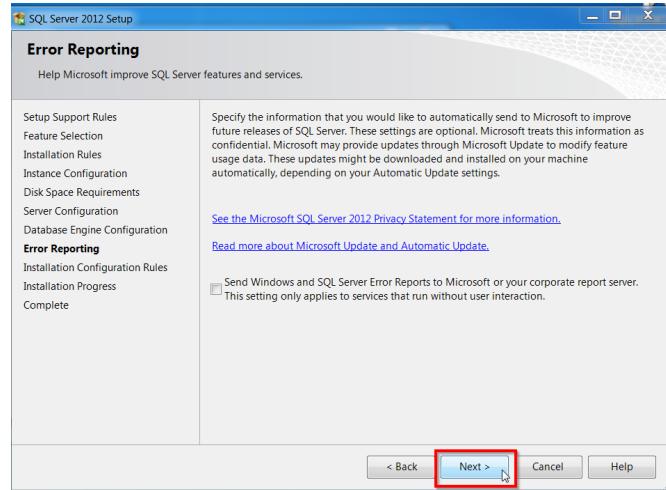
9

Enter a password, confirm it and then click Next. This password is used to login to MsSQL management studio.



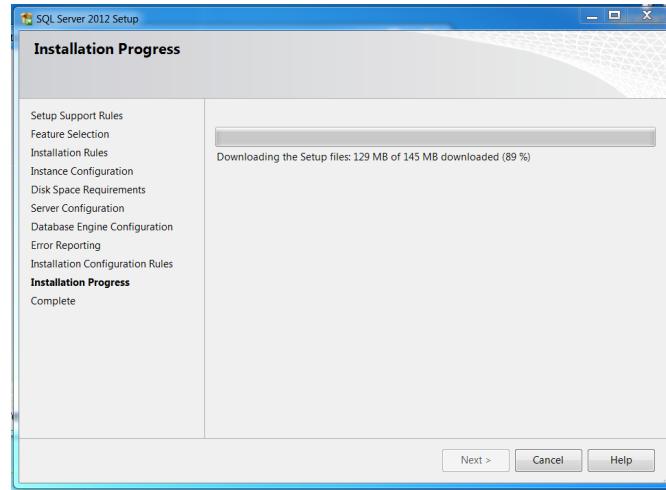
10

Click **Next** to begin the installation of MsSQL.



11

MsSQL will now be installed.



You have now successfully installed MsSQL. You can now configure it for usage

Configure MsSQL

Table of Contents

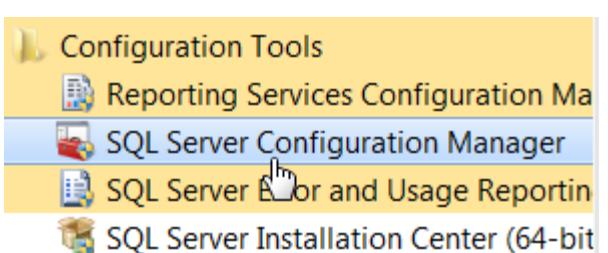
▼ Expand / collapse

- Introduction
- Configure TCP/IP Settings
- Create New MsSQL database
- Create a User Account

Introduction

You must configure MsSQL, so that tribefire GME can connect to it, including creating a user account, a database and some tables, as well as adding some data to these tables. In addition to this, you must also enable TCP/IP for MsSQL and set the open port through which tribefire can access the database.

Configure TCP/IP Settings

Step #	Task
1	<p>From Microsoft SQL Server 2012 in your start menu, select <i>Configuration Tools...</i></p>  <p>...and then select <i>SQL Server Configuration Manager</i>.</p> 
2	<p>The SQL Server Configuration Manger will then open.</p> <p>Select <i>SQL Server Network Configuration</i> to reveal the protocols for your new server instance.</p> 

3

Right Click on TCP/IP to display the context menu and click **Enable**.

Protocol Name	Status
Shared Mem...	Enabled
Named Pipes	Disabled
TCP/IP	Disabled

Enable
Disable
Properties
Help

You will be warned that these changes will not take effect until after you have restarted the MsSQL service. Click **OK** to confirm.



4

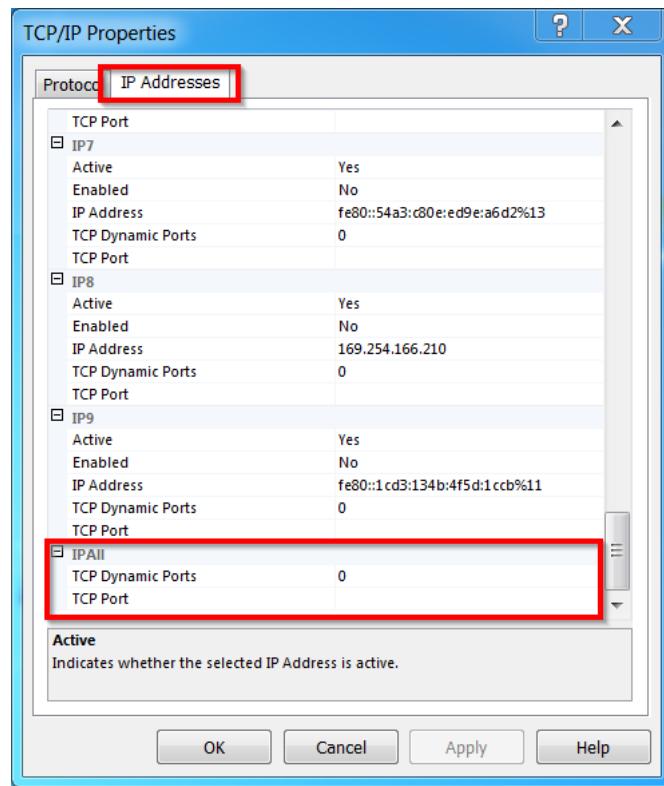
Right-Click again on the TCP/IP and this time, click **Properties**.

Protocol Name	Status
Shared Mem...	Enabled
Named Pipes	Disabled
TCP/IP	Enabled

Enable
Disable
Properties
Help

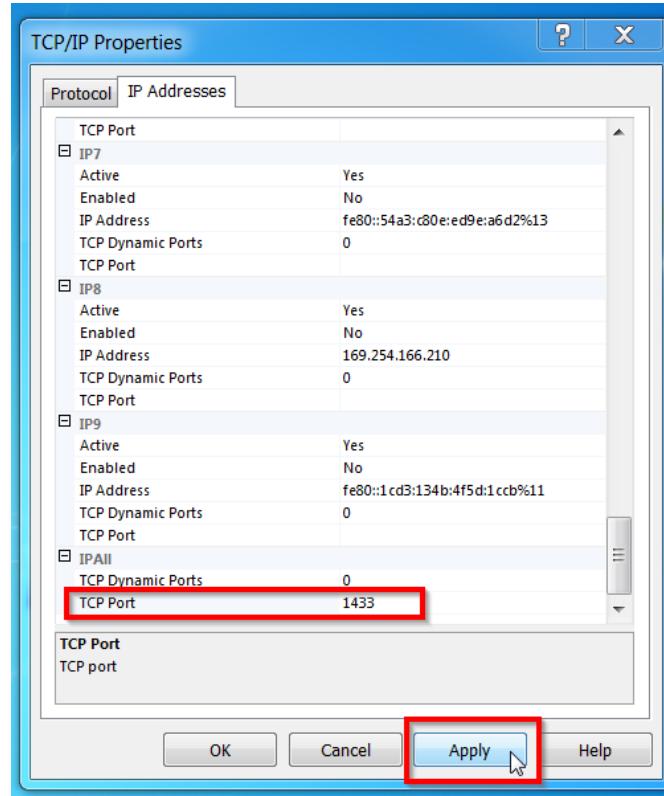
5

Click on the **IP Addresses** tab and scroll down till you find the section entitled **IPAll**.



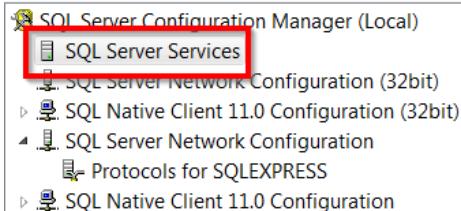
6

Enter the value **1433** to the property **TCP Port** and click **Apply** and then **OK**.



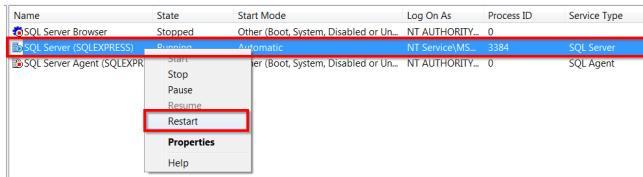
7

Select SQL Server Services from the left-hand panel, located under the heading SQL Server Configuration Manager.



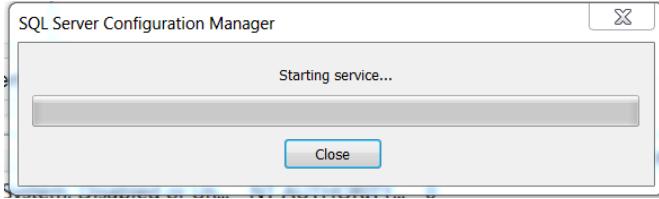
8

On the SQL Server service right click and select **Restart**.



9

The service will be restarted and your changes will be saved.



Create New MsSQL database

You must also create a database with some tables before you can connect to tribefire GME.

Step #	Task
1	<p>From your windows start menu, select Microsoft SQL Server 2012 and then <i>SQL Server Management Studio</i>.</p> <p>The screenshot shows the Microsoft SQL Server 2012 start menu. It includes options like 'Download Microsoft SQL Server Com...', 'Import and Export Data (32-bit)', 'Import and Export Data (64-bit)', 'SQL Server Management Studio' (which is highlighted with a red box), 'Analysis Services', 'Configuration Tools', 'Integration Services', and 'Performance Tools'.</p> <p>i For more information on the SQL Server Management Studio, please see Microsoft's official documentation</p>

2

Login into your MsSQL server.

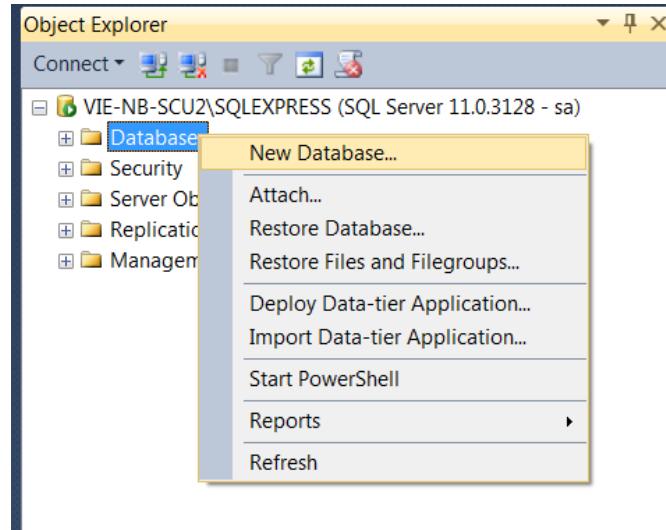
i To login to your SQL server, you should use the password you defined during the installation process and the user name **sa**.



3

The Microsoft SQL Server Management Studio will open. At the left-hand side of the studio is the Object Explorer.

Right click on the **Database** folder, found in the Object Explorer, and select **New Database**.

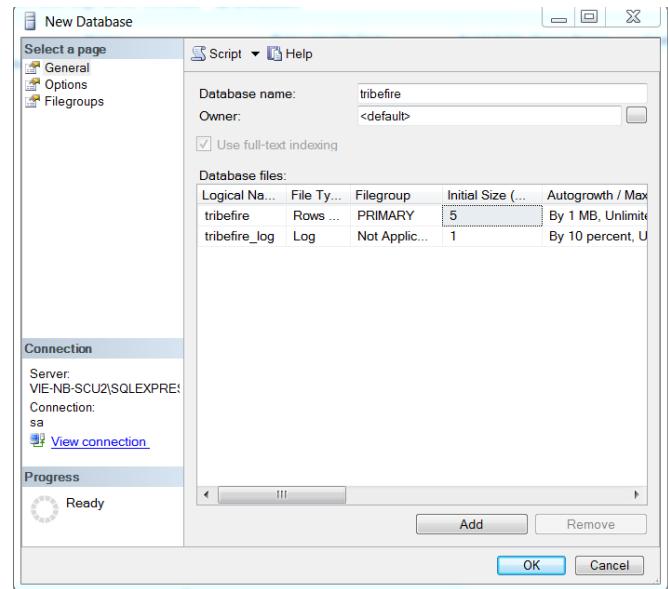


4

The New Database window will appear.

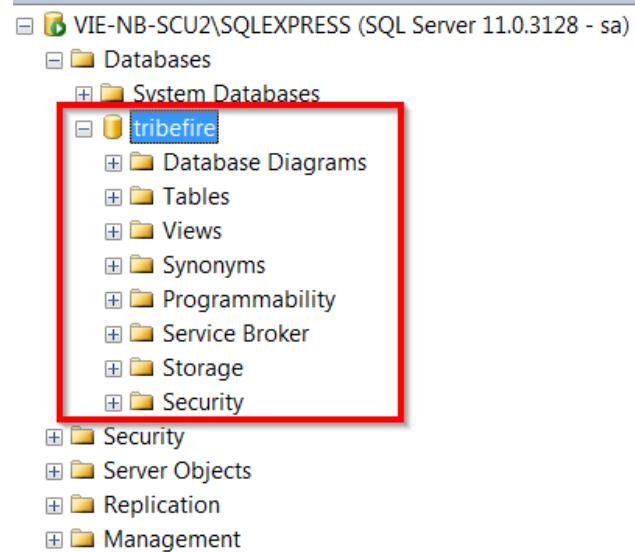
Enter the name of you would like your database to have in the Database Name field and click **OK**.

In this tutorial, the database will be named *tribefire*.



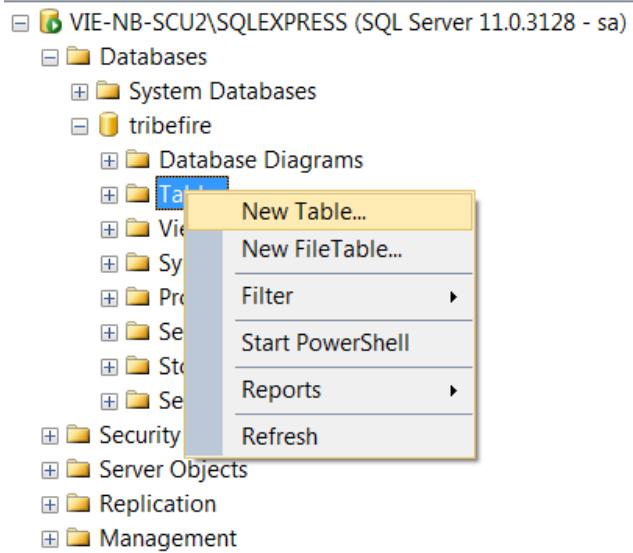
5

The new database will be created. We now have to create some tables.



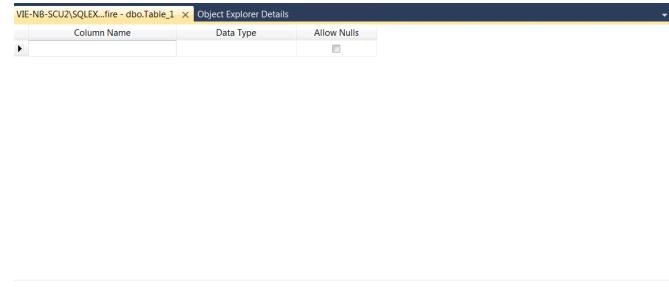
6

On the tables folder, right-click and select **New Table**.



7

The New Table editor will be displayed in the central panel of the Server Management Studio.



8

Normally, a database will have many tables with many columns, all representing a more complex modeling of data. However, for the purposes of this tutorial, we will only create one table with just a few columns.

The first column to define is an ID which will be used as a primary key.

Enter the name **ID** and the data type **int**.

Column Name	Data Type	Allow Nulls
ID	int	<input checked="" type="checkbox"/> <input type="checkbox"/>

Right Click on the **Column** to display a context-menu

Column Name	Data Type	Allow Nulls
ID	int	<input checked="" type="checkbox"/> <input type="checkbox"/>

Set Primary Key
Insert Column
Delete Column
Relationships...
Indexes/Keys...
Fulltext Index...
XML Indexes...
Check Constraints...
Spatial Indexes...
Generate Change Script...
Properties

then select the option *Set Primary Key*. The key icon next to Column Name indicates that the primary key has been set.

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/> <input checked="" type="checkbox"/>

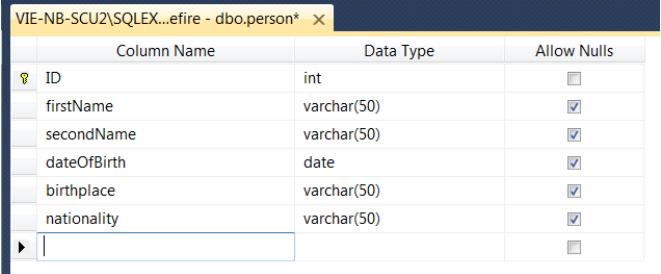
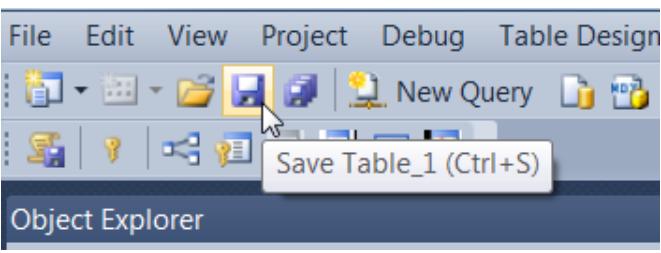
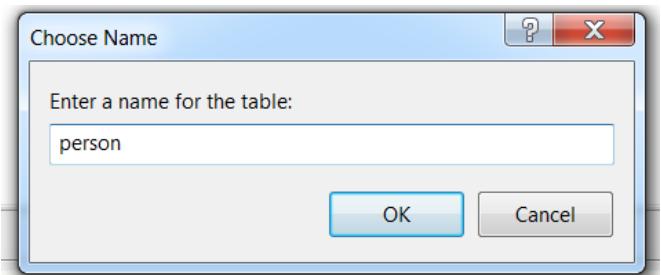
9

In the Column Properties section, located underneath the column definitions, scroll down till you find the value *Identity Specification*.

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/> <input checked="" type="checkbox"/>

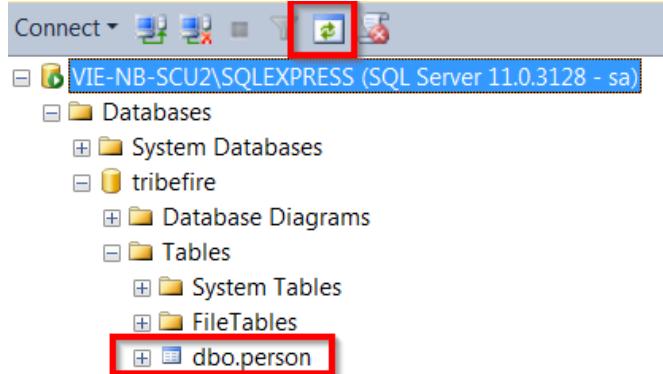
Column Properties
Table Designer

- Collation <database default>
- Computed Column Specification
- Condensed Data Type int
- Description
- Deterministic Yes
- DTS-published No
- Full-text Specification No
- Has Non-SQL Server Subscriber No
- Identity Specification No
- Indexable Yes
- Is Columnset No
- Is Sparse No
- Merge-published No
- Not For Replication No
- Replicated No
- RowGuid No
- Size 4

10	Double Click on Identity Specification to open it. 
11	Select Yes for the property (Is Identity). 
12	<p>You have now configured your primary key, and can now enter some more properties for your table.</p> <p>In the screenshot below, you will notice that there are two different data types used. For string properties, like first name, second name and so on, we use the data type varchar, which will be translated as a String in tribefire. The number in brackets specify the maximum length of that property.</p> 
13	<p>Click Save.</p>  <p>Give your table a name and click OK.</p> 

14

Click **Refresh** and you will see that your table has been added to your database.



Create a User Account

The last step when configuring MsSQL in preparation for a tribefire GME connection is to create a valid user account that GME can use to access this database.

Step #	Task
1	<p>Select Microsoft SQL Server 2012 from your windows start menu and then <i>SQL Server Management Studio</i>.</p>  <p>i For more information on the SQL Server Management Studio, please see Microsoft's official documentation</p>

2

Login into your MsSQL server.

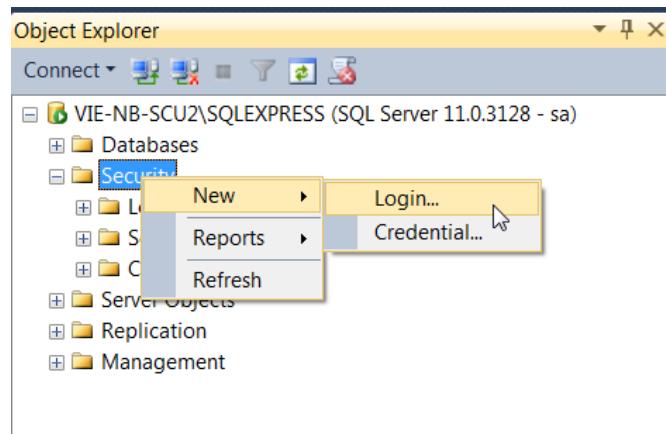
i To login to your SQL server, you should use the password you defined during the installation process and the user name **sa**



3

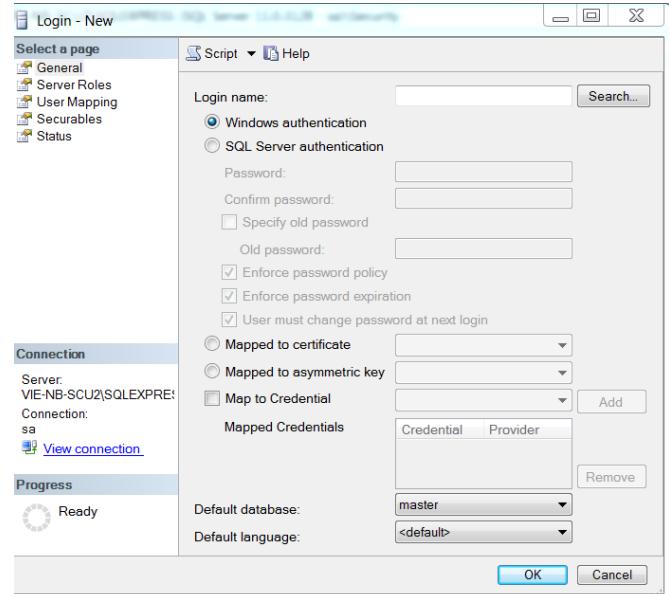
The Microsoft SQL Server Management Studio will open.

At the left-hand side of the studio is the Object Explorer. Right-click on the **Security** folder, select **New** and then **Login**.



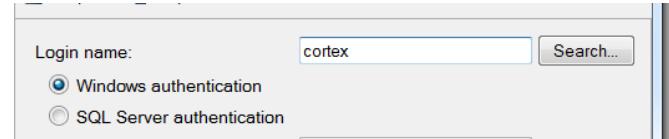
4

The New Login window will appear.

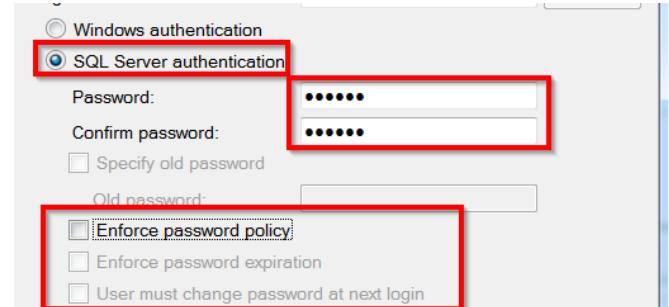


5

Give the new user a name.



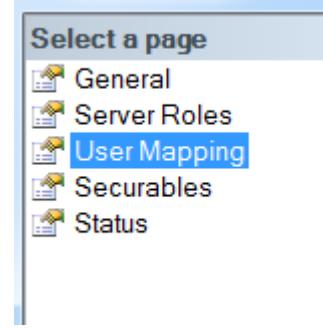
6

Select the SQL Server authentication option and give the new user a password. Also deselect the **Enforce password policy** option.

! Take a note of the user name and password that you have entered here. They will be required when [creating a new Connector](#).

7

Select the User Mapping page.



8

Select the tribefire database and then give your new user the roles: db owner and public.

Users mapped to this login:

Map	Database	User	Default Schema
<input type="checkbox"/>	master		
<input type="checkbox"/>	model		
<input type="checkbox"/>	msdb		
<input type="checkbox"/>	tempdb		
<input checked="" type="checkbox"/>	tribefire	cortex	

Guest account enabled for: tribefire

Database role membership for: tribefire

<input type="checkbox"/>	db accessadmin
<input type="checkbox"/>	db backupoperator
<input type="checkbox"/>	db datareader
<input type="checkbox"/>	db datawriter
<input type="checkbox"/>	db ddladmin
<input type="checkbox"/>	db denvdatareader
<input type="checkbox"/>	db denydatawriter
<input checked="" type="checkbox"/>	db owner
<input type="checkbox"/>	db securityadmin
<input checked="" type="checkbox"/>	public

OK **Cancel**

9

Select the Server Roles page.

Select a page

- General
- Server Roles**
- User Mapping
- Securables
- Status

10

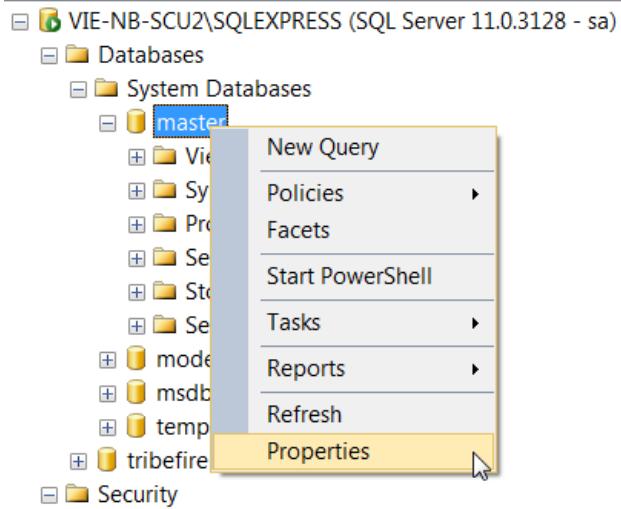
Give your new user the role *public* and click **OK** to save the new user.

Server roles:

<input type="checkbox"/>	bulkadmin
<input type="checkbox"/>	dbcreator
<input type="checkbox"/>	diskadmin
<input type="checkbox"/>	processadmin
<input checked="" type="checkbox"/>	public
<input type="checkbox"/>	securityadmin
<input type="checkbox"/>	serveradmin
<input type="checkbox"/>	setupadmin
<input type="checkbox"/>	sysadmin

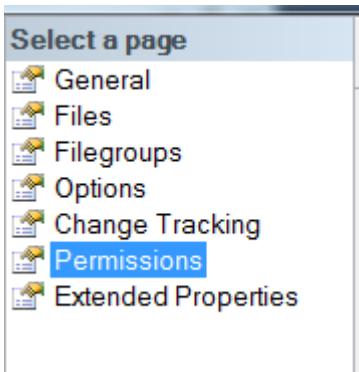
11

Open the System Databases and then right-click on **master** to open the context menu. Select **Properties**.



12

The Database properties window for master will appear. Select the page *Permissions*.



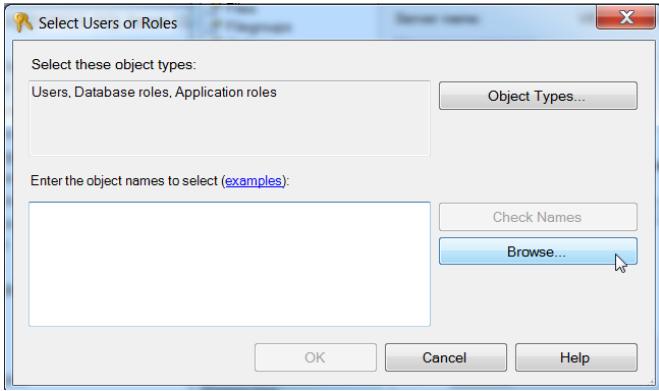
13

Click the **Search** button.

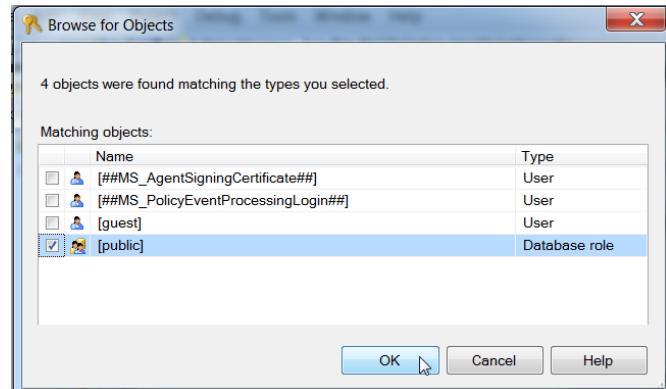


14

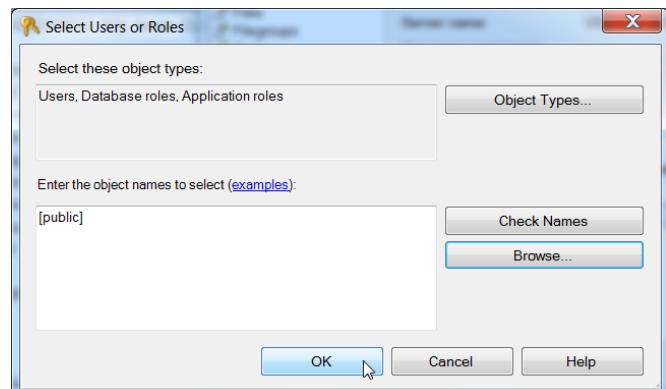
The Select Users or Roles window will appear. Click the browse button to search for users.



15

Select the role *Public* and then click **OK**.

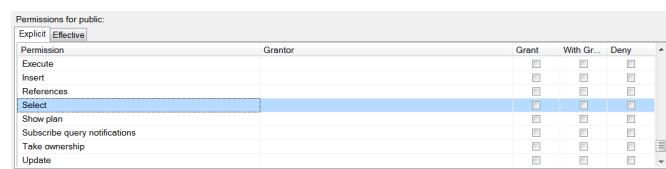
16

Click **OK**

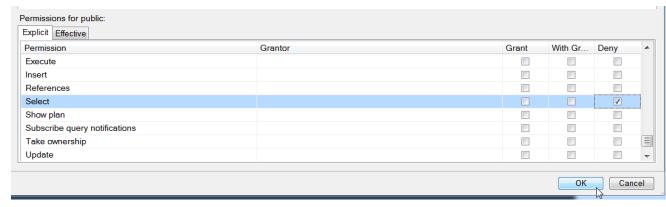
17

Select the role *public*.

18

In permissions scroll down till you find the entry **Select**.

19

Check the **Deny** box and then click **OK**

You have now successfully configured MsSQL in preparation for a tribefire connection. You can now create a connector.

Creating a MsSQL Connection

Table of Contents

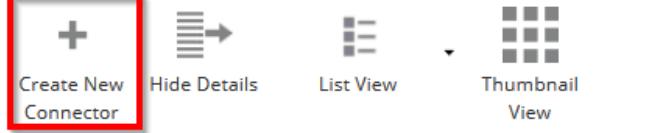
- ▼ Expand / collapse
 - Introduction
 - Task

Introduction

This step shows you how to create a connection that give tribefire GME access to the MsSQL database server. The connection functions by defining the location of the database server and the user account GME should use to gain access to the server, as well as the database which should be used.

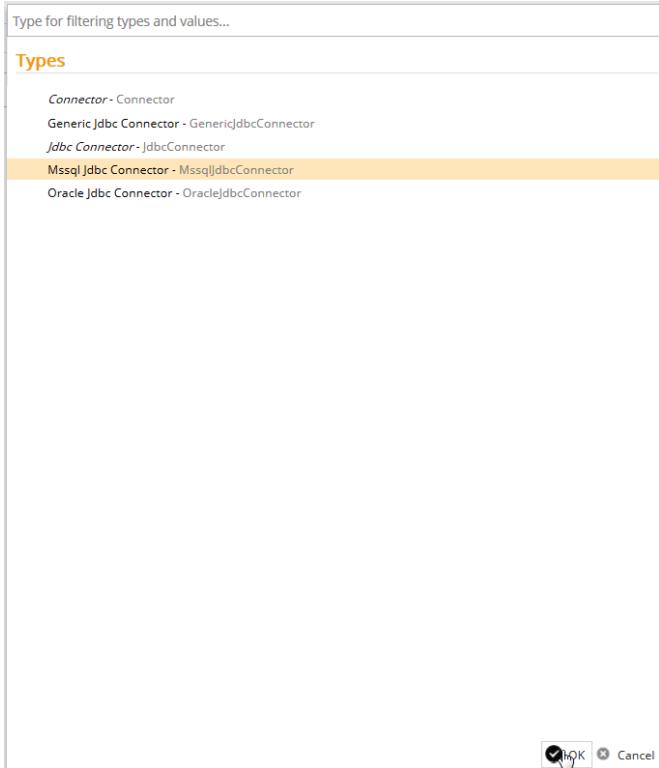
Once connected GME can then be used to create prototypes of the tables found in the database in the form of a db schema. In the next step, we will use this db schema to create a new Model in tribefire which can then be used to access the data found in these tables.

Task

Step #	Task
1	<p>Login to tribefire GME using your valid credentials.</p> 
2	<p>Click the <i>Connections</i> icon.</p> 
3	<p>Click the Create New Connector from the buttons panel.</p> <p>If there are any connections highlighted, you will have to deselect them before you can create a new connector; to deselect the connector, hold down CTRL and click the highlighted item.</p> 

4

The New Connector window will appear. Select the MsSQLJdbcConnector option and click **OK**.



5

Fill in the appropriate details and click **Apply**.



The main information required

Property	Description	Example Value
Database	The name of the database inside MsSQL that should be accessed.	tribefire
Driver	The name of the driver used to connect to the database.	MicrosoftJdbc4 Driver
Instance	The name of the MsSQL service instance that should be accessed.	SQLEXpress
Version	The version of MsSQL being used.	SqlServer2012
Host	The location of the MsSQL server. If running on the same computer, the value will be localhost. Otherwise, it will be the IP address of the computer on which MsSQL is installed.	localhost
Port	The open port through which MsSQL will be accessed.	1433
User	The user name of the account which has permissions to access and manipulate database, as configured in the previous step.	cortex
Password	The password of the account which has permissions to access and manipulate database, as configured in the previous step.	cortex

		<p>With Host</p> <table border="1"> <tr><td>Host</td><td>localhost</td></tr> <tr><td>Port</td><td>1433</td></tr> </table> <p>With Account</p> <table border="1"> <tr><td>Password</td><td>cortex</td></tr> <tr><td>User</td><td>cortex</td></tr> </table> <p>Mssql Jdbc Connector</p> <table border="1"> <tr><td>Database</td><td>tribefire</td></tr> <tr><td>Driver</td><td>MicrosoftJdbcDriver</td></tr> <tr><td>Instance</td><td>SQLEXPRESS</td></tr> <tr><td>Version</td><td>SqlServer2012</td></tr> </table> <p>Deployable</p> <table border="1"> <tr><td>Deployment State</td><td>∅</td></tr> <tr><td>Description</td><td>∅</td></tr> <tr><td>External Id</td><td>MsSQL.Connection</td></tr> <tr><td>Hardwired</td><td>□</td></tr> <tr><td>Name</td><td>MsSQL.Connection</td></tr> </table>	Host	localhost	Port	1433	Password	cortex	User	cortex	Database	tribefire	Driver	MicrosoftJdbcDriver	Instance	SQLEXPRESS	Version	SqlServer2012	Deployment State	∅	Description	∅	External Id	MsSQL.Connection	Hardwired	□	Name	MsSQL.Connection
Host	localhost																											
Port	1433																											
Password	cortex																											
User	cortex																											
Database	tribefire																											
Driver	MicrosoftJdbcDriver																											
Instance	SQLEXPRESS																											
Version	SqlServer2012																											
Deployment State	∅																											
Description	∅																											
External Id	MsSQL.Connection																											
Hardwired	□																											
Name	MsSQL.Connection																											
		You must also enter a name and external Id for your connection.																										
6		<p>Click Save to persist your new Connector in tribefire.</p> 																										
7		<p>Deploy your connection by selecting Deploy from the buttons panel.</p>  <p>If successful, you will receive the following message:</p> <p>saved successful ✘</p>																										
8		<p>You can test the connection to make sure you have configured the connector properly. To do so, click the Test Connection from the buttons panel.</p>  <p>If successful, you will receive the following message.</p> <p>Successfully tested connection! ✘</p>																										

9

Import the tables from the database by selecting the **Update DB Schema** button.



A new Schema will be created, this contains all tables found the database.

Standard Identifiable Entity

Id 8

Mssql Jdbc Connector

Database	tribefire
Driver	MicrosoftJdbc4Driver
Instance	SQLEXPRESS
Version	SqlServer2012

Jdbc Connector

- Db Schemas
- Db Schema (12)

Deployable

Deployment State	deployed
Description	Ø
External Id	MsSQL.Connection
Hardwired	<input type="checkbox"/>
Name	MsSQL.Connection

10

Click on the schema to display its properties.

Status

Deployment State

deployed

MssqlJdbcConnector's Properties

Database

tribefire

Driver

MicrosoftJdbc4Driver

Instance

SQLEXPRESS

Version

SqlServer2012

– Db Schemas

- dbo

– Properties

∅

Connection

Host

localhost

Port

1433

Account

User

cortex

Password

General

Id

33

External Id

MsSQL.Connection

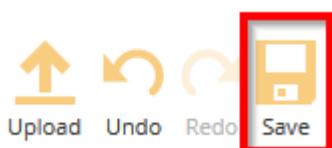
Name

MsSQL.Connection

– Description

∅

11

Click **Save** to persist the changes

You have now successfully configured a MsSQL Connection. The next step is to create a Hibernate Access based on this connection

Creating a MsSQL access

Table of Contents

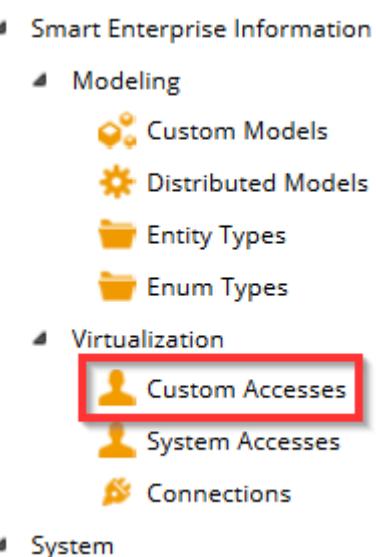
- ▼ Expand / collapse
 - Introduction
 - Task

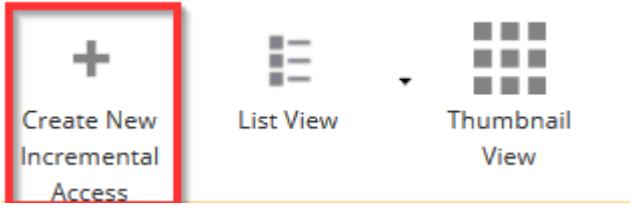
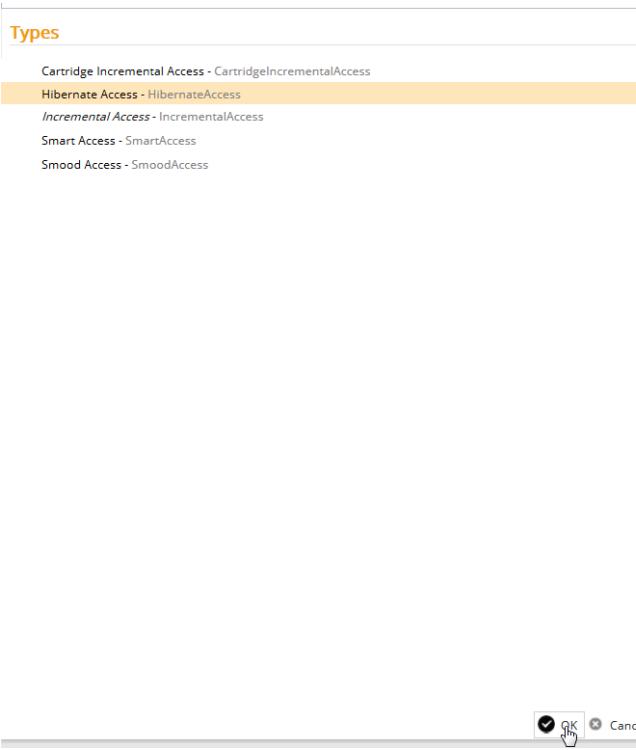
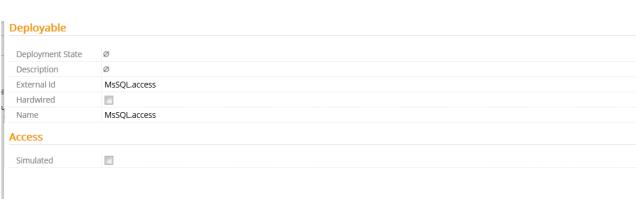
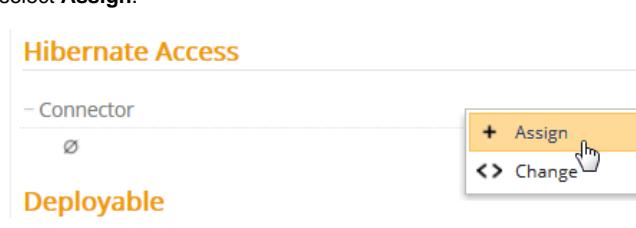
Introduction

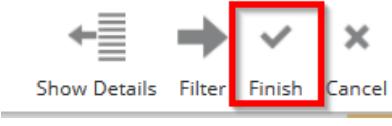
After creating a connection to your MsSQL database, the next step is to create an access that will make use of the rough types ([created by the Create Db Schema button](#)). The access converts these rough types into entities and creates a new model in tribefire GME based on this data.

In tribefire GME there is no specific MsSQL access. Instead, there is a standard integration access, called Hibernate Access, which is the standard access used to connect to a jdbc database.

Task

Step #	Task
1	<p>Login to tribefire using valid credentials.</p> 
2	<p>Click the Custom Accesses link found in the left-hand menu panel of Control Center.</p>  <ul style="list-style-type: none"> ▲ Smart Enterprise Information ▲ Modeling <ul style="list-style-type: none"> ◆ Custom Models ◆ Distributed Models ◆ Entity Types ◆ Enum Types ▲ Virtualization <ul style="list-style-type: none"> ◆ Custom Accesses ◆ System Accesses ◆ Connections ▲ System

3	<p>Click the Create New Incremental Access from the buttons panel.</p> <p>If there is an access highlighted, you will have to deselect it. To do so, hold down CTRL and then click the highlighted item.</p> 
4	<p>The New Access window will appear. Select the Hibernate Access and click OK.</p> 
5	<p>Enter a name, External Id, and, if you wish, a Description. Once you have entered your information click Apply.</p> 
6	<p>The next step is to add the correct connector to your integration access. To do so, either double click on the Connection property or hover over it and select the context-menu icon that appears. Then select Assign.</p> 

7	<p>Select the connector that we configured in the previous step...</p>  <p>...and click Finish.</p> 
8	<p>The Connector will be added to your access.</p> <h3>Hibernate Access</h3> <ul style="list-style-type: none"> – Connector <p>MsSQL.Connection (MsSQL.Connection)</p>
9	<p>Click Save to persist this access in tribefire GME.</p> 

You have now correctly configured an Hibernate Access. The next step shows you how to generate a model from the connection and how to deploy it.

Automatically create a new Model from your MsSQL database

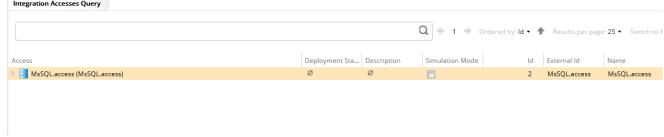
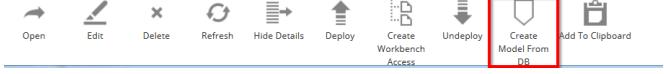
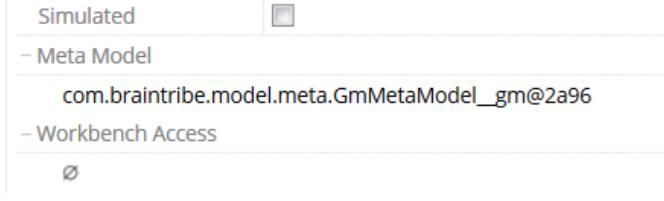
Table of Contents

- ▼ Expand / collapse
 - Introduction
 - Task

Introduction

After you have created an access, the next step is to use this access to create a new model in tribefire based on the rough types imported from your database. This will automatically generate the new entity types and properties needed for tribefire to manipulate and query data.

Task

Step #	Task
1	Select the access that you configured in the previous step. 
2	Click the Create Model From DB from the buttons panel. 
3	A new meta model will be created. Click it to open the model for editing. 
4	When first created the new meta model appears empty. 
5	Click the refresh button to complete the model's creation. 

6

The model's entity types will be created and displayed under the field entitled **Entity Types**.

Gm Meta Model

<i>Id</i>	14
-----------	----

Name	Ø
------	---

- Artifact Binding

Ø

- Base Type

Ø

- Entity Types

- GenericEntity
- Person

- Enum Types

Ø

- Meta Data

Ø

- Simple Types

Ø

7

The last step is to give your model a new name.

- Name

MsSQLPersonModel

8

Click **Save** to persist this model in tribefire.



9

The last step is to deploy the model. Select your hibernate access (in screenshot below, MsSQL.access) option from the tabs at the top of the assembly panel (or simply reopen the Integration Accesses link)...

Integration Accesses Query	MsSQLAccess (MsSQL.access)	Model	Entity Types	Deployment Status	Description	Simulation Mode	Id	External Id	Name
HibernateAccess	MsSQLAccess (MsSQL.access)			0	0	0	2	MsSQLAccess	MsSQL.access

...and select **Deploy** from the buttons panel.



10

Your access has been deployed.

	HibernateAccess	Deployment Sta...	Description	Simulation Mode
	MySQLAccess (MySQLAccess)	deployed	Ø	<input type="checkbox"/>

You have now successfully generated a model automatically from a db schema and deployed it. The next step is to query this access and display the results in tribefire GME.

Query your new MsSQL Model in tribefire GME

Table of Contents

▼ Expand / collapse

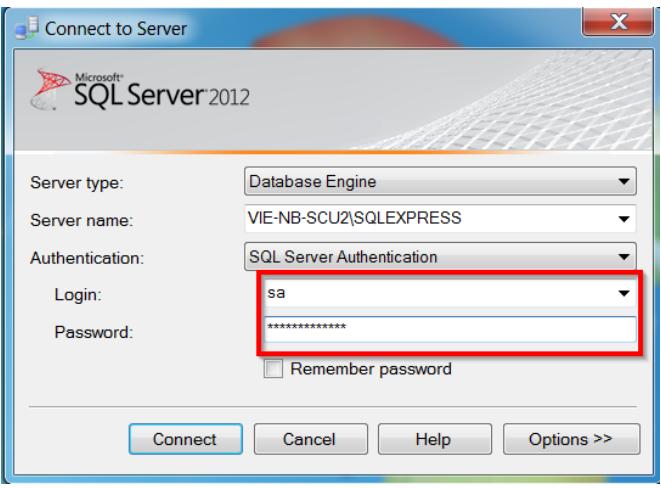
- Introduction
- Creating data in MsSQL
- Viewing Data in tribefire GME
- Add Data Using tribefire Explorer

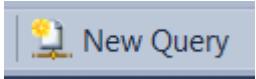
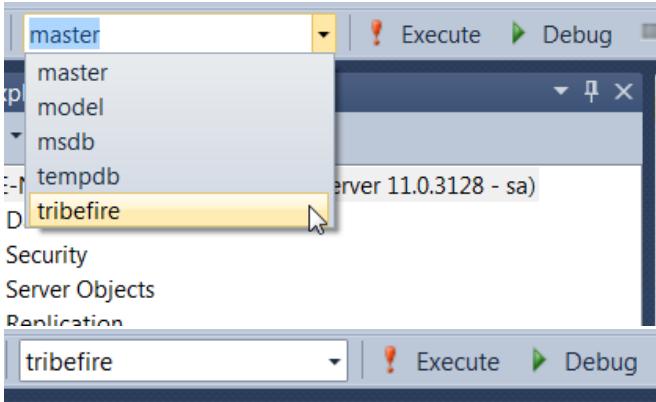
Introduction

After creating and deploying an access, you can now use it to view the data within tribefire. The access allows you to view entity types and data without any extra configuration. However, since we some data to query first, we need to add some using MsSQL—adding data using tribefire Explorer is shown at the end of this guide.

Creating data in MsSQL

Most actual databases are much more complex and contain much more data than shown here. We will only add a few entries into the database, enough to show you tribefire's functionality when connecting to tribefire.

Step #	Task
1	<p>From your Start Menu, select Microsoft SQL Server 2012 and then SQL Server Management Studio.</p>  <p>The screenshot shows the Windows Start Menu with the Microsoft SQL Server 2012 folder expanded. Inside, there are several options: 'Download Microsoft SQL Server Com...', 'Import and Export Data (32-bit)', 'Import and Export Data (64-bit)', 'SQL Server Management Studio' (which is highlighted with a red box), 'Analysis Services', 'Configuration Tools', 'Integration Services', and 'Performance Tools'.</p>
2	<p>Login with the user name sa and the password you defined during installation.</p>  <p>The screenshot shows the 'Connect to Server' dialog box. It has fields for 'Server type:' (set to 'Database Engine'), 'Server name:' (set to 'VIE-NB-SCU2\SQLEXPRESS'), 'Authentication:' (set to 'SQL Server Authentication'), 'Login:' (set to 'sa'), and 'Password:' (redacted). A red box highlights the 'sa' login field. At the bottom are 'Connect', 'Cancel', 'Help', and 'Options >' buttons.</p>

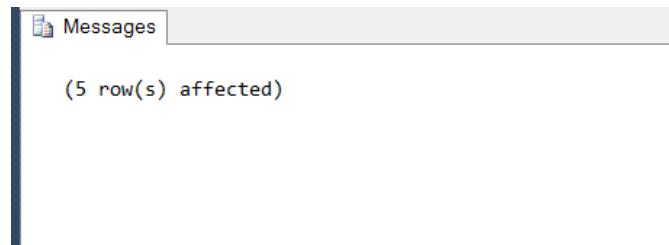
3	<p>From the toolbar select the option <i>NewQuery</i>.</p> 
4	<p>Underneath the toolbar is a drop-down list. This allows you to select on which database this query should be based. Select the database you created during the configuration of MsSQL.</p> 
5	<p>You can now enter a query which allows you to insert data to your database.</p> <pre>INSERT into dbo.person (firstName,secondName,dateOfBirth,birthplace,nationality) VALUES ('Robert', 'Smith', '1986-08-23','Manchester','British'), ('Lisa', 'Robertson', '1984-09-26','London', 'British'), ('Hans', 'Hueber', '1979-04-30','Vienna', 'Austrian'), ('Maria', 'Goesser', '1990-04-12','Graz', 'Austrian'), ('Peter', 'Stubbs', '1992-03-18','Detroit', 'American');</pre> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>If you created the same table during the configuration of MsSQL, you can use copy and paste from the example below.</p> <p>Click here to expand...</p> <pre>INSERT into dbo.person (firstName,secondName,dateOfBirth,birthplace,nationality) VALUES ('Robert', 'Smith', '1986-08-23','Manchester','British'), ('Lisa', 'Robertson','1984-09-26','London', 'British'), ('Hans', 'Hueber', '1979-04-30','Vienna', 'Austrian'), ('Maria', 'Goesser','1990-04-12','Graz','Austrian'), ('Peter', 'Stubbs', '1992-03-18','Detroit','American');</pre> </div>

6

Click the Execute button found in the toolbar and your values will be entered.

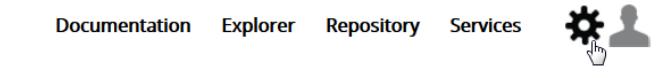
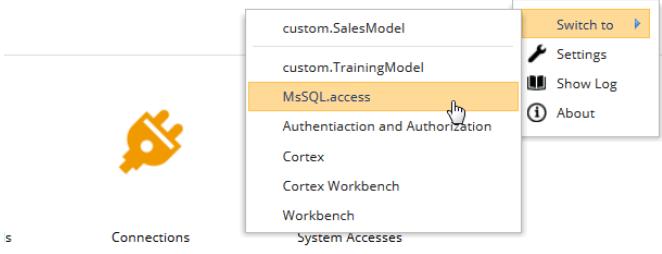


...if successful, you will receive the following message



Viewing Data in tribefire GME

You can now view the data you have just entered by using your access created in the last step.

Step #	Task
1	<p>Login into tribefire with your credentials.</p> 
2	<p>At the top right of the tribefire Control Center there is a cog icon which allows you to select between your deployed accesses. Click on the icon.</p> 
3	<p>From the drop-down list, select the access you created.</p> 

4

A new tab will be opened in your browser.

5

Your integration access will open. At the left-hand side of your access, there will be a list of all available entity types as defined by the database tables you are connected to. Select the entity type which you added data to in the previously.

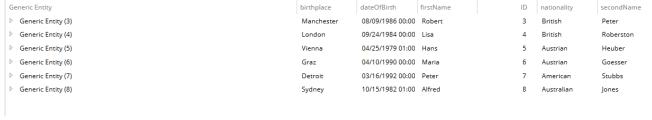
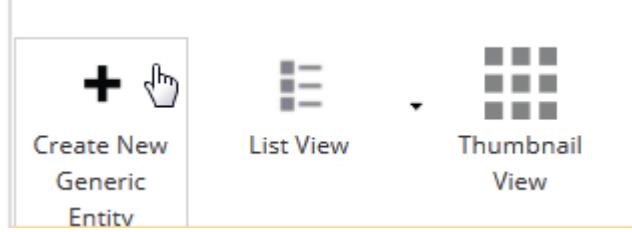
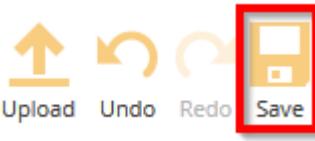
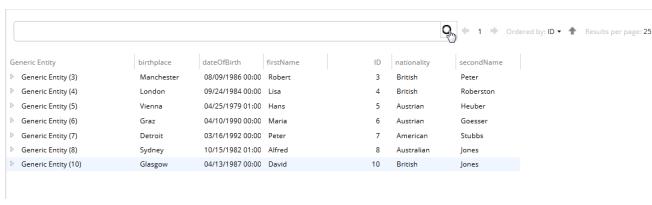
6

The data you added in MsSQL will now be displayed.

Generic Entity	birthplace	dateOfBirth	firstName	ID	nationality	secondName
Generic Entity (3)	London	08/09/1986 00:00	Robert	3	British	Peter
Generic Entity (4)		09/24/1984 00:00	Lisa	4	British	Robertson
Generic Entity (5)	Vienna	04/25/1979 01:00	Hans	5	Austrian	Heuber
Generic Entity (6)	Graz	04/10/1990 00:00	Maria	6	Austrian	Goosser
Generic Entity (7)	Detroit	03/16/1990 00:00	Peter	7	American	Stubbs
Generic Entity (8)	Sydney	10/15/1982 01:00	Alfred	8	Australian	Jones

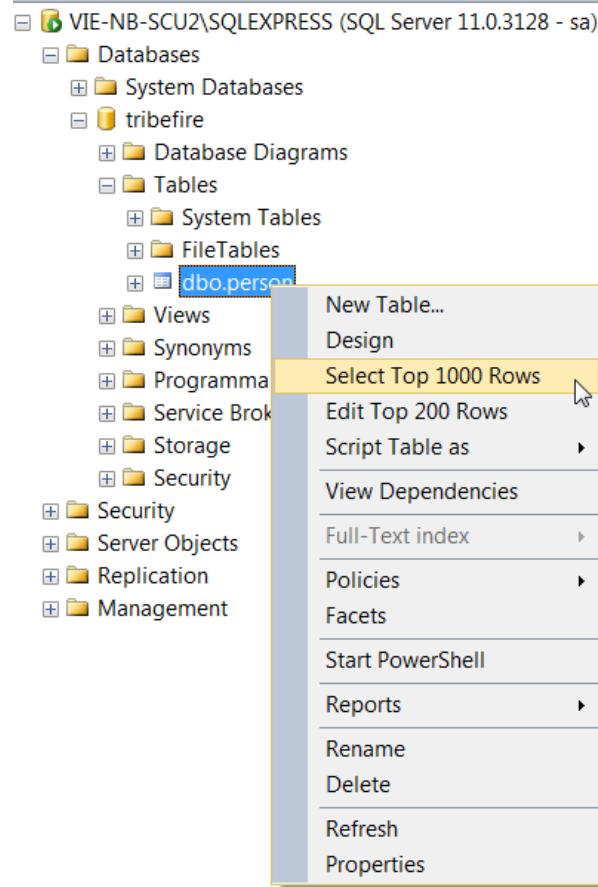
Add Data Using tribefire Explorer

You can also add data to your MsSQL database by using tribefire Explorer.

Step #	Task
1	Deselect any currently selected instances. 
2	Click the Create New Generic Entity button. 
3	Add some information for this new person...  ...and then click Apply . 
4	Click Save to persist your changes. 
5	Click the magnifying glass icon to refresh your Person Query and your new data will be displayed. 

6

If we were to view the database in MsSQL, we would see the new entry has been added to the database. To do so, right click on your table in MsSQL and select the *Select Top 1000 Rows* option.



7

All entries in this table will be shown, with your newly added entry also displayed.

The screenshot shows the SSMS Results grid displaying the data from the 'dbo.person' table. The grid shows 8 rows of data, including the newly added row (ID 10, firstNa... 3, secondNa... Jones, dateOfBirth 1987-04-15, birthplace Glasgow, nationality British). The columns are labeled ID, firstNa..., secondNa..., dateOfBirth, birthplace, and nationality.

	ID	firstNa...	secondNa...	dateOfBirth	birthplace	nationality
1	3	Robert	Peter	1986-08-11	Manchester	British
2	4	Lisa	Roberston	1984-09-26	London	British
3	5	Hans	Heuber	1979-04-27	Vienna	Austrian
4	6	Maria	Goesser	1990-04-12	Graz	Austrian
5	7	Peter	Stubbs	1992-03-18	Detroit	American
6	8	Alfred	Jones	1982-10-17	Sydney	Australian
7	10	David	Jones	1987-04-15	Glasgow	British

Configuring and setting up a connection to MySQL

Table of Contents

▼ Expand / collapse

- Introduction

Child pages

▼ Expand / collapse

- Downloading and installing MySQL
- Configure MySQL
- Create a MySQL connection
- Create a MySQL access
- Automatically create a new Model from your MySQL database
- Query your new MySQL Model in tribefire GME

Introduction

Using tribefire Control Center, you can create a connection to a MySQL database, allowing you to manipulate the data stored there. Once you have created a connection, you can import the database schema and automatically generate models from these tables, which gives you the opportunity to create, edit, delete entries in the database, as well as being able to execute queries from within tribefire Explorer. In addition to this, you can combine different accesses together to create a new access from disparate repositories, giving you a fully 360 degree picture of your data—this is done through the use of a Smart Enterprise Information Access.

! To make sure that you can properly connect to your MySQL server, you must make sure that you have the correct drivers in your tribefire host's library directory.

You can either use the [org.mariadb.jdbc.Driver](#) or the [com.mysql.jdbc.Driver](#) driver.

Download either driver and save it to the directory:

Tribefire_Installation_Directory/tribefire/host/lib

Then, when creating a connector, [enter the corresponding name](#) (either `org.mariadb.jdbc.Driver` or `com.mysql.jdbc.Driver`) into the Driver property field.

Once you have added your drivers, make sure that you restart your tribefire host.

There are several steps required when configuring tribefire GME so that it can be connected to a MySQL.

- Downloading and Installing MySQL
- Configure MySQL
- Create a MySQL connection
- Create a MySQL access
- Automatically create a new Model from your MySQL database
- Query new Model and using integration access to add data

Downloading and installing MySQL

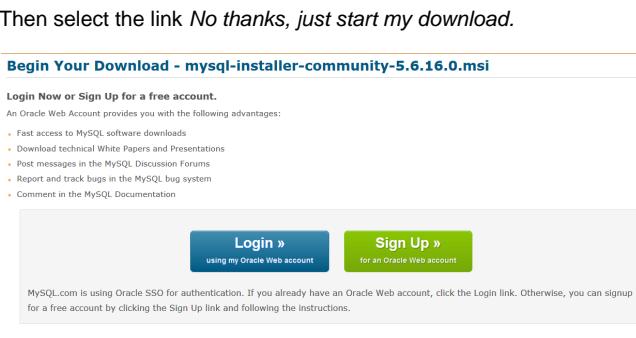
Table of Contents

- ▼ Expand / Collapse
 - Introduction
 - Task

Introduction

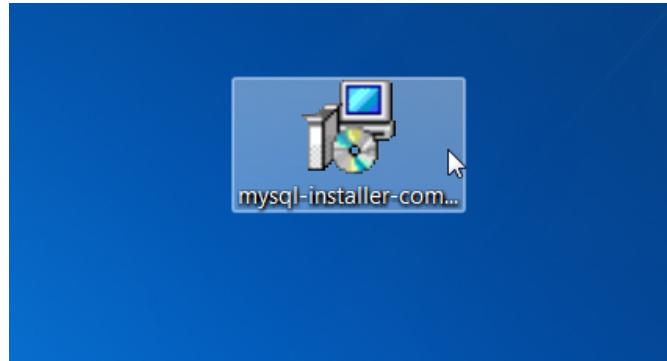
Before we can create a connection to a MySQL server, we must download and install an instance of it first . If you already have a version of MySQL install you can skip this stage.

Task

Step #	Task
1	<p>Download the MySQL Community Server from the MySQL website. Select your platform and then select download.</p> 
2	<p>Select the correct version and then click Download.</p> 
3	<p>Then select the link <i>No thanks, just start my download</i>.</p>  <p>MySQL will then be downloaded.</p>

4

Double-click the installation file to begin setup.



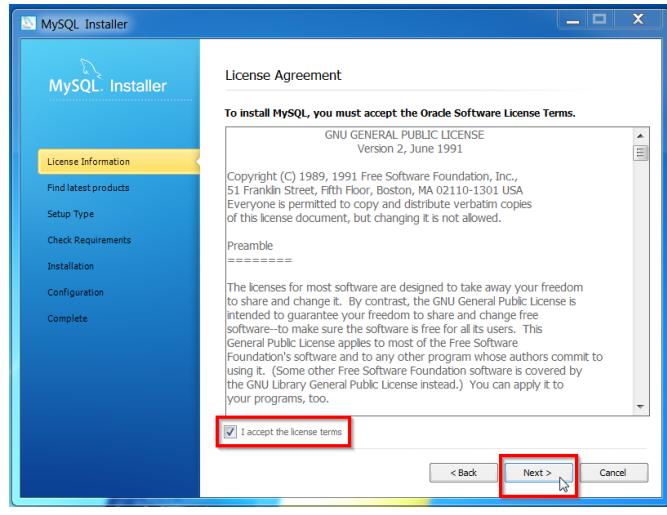
5

Select *Install MySQL Products*.



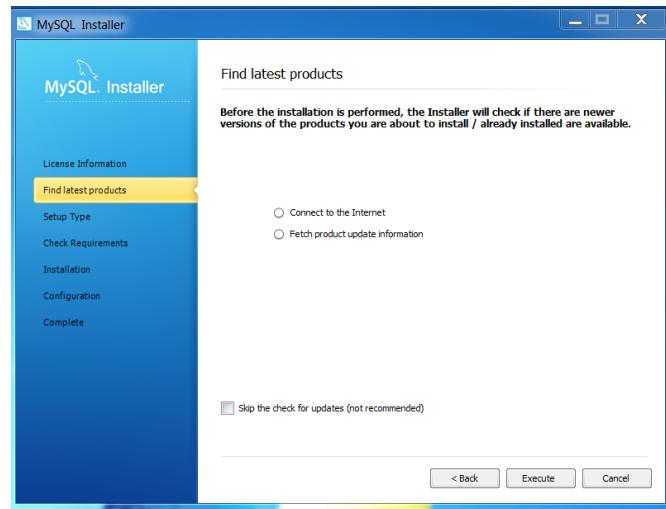
6

Check the *I accept the license terms* and then click **Next**.



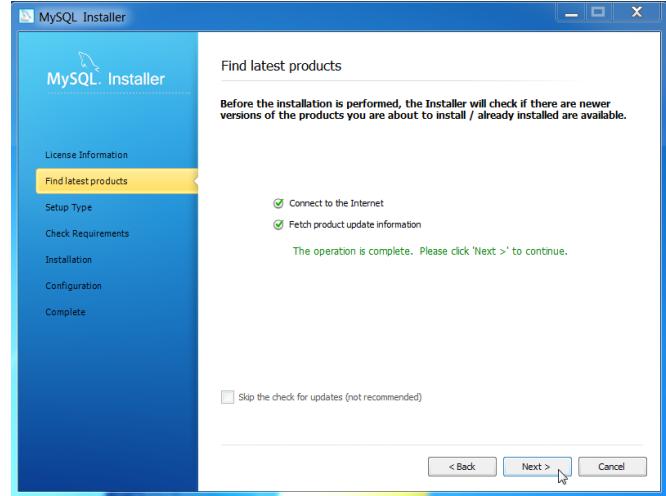
7

Click **Execute** and the MySQL installer will check for new updates. If you are certain that you have the latest version, you can check the *Skip the check for updates* option.



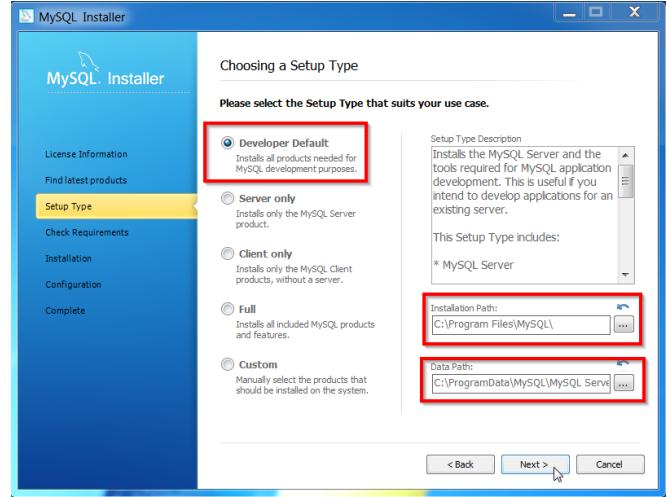
8

Once the MySQL has finished checking for updates, select **Next**.

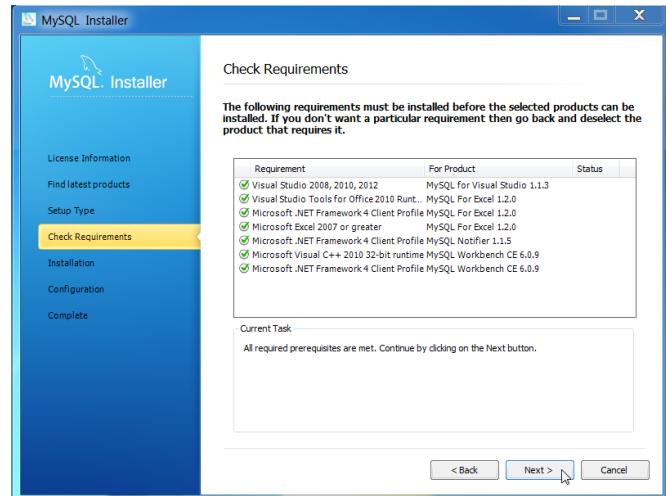


9

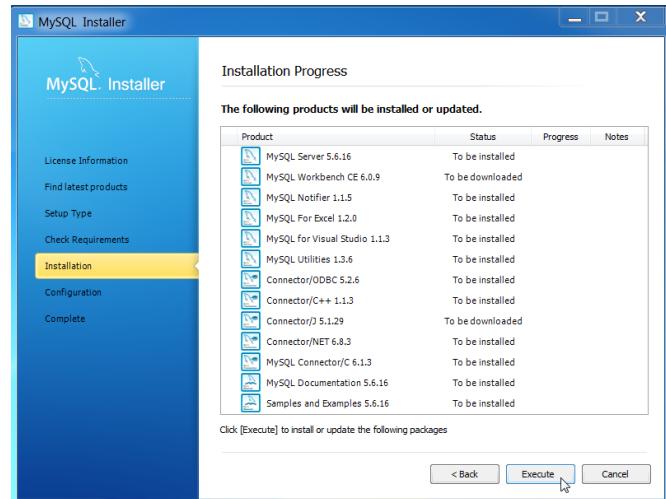
Click **Next**. If you wish to change the installation path or data path, you can do so here.



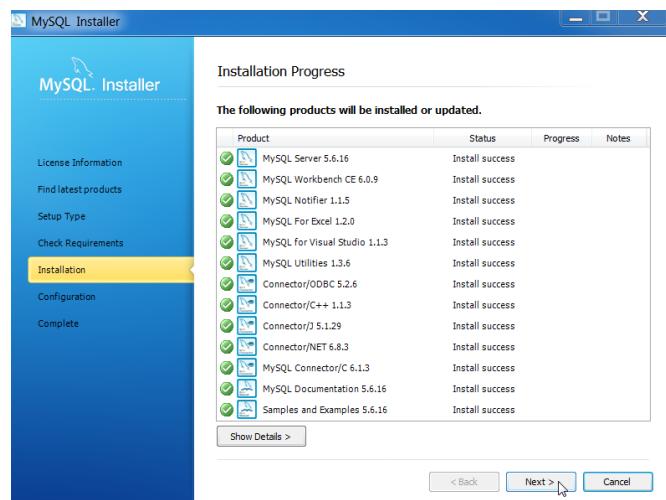
10

Click **Next** to install the required components.

11

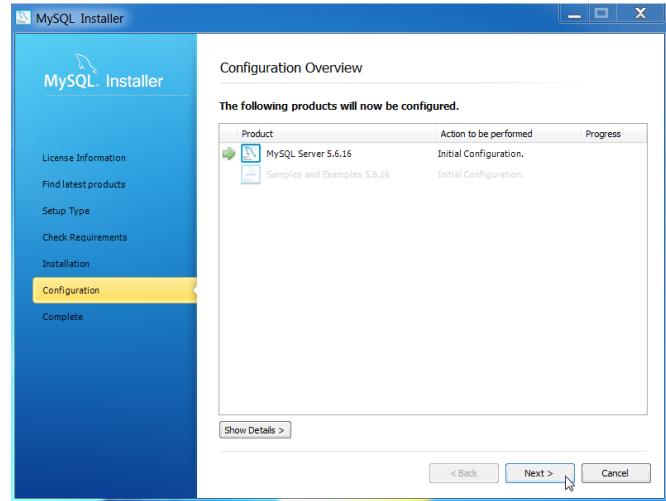
Click **Execute** and the components will be installed.

12

Once all components have been installed click **Next**.

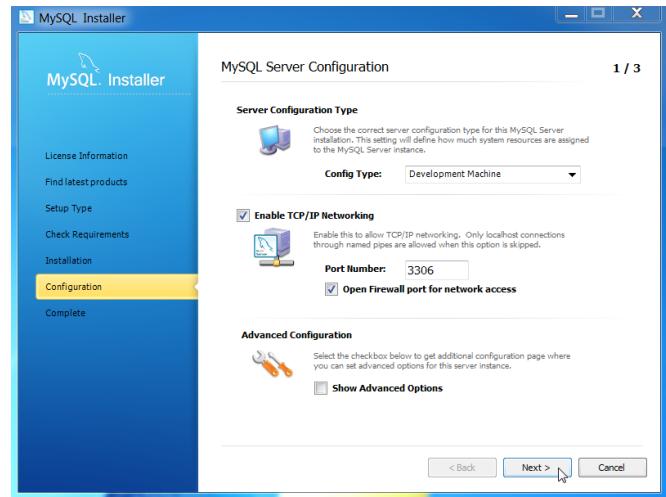
13

Click **Next**.



14

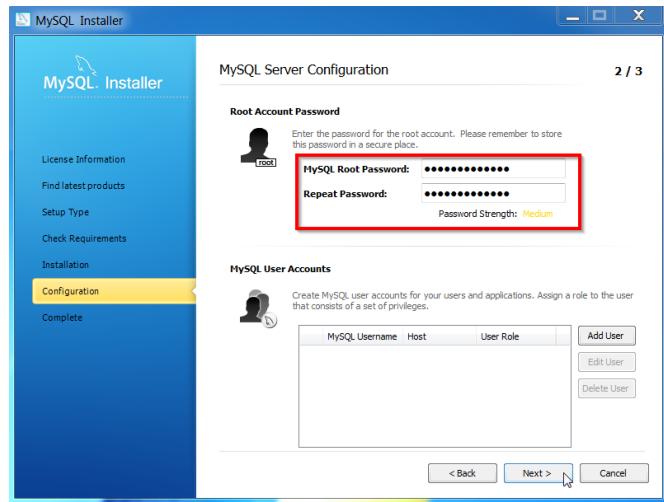
Leaving the default options, Click **Next** to continue.



The default port used by MySQL is 3306. This value will be used later when creating a **MySQL connector**. If, for any reason, you have to change this port value, make sure you take a note of the new value for later use.

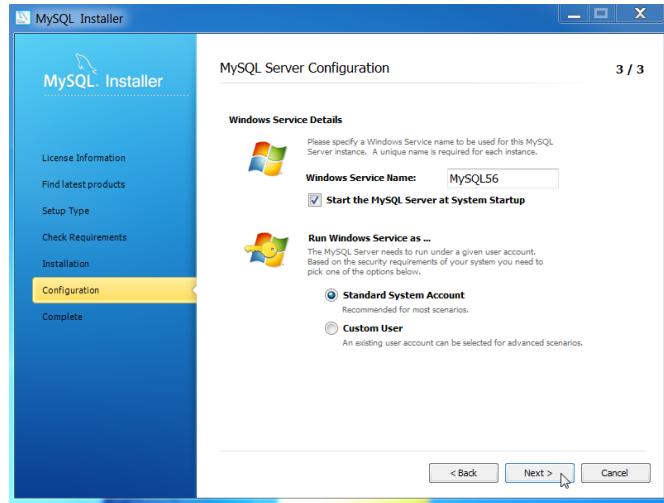
15

Enter a password that will be used when logging into the MySQL Workbench. You will need this password later when configuring the SQL server. Once you have entered a valid password, click **Next**.



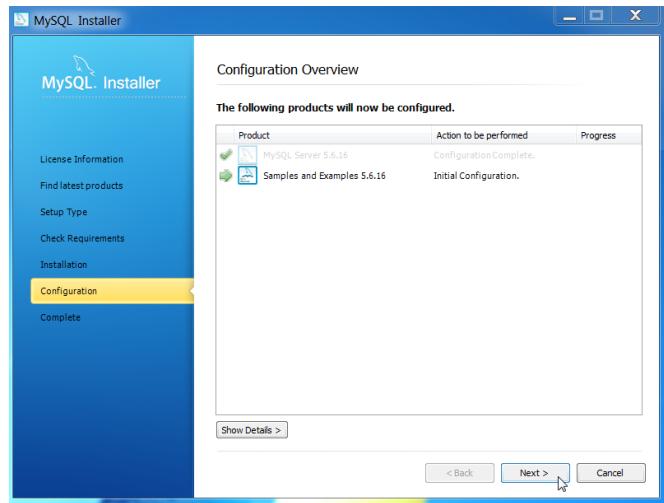
16

Click **Next** to complete the installation.



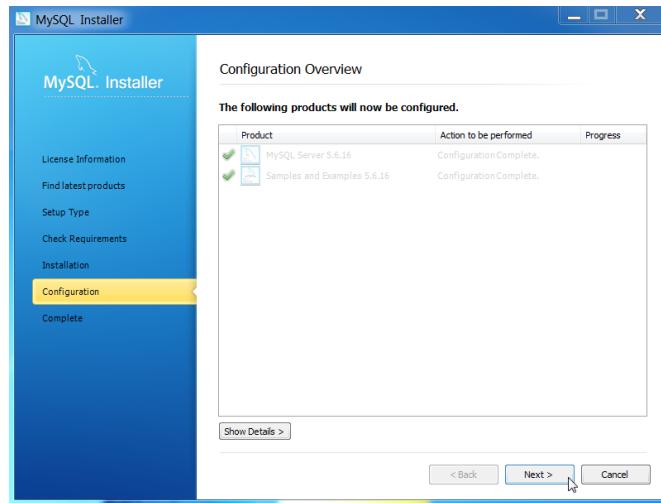
17

The MySQL server will be installed. After installation is complete, click **Next**.



18

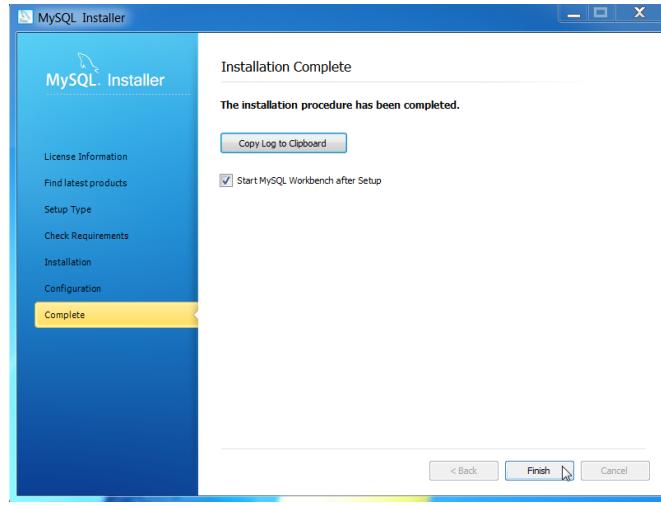
The SQL database will also be installed. After installation, click **Next** to continue.



19

The installation of MySQL is now complete. You can either start the MySQL workbench immediately, in which case click **Finish**, or uncheck the *Start MySQL Workbench after setup* option and then click **Finish** to start the MySQL later.

The MySQL workbench is used to configure the database server, a step which is required before we can create a connection in tribefire Control Center.



You have now successfully installed MySQL. You can now configure MySQL to allow for a connection to tribefire.

Configure MySQL

Table of Contents

▼ Expand / collapse

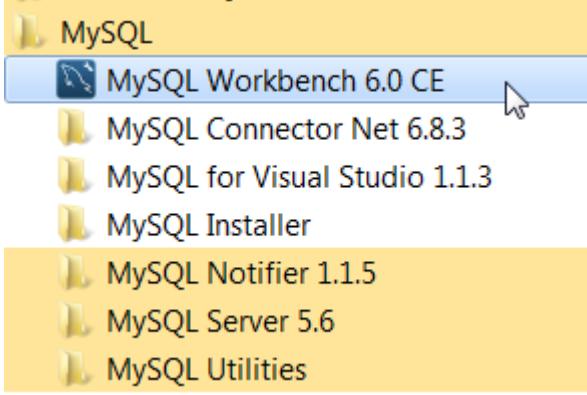
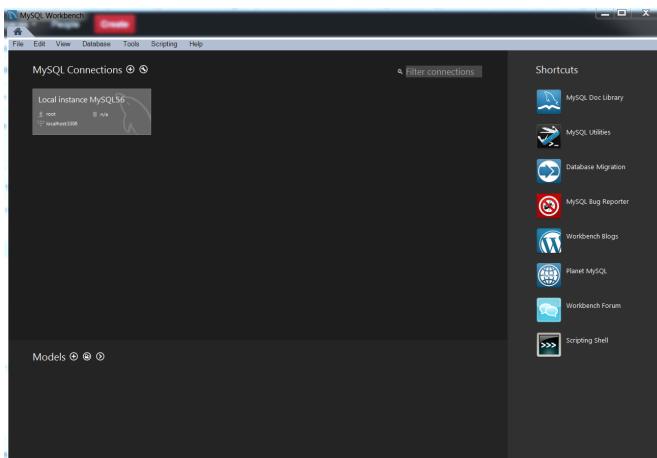
- Introduction
- Create new MySQL Database
- Create a new database table
- Create a New User Account

Introduction

After installing MsSQL, you will have to configure it, so that tribefire can connect to the database server. This includes creating a user account to be used by tribefire to access MsSQL, and a database and some tables which contain some data.

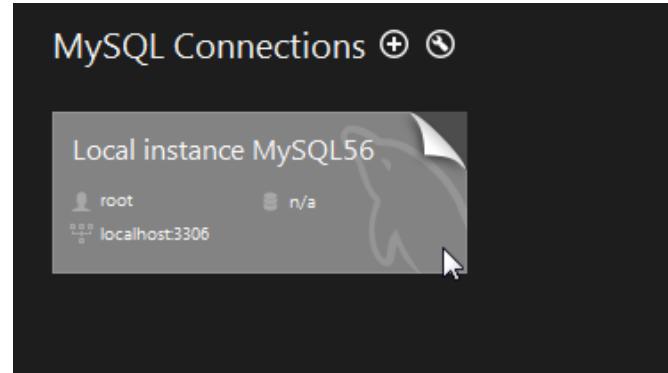
Create new MySQL Database

The first step when configuring the MySQL server is to create a new database. In MySQL the database is referred to as a Schema. This page will use both terms interchangeably.

Step #	Task
1	<p>From your start menu, select MySQL and then <i>MySQL workbench</i>.</p> 
2	<p>The MySQL workbench will be opened. From here you can configure user accounts and databases.</p>  <p>i for more information on the configuration and usage of MySQL Workbench.</p>

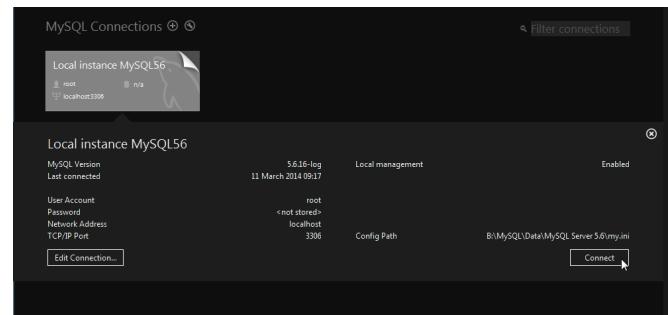
3

Click the label *Local Instance*. In the screenshot below the instance of this MySQL database server is *MySQL56*.



4

Click **Connect**.



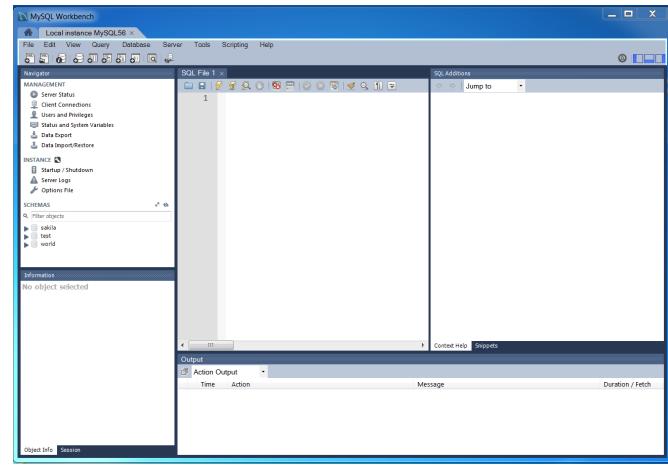
5

Enter the password that you defined in the installation stage and click **OK**.



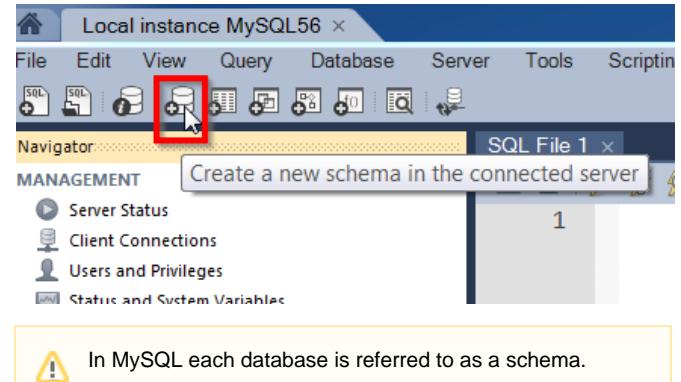
6

The main configuration window for this instance of MySQL will open.



7

From the toolbar select the new database icon .



8

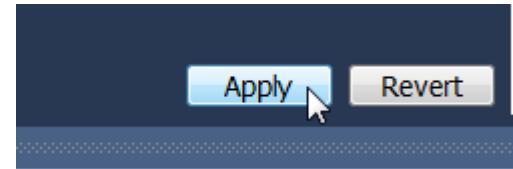
The New Schema configuration panel will be displayed. Enter a name for your database.

In this example, we will use the name tribefire.



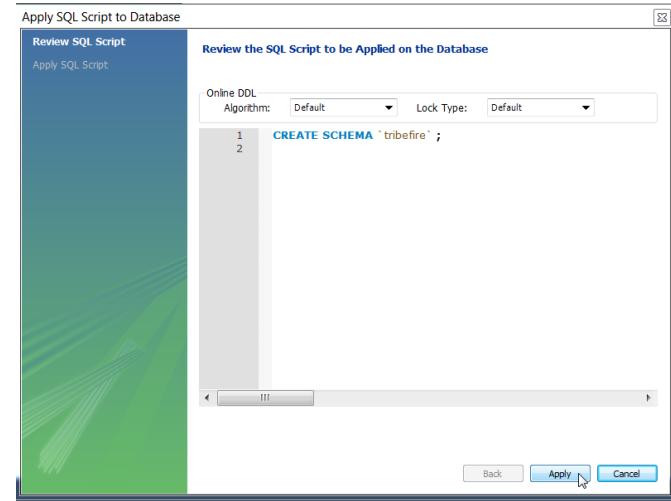
9

Click the **Apply** button found at the bottom of the panel.

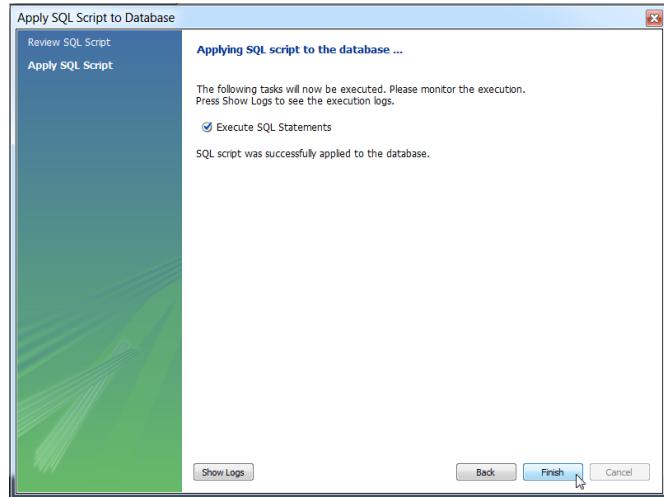


10

You will be asked to review the schema you are creating. Click **Apply** to continue.

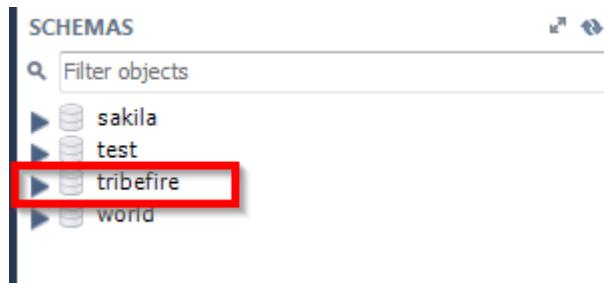


11

Click **Finish** to complete the creation of your new database.

12

Your new database will be displayed in the Schemas section.



Create a new database table

The next step after the creation of the database is to add a new table to it. Normally, a database will have many tables with many columns, all representing a more complex modeling of data. However, for the purposes of this tutorial, we will only create one table with just a few columns.

Step #	Task
1	<p>Either double-click the newly created database or click on the expand icon to open the new database.</p>
2	<p>Right-click on the property Tables and then select Create Table.</p>

3

The new Table panel will be displayed. Give your table a name.

4

Next double click the first column (PK) so that a new Primary Key is created.

Column Name	Datatype	PK	N.	U.	B..	U.	Z..	AI	Default
idperson	INT	<input type="checkbox"/>							



When creating a table that will be imported into tribefire, it must contain a primary key, otherwise the import will fail.

5

Give this new column the name ID. We need to make sure that this ID has three properties checked: PK, NN and AI.

- PK stands for Primary Key and defines this column as the Primary Key.
- NN stands for Non-Nullable and means that data must be entered in this column when creating a new entry.
- AI stands for Auto-Increment and means a number will automatically be added when creating a new entry.

Once you have checked these three boxes, select the TAB key.

Column Name	Datatype	PK	NN	U..	B..	U.	Z..	AI	Default
ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				

6

Double-click the second column to create it.

Column Name	Datatype	P..	N..	U..	B..	U.	Z..	AI	Default
ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
new_tablecol	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

7

Give it the value firstName and then press the TAB key.

Column Name	Datatype	PK	NN	U..	B..	U.	Z..	AI	Default
ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
firstName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

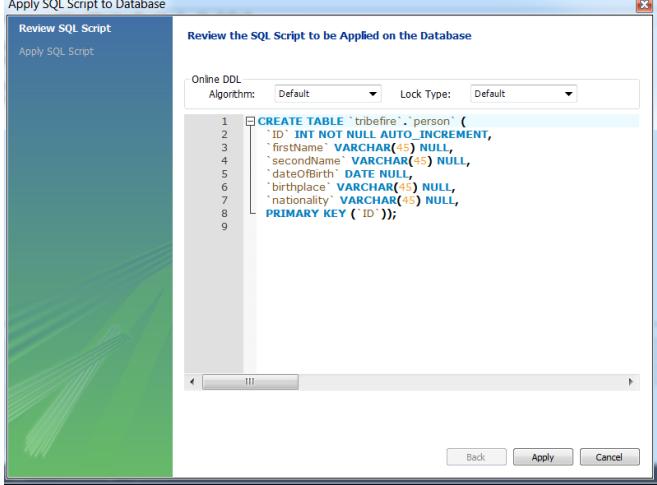
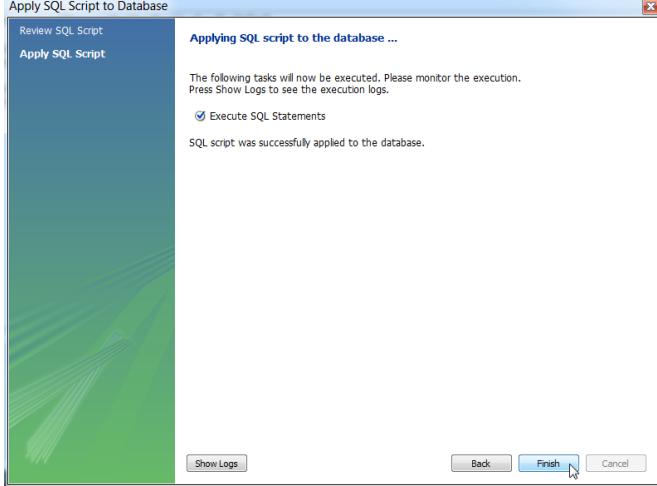
8

Continue this process until you have added all the columns you need.

Column Name	Datatype	PK	NN	U..	B..	U.	Z..	AI	Default
ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
firstName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
secondName	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
dateOfBirth	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
birthplace	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
nationality	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



Note that the column dateOfBirth has the data type Date.

9	<p>Click Apply to save the changes.</p> 
10	<p>You will be asked to review the changes made. Click Apply to continue.</p> 
11	<p>Click Finish to save your new table.</p> 

You new table will be added to the database.

Table: person

Columns:

ID	int(11) AI PK
firstName	varchar(45)
secondName	varchar(45)
dateOfBirth	date
birthplace	varchar(45)
nationality	varchar(45)

Create a New User Account

The last step when configuring MySQL is to create a new user account. This will allow tribefire to access the database.

Step #	Task
1	<p>From the navigator panel of MySQL Workbench, select <i>Users and Privileges</i>.</p>

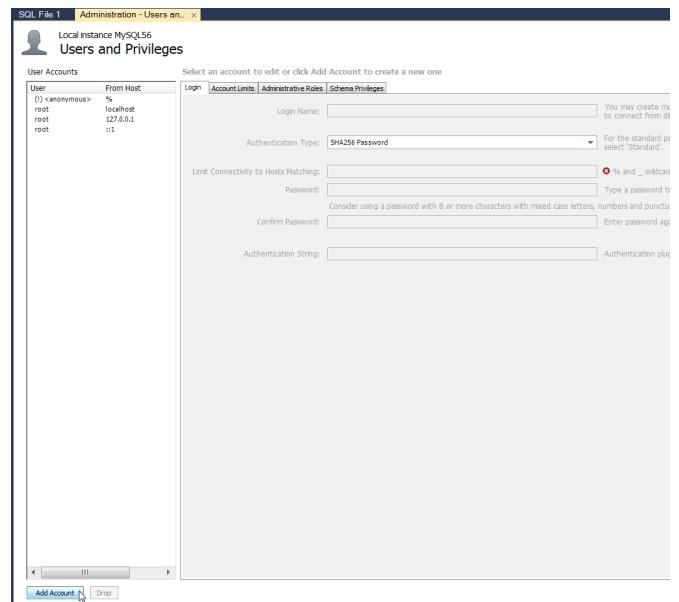
2

Enter the password that you defined during the installation stage and click **OK**.



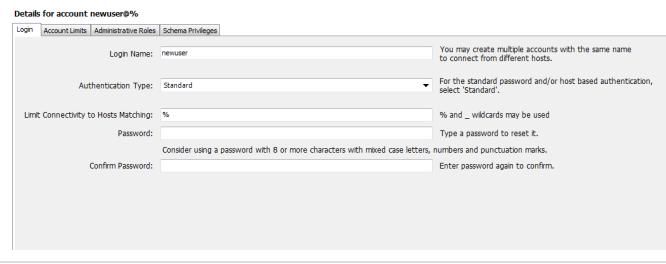
3

The User and Privileges panel will be displayed. Click the **Add Account** button located at the bottom of this panel.



4

A new user will be created.



5

Enter a Login Name and a valid password for this account.

Login Name: cortex
Authentication Type: Standard
Limit Connectivity to Hosts Matching: %
Password: cortex
Confirm Password: cortex



Take a note of the user name and password that you have entered here. They will be required when creating a new Connector.

6

Select the *Administrative Roles* tab and the check the DBManager box. The DBDesigner and BackupAdmin boxes will automatically be checked.

Role	Description
<input type="checkbox"/> DBA	grants the rights to perform all tasks
<input type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server
<input type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user proce...
<input type="checkbox"/> UserAdmin	grants rights to create users logins and reset passwords
<input type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server an...
<input type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server
<input checked="" type="checkbox"/> DBManager	grants full rights on all databases
<input checked="" type="checkbox"/> DBDesigner	rights to create and reverse engineer any database sche...
<input type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication
<input checked="" type="checkbox"/> BackupAdmin	minimal rights needed to backup any database

7

Located at the bottom-right of the User and Privileges panel, click **Apply** and your new user will be saved.

Revoke All Privileges Expire Password Revert **Apply** Refresh

You have now successfully configured MySQL in preparation for a tribefire connection. You can now use tribefire Control Center to create a new connector.

Create a MySQL connection

Table of Contents

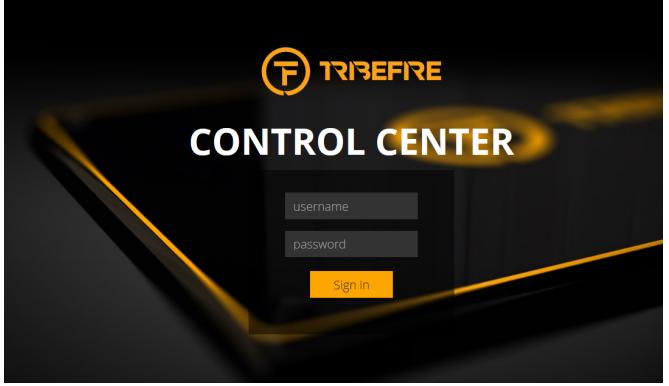
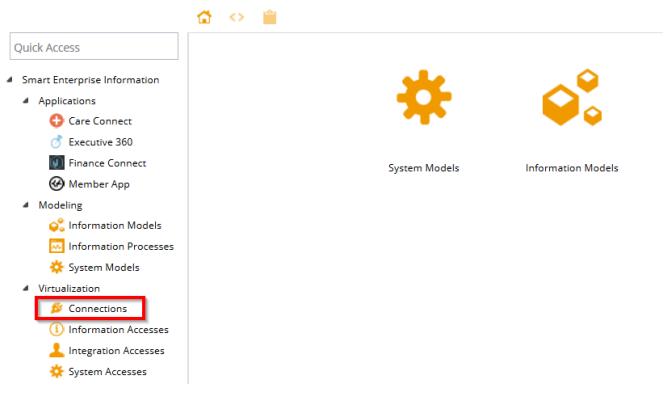
- ▼ Expand / collapse
 - Introduction
 - Task

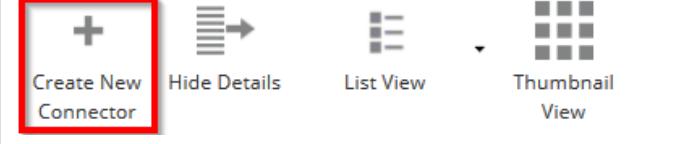
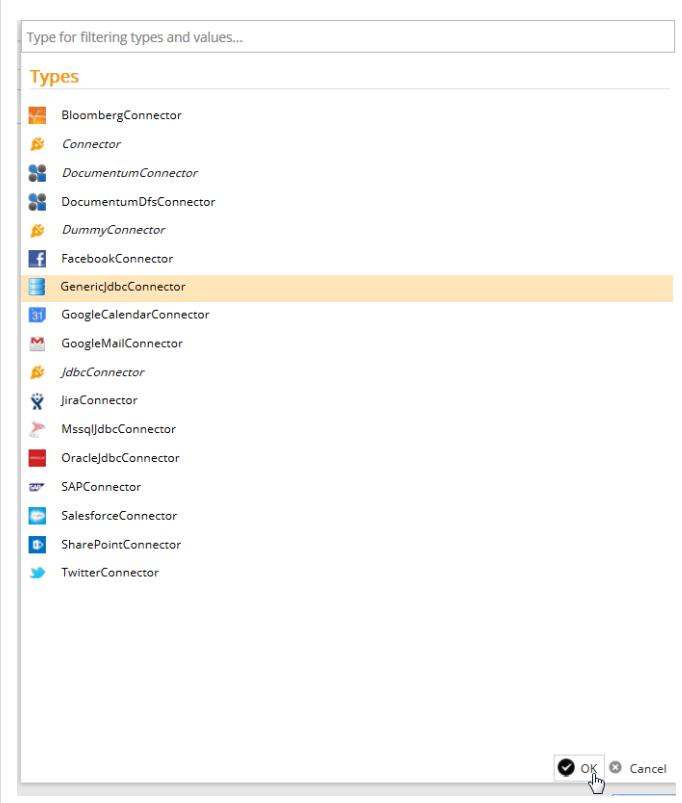
Introduction

The next step is to create a new connection that gives tribefire GME access to the MySQL database server we have just configured. You define the location of the database server, and through which port it should be accessed, and the user account tribefire should use to login to MySQL. You must also set the name of the database that should be used.

Once connected, the Control Center can then be used to create prototypes, known as a schema in tribefire, of the tables found in the database.

Task

Step #	Task
1	<p>Login to tribefire Control Center using your valid credentials.</p> 
2	<p>Click the Connections link found in the left-hand menu panel of Control Center.</p> 

3	<p>Click the Create New Connector from the buttons panel.</p> <p>If there are any connections highlighted, you will have to deselect them before you can proceed. To do so, hold down CTRL and then click the highlighted object.</p> 
4	<p>The New Connector window will appear. Select the GenericJdbcConnector option and click OK.</p> 

5

The GenericJdbcConnector's properties window will be displayed. Here we must enter the information required to connect to your MySQL server.



Configuration Information

Property	Description	Tutorial Example Value
Url	<p>The URL is used to connect to the MySQL server.</p> <p>For more information on the syntax and configuration of the Jdbc URLs</p>	<p>jdbc:mysql://localhost:3306/tribefire</p>   localhost is the location of the MySQL server 3306 is the open port tribefire is the name of the database we created
Driver	The driver used to connect to the database	org.mariadb.jdbc.Driver
User	The name of the user account defined in the previous step	cortex
Password	The password of the user account defined in the previous step	cortex
External ID	The external ID of the Connection	MySQL.connection
Name	A name for this connection	mySQLConnection



You do not have to enter any information into the *Hibernate Dialect* or *Hibernate Dialect Class* fields. This will be automatically configured after deployment.

6	<p>Enter the details for your new Connector and click Apply.</p>
7	<p>Click Save to persist the changes in tribefire.</p>
8	<p>The next step is to deploy your connection. Click Deploy from the buttons panel at the bottom of GME.</p>
9	<p>You model will now be deployed.</p>
10	<p>You can now test the connection by clicking the Test Connection button.</p> <p>If successful, you will receive a success message.</p>
11	<p>The next step is to import the tables from your database by using Update DB Schema located in the buttons panel.</p>

12

Clicking **Update DB Schema** will load the rough types into tribefire. They will be displayed in the Jdbc Connector section of the details panel.

Jdbc Connector

– Db Schemas

● Db Schema (36)

You have now successfully created a connection and uploaded the database tables from your MySQL server. You can now create an integration that will make use of this schema to create a model.

Create a MySQL access

Table of Contents

▼ Expand / collapse

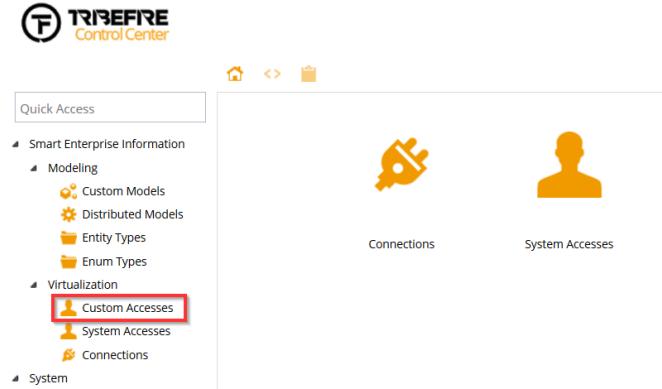
- Introduction
- Create a MySQL Integration Access

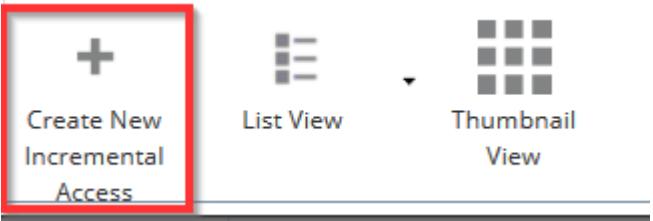
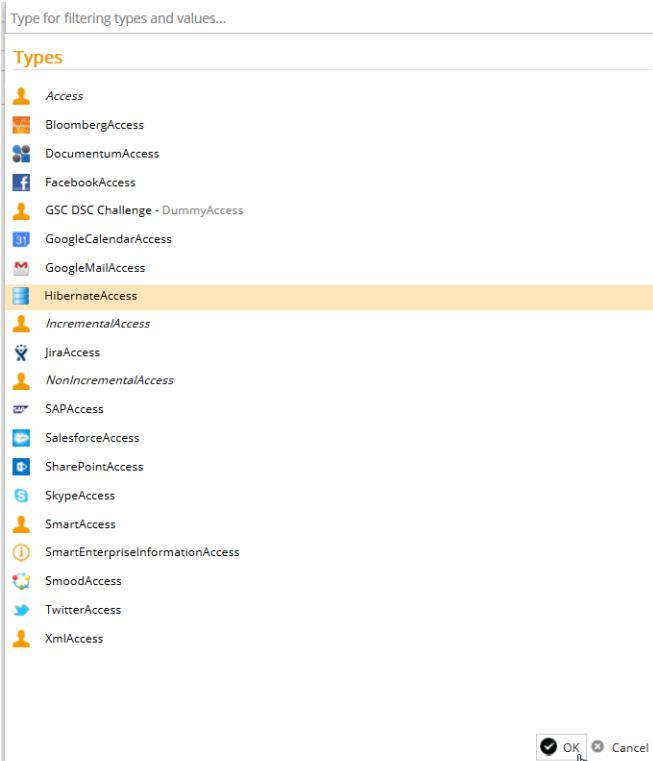
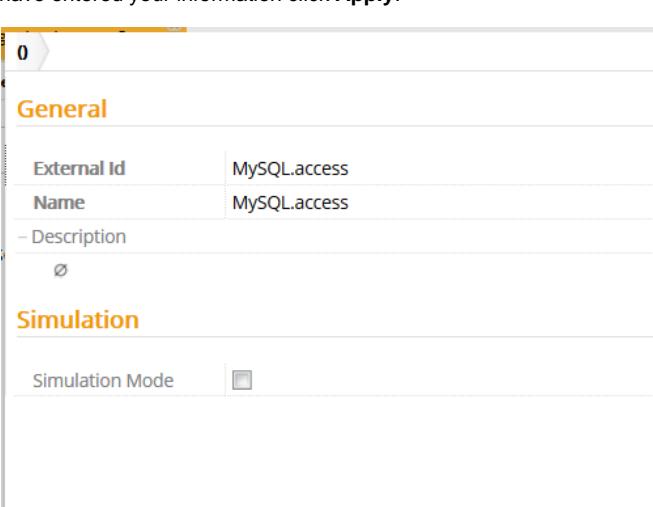
Introduction

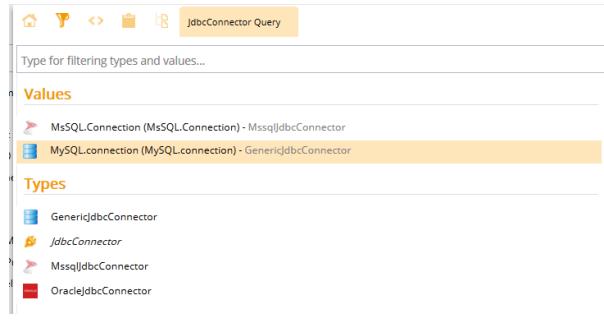
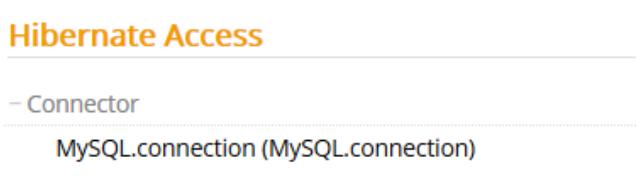
After connecting to your MySQL database, you can now create an access that will make use of this connection to convert the rough types extracted from the database to create a new model in tribefire.

There is no specific MySQL access in tribefire. Therefore, you must use the standard access **Hibernate Access** for all database connections.

Create a MySQL Integration Access

Step #	Task
1	<p>Login to tribefire with your valid credentials.</p> 
2	<p>Click the link <i>Custom Accesses</i> found in the left-hand menu panel of Control Center.</p> 

3	<p>Click the Create New Incremental Access from the buttons panel.</p> 
4	<p>The New Access window will appear. Select the Hibernate Access and click OK.</p> 
5	<p>Enter a name, External Id, and, if you wish, a Description. Once you have entered your information click Apply.</p> 

6	<p>Persist your changes by clicking Save.</p> 
7	<p>The next step is to add the correct connector to your integration access. To do so, either double click on the Connection property or hover over it and select the context-menu icon that appears. Then select Assign.</p> 
8	<p>Select the connector that we configured in the previous step...</p>  <p>...and then click Finish.</p> 
9	<p>The Connector will be added to your integration access.</p> 
10	<p>Click Save to persist this integration access in tribefire GME.</p> 

You have now correctly configured an Hibernate Integration Access. The next step shows you how to generate a model from the connection and how to deploy it.

Automatically create a new Model from your MySQL database

Table of Contents

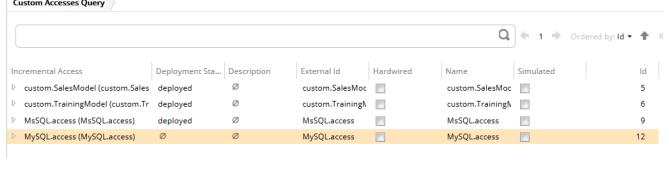
▼ Expand / collapse

- Introduction
- Task

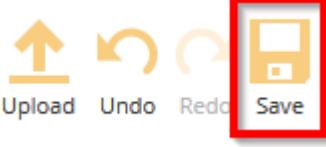
Introduction

After creating an access, you can now create a new model in tribefire, based on the rough types imported from your database. This will automatically generate the new entity types and properties needed for tribefire to manipulate and query data.

Task

Step #	Task
1	Select the access that you configured in the previous step. 
2	Click Create Model From DB from the buttons panel. 

<p>3</p>	<p>A new meta model will be created.</p> <p>General</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"><i>Id</i></td><td style="padding: 5px;">5</td></tr> <tr> <td style="padding: 5px;"><i>External Id</i></td><td style="padding: 5px;">MySQL.access</td></tr> <tr> <td style="padding: 5px;"><i>Name</i></td><td style="padding: 5px;">MySQL.access</td></tr> <tr> <td colspan="2" style="padding: 5px;"><i>- Description</i></td></tr> <tr> <td colspan="2" style="padding: 5px; text-align: center;">Ø</td></tr> <tr> <td colspan="2" style="padding: 5px;">- Model</td></tr> <tr> <td colspan="2" style="padding: 5px; border: 2px solid red; background-color: #ffffcc;">com.braintribe.model.meta.GmMetaModel__gm@2215</td></tr> </table> <p>Status</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"><i>Deployment State</i></td><td style="padding: 5px;">Ø</td></tr> </table> <p>Simulation</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"><i>Simulation Mode</i></td><td style="padding: 5px; text-align: center;">□</td></tr> </table> <p>HibernateAccess's Properties</p> <p><i>- Workbench Access</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Ø</td></tr> </table> <p>Connection</p> <p><i>- Connection</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">MySQL.connection (MySQL.connection)</td></tr> </table>	<i>Id</i>	5	<i>External Id</i>	MySQL.access	<i>Name</i>	MySQL.access	<i>- Description</i>		Ø		- Model		com.braintribe.model.meta.GmMetaModel__gm@2215		<i>Deployment State</i>	Ø	<i>Simulation Mode</i>	□	Ø	MySQL.connection (MySQL.connection)
<i>Id</i>	5																				
<i>External Id</i>	MySQL.access																				
<i>Name</i>	MySQL.access																				
<i>- Description</i>																					
Ø																					
- Model																					
com.braintribe.model.meta.GmMetaModel__gm@2215																					
<i>Deployment State</i>	Ø																				
<i>Simulation Mode</i>	□																				
Ø																					
MySQL.connection (MySQL.connection)																					
<p>4</p>	<p>Click on the newly created meta model to open it. You will notice that its properties are empty.</p> 																				
<p>5</p>	<p>Click the refresh button to complete the model's creation.</p> 																				

6	<p>The entities will be created and displayed in the field Entity Types.</p> <p>General</p> <table border="0" style="width: 100%;"> <tr> <td style="width: 10%;"><i>Id</i></td><td style="width: 10%;">6</td></tr> <tr> <td colspan="2">– Name</td></tr> <tr> <td colspan="2">∅</td></tr> </table> <p>Model's Properties</p> <table border="0" style="width: 100%;"> <tr> <td style="width: 10%; vertical-align: top;"> – baseType ∅ </td><td style="width: 90%;"> – Entity Types <ul style="list-style-type: none"> • GenericEntity • Person </td></tr> <tr> <td colspan="2">– Enum Types ∅</td></tr> <tr> <td colspan="2">– Meta Data ∅</td></tr> <tr> <td colspan="2">– simpleTypes ∅</td></tr> </table>	<i>Id</i>	6	– Name		∅		– baseType ∅	– Entity Types <ul style="list-style-type: none"> • GenericEntity • Person 	– Enum Types ∅		– Meta Data ∅		– simpleTypes ∅	
<i>Id</i>	6														
– Name															
∅															
– baseType ∅	– Entity Types <ul style="list-style-type: none"> • GenericEntity • Person 														
– Enum Types ∅															
– Meta Data ∅															
– simpleTypes ∅															
7	<p>The next step is to give your model a new name.</p> <table border="0" style="width: 100%;"> <tr> <td style="width: 10%; vertical-align: top;"> – Name MySQLPersonModel </td><td style="width: 90%;"></td></tr> </table>	– Name MySQLPersonModel													
– Name MySQLPersonModel															
8	<p>Click Save to persist this model in tribefire.</p> <div style="text-align: center; margin-top: 10px;">  </div>														
9	<p>The last step is to deploy the model. Select your access (in this case, MySQL.access) option from the tabs at the top (or simple reopen the Integration Accesses link)...</p> <div style="margin-top: 10px;">  </div> <p>...and select Deploy from the buttons panel.</p> <div style="text-align: center; margin-top: 10px;">  </div>														

10

Your access has been deployed.

Integration Accesses Query	MySQL_Access (MySQL_Access)	Model Model
HibernatesAccess	MySQL_Access (MySQL_Access)	Deployment Status Description Simulation Mode ID External ID Name Deployed MySQLAccess MySQLAccess

You have now successfully automatically generated a model from a db schema and deployed it. The next step is to query this access, displaying the results in tribefire Explorer.

Query your new MySQL Model in tribefire GME

Table of Contents

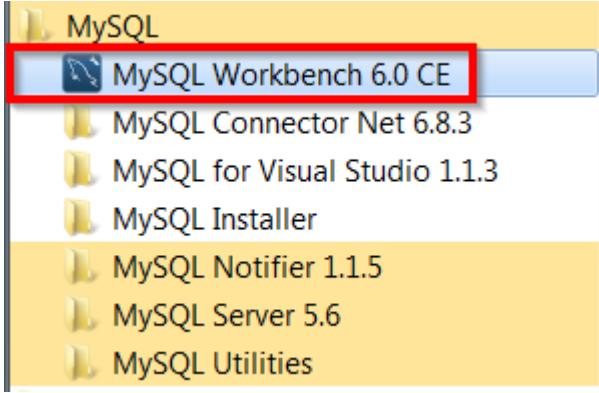
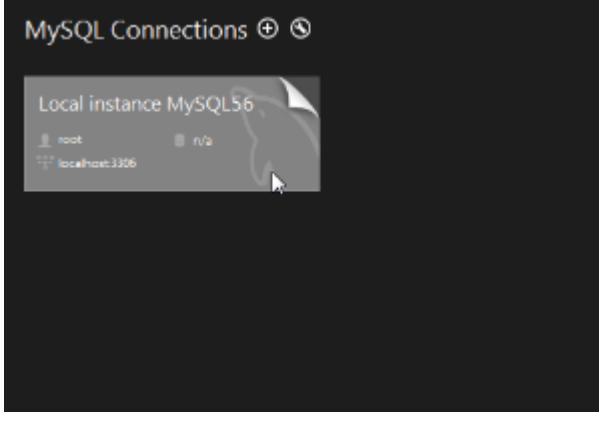
▼ Expand / collapse

- Introduction
- Creating Data in MySQL
- View Data in tribefire GME
- Add Data to tribefire

Introduction

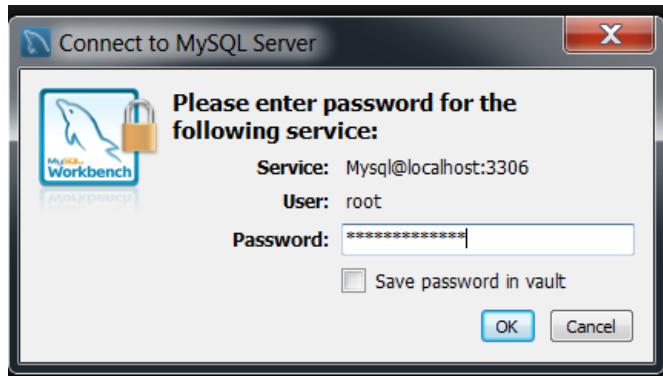
After creating and deploying an integration access, you can now use it to view the data within tribefire GME. The access by standard allows you to access any of the entity types and view the data without any extra configuration. However, since we only created a new table and didn't actually add any data, we will have to do that first.

Creating Data in MySQL

Step #	Task
1	<p>From start menu, select the MySQL Workbench tool.</p> 
2	<p>Click on your MySQL instance.</p> 

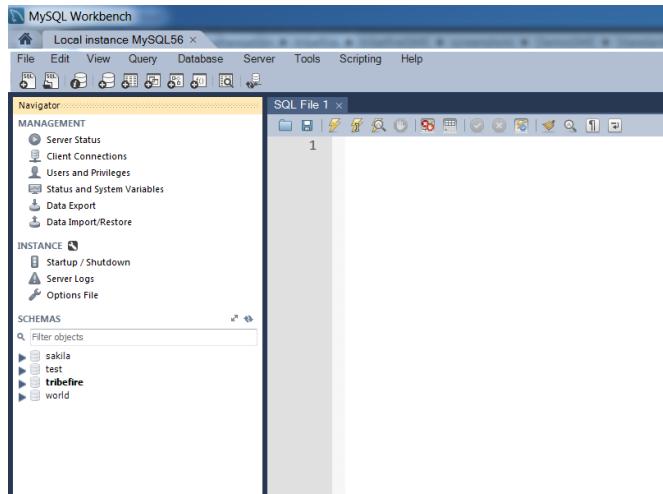
3

Enter the password you defined when configuring MySQL.



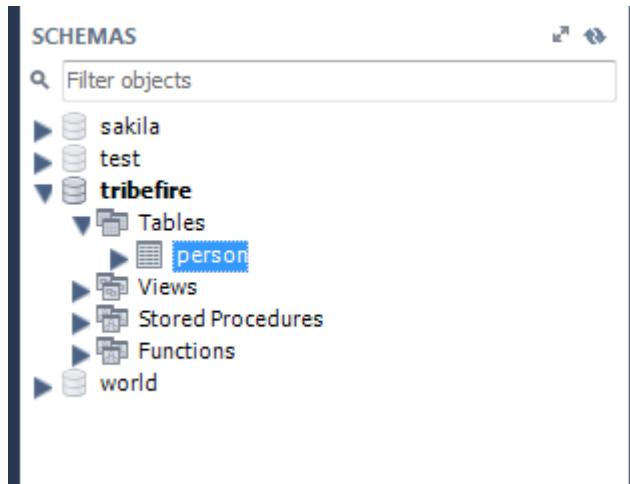
4

The main configuration window for this instance of MySQL will open.



5

From the schema section, select the database that you configured and then the table. You can do this by using the expand icon or double click on the object.



6

Right-click on the person table and select the *Select Rows - Limit 1000* option.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows 'sakila', 'test', 'tribefire' (selected), and 'world'. Under 'tribefire', there are 'Tables', 'Views', 'Stored', 'Functions', and 'world'. The 'Tables' node has a child 'person'. The main panel shows 'Information' for the 'Table: person'. Below it, 'Columns:' are listed: ID (int), firstName (varchar(45)), secondName (varchar(45)), dateOfBirth (date), birthplace (varchar(64)), and nationality (varchar(15)). A context menu is open over the 'person' table, with the 'Select Rows - Limit 1000' option highlighted in blue.

7

This will execute a query showing all the entries in this table. However, since no data has been added, no results will be shown.

The screenshot shows the MySQL Workbench SQL editor with the query 'SELECT * FROM tribefire.person;' entered. The results pane below shows an empty table with columns ID, firstName, secondName, dateOfBirth, birthplace, and nationality.

8

Double click on the column *firstName* to enter data. Press the TAB key to navigate between the different columns.

Enter some data.

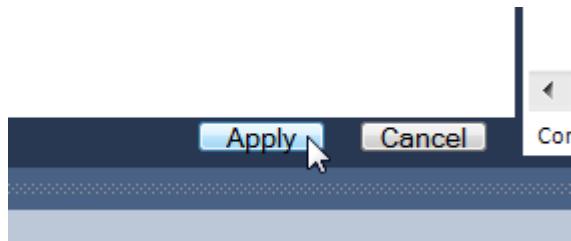
The screenshot shows the MySQL Workbench SQL editor with the same query. The results pane now displays a single row of data: ID is null, firstName is 'Robert', secondName is 'Smith', dateOfBirth is '1986-08-23', birthplace is 'Manchester', and nationality is 'British'. The cursor is currently in the 'secondName' column.

9

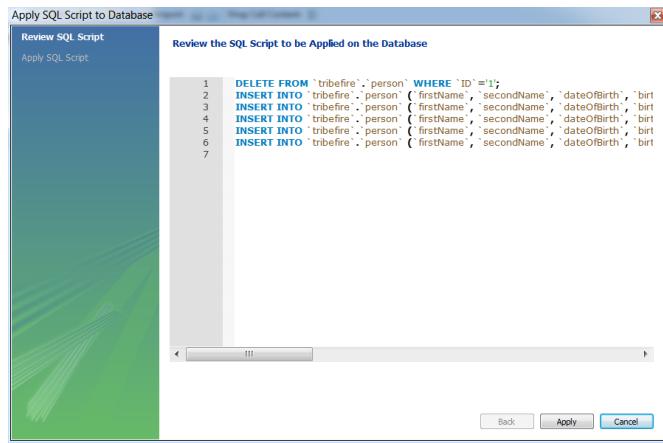
Continue until you have filled in 5 or 6 rows of data.

ID	firstName	secondName	dateOfBirth	birthplace	nationality
NULL	Robert	Smith	1986-08-23	Manchester	British
NULL	Lisa	Robertson	1984-09-26	London	British
NULL	Hans	Hueber	1979-04-30	Vienna	Austrian
NULL	Maria	Goesser	1990-04-12	Graz	Austrian
*	Peter	Stubbs	1992-03-18	Detroit	American

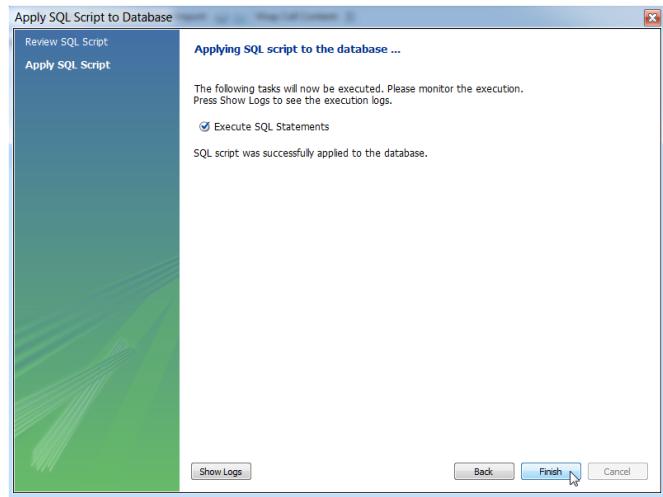
10

Once you have added your data, click the **Apply** button found at the bottom of the panel.

11

You will be asked to review the SQL statement pertaining to the creation of the data. Click **Apply** to continue.

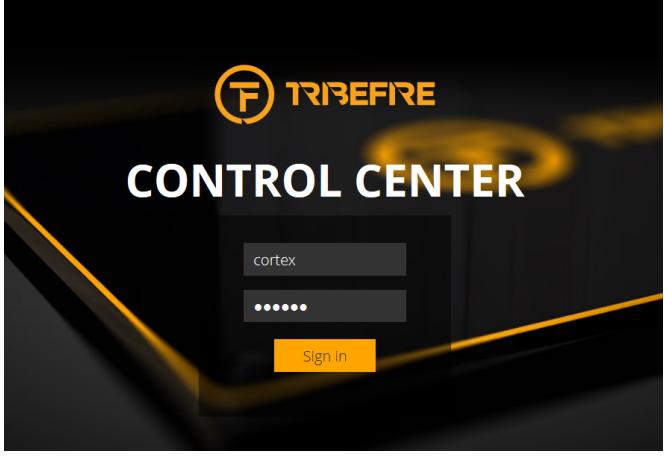
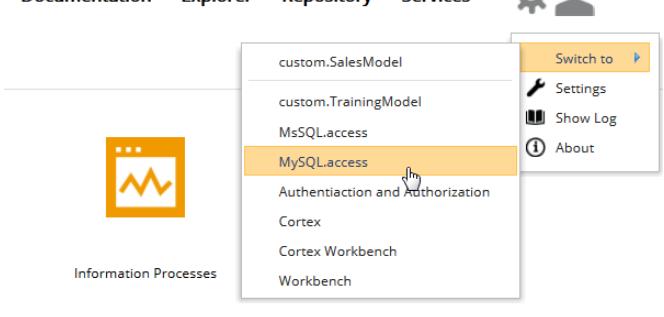
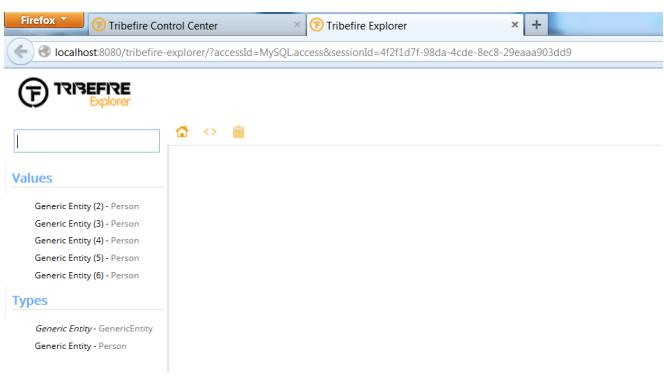
12

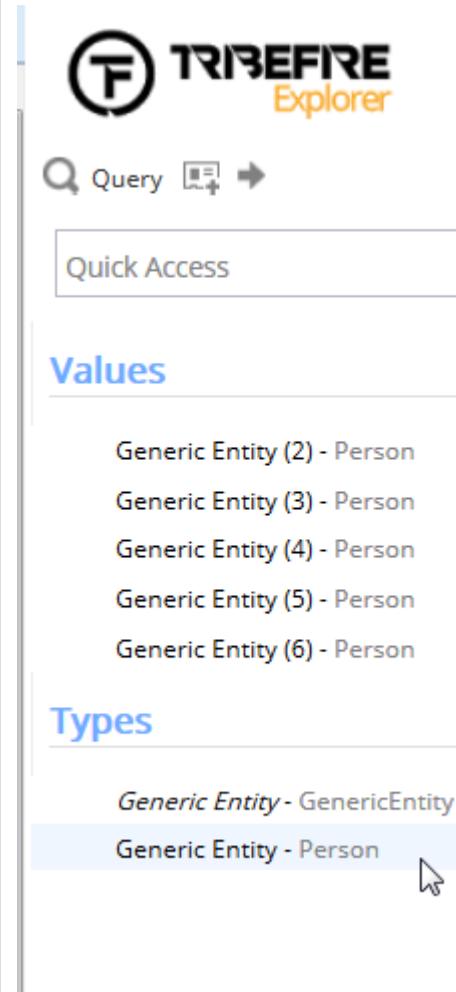
Click **Finish** to complete the creation of data.

You have now successfully created your data.

View Data in tribefire GME

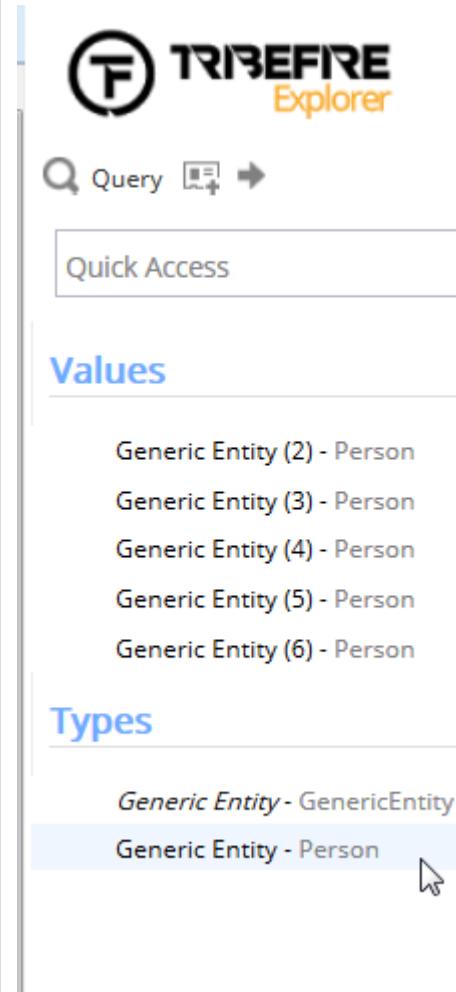
You can now view and add data to this table through tribefire Explorer

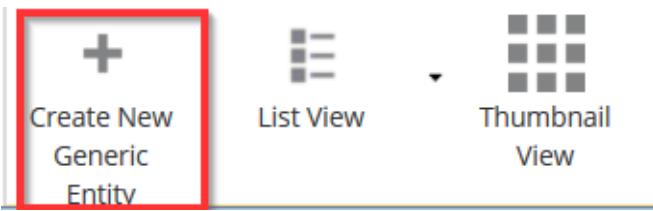
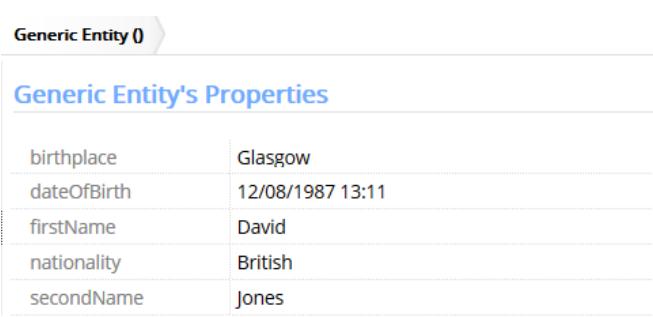
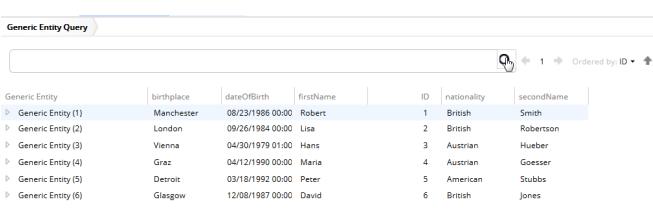
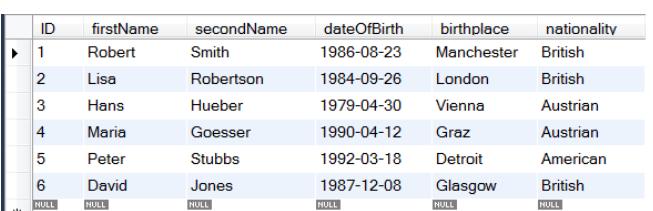
Step #	Task
1	<p>Login to tribefire Control Center with your credentials.</p> 
2	<p>At the top-right hand side of tribefire GME there is a cog icon. Click on it to view the different accesses that are currently deployed.</p> 
3	<p>A drop-down list will appear. Select the access that you deployed in the previous stage.</p> 
4	<p>Tribefire Explorer will open in a new tab.</p> 

5	<p>Once tribefire Explorer has opened, you will see a list of available entity types displayed in the left-hand panel of tribefire GME. Select the person entity</p> 																																										
6	<p>The data that you entered into MySQL should now be displayed.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Generic Entity</th> <th>birthplace</th> <th>dateOfBirth</th> <th>firstName</th> <th>ID</th> <th>nationality</th> <th>secondName</th> </tr> </thead> <tbody> <tr> <td>Generic Entity (1)</td> <td>Manchester</td> <td>08/23/1986 00:00</td> <td>Robert</td> <td>1</td> <td>British</td> <td>Smith</td> </tr> <tr> <td>Generic Entity (2)</td> <td>London</td> <td>09/26/1984 00:00</td> <td>Lisa</td> <td>2</td> <td>British</td> <td>Roberson</td> </tr> <tr> <td>Generic Entity (3)</td> <td>Vienna</td> <td>04/30/1979 01:00</td> <td>Hans</td> <td>3</td> <td>Austrian</td> <td>Hueber</td> </tr> <tr> <td>Generic Entity (4)</td> <td>Graz</td> <td>04/12/1990 00:00</td> <td>Maria</td> <td>4</td> <td>Austrian</td> <td>Goesser</td> </tr> <tr> <td>Generic Entity (5)</td> <td>Detroit</td> <td>03/18/1992 00:00</td> <td>Peter</td> <td>5</td> <td>American</td> <td>Stubbs</td> </tr> </tbody> </table>	Generic Entity	birthplace	dateOfBirth	firstName	ID	nationality	secondName	Generic Entity (1)	Manchester	08/23/1986 00:00	Robert	1	British	Smith	Generic Entity (2)	London	09/26/1984 00:00	Lisa	2	British	Roberson	Generic Entity (3)	Vienna	04/30/1979 01:00	Hans	3	Austrian	Hueber	Generic Entity (4)	Graz	04/12/1990 00:00	Maria	4	Austrian	Goesser	Generic Entity (5)	Detroit	03/18/1992 00:00	Peter	5	American	Stubbs
Generic Entity	birthplace	dateOfBirth	firstName	ID	nationality	secondName																																					
Generic Entity (1)	Manchester	08/23/1986 00:00	Robert	1	British	Smith																																					
Generic Entity (2)	London	09/26/1984 00:00	Lisa	2	British	Roberson																																					
Generic Entity (3)	Vienna	04/30/1979 01:00	Hans	3	Austrian	Hueber																																					
Generic Entity (4)	Graz	04/12/1990 00:00	Maria	4	Austrian	Goesser																																					
Generic Entity (5)	Detroit	03/18/1992 00:00	Peter	5	American	Stubbs																																					

Add Data to tribefire

You can also enter data using tribefire Explorer which will be stored in the MySQL database.

Step #	Task
1	<p>Deselect any currently selected instances from the person query.</p> 

2	<p>Click the Create New Generic Entity button.</p> 
3	<p>Enter some data for this new person.</p> 
4	<p>Click Save to persist this change in tribefire, and also to store it in the MySQL database.</p> 
5	<p>Refresh the Person Query with the magnifying glass and you should see that your new entry has been added.</p> 
6	<p>If we were to view the data in MySQL, by using the Select Rows - Limit 1000 option, like we did previously, we can see it has been added to the database.</p> 

Create a new Access

Table of Contents

- ▼ Expand / collapse
 - Introduction
 - Create a New access

Introduction

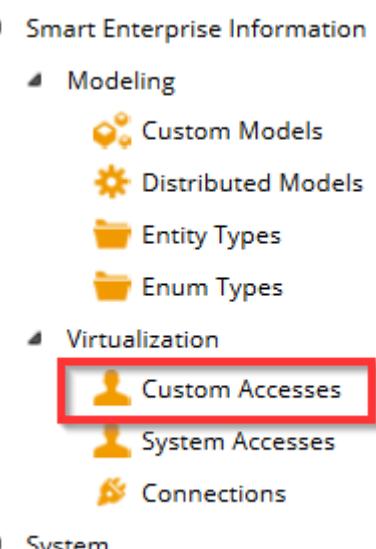
An access is needed when connecting data from an underlying repository to a model. After creating a new access, you can access it by selecting the cog icon. This will display a list of all available accesses available in tribefire, and which you have permission to access, from which you can select the one you need.

Depending on the type of repository from which your Model will be, you will create a the corresponding Access. This document will show you how to create an Access using the Smood database. However, regardless of what type of Access you wish to create, the method remains the same.

 Before you can create a new access, you must already have a model ready in tribefire. You can review which models you have by selecting Distributed Models , Custom Models or import a new model from a Zargo file.

Create a New access

Once you have a model ready, you can now create a new integration access.

Step #	Task
1	<p>Click Custom Accesses entry on the workbench panel.</p>  <ul style="list-style-type: none"> ▲ Smart Enterprise Information ▲ Modeling <ul style="list-style-type: none">  Custom Models  Distributed Models  Entity Types  Enum Types ▲ Virtualization <ul style="list-style-type: none">  Custom Accesses  System Accesses  Connections ▲ System

2

The Custom Accesses Query will then be displayed. This page displays all the Custom Accesses that have already been defined for your instance of tribefire.

If an Integration Access is already selected when the query opens, deselect it.

i You can deselect an object in the main details panel by holding down CTRL and clicking on the object you wish to select.

Once deselected, you will notice that the buttons displayed in the buttons panel have changed. Select the Create New Access button

ID	External ID	Name
3	BTModel	BTModel
6	SalesModel	SalesModel
1	SampleAccess	SampleAccess

3

The Select Access Type window will be displayed. There are many different types Access types available, in this documentation we will use the SmoodAccess type.

Select the SmoodAccess option and click Ok.

Types

- Access
- BloombergAccess
- DocumentumAccess
- FacebookAccess
- GSC DSC Challenge - DummyAccess
- GoogleCalendarAccess
- GoogleMailAccess
- HibernateAccess
- IncrementalAccess
- JiraAccess
- NonIncrementalAccess
- SAPAccess
- SalesforceAccess
- SharePointAccess
- SkypeAccess
- SmartAccess
- SmartEnterpriseInformationAccess
- SmoodAccess**
- TwitterAccess
- XmlAccess

4

The Access properties window will appear, allowing you to define the properties for the new Smood Access. At this stage, we need only define the External Id and Name fields, and also a description if you wish.

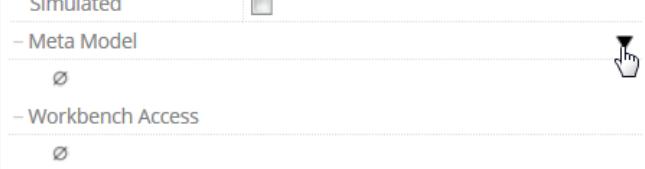
Field	Description
Name	Used in tribefire to refer to the particular access
External Id	Used to access this Access, either from an external app, a URL call in a browser or by the accesses panel
Description	Used only for reference purposes. This helps other users understand what this access does

5

Enter any name you wish for these two properties and click Apply

6

The new Smood Access will be created. However, until the save button is clicked it will not be persisted in tribefire. Before, clicking save there is one more property that must be configured for this Integration Access; we must link this Access with a model. We do this by adding an existing model to the model field in the properties panel.

7	<p>To configure this property, mouse hover over the <i>Meta Model</i> field so that the context-menu icon appears ()</p> <p>Access</p>  <p>The screenshot shows a list of access configurations under the heading 'Access'. It includes sections for 'Simulated' (with a dropdown arrow), 'Meta Model' (with a context-menu icon), 'Workbench Access' (empty), and another 'Workbench Access' section (empty). A small yellow arrow points to the context-menu icon on the right side of the 'Meta Model' row.</p>
8	<p>Clicking on the icon will display the context-menu. As the name suggests, the menu options which are displayed depend on the context. Click Assign.</p> <p>Access</p>  <p>The screenshot shows the same 'Access' configuration interface as above, but with a context-menu open over the 'Meta Model' section. The menu items '+ Assign' and '<> Change' are visible, with '+ Assign' being highlighted by a yellow box and a cursor icon pointing to it.</p>

9

Selecting Assign will display the Selection Constellation, allowing you to select an available model. You can either select your model by using the Filter Page (this displays an unordered list of models) or clicking the Model Query tab at the top of the window. This functions like any other query, allowing you to use the search panel to further refine the search.

Filter panel

The screenshot shows the 'Model Query' interface. At the top, there are icons for Home, Filter, Refresh, Save, and Import. A yellow box highlights the 'Model Query' tab. Below the tabs is a search bar. Underneath are two sections: 'Values' and 'Types'. The 'Values' section lists several models: BasicDeploymentWorkbenchModel#1.0 - GmMetaModel, BtTrainingModel#1.0 - GmMetaModel, BtTraining_WorkbenchModel - GmMetaModel, StevesModel-1.0 - GmMetaModel, and WorkbenchModel#2.0 - GmMetaModel. The 'Types' section has one item: Model - GmMetaModel, which is also highlighted with a yellow box. Below these are sections for 'Actions' and a search bar.

Model Query Panel

The screenshot shows the 'Model Query' interface with a search bar at the top. Below it is a table with columns for 'Model', 'Name', and 'Id'. The table lists several models: WorkbenchModel#2.0, BasicDeploymentModel#1.0, BasicDeploymentWorkbenchModel#1.0, BtTrainingModel#1.0, SalesModel#1.0, BtTraining_WorkbenchModel, and StevesModel-1.0. The 'Name' column shows the full class name for each model. The 'Id' column shows numerical values from 1 to 8. A yellow box highlights the 'BtTrainingModel#1.0' entry in the table.

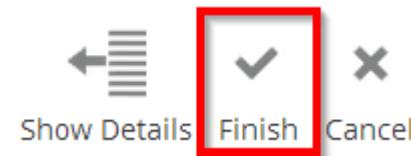
10

Regardless of the method, select a Model you would like this access to be associated with.

The screenshot shows the 'Model Query' interface with a search bar at the top. Below it is a table with columns for 'Model', 'Name', and 'Id'. The table lists several models: WorkbenchModel#2.0, BasicDeploymentModel#1.0, BasicDeploymentWorkbenchModel#1.0, BtTrainingModel#1.0, SalesModel#1.0, BtTraining_WorkbenchModel, and StevesModel-1.0. The 'Name' column shows the full class name for each model. The 'Id' column shows numerical values from 1 to 8. A yellow box highlights the 'BtTrainingModel#1.0' entry in the table.

11

Once you have selected the model, click the Finish button to complete the action.



12

The Model will now be linked to the Access. Click the Save button to persist your new Access to tribefire. If successful a message will be displayed at the top of tribefire.

saved successful 

Your new access has been created.

Create a Simple Query

Table of Contents

Expand / collapse

- Introduction
- Create new Simple Query Action
- Placing Simple Query Action in your main access
 - Button in another context
 - Left-hand Panel link

Introduction

A simple query action allows you to build simple queries based on a model's type. Building and executing this type of query will display all data associated with this type (for example, all employee data of an employee type). To refine these search results you must use the search panel after the query has been executed.

 If you would like to execute a query that places restrictions on a query during execution, you can use the Template Query Action object

Create new Simple Query Action

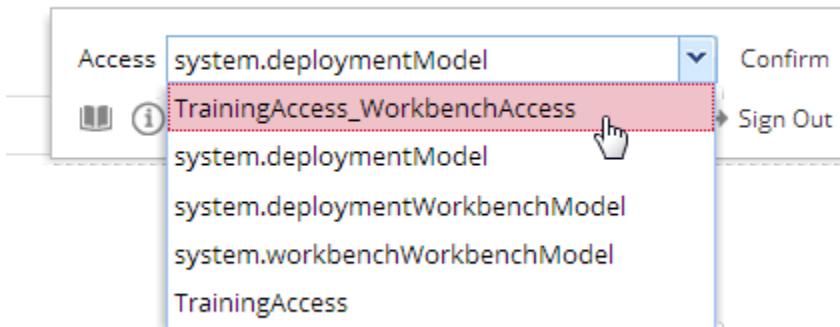
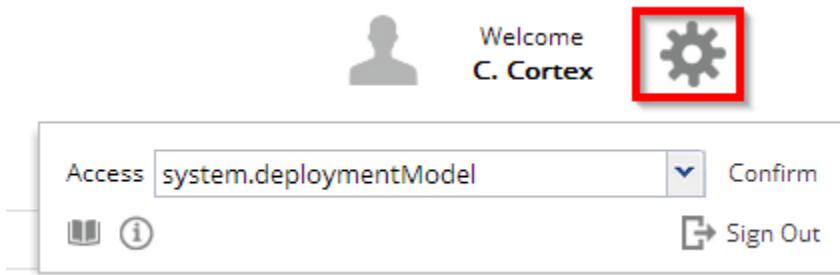
 Because of tribefire cortex's flexible and self-descriptive nature, the screenshots below will look different to the way your accesses are displayed. The following screenshots are used merely to give an example of the Simple Query Action's functionality.

This documentation uses the following parameters

- System Model - TrainingModel#2.0
- Integration Access (SMOOD) - TrainingAccess
- Workbench Access - TrainingAccess_WorkbenchAccess

To create a simple query action for your main Access, you will first have to [create a workbench access](#).

Navigate to your workbench access by selecting it from the cog icon at the top-right hand portion of tribefire GME.



General

Click Confirm to load your WorkbenchAccess

The left-hand side of the tribefire Explorer interface is the Workbench panel. It displays all the types and values associated with this WorkbenchAccess. You can search these types by using the search field (labeled *Quick Access*). The panel shows a list of types under the 'Types' section, including AbsenceInformation, AbsentingManipulation, AbstractJunction, AbstractOperation, AbstractResourceBundle, AbstractSecurityPolicy, AbstractTypeCondition, ActionRequest, ActionResponse, AdaptiveIcon, Address, AnyTypeCondition, AnyTypeMatch, AtomicManipulation, and Attendee. There are also sections for 'Values' and 'Actions'.

To the left-hand side of tribefire Explorer is the Workbench panel. This displays all the types and values associated with this WorkbenchAccess. You can search these types by using the search field (labeled *Quick Access*).

In the *Quick Access* search panel enter: SimpleQueryAction

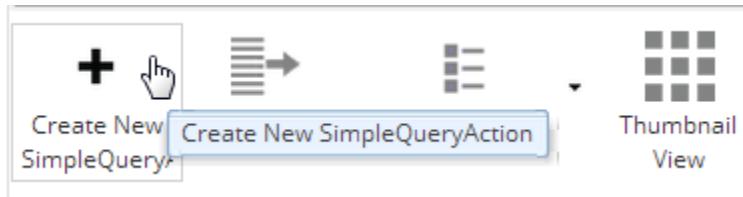
The search results for 'SimpleQueryAction' show one result: 'SimpleQueryAction'. This result is selected and highlighted in orange. A warning message box states: 'The name of this object is SimpleQueryAction. Because the search only finds results based on name, you must enter this exactly; entering Simple Query Action (with spaces) won't work.'

Double click on the SimpleQueryAction result that appears. This will execute a query and will display all, if any, Simple Queries which already exist in this workbench access.

The search results for 'SimpleQueryAction' show one result: 'SimpleQueryAction (1)'. This result is selected and highlighted in orange. The properties panel on the right shows 'SimpleQueryAction's Properties' with fields: displayName (TrainingQuery), id (1), typeSignature (braintribe.model.TrainingQuery), and resources (10).

To create a new SimpleQueryAction, click on the Create New SimpleQueryAction button located at the bottom of tribefire GME. If any SimpleQueryActions have already been created, you must first deselect the currently selected QueryAction.

- To deselect an Object in tribefire Explorer, hold down the CTRL key and click on the selected object.



After clicking the create new SimpleQueryAction button, the GIMA will be displayed.

You will be prompted to enter two parameters: displayName and typeSignature.

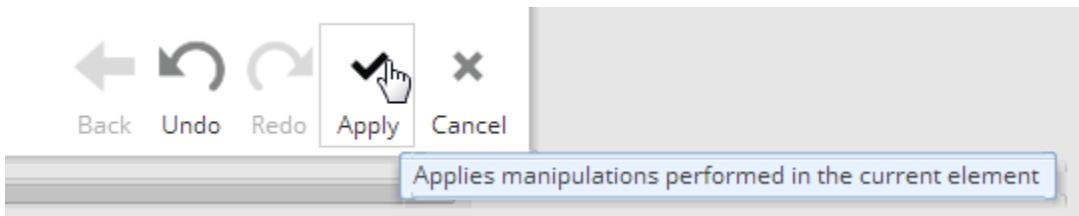
Property	Description	Example
displayName	The name of the SimpleQueryAction, which will be displayed in your main access	Attendee
typeSignature	The full name of the entity that you wish your Query to search on.	braintripe.model.Attendee

- If you are unsure of the typeSignature for the entity you want to base your query on, you can view it using tribefire Control Center by navigating to the entity you query. Included in the entity's properties is the field Type Signature.

Enter the correct variables into the parameters...

The screenshot shows the 'SimpleQueryAction ()' dialog. In the 'SimpleQueryAction's Properties' section, there are two fields: 'displayName' set to 'Attendee' and 'typeSignature' set to 'braintribte.model.Attendee'. The 'typeSignature' field has a red underline, indicating it is being edited.

...and click Apply.



Your new SimpleQueryAction will be created.

The screenshot shows the GME workspace with a new element named 'SimpleQueryAction (15)' selected. The details panel on the right shows its properties: displayName is 'Attendee', id is '15', and typeSignature is 'braintribte.model.Attendee'. The 'SimpleQueryAction's Properties' section also lists these values. Other properties shown are 'icon' (empty) and 'inplaceContextCriterion' (empty).

To persist this query action in tribefire, you must click on the save button located in the details panel to the bottom-right of tribefire GME



Placing Simple Query Action in your main access

You have two main possibilities when it comes to displaying your Simple Query Action in your main access, either in the left-hand panel or as a button in a specific context.

Button in another context

To place you this Simple Query Action in another context you use the `inplaceContextCriterion`. This determines the location of where this `SimpleQueryAction` should be placed.

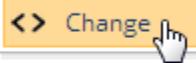
Use to mouse to hover over the `inplaceContextCriterion` property so that the context-menu icon appears..

SimpleQueryAction's Properties

displayName	Attendee
<i>id</i>	15
typeSignature	braintribe.model.Attendee
- icon	∅
- inplaceContextCriterion	∅ 

...select change

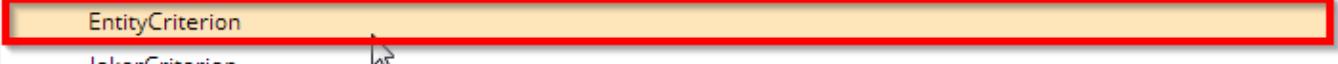
- inplaceContextCriterion

∅	 Change 
Changes the given property value	

 You can also double-click on a property's field to automatically open the change window.

The Criterion selection will be displayed. Select the property *EntityCriterion* from the Types sub-section

Types

- BasicCriterion
- ConjunctionCriterion
- DisjunctionCriterion
- EntityCriterion** 
- JokerCriterion 
- JunctionCriterion
- ListElementCriterion

Click finish and a new EntityCriterion will be saved to the inplaceContextCriterion

- inplaceContextCriterion

EntityCriterion ()

Double Click on the EntityCriterion property to open it for editing

	Strategy	Type Signature	Id
EntityCriterion	∅	∅	∅
EntityCriterion ()	∅	∅	∅
	/d	∅	
	Strategy	∅	
	Type Signature	∅	

We must now add a new Type Signature to define this EntityCriterion. The type signature that is entered here is where the Simple Action Query will be placed in your main access.

⚠ In this example, we shall add the typeSignature braintribe.model.Training. This is the full name of the Training entity belonging to the TrainingModel.

Enter the appropriate value in Type Signature...

EntityCriterion's Properties

<i>Id</i>	∅
Strategy	∅
Type Signature	braintribe.model.Training

...and then click save to persist your changes.



Your newly created Simple Action Query will now be placed as a button in the context which you defined in your main access:

clicking the button will execute the search:

The screenshot shows the 'Attendee Query' interface. At the top, there are tabs for 'TrainingQuery Query' and 'Attendee Query'. Below the tabs is a search bar with a magnifying glass icon and a results counter '1'. To the right of the search bar are buttons for 'Ordered by: Id' (with a dropdown arrow), 'Results per page: 25' (with a dropdown arrow), and 'Switch to Advanced'. The main area is titled 'Attendee' and contains a table with the following data:

	Email Address	First Name	Job Title	Second Name	Id
Attendee (2)	johnsmith@email.com	John	Programmer	Smith	1
Attendee (5)	maryAnn@email.com	Mary	Support	Ann	2
Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy	3
Attendee (7)	TonyStark@Iman.com	Tony	Engineer	Stark	4
Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper	5
Attendee (11)	DavidSilva@msn.com	David	Support	Silva	6
Attendee (12)	OliviaFog@Iman.com	Olivia	Technical Writer	Fog	7

Left-hand Panel link

To place your newly created SimpleQueryAction as a link on the left-hand panel of your main access you must create a new folder.

In the search main of your workbench access enter *Folder...*

The screenshot shows the search interface with a search bar containing the text 'folder'. Above the search bar are icons for 'Query', 'List', and 'Advanced'. Below the search bar is a section titled 'Types' with a 'Folder' button highlighted in orange. Underneath 'Folder' is the 'FolderContent' link.

...and double click on the Folder result to execute the Folder Query. This will display all the folders, if any, that have been created in this workbench access.

The screenshot shows the 'Folder Query' interface. At the top, there are tabs for 'TrainingQuery Query' and 'Folder Query'. Below the tabs is a search bar with a magnifying glass icon and a results counter '1'. To the right of the search bar are buttons for 'Ordered by: id' (with a dropdown arrow), 'Results per page: 25' (with a dropdown arrow), and 'Switch to Advanced'. The main area is titled 'Folder' and contains a table with the following data:

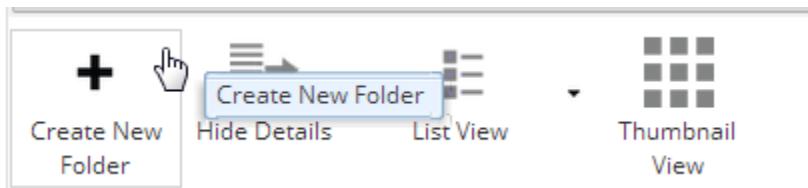
	displayName	id	name
Folder (1)	Queries	1	queries
Folder (2)	Training	2	training
Folder (4)	Company	4	company
Folder (5)	Address	5	address
Folder (6)	Resources	6	Resources

If any Folders have already been created, you must first deselect the currently selected Folder.



To quickly deselect an Object in tribefire GME, hold down the CTRL key and click on the selected object.

Click on the Create New Folder to display the create window.



displayName	Attendee
name	attendee

You will be prompted to enter information for two parameters: displayName and name.

Property	Description	Example
displayName	The information entered here will be used when displaying the folder in your main access	Attendee
name	The information entered here will be used as reference for this folder	attendee

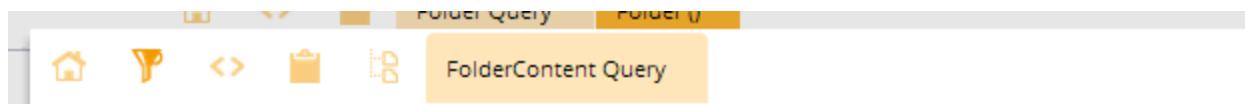
Enter the correct information and click Apply. A new folder will be created.

displayName	Attendee
id	Ø
name	attendee

In the properties panel you can configure this folder. The different properties allow you to enter security protocols, an icon, and any subfolders. However, since the scope of this document is only to add a Simple Query Action to this folder, we need only concern ourselves with one property at the moment: content.

- content	Ø
-----------	---

Double Click the content field (where the circle with a line through is displayed) or mouse hover over the property till the context-menu icon is display and click change, so that the Selection Constellation appears.



The screenshot shows a software interface with a toolbar at the top containing icons for Home, Filter, Copy, Paste, and others. The 'FolderContent Query' tab is highlighted in yellow. Below the toolbar is a search bar with placeholder text 'Type for filtering types and values...'. Underneath the search bar, there are two sections: 'Values' and 'Types'. The 'Values' section lists several action types: CustomAction (11) - CustomAction, SimpleInstantiationAction (5) - SimpleInstantiationAction, SimpleInstantiationAction (6) - SimpleInstantiationAction, SimpleInstantiationAction (8) - SimpleInstantiationAction, and TemplateInstantiationAction (9) - TemplateInstantiationAction. The 'Types' section lists various action types: CustomAction, FolderContent, HyperlinkAction, InstantiationAction, ModelLinkAction, PrototypeQueryAction, QueryAction, SimpleInstantiationAction, SimpleQueryAction, TemplateBasedAction, TemplateCustomAction, TemplateInstantiationAction, TemplateQueryAction, WidgetOpenerAction, and WorkbenchAction.

Values

- CustomAction (11) - CustomAction
- SimpleInstantiationAction (5) - SimpleInstantiationAction
- SimpleInstantiationAction (6) - SimpleInstantiationAction
- SimpleInstantiationAction (8) - SimpleInstantiationAction
- TemplateInstantiationAction (9) - TemplateInstantiationAction

Types

- CustomAction
- FolderContent*
- HyperlinkAction
- InstantiationAction*
- ModelLinkAction
- PrototypeQueryAction
- QueryAction*
- SimpleInstantiationAction
- SimpleQueryAction
- TemplateBasedAction*
- TemplateCustomAction
- TemplateInstantiationAction
- TemplateQueryAction
- WidgetOpenerAction*
- WorkbenchAction*

You can then search and select the SimpleQueryAction that is required.

- However, because this view is not particularly clear for selecting the instance you wish, you can also click on the FolderContentQuery tab at the top of the window. This will display all possible content that can be attached to this folder, along with information pertaining to each object.

The screenshot shows a list of 'FolderContent Query' items. The 'SimpleQueryAction (15)' item is highlighted with a red box. The list includes:

displayName	id
TrainingQuery	1
CompanyQuery	3
AddressQuery	4
Create New Attendee	5
Create New Company	6
Create New Address	7
Create New Training	8
Create New Address	9
resources	10
testAction	11
All Braintribe Employees	14
Attendee	15

Select the appropriate SimpleQueryAction and click finish. The query will now be added to the content property of your newly created folder.

Folder's Properties

displayName Attendee

id Ø

name attendee

– content

SimpleQueryAction (15)

– Denied For

Click save to persist this change to tribefire.



Your newly created Simple Action Query will no be placed on the left-hand menu in your main access:



Quick Access

Attendee

Queries

Address

Company

Training

Resources

Create a Template Query Action

Table of Contents

Expand / collapse

- Introduction
- Open Your Workbench
- Create a New Folder
- Create New Template Query

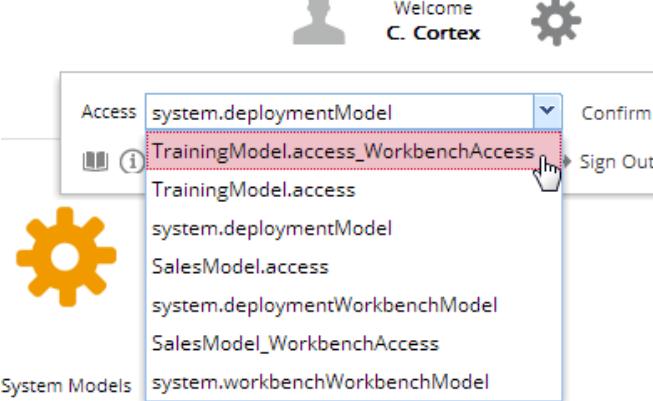
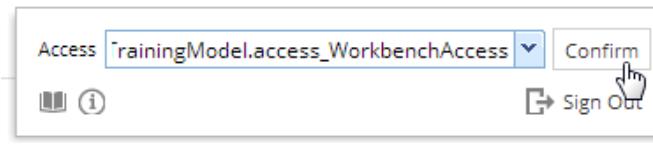
Introduction

When viewing your accesses in Explorer, you can create custom queries to display specific information. In tribefire GME there are two main types of queries: simple queries and template queries. A simple query is limited in its actions and will display all the information based on one entity type. With this query there are no options allowing you to filter this data and produce specific results. If you wish to filter your data, you must create a template query.

 To create a template query you must create a [Workbench](#) for your main access first. It is through this Workbench that any configuration, such as creating queries, are done.

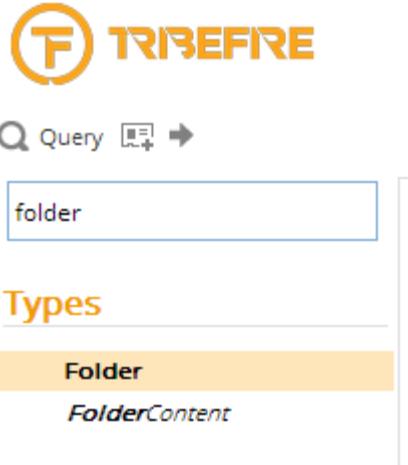
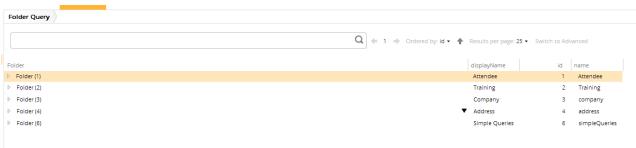
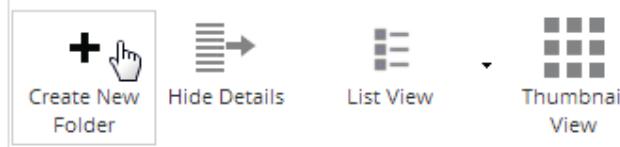
Open Your Workbench

The first step after creating your workbench is to access it so that you can create the query.

Step #	Task
1	<p>Select the cog icon at the top-right of tribefire GME</p> 
2	<p>Select the workbench that you created. That is, the workbench that will be used to configure the main access you wish the query to use...</p>  <p>...and click confirm. Your workbench access will be created.</p> 

Create a New Folder

The next step is to create a folder. The folder is the object which is displayed in the left-hand panel of your Data View.

Step #	Task
1	<p>In the search box, type the word <i>Folder</i>. The search field will dynamically search as you type and will display the <i>Folder</i> option</p> 
2	<p>Select the <i>Folder</i> option to open it. If you have any previously created folder, they will be displayed here.</p> 
3	<p>Deselect any highlighted folders... ...and click the Create New Folder Button.</p> 
4	<p>The GIMA will appear.</p> 

5	<p>Enter a displayName and a Name for your new folder.</p> <p>Folder's Properties</p> <table border="1"> <tr> <td>displayName</td> <td>The Display Name for your Query</td> </tr> <tr> <td>name</td> <td>The Name for your Query</td> </tr> </table> <p>i The display name is what will be shown in Explorer and the name is the internal reference for this folder instance.</p>	displayName	The Display Name for your Query	name	The Name for your Query
displayName	The Display Name for your Query				
name	The Name for your Query				
6	<p>Click Apply to create the folder...</p> <p>Back Undo Redo Apply Cancel</p> <p>...and then Save to persist it</p> <p>Upload Undo Redo Save</p>				

Create New Template Query

Step #	Task																				
1	<p>Select the context-menu icon for the content field...</p> <p>Folder's Properties</p> <table border="1"> <tr> <td>displayName</td> <td>The Display Name for your Query</td> </tr> <tr> <td><i>id</i></td> <td>Ø</td> </tr> <tr> <td>name</td> <td>The Name for your Query</td> </tr> <tr> <td>- content</td> <td>Ø</td> </tr> <tr> <td>- Denied For</td> <td>∅</td> </tr> </table> <p>...click to display the context-menu and then select change</p> <p>Folder's Properties</p> <table border="1"> <tr> <td>displayName</td> <td>The Display Name for your Query</td> </tr> <tr> <td><i>id</i></td> <td>Ø</td> </tr> <tr> <td>name</td> <td>The Name for your Query</td> </tr> <tr> <td>- content</td> <td>Ø</td> </tr> <tr> <td>- Denied For</td> <td>∅</td> </tr> </table>	displayName	The Display Name for your Query	<i>id</i>	Ø	name	The Name for your Query	- content	Ø	- Denied For	∅	displayName	The Display Name for your Query	<i>id</i>	Ø	name	The Name for your Query	- content	Ø	- Denied For	∅
displayName	The Display Name for your Query																				
<i>id</i>	Ø																				
name	The Name for your Query																				
- content	Ø																				
- Denied For	∅																				
displayName	The Display Name for your Query																				
<i>id</i>	Ø																				
name	The Name for your Query																				
- content	Ø																				
- Denied For	∅																				

2

The Selection Constellation will appear. Select the option TemplateQueryAction...

The screenshot shows a selection interface with various icons at the top: a house, a magnifying glass, a double arrow, a folder, and a trash can. The 'FolderContent Query' icon is highlighted with a yellow background. Below this is a search bar with the placeholder 'Type for filtering types and values...'. Under the heading 'Values', there is a list of four SimpleQueryAction items. Under the heading 'Types', there is a long list of action types, with 'TemplateQueryAction' highlighted by a yellow background and a cursor pointing at it. Other listed types include CustomAction, FolderContent, HyperlinkAction, InstantiationAction, ModelLinkAction, PrototypeQueryAction, QueryAction, SimpleInstantiationAction, SimpleQueryAction, TemplateBasedAction, TemplateCustomAction, TemplateInstantiationAction, WidgetOpenerAction, and WorkbenchAction.

...and click Finish.



3

A new TemplateQueryAction will be created.

Folder's Properties

displayName	The Display Name for your Query
<i>id</i>	Ø
name	The Name for your Query
– content	
TemplateQueryAction ()	
– Denied For	
Ø	

4

Click on TemplateQueryAction to open it.

The screenshot shows a detailed properties dialog for a TemplateQueryAction. It includes sections for 'TemplateQueryAction's Properties' (with fields for displayName, formula, id, name, implementationCriteria, and template), 'TemplateQueryAction' (with fields for content, id, name, and type), and 'TemplateQueryAction (1)' (with fields for content, id, name, and type). The 'TemplateQueryAction (1)' section is expanded, showing its properties: content (TemplateQueryAction (1)), id (Ø), name (TemplateQueryAction (1)), and type (TemplateQueryAction).

5

Click the context-menu icon for the template property and then select change.

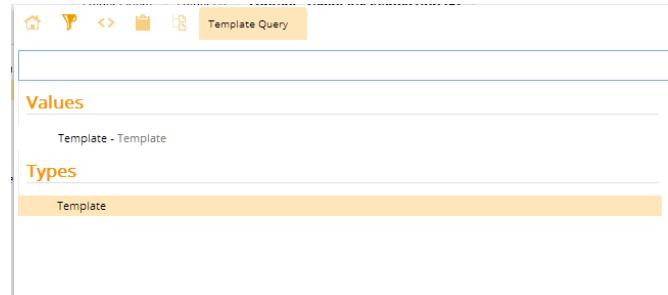
TemplateQueryAction's Properties

displayName	Ø
forceFormular	<input type="checkbox"/>
<i>id</i>	5
- icon	Ø
- inplaceContextCriterion	Ø
- template	Ø

<> Change 

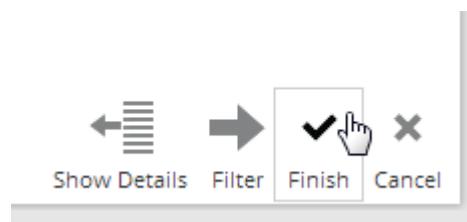
6

Select the type Template...



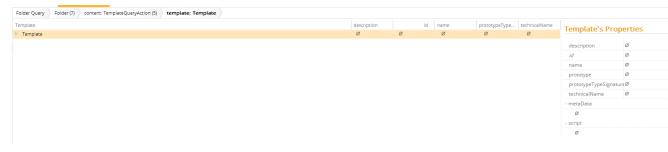
The screenshot shows a selection dialog with a toolbar at the top featuring icons for home, search, filter, and a 'Template Query' button. Below the toolbar, there are two sections: 'Values' and 'Types'. Under 'Values', there is a single entry: 'Template - Template'. Under 'Types', there is a list of items, with 'Template' highlighted by a yellow bar. At the bottom of the dialog are four buttons: 'Show Details', 'Filter', 'Finish', and 'Cancel'. A hand cursor is hovering over the 'Finish' button.

...and then click Finish.



7

Click on Template to open it.



The screenshot shows a list view titled 'Template' with a single item named 'Template'. The list includes columns for 'description', 'id', 'name', 'prototype', and 'templateName'. To the right of the list is a table titled 'Template's Properties' with several rows of properties. A hand cursor is hovering over the 'Template' item in the list.

8

Click on the context-menu icon for the property prototype and then select Change to Existing.

Template's Properties

description	Ø
<i>id</i>	Ø
name	Ø
prototype	Ø
prototypeTypeSignature	Ø
technicalName	Ø
- metaData	
	Ø
- script	
	Ø

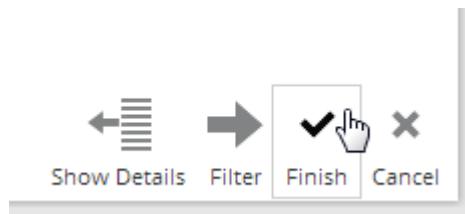
Change to Existing

Change to New

9

The Selection Constellation will appear. Search for the type EntityQuery and then select it...

...then select Finish.



10

Click on EntityQuery to open it.

11

Click the Save button to persist the query action.



You have now created the basis for a template query. The next step is define a query that you wish to use.

For more information on how to define a query and the different types of queries and configuration available.

Create Custom Workbench Access

Table of Contents

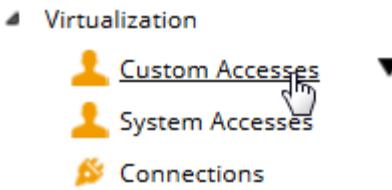
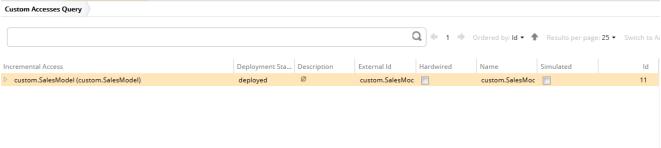
Expand / collapse

- Introduction
- Creating a Workbench

Introduction

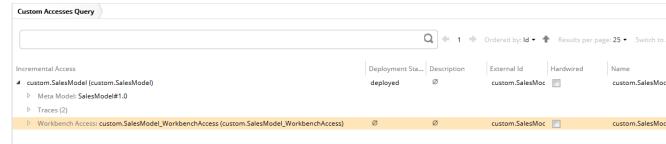
A workbench is an access which allows you to manipulate how other accesses behave. Once you have created a new workbench, you can use it to add links, queries and actions to the main access that it is associated with. The setting up of a workbench is a simple process which requires no configuration. You must only click a button and then deploy it. After deployment, the workbench will appear in the access selection list, just like any other access that you have defined.

Creating a Workbench

Step #	Task
1	<p>Click the Custom Accesses link on the Workbench panel.</p> 
2	<p>The corresponding query will then be opened. Select the access that you would like to create a workbench for.</p> 
3	<p>Click the Create Workbench Access button located at the bottom of tribefire Control Center...</p>  <p>A new Workbench Access will be created.</p>

4

Click the expand icon  on the access you've just created the workbench for to see further properties. You will see that a workbench has been created for this access.



The screenshot shows a table titled 'Custom Accesses Query'. It lists several entries under 'Incremental Access'. One entry is highlighted in orange: 'custom.SalesModel (custom.SalesModel) > Workbench Access custom.SalesModel_WorkbenchAccess (custom.SalesModel_WorkbenchAccess)'. The 'Deployment Status' column shows 'deployed' for this entry. The 'External Id' column contains 'custom.SalesMod' and 'custom.SalesMod'. The 'Name' column also contains 'custom.SalesMod'.



When creating a new workbench access, it will always follow the same naming convention. That is, the external id and the name will be the same and will be the `accessName_WorkbenchAccess`.

5

The next step is to deploy the workbench access. With the newly created workbench selected, click the deploy button located at the bottom of the page...

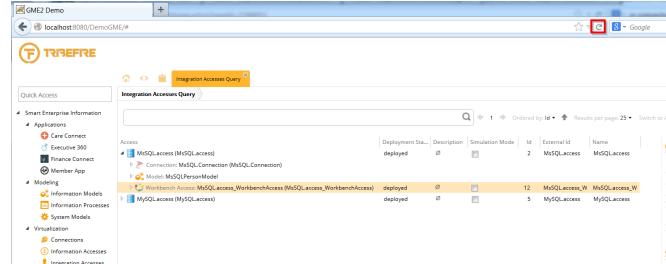


...if your workbench is deployed, you will receive a success message.

Successfully executed Action. 

6

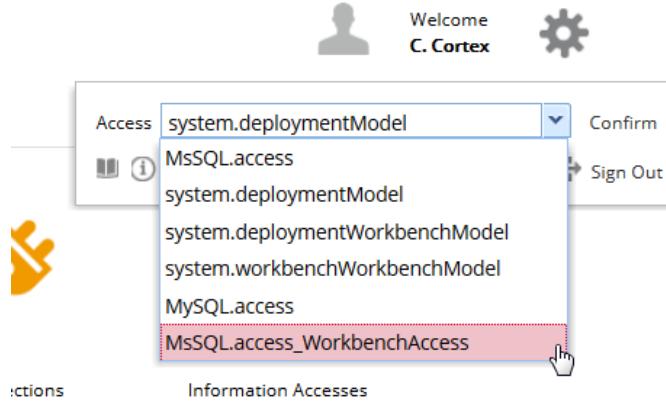
Refresh tribefire GME and then login.



The screenshot shows a table titled 'Integration Accesses Query'. It lists several entries under 'Access'. One entry is highlighted in orange: 'MsSQL.access (MsSQL.access) > Connection MsSQL_Connection (MsSQL_Connection) > Access MsSQL_Previewer (MsSQL_Previewer)'. The 'Deployment Status' column shows 'deployed' for this entry. The 'External Id' column contains '2' and 'MsSQL.access'. The 'Name' column also contains 'MsSQL.access'.

7

Your newly created Workbench Access will be available in the access selection list.



The screenshot shows a dropdown menu for 'Information Accesses'. The options listed are: 'Access system.deploymentModel', 'MsSQL.access', 'system.deploymentModel', 'system.deploymentWorkbenchModel', 'system.workbenchWorkbenchModel', 'MySQL.access', and 'MsSQL.access_WorkbenchAccess'. The 'MsSQL.access_WorkbenchAccess' option is highlighted with a red box and has a cursor pointing at it.

Create Model From Zargo File

Table of Contents

▼ Expand / collapse

- Introduction
- Import a Zargo File
- Create Model from a Zargo File
- Verifying That A New Model has been Created

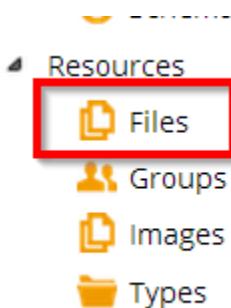
Introduction

If you have a Model that was built in Zargo, then you can upload it directly to tribefire Control Center and it will automatically convert it into either a [System Model](#) or an [Information Model](#), depending on the context of the Model.

Import a Zargo File

The first step in creating a Model from a Zargo file is to upload it into the tribefire system.

Click on the Files link in the left-hand menu.



i Although in this documentation we will use the Files Query to upload a file, you can import a file from any of the Property panels displayed in tribefire. At the bottom of every Detail Panel are a series of buttons, including Upload. Clicking this button will display the Upload window, which you can use to upload a file, image or Zargo file.

The Files Query will be executed and all files available to this instance of tribefire will be displayed.

Files Query

RawResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resolver URI	Width In Cm
MetaDataExtract-com.braintribe.r	1831693	0.00	7cd5803d553464	application/xml	0	Ø	0.00
samplePDF.pdf	29309	0.00	a9449436dd70e930eacac8ab229383b2	application/pdf	0	Ø	0.00

RawResource's Properties

<i>/</i>	RN_110
File Size	29309
Height In Cm	0.00
Md5	a9449436dd70e930eacac8ab229383b2
Mime Type	application/pdf
Page Count	0
Resolver URI	Ø
Width In Cm	0.00

- Resource Source
StaticSource (RS_110)

- Security Policy
Ø

- Tags
Ø

Info

Created	02/20/2014 09:45
Creator	cortex
Name	samplePDF.pdf

Open Edit Delete Refresh Hide Details Download Resource Import Model from Zargo Add To Clipboard List View Thumbnail View

Upload Undo Redo Save

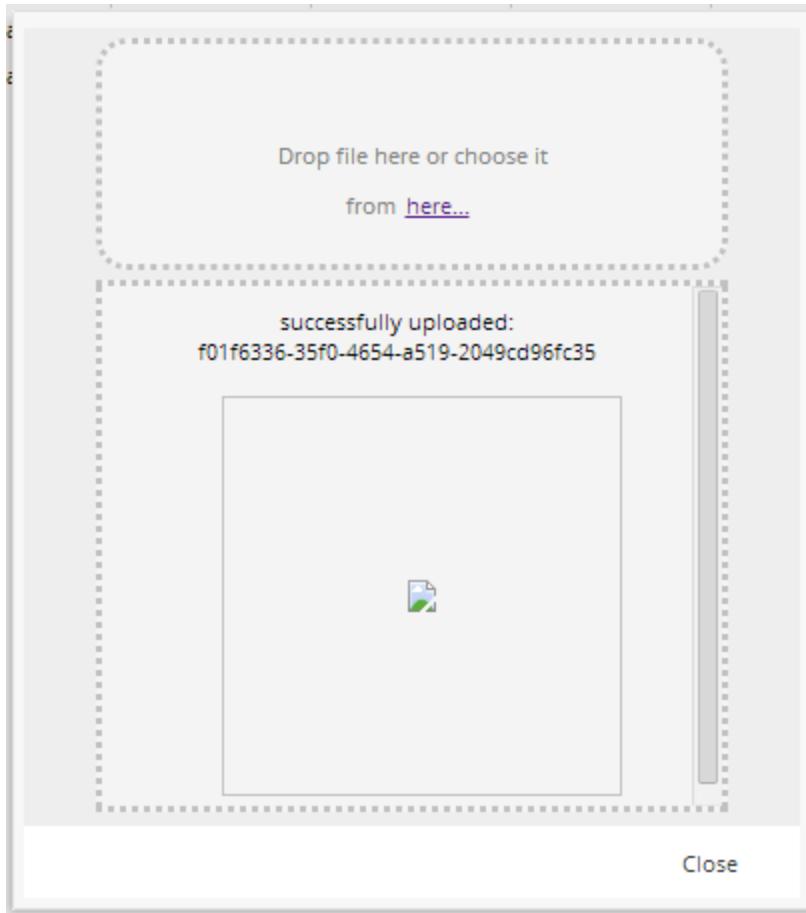
In the details panel, located at the right-hand side of tribefire GME, there are four buttons: Upload, Undo, Redo and Save. Click the Upload button to upload a file.



This will display the file upload window. You can upload a file in one of two ways: either drag the file you wish to upload and drop it on this window or you can click the [here...](#) link and browse to the file you want. Uploading the Zargo file works in the exact same way as any other file.



Once you have uploaded your Zargo file, and it has been accepted by tribefire, you will receive a success message.



Click the close button to close the upload window.

Create Model from a Zargo File

After closing the File Upload Window the Files Query will be displayed again. Click on the Magnifying glass icon to refresh the search. Your newly uploaded model will have been added to tribefire. The files query will display all the relevant information about this file. Including, what type of file it is, its size, its content ID, who uploaded it and the name of the file.

Files Query							
RawResource	File Size	Height in Cm	Mds	Mime Type	Page Count	Resolver URI	Width In Cm
MetaDataExtract-com.braintribe.r	1831693	0.00	7cd5803d553464	application/xml	0	Ø	0.00
samplePDF.pdf	29309	0.00	a9449436dd70e9	application/pdf	0	Ø	0.00
SalesModel-1.0.zargo	56429	0.00	e2171d6eef310a	application/zip	0	Ø	0.00

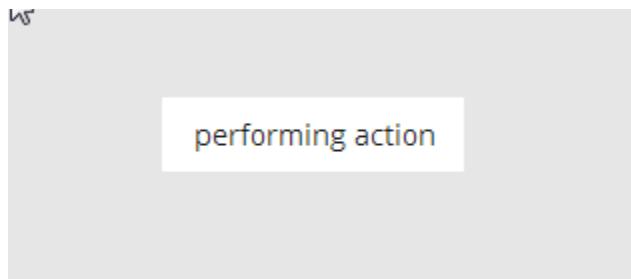
Select the newly uploaded Zargo file by clicking on it....

RawResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resolver URI	Width In Cm
MetaDataExtract-com.braintribe.r	1831693	0.00	7cd5803d553464	application/xml	0	Ø	0.00
samplePDF.pdf	29309	0.00	a9449436dd70e9	application/pdf	0	Ø	0.00
SalesModel-1.0.zargo	56429	0.00	e2171d6eef310a	application/zip	0	Ø	0.00

...and then click the button **Import Model from Zargo**.



You will receive a performing action display and then either a success message or a failed message, depending on the status of the execution.



Successfully executed Action. ✘

If successful your mode has been created.

Verifying That A New Model has been Created

You can check whether your model has been created or not by clicking either on Custom Models or Distributed Models, depending on the context of your Model, displayed in the left-hand menu.

-  [Custom Models](#)
-  [Distributed Models](#)
-  [Entity Types](#)
-  [Enum Types](#)

These links will display all models that you can use in this instance of tribefire. Your newly created model should be located there in one of the links.

Custom Models Query

Q 1 Ordered by: Id ▼ ↑ Results per page: 25 ▼ Switch to A

	Name	Id
▶ personModel	personModel	14
▶ EvnModel#2.0	com.braintribe.evn	13
▶ SalesModel#1.0	com.braintribe.sm	10

Create New Properties

Table of Contents

▼ Expand / collapse

- Introduction
- Adding a new Property

Introduction

Each Model contains a number of types and enums. Attached to these types are properties, which can be either Simple or Complex. Simple properties are Strings, Booleans, Integers, Long and Date. They are basically any type which holds simple information and are not related to another type. Complex properties are properties which relate to another type. For example, if you have a company type it will likely have a property for employees. The employee would be another type attached to the Model. A third type also exists: the collection type, such as a map or list, for example.

a simple property type - in this example, we have a property called firstName of the simple type String

The screenshot shows a software interface for managing properties. At the top, it says "GmProperty's Properties". Below that is a table with four rows:

<i>Id</i>	3511
Is Id	<input type="checkbox"/>
isOverlay	<input type="checkbox"/>
Name	firstName

Below the table are several sections:

- Entity Type: Attendee
- Meta Data: \emptyset
- Type: string

The "Type" field is highlighted with a red border.

a complex property type - in this example we have a property called Company which is of the type Company

GmProperty's Properties

<i>Id</i>	3509
Is Id	<input type="checkbox"/>
isOverlay	<input type="checkbox"/>
Name	company
- Entity Type	
Attendee	
- Meta Data	
Ø	
- Type	
Company	

if we look at the entity types for this Model, we can see that Company is another entity type, complete with its own properties

- ▲ Entity Types (5)
 - ▷ Address
 - ▷ Company
 - ▷ StandardIdentifiable
 - ▷ Attendee
 - ▷ Training

After uploading a model and deploying it, there exist the possibility of adding new properties to any entity type. You can add any new property by navigating to the Models link at the left-hand side of Control Center and selecting either Custom Models or Distributed Models, depending on the type of Model that you wish to edit.

Adding a new Property

In this example we will add a new Property *Date of Birth* to the entity type Attendee of the Model *Training Model*. This will be of the simple type DOB.

Navigate to the Model which you wish to add the property to.

System Models Query

Model

- ▷ btTrainingAccess_WorkbenchModel
- ▷ WorkbenchModel#2.0
- ▷ joinModel.access_WorkbenchModel
- ▷ TrainingModel#2.0
- ▷ training.access_WorkbenchModel
- ▷ JoinModel#1.0
- ▷ BasicDeploymentWorkbenchModel#1.0
- ▷ BasicDeploymentModel#1.0

Click on the expand icon to see this models properties. Expand the Entity types properties and then expand the entity type to which you would like to add your new property.

▲ TrainingModel#2.0

baseType: Ø

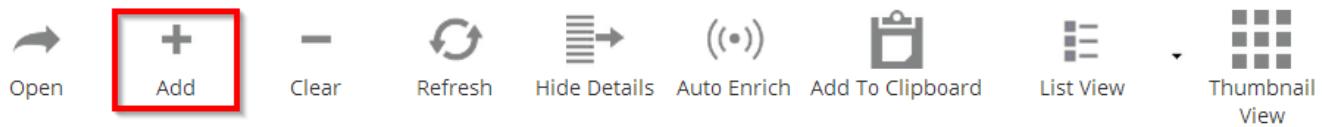
▲ Entity Types (4 ...)

- ▷ Address
- ▷ Company
- ▷ StandardIdentifiable
- ▲ Attendee
 - ▷ Artifact Binding:
 - ▷ Meta Data

▲ Properties (7)

- ▷ firstName
- ▷ secondName
- ▷ jobTitle
- ▷ emailAddress
- ▷ company
- ▷ address
- ▷ attended

At the button of tribefire GME is the buttons panel. Click Add to display the create new property window.



You can search for a preexisting property by using the search bar at the top of the Selection Constellation. However, since we want to create a new property, we select the GmProperty object display in the Types section.

The screenshot shows the tribefire GME interface with a search bar at the top labeled 'GmProperty Query'. Below the search bar, there are three main sections: 'Values' (containing descriptions, id, inverseManipulation, password, selector), 'Types' (containing GmProperty, which is highlighted with a red box), and 'Actions' (containing My Access Query and My Query). A red box highlights the search bar. Two arrows point from the 'GmProperty' entry in the 'Types' section to the search bar and the 'Add' button in the toolbar below.

Select GmProperty and then click the Add and Finish button. A new property will be created

The screenshot shows the tribefire GME interface with a 'Model' tree on the left. The tree includes 'TrainingModel#2.0' and its sub-entities: Entity Types (Address, Company, StandardIdentifiable), Attendee, and Properties (firstName, secondName, jobTitle, emailAddress, company, address, attended). A new property field is also listed. On the right, a 'GmProperty's Properties' dialog is open, showing fields for Id (4), Is Id (checkbox), isOverlay (checkbox), Name (checkbox), Entity Type (checkbox), Meta Data (checkbox), and Type (checkbox).

Enter the information for this new property in the Properties panel. The items highlighted in bold are mandatory properties and so must be defined.

Property	Description	Accepted Value
ID	The id for this new property. It is automatically generated when you save this property. Therefore, no value should be entered here	NA
Is ID	Defines whether this property should be used as an ID for this entity type	True/False
isOverlay	Whether this property which belongs to the entity type has been inherited from a super type or not. If set to true this is the case. If set to false the property is generated out of its entity type.	True/False
Name	The name of this property	Any valid string that describes this property
Entity Type	What entity type this property should belong to	Double click on this entity and chose the required entity type
Meta Data	Data which further describes this property. Such as how the name should be displayed in the columns in the data view	Double click to open and add new metadata
Type	Which type this property is, you can choose between simple (either string, boolean, date, etc.) or complex (an other entity type)	Either a simple or complex type

Enter your information and click the Save button to persist this new property.

GmProperty's Properties

<i>Id</i>	3795
Is Id	<input type="checkbox"/>
isOverlay	<input type="checkbox"/>
Name	dateOfBirth
- Entity Type	
Attendee	
- Meta Data	
∅	
- Type	
date	

After adding this new property, you will have to restart your tribefire host before the changes can take effect.

Create Simple Instantiation Action

Table of Contents

Expand / collapse

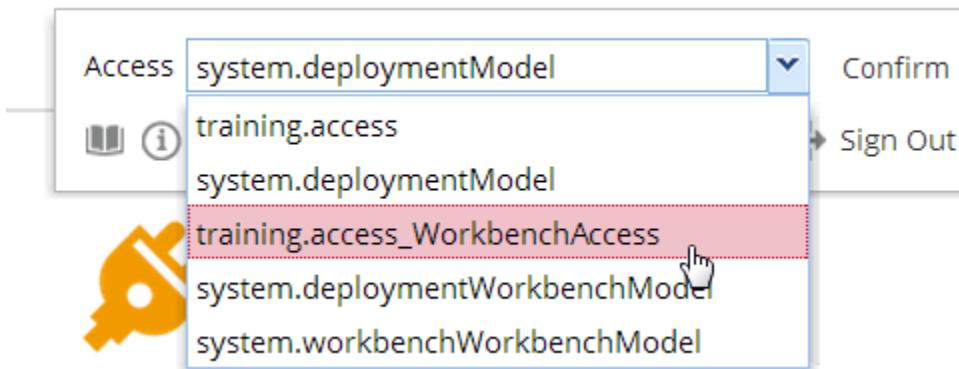
- Introduction
- Create a New SimpleInstantiationAction
- Simple Instantiation Action Additional Properties

Introduction

The SimpleInstantiationAction allows you to create a new instance of an entity type. You need only configure the TypeSignature to use this object.

Create a New SimpleInstantiationAction

You can create a new Simple Instantiation Action object by navigating to the workbench action for the access you would like to create the Simple Instantiation Action.



Select the access and then click confirm to load your workbench.

At the left-hand side of tribefire Explorer there is a Workbench panel. At the top of this panel is a search field (labeled Quick Search). Enter the search value *SimpleInstantiationAction* and double click on the entry that is returned.

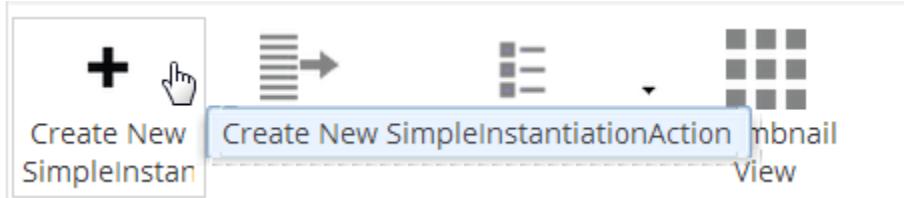
This will open the SimpleInstantiationAction query, which will display all, if any, Simple Instantiation Actions already configured.

If there are any already configured, and it is selected, indicated by an orange background, you will have to deselect it.

Deselect Tip

To quickly deselect an object in tribefire, hold down the CTRL key and then click on the selected object. This object should now be deselected.

Once all objects have been deselected, you can click on the **Create New Simple Instantiation** button located at the bottom of tribefire.



The Create New SimpleInstantiationAction window will be displayed.

displayName	
typeSignature	∅
withSubTypesSelection	<input type="checkbox"/>

There are three properties which you can configure:

Property	Description	Accepted Value
displayName	The name that is displayed when this action is executed.	Any valid name
typeSignature	The name of the entity type that should be instantiated. That is, what type will this new instance belong to.	A valid typeSignature
withSubTypesSelection	If the type that is being instantiated has subtypes, this defines whether they are selectable	True / False

Enter your valid information into these properties...

SimpleInstantiationAction ()

SimpleInstantiationAction's Properties

displayName	New Company
typeSignature	braintribe.model.Company
withSubTypesSelection	<input type="checkbox"/>

...and click **Apply** to create this new SimpleInstantiationAction object. It will be displayed in a new tribefire window

SimpleInstantiationAction	displayName	id	typeSignature	withSubTypesS...
SimpleInstantiationAction ()	New Company	Ø	braintribe.model	<input checked="" type="checkbox"/>

SimpleInstantiationAction's Properties	
displayName	New Company
/id	Ø
typeSignature	braintribe.model.Company
withSubTypesSelection	<input checked="" type="checkbox"/>
-icon	Ø
-inplaceContextCriterion	Ø

Simple Instantiation Action Additional Properties

In addition to the three main properties defined above, there are also two other properties which you can configure, although they are not mandatory. They are icon and inplaceContextCriterion.

Property	Description	Accepted Values
Icon	Allows you to define an icon object which will be placed next to this action	Any correctly configured icon object
inplaceContextCriterion	Allows you to define a context where this action will be displayed	A valid Entity Criterion

Creating User Roles and Groups

Table of Contents

▼ Expand / collapse

- Creating User Roles and Groups
 - Introduction
 - Creating a New Role
 - Creating a New Group

Creating User Roles and Groups

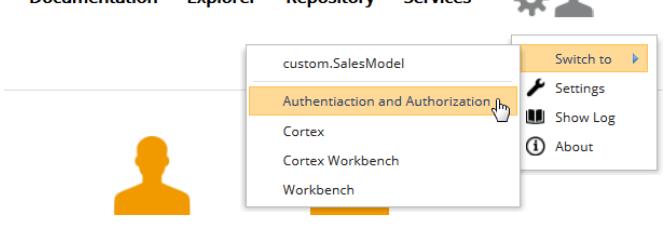
Introduction

tribefire allows you to create user roles and groups which can then be used to determine the rights and permissions a certain user has when performing certain tasks. Normally, a role is created which define which tasks and actions are allowed, and these roles are then gathered together into groups. Examples of groups include Administration or guests.

You can create new roles and groups by selecting Authentication and Authorization in tribefire Control Center. The creation of roles is handled by the Role entity type, which has the type signature - com.braintribte.model.user.Role and the creation of groups is handled by the Group entity type, which has the type signature - com.braintribte.model.user.Group.

 There are two meanings of the word Group in tribefire. One refers to the creation of groups which allows you to gather together properties in logical sections and the other to user groups. Each, however, is handled by a separate entity type, where the type signature of the Property Group entity type is com.braintribte.model.meta.data.display.Group.

Creating a New Role

Step #	Task
1	<p>Select <i>Authentication and Authorization</i> from the list of available accesses.</p> 

In the Quick Access search field, enter the term *role...*

role

Values

- Role (john.smith.role.a) - Role
- Role (mary.williams.role.b) - R...
- Role (robert.taylor.role.a) - Role
- Role (steven.brown.role.b) - R...
- Role (steven.brown.role.c) - R...

Types

Role

RoleSelector

and select Role to display its query.

role

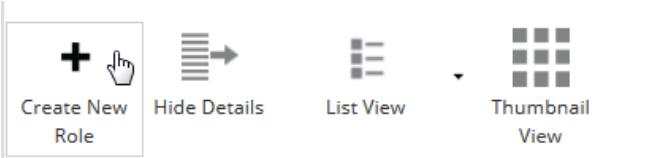
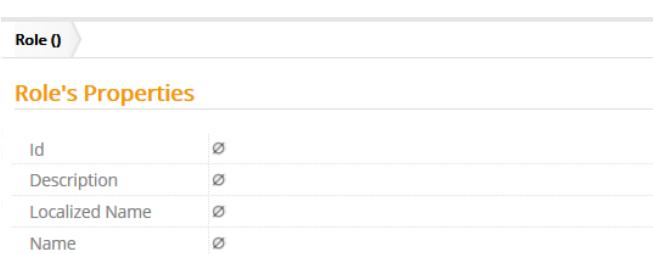
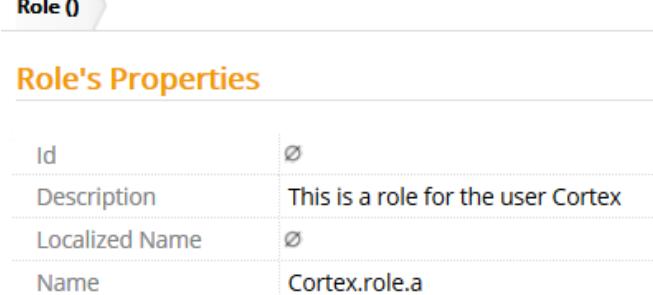
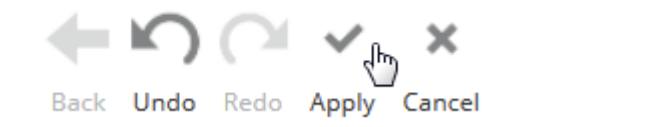
Values

- Role (john.smith.role.a) - Role
- Role (mary.williams.role.b) - R...
- Role (robert.taylor.role.a) - Role
- Role (steven.brown.role.b) - R...
- Role (steven.brown.role.c) - R...

Types

Role

RoleSelector

3	<p>Deselect any existing roles using the space bar and click the Create New Role button.</p> 								
4	<p>The GIMA will be displayed, allowing you to enter details of your new role. There are three editable properties - Description, Localized Name and Name - which you can use to define your new role.</p> <p>Description allows you to describe what this role should do. Name is what your role should be called and will be used as the ID for referencing this role. If you wish, you can also add localized names, so that the roles appear in a relevant target language. You do this using the Localization Panel.</p> <p>Enter a name for your role.</p>  <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Id</th> <td style="padding: 2px;">Ø</td> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">Description</td> <td style="padding: 2px;">Ø</td> </tr> <tr> <td style="padding: 2px;">Localized Name</td> <td style="padding: 2px;">Ø</td> </tr> <tr> <td style="padding: 2px;">Name</td> <td style="padding: 2px;">Ø</td> </tr> </tbody> </table>	Id	Ø	Description	Ø	Localized Name	Ø	Name	Ø
Id	Ø								
Description	Ø								
Localized Name	Ø								
Name	Ø								
5	<p>Once you have enter information for your role,</p>  <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Id</th> <td style="padding: 2px;">Ø</td> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">Description</td> <td style="padding: 2px;">This is a role for the user Cortex</td> </tr> <tr> <td style="padding: 2px;">Localized Name</td> <td style="padding: 2px;">Ø</td> </tr> <tr> <td style="padding: 2px;">Name</td> <td style="padding: 2px;">Cortex.role.a</td> </tr> </tbody> </table> <p>click Apply.</p>  <p style="text-align: center;"> ← Back Undo Redo ✓ Apply ✗ Cancel </p>	Id	Ø	Description	This is a role for the user Cortex	Localized Name	Ø	Name	Cortex.role.a
Id	Ø								
Description	This is a role for the user Cortex								
Localized Name	Ø								
Name	Cortex.role.a								

6

Then click **Save** to persist your changes.



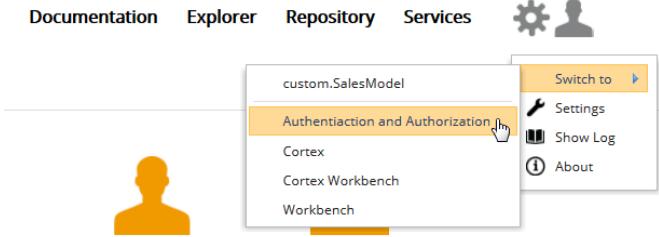
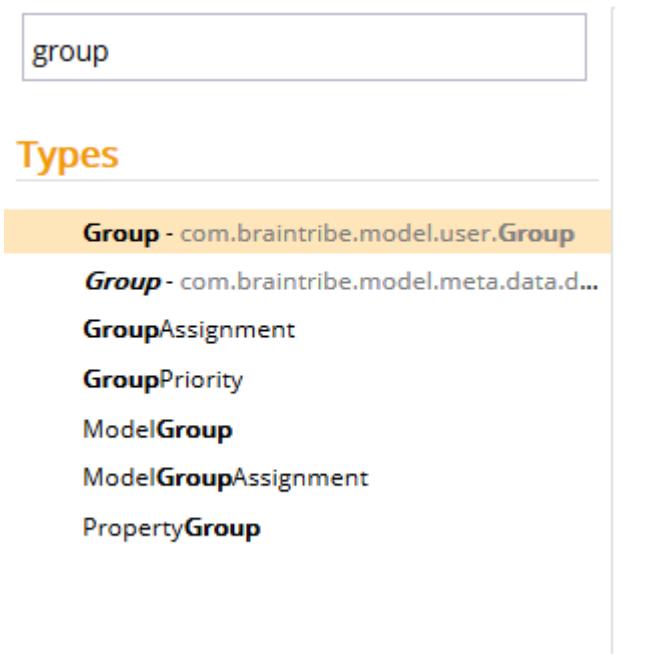
You have now created a new role which can be assigned to a user or a group.

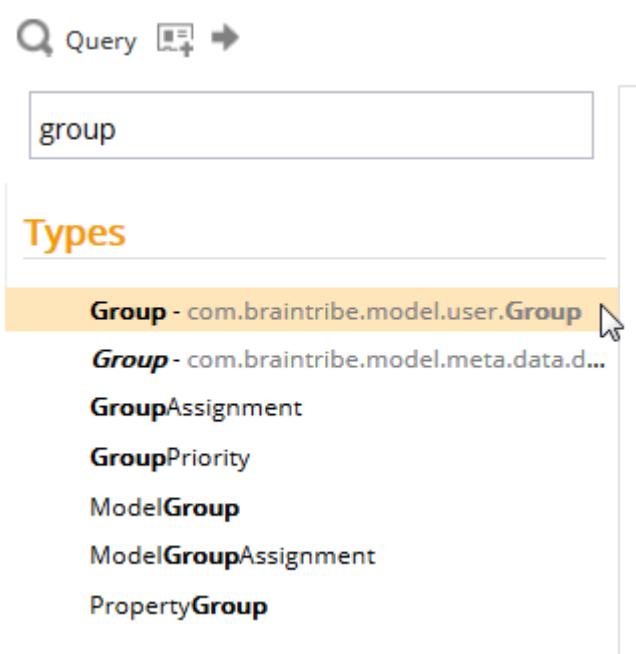
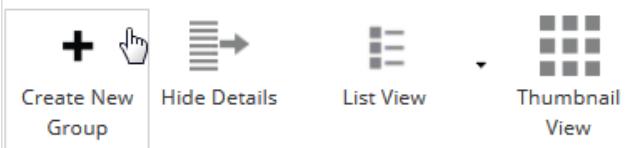
Role	Description	Localized Name	Name	Id
Role (Cortex.role.a)	This is a role for the user Cortex.	Cortex.role.a	Cortex.role.a	Cortex.role.a

Role's Properties

ID	Cortex.role.a
Description	This is a role for the user Cortex.
Localized Name	Cortex.role.a
Name	Cortex.role.a

Creating a New Group

Step #	Task
1	<p>Select <i>Authentication and Authorization</i> from the list of available accesses.</p> 
2	<p>In the Quick Access search field enter the term <i>Group</i>.</p>  <p>Types</p> <ul style="list-style-type: none"> Group - com.braintribe.model.user.Group Group - com.braintribe.model.meta.data.d... GroupAssignment GroupPriority ModelGroup ModelGroupAssignment PropertyGroup

3	<p>There are two types of group in tribefire. Select the group with the type signature com.braintripe.model.user.Group.</p> 
4	<p>Deselect any groups and click the Create New Group button.</p> 

5

The GIMA will appear, allowing to create a new group instance. There are various priorities that can be defined for this group.

Description allows you to describe what this group does, email defines an email address linked to this account, and localized name allows you to define the name in different target languages. The Name property is used when referencing the group and should be given a logical name according to its function, for example, admin, guests, and so on.

Conflict Priority is used when two groups are in conflict (that is, there are two groups that could be used and tribefire must decide between them). The property should be defined with a number between 0 and 1, and the winner of the conflict will be the group with the highest value.

Group 0

Group's Properties

Id	Ø
Conflict Priority	0.00
Description	Ø
Email	Ø
Localized Name	Ø
Name	Ø

6

Once you have entered information for your group,

Group 0

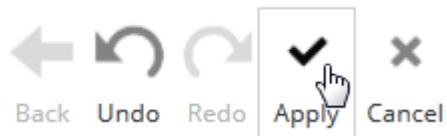
Group's Properties

Id	Ø
Conflict Priority	0.00
Description	A group for developers
Email	dev@email.com
Localized Name	Ø
Name	Developers

click **Apply**.



Then click **Save** to persist your changes.



You have now created a new group.

Group's Properties

Id	Developers
Conflict Priority	0.00
Description	A group for developers
Email	dev@email.com
Localized Name	Ø
Name	Developers
– Image	Ø
– Roles	Ø
– Users	Ø

Deploy an Access

! Work in progress!

Table of Contents

- ▼ Expand / collapse
 - Introduction
 - Deploying an Access
 - Checking the Deployment Report

Introduction

An access is needed to link a Model to an underlying repository, be it external or internal with tribefire's Smood database. It is also required when you wish to configure, add or create queries or actions relating to this Model. Once you have created your access, you must then deploy it. Deploying an access means that the link is created and you can begin to manipulate your access, either directly or through a workbench access.

Click for instructions on creating an access

Deploying an Access

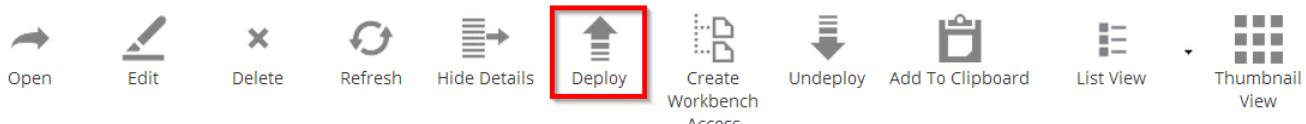
Select the [custom accesses](#) link from the workbench panel.

The screenshot shows the tribefire Workbench interface. On the left, there is a sidebar titled 'Quick Access' with several collapsed sections: 'Smart Enterprise Information' (Modeling, Virtualization), 'System', and 'Actions'. Under 'Virtualization', the 'Custom Accesses' link is highlighted with a red box. The main area displays icons for 'Information Accesses', 'Images', 'Custom Models', 'Distributed Models', 'Connections', 'System Accesses', and 'Information Processes'. Below these icons, there are links to 'Information Accesses', 'Images', 'Custom Models', 'Distributed Models', 'Connections', 'System Accesses', and 'Information Processes'.

The Custom Accesses Query will then be displayed. This page displays all the Accesses that have already been defined for your instance of tribefire. Select one you would like to deploy.

Access	Deployment St...	Description	Simulation Mode	Id	External Id	Name
▶ newSmoodAccess (newSmoodAccess)	∅	A example Smoo...	<input type="checkbox"/>	8	newSmoodAcces...	newSmoodAccess
▶ BTModel (BTModel)	∅	∅	<input type="checkbox"/>	3	BTModel	BTModel
▶ (i) SalesModel (SalesModel)	∅	∅	<input checked="" type="checkbox"/>	6	SalesModel	SalesModel
▶ SampleAccess (SampleAccess)	deployed	∅	<input type="checkbox"/>	1	SampleAccess	SampleAccess

Click the **Deploy** button.

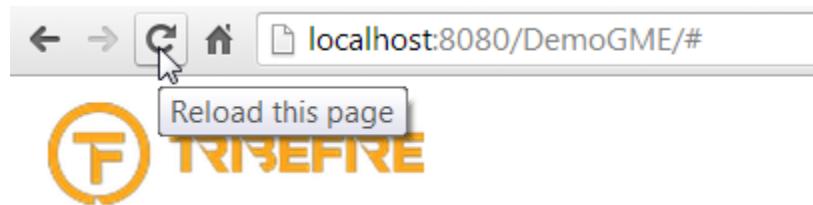


If successful a message will be displayed and the Deployment State of your Access will now read deployed

Successfully executed Action. ✘

Access	Deployment St...	Description	Simulation Mode	Id	External Id	Name
newSmoodAccess (newSmoodAccess)	deployed	A example Smoo...	<input type="checkbox"/>	8	newSmoodAcces	newSmoodAcce
BTModel (BTModel)	∅	∅	<input type="checkbox"/>	3	BTModel	BTModel
SalesModel (SalesModel)	∅	∅	<input checked="" type="checkbox"/>	6	SalesModel	SalesModel
SampleAccess (SampleAccess)	deployed	∅	<input type="checkbox"/>	1	SampleAccess	SampleAccess

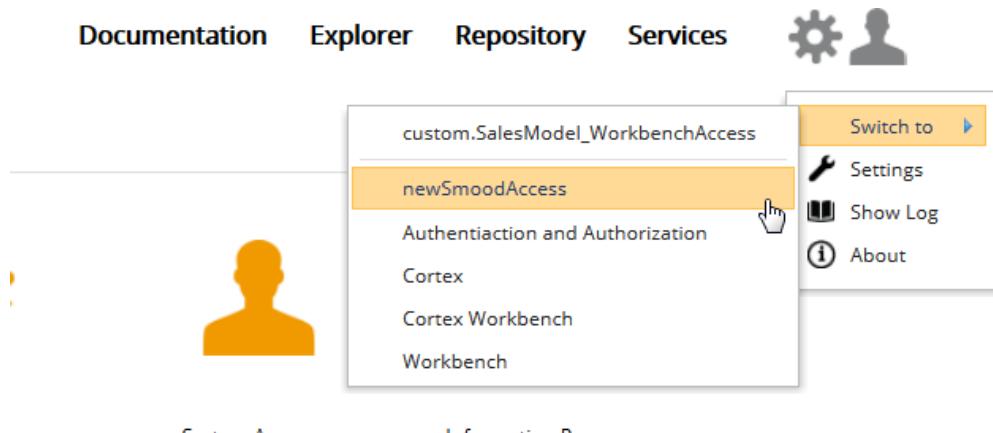
Now that you have deployed your Access, you can now begin to use it. To do so, you must first refresh your browser.



After your browser has reloaded, your Access can be accessed by clicking on the cog icon next to the under information panel.

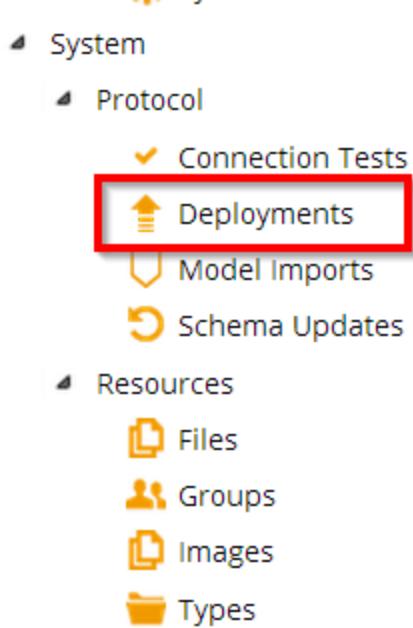


This displays a list of all deployed accesses. You can now select the access you deployed, and it will open in a new tab, in Explorer mode.



Checking the Deployment Report

Each time a deployment attempt is made in tribfire, it is logged. You can view the log by selecting the Deployments link in the left-hand menu.



This will display a list of every deployment attempt and metadata relating to it.

Deploy	Execution End	Mode		Id	Execution Start	State	Code	Result	Detailed Message
▶ ► executed success	02/14/2014 14:03	synchron		5	02/14/2014 14:03	executed	success	∅	
▶ ► executed failure	02/14/2014 14:45	synchron		9	02/14/2014 14:45	executed	failure	Deployment ac	
▶ ► executed failure	02/14/2014 14:46	synchron		10	02/14/2014 14:46	executed	failure	Deployment ac	
▶ ► executed failure	02/17/2014 09:28	synchron		11	02/17/2014 09:28	executed	failure	Deployment ac	
▶ ► executed success	02/17/2014 14:52	synchron		12	02/17/2014 14:52	executed	success	∅	
▶ ► executed success	02/17/2014 14:52	synchron		14	02/17/2014 14:52	executed	success	∅	
▶ ► executed failure	02/18/2014 11:13	synchron		15	02/18/2014 11:13	executed	failure	Deployment ac	
▶ ► executed failure	02/18/2014 11:14	synchron		16	02/18/2014 11:14	executed	failure	Deployment ac	
▶ ► executed success	02/18/2014 15:21	synchron		19	02/18/2014 15:21	executed	success	∅	
▶ ► executed success	02/19/2014 16:51	synchron		25	02/19/2014 16:51	executed	success	∅	
▶ ► executed success	02/19/2014 16:51	synchron		27	02/19/2014 16:51	executed	success	∅	

For more information on the Deployment Logs

Extract Meta Data

Extract Meta Data

If your model has metadata attached to it, you can use the Extract Meta Data button to archive it to tribefire host. The metadata is automatically extracted and the resulting XML file is placed in the files query. You can either use this file on the same instance of tribefire, and on another model, or download the resource for use in a different instance of tribefire.

To extract metadata, select the model which you would like to use and then click the Extract Meta Data button.



It will then be placed in the Files element, allowing you to download it. The new resource has the ending XML.

Raw Resource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resolver URI	Width In Cm	Id
Raw Resource (140428133429-83400174-1296-45ft	46365	0.00	0ac664dc3fd84c5	application/zip	0	Ø	0.00	140428133429-8:
Raw Resource (140428162109-1eecab32-757d-4dfc	8443	0.00	2db21b85b037d8	application/zip	0	Ø	0.00	140428162109-1e:
Raw Resource (140428174337-ece43f63-8bea-43cc	24014	0.00	83b7c6e7295d3a	application/zip	0	Ø	0.00	140428174337-e:
Raw Resource (140429181222-93c73d44-25df-45b6	304225	0.00	aa7eb09a267440	application/xml	0	Ø	0.00	140429181222-9:

Usage Example

In the following example we have a model called personModel. The model has two types, each with their own properties associated with it, along with various metadata properties.

Model

personModel

▷ baseType: Ø

▷ Entity Types (2)

▷ GenericEntity

▷ Person

▷ Artifact Binding: Ø

▷ Meta Data (2)

▷ Database Mapping (7300)

▷ EntityEmphasis (7314)

▷ Properties (6)

▷ ID

▷ firstName

▷ Entity Type: Person

▷ Meta Data (2)

▷ PropertyMapping (7301)

▷ PropertyDisplayInfo (7308)

▷ Type: Ø

▷ secondName

▷ dateOfBirth

▷ birthplace

▷ nationality

▷ Super Types (1)

Enum Types Ø

Meta Data Ø

simpleTypes Ø

Select a model...

System Models Query

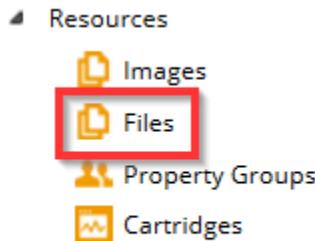
Model	Name	Id
WorkbenchModel#2.0	com.braintribe.mc	1
BasicDeploymentModel#1.0	com.braintribe.mc	2
BasicDeploymentWorkbenchModel#1.0	com.braintribe.mc	3
personModel	personModel	6

...and then click **Extract Meta Data**



If successful, you will receive a message confirming the action's success.

Select Files from the Workbench panel.



You can find the extracted metadata file here, with the ending XML.

Raw Resource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resolver URI	Width In Cm	Id
Raw Resource (140428133429-83400174-1296-45ff)	46365	0.00	0ac664dc3fd84c5	application/zip	0	Ø	0.00	140428133429-83400174-1296-45ff
Raw Resource (140428162109-1eecab32-757d-4dfc)	8443	0.00	2db21b85b037d8	application/zip	0	Ø	0.00	140428162109-1eecab32-757d-4dfc
Raw Resource (140428174337-ece43f63-8bea-43cc)	24014	0.00	83b7c6e7295d3a	application/zip	0	Ø	0.00	140428174337-ece43f63-8bea-43cc
Raw Resource (140429181222-93c73d44-25df-45bf)	304225	0.00	aa7eb09a267440	application/xml	0	Ø	0.00	140429181222-93c73d44-25df-45bf

You can download this file, for use in a different version of tribefire, or for archival purposes, by clicking the **Download Resource** button.



Importing Data from Microsoft Excel

Table of Contents

▼ Expand / collapse

- Importing Data from Microsoft Excel
 - Introduction
 - Detailed Description on the Functionality and Usage of Excel Import Tool
 - Configure the Excel Spreadsheet
 - Mapping Properties to Headings
 - Excel Column Mapping
 - Excel Reference Column Mapping
 - Excel Identity Management Property
 - Creating and Executing the Import Tasks
 - Creating Import Task
 - Executing Import tasks

Importing Data from Microsoft Excel

Introduction

It is possible to **import data** from an **Excel** spreadsheet directly into **tribefire** by mapping the appropriate properties of an entity, which is done through the use of metadata. There are three property metadata that are required when importing from Excel: **ExcelColumnMapping**, **ExcelIdentityManagementProperty** and **ExcelReferenceColumnMapping**. After **mapping** properties to columns, you must then create and **execute** a import **task** for each entity you would like to import data to.

The Excel spreadsheet used must be saved as a *.XLS file.

The purpose of this user guide is to show you how to properly configure your model for an Excel data import.

Detailed Description on the Functionality and Usage of Excel Import Tool

This guide makes the **distinction** between properties of a **simple type** and a **complex type**. Properties which are **simple** contain information like **strings**, **dates** and **numbers**. **Complex** types, on the other hand, are properties which **refer** to another **entity**. This means that the property is used to refer to another entity in a model. In the example below, the Attendee has a property called company. The type of this property is the entity Company, meaning that it is a complex type. There is a **third** type of property, a **collection** type, such as **maps** or **lists**, however, they are **not** supported by the Excel import tool.

You can tell which type a property is by inspecting it in the Control Center.

▼ [Property Type Examples](#)

Simple Property Type

Gm Property

<i>Id</i>	2729
Is Id	<input type="checkbox"/>
Is Overlay	<input checked="" type="checkbox"/>
Name	title
– Entity Type	
Training	
– Meta Data	
∅	
– Type	
string	

A property which is of a simple type can only hold simple data. That is, data which is not linked to an entity of any other type. These types include Strings, Numbers and Dates.

A complete list of the simple types supported in tribefire.

Complex Property Type

- ▲ TrainingModell#2.0
 - ▷ Artifact Binding: Binding Artifact (1820)
 - ▷ Base Type: ∅
 - ▲ Entity Types (4 ...)
 - ▷ Training
 - ▷ Company
 - ▷ StandardIdentifiable
 - ▷ Attendee
 - ▷ Enum Types (1)
 - ▷ Meta Data (0)
 - ▷ Simple Types (0)

Gm Property

<i>Id</i>	2735
Is Id	<input type="checkbox"/>
Is Overlay	<input checked="" type="checkbox"/>
Name	company
– Entity Type	
Attendee	
– Meta Data	
∅	
– Type	
Company	

You will notice that the property called company belongs to the entity type Attendee, as indicated by the image to the right. This property's type is Company, and refers to the Company entity, also contained in the model.

Collection Property Type

Gm Property

<i>Id</i>	2722
Is Id	<input type="checkbox"/>
Is Overlay	<input type="checkbox"/>
Name	attendee
– Entity Type	
Training	
– Meta Data	
\emptyset	
– Type	
Attendee>	

The property attendee, which belongs to the Training entity, is of the type Attendee. However, instead of being linked to the entity directly, which would make it a complex type, we want this property to hold many attendees. Thus, we make this a collection type. You can determine if the property is of a collection type by the greater than symbol (>) at the end of the type description.

Mapping a **simple** property requires only the **configuration** of a specific property in an entity to a column in Excel, created by attaching a **ExcelColumnMapping** metadata to the target property. This metadata is used to define **which column** in the **Excel** spreadsheet this **property** represents. For example, the simple property firstName, belonging to the entity Attendee, is represented by a column in the Excel spreadsheet called First Name. Therefore, we configure the ExcelColumnMapping's Column Name field with the value First Name.

Mapping a complex type requires a **further** step. Because a complex type **represents** an instance of **another entity**, we require a way of recognizing which instance the property should represent; akin to a **foreign key** in a database. Like a foreign key, tribefire requires that a link between two entities are established. This is done through the use of two metadata: **ExcelIdentityManagementProperty** and **ExcelReferenceColumnMapping**.

Excel Reference Column Mapping functions in a similar manner to Excel Column Mapping, in that there is the property Column Name where you enter the name of the column in Excel that this property represents. There is also an additional property, called **Reference Property**. This is used to create the **connection** between the **two entities**. The main entity, to which the **complex property belongs**, can be thought of as the **parent entity**, whereas the entity which the **property represents** is the **child entity**. Since the purpose of the Reference Property is to identify a particular instance of the child entity, the **data contained** in the Excel spreadsheet, and to which the **property** is mapped, should be **equal** to the information in the **child entity**.

Consider the following example, which will be used below. The entity Attendee has a property called company, representing where this attendee works. The Excel worksheet which represents the Attendee entity, has the column company. The property and column are linked by entering the columns name, in this case company, in the field called Column Name. However, because this is not a simple property, that is, it is not a string, but representative of a instance of the entity company, we need to use a unique field to link these two entities. The entity Company has various properties, including companyName. The data contained here is equal to the data contained in the company column in Excel; both can have the value 'Braintripe', for example. These two properties should be linked, as they contain the same information and therefore, can be used to indicate a specific instance in the child entity (Company).

In addition to this, you **must** also use the **Excel Identity Management Property** metadata to enforce this **connection** between a parent and child entity. This metadata should be attached on the **child entity**, to a property that is considered unique. Using this metadata will **disallow any repeated** occurrences of data in the property. In the example above, the ExcelIdentityManagementProperty would be attached to the property companyName.

An example of the relationship between a parent (Attendee) and child entity.



The screenshot above shows an example configuration of a complex property. The entity Attendee includes the property company. The [ExcelReferenceColumnMapping](#) metadata is attached to it:

Excel Reference Column Mapping

– Reference Property

companyName

Excel Column Mapping

Column Name	Company
-------------	---------

The Reference Property field is defined as companyName, a property belonging to the entity Company, meaning this is the property that will be used to find the specific instance referenced in the Excel spreadsheet. Because we use this to find specific instances, it must be a unique value. Hence, we attach that property with the ExcelIdentityManagementProperty (see the screenshot above).

⚠ The Excel Identity Management Property only restricts properties as unique on data imported from Excel. In all other instances, you can still enter duplicate values.

i The model which this guide uses to import data to is called Training. It has four main entities: Company, Address, Training and Attendee. The Excel spreadsheet is called Bt-TrainingData.xls and has four worksheets, each representing an entity. In each sheet columns are defined and the name of these columns will be mapped to the properties of each entity in the model.

You can download the BtTrainingData.xls and Training-2.0.Zargo used in this tutorial.

[BtTrainingData.xls](#)

[Training-2.0.zargo](#)

You will have to [create a model](#) from this Zargo file.

This guide will use the entity type Attendee to show you how to set up tribefire for an Excel import.

Configure the Excel Spreadsheet

Each entity that you wish to import data to should be represented by a worksheet. There are no rules governing the name of each worksheet; they can be named anything you like. However, it is recommended that you observe the following naming convention, *each worksheet named after the entity it represents*.

The first row of each worksheet should contain headings that represent properties in the corresponding entity in tribefire. The following rows then contain the data that will be imported.

! Each column name should be unique through out the Excel spreadsheet. This means the name of each column should not be repeated, either in the current worksheet or any other worksheets belonging to the spreadsheet.

The Attendee worksheet

	A	B	C	D	E	F	G
1	First Name	Last Name	Address	Company	Email Address	Job Title	Social Security Number
2	Robert	Jones	103	Braintribe	Robert.Jones@braintribe.com	Developer	665-5665-AT
3	Lisa	Steiner	104	tribefire	Lisa.Steiner@tribefire.com	Project Manager	998-9999-DE
4	David	Weller	105	TechSphere	David.Weller@TechSphere.com	Technical Writer	489-9887-TO
5	Maria	Collins	106	tribefire	Maria.Collins@tribefire.com	Accountant	339-4916-DE
6	Paula	Hill	107	TechSphere	Paula.Hill@TechSphere.com	Developer	789-8973-RT
7	James	Osborne	108	Braintribe	James.Osborne@braintribe.com	Administrator	263-6631-AT
8	Stephen	MacGarry	109	tribefire	Stephen.MacGarry@tribefire.com	Developer	554-5668-PR
9	June	Bauer	110	TechSphere	June.Bauer@TechSphere.com	Project Manager	467-8250-RT
10	Penelope	O'Donnell	111	Braintribe	Penelope.O'Donnell@braintribe.com	Technical Writer	654-9876-AT
11							

The Attendee Entity in tribefire

- **Attendee**
- Properties (7)
 - ▷ **firstName**
 - ▷ **secondName**
 - ▷ **jobTitle**
 - ▷ **emailAddress**
 - ▷ **company**
 - ▷ **address**
 - ▷ **socialSecurityNumber**

You should create a worksheet, headings and data for each entity you intend to use.

Mapping Properties to Headings

The next step is to map the properties to a column in Excel, configured through the use of metadata. There are three metadata used, with each section of the user guide showing you to define each one.

For more information on the function of each metadata and how they work in conjunction with each other, see the [detailed description](#) above.

Excel Column Mapping

The metadata `ExcelColumnMapping` is used to map [simple property types](#), for example, Strings, Dates or Numbers. If your property is of a complex type, you will must use `ExcelReferenceColumnMapping`.

An example of a simple type

GmProperty's Properties

<i>Id</i>	4275
<i>Is Id</i>	<input type="checkbox"/>
<i>isOverlay</i>	<input type="checkbox"/>
<i>Name</i>	firstName
- Entity Type	
Attendee	
- Meta Data	
∅	
- Type	
string	

Step #	Task

1	<p>Open the entity whose property you wish to map.</p> <pre> System Models Query TrainingModel#2.0 Model ↳ TrainingModel#2.0 ↳ baseType: ∅ ↳ Entity Types (4 ...) ↳ Company ↳ Address ↳ Training ↳ Attendee ↳ Properties (7) ↳ firstName ↳ secondName ↳ jobTitle ↳ emailAddress ↳ company ↳ address ↳ socialSecurityNumber ↳ Meta Data ∅ ↳ Property Meta Data ∅ ↳ Super Types (1) </pre>
2	<p>Collapse the property you would like to add metadata to and select the option <i>Meta Data</i>.</p> <pre> Attendee ↳ Properties (7) ↳ firstName ↳ Meta Data (0) ↳ secondName ↳ jobTitle </pre>
3	<p>Click Add.</p>

4

Select the ExcelColumnMapping type...

The screenshot shows a software interface for selecting a type. At the top, there are icons for Home, Filter, Refresh, Save, and Undo. A yellow bar labeled "PropertyMetaData Query" is visible. Below is a search bar with placeholder text "Type for filtering types and values...". Under "Values", there is a list of items. Under "Types", a list of class names is shown, with "ExcelColumnMapping" highlighted by a yellow background and a cursor icon pointing at it. Other items in the list include AmbiguousPropertyAssignment, AnalyzedProperty, BidirectionalProperty, CascadingDelete, CollectionElementCountConstraint, CorrelationPropertyAssignment, DistinctPropertyAssignment, ExcelIdentityManagementProperty, ExcelReferenceColumnMapping, and FormattingMetaData. Below the list is a button labeled "...and click Add and Finish". At the bottom are buttons for Show Details, Filter, Add, Add and Finish (which has a hand cursor icon over it), and Cancel.

5

A newly created instance of ExcelColumnMapping will be added to your property's metadata.

The screenshot shows a tree view of property metadata. At the top, there is a navigation bar with icons for Back, Forward, Filter, Add, Add and Finish (with a hand cursor icon), and Cancel. Below the navigation bar is a list of properties under an "Attendee" folder. The "Properties (7)" node is expanded, showing the "firstName" node, which in turn has an "Meta Data (1)" node. This "Meta Data (1)" node is highlighted with a yellow background, and its child node "Excel Column Mapping:" is also highlighted. The "secondName" node is also visible under "Properties (7)".

6

To configure this metadata property, you must enter the column name from Excel into the Column Name property of this metadata.

In this case, the Excel worksheet, see above, has the column *First Name* defined.

First Name
Robert
Lisa
David
Maria
Paula
James
Stephen
June
Penelope

In the Column Name we simply enter *First Name*.

ExcelColumnMapping's Properties

Column Name	First Name
-------------	------------

7

Click **Save** to persist your changes.



8

You should repeat these steps for each property that is of a **simple type**, so that every simple type property in each entity that you wish to import data to has a ExcelColumnMapping metadata attached to it.

In the example above the properties `firstName`, `secondName`, `emailAddress`, `jobTitle` and `socialSecurityNumber` are all examples of simple types, and thus have a ExcelColumnMapping metadata configured.

Excel Reference Column Mapping

While the metadata ExcelColumnMapping is used to create a connection between simple properties in tribefire to data in an Excel spreadsheet, ExcelReferenceColumnMapping links the data to complex properties. When data is imported, rather than simple writing the data to the property, as is what happens with simple types, tribefire will use the found data as a key to find the particular instance of the entity the property represents.

The configuration of this metadata is similar to the previous step—you configure the property Column Name with the corresponding column in Excel—you also use the property Reference Property to define the relationship between the two entities.

For more information on the functionality of this metadata, and how it works in conjunction with the other metadata, see the [detailed description](#) above.

An example of a complex property type

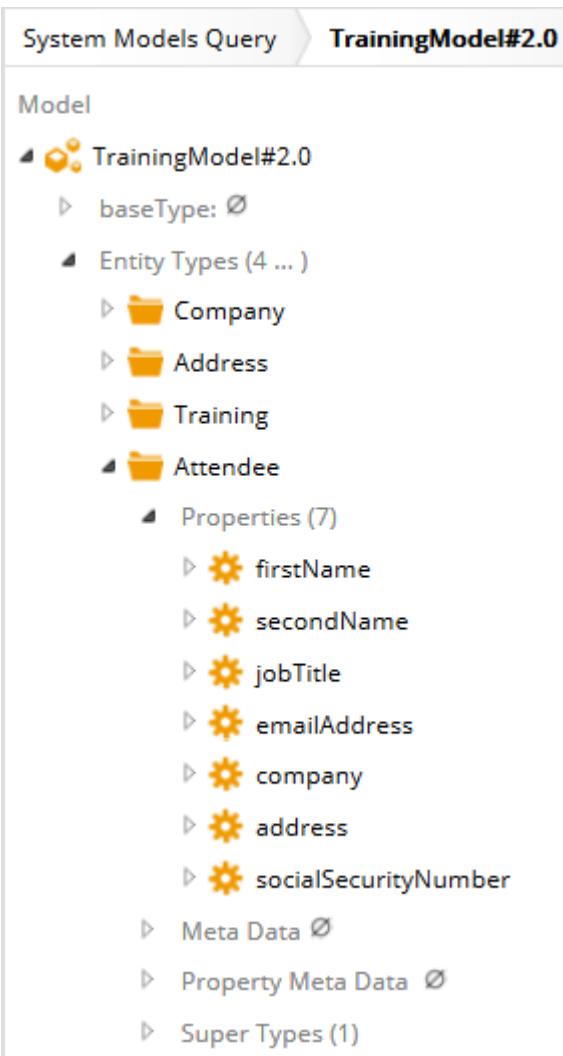
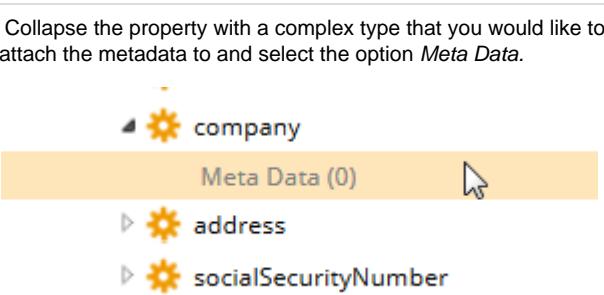
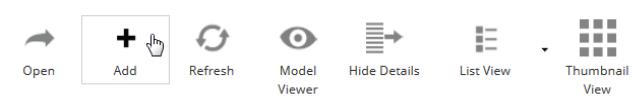
GmProperty's Properties

<i>Id</i>	4278
<i>Is Id</i>	<input type="checkbox"/>
<i>isOverlay</i>	<input type="checkbox"/>
<i>Name</i>	company
–  Entity Type	
Attendee	
– Meta Data	
–  Type	
Company	

As well as configuring the ExcelReferenceColumnMapping metadata, you must also use ExcelIdentityManagementProperty to identify the unique property of an entity. This metadata restricts the property to which it is attached, meaning that data coming from the Excel spreadsheet must be unique in this property (and column). If you do not configure this, the property which has been configured with the ExcelReferenceColumnMapping metadata, will return an error and no data will be imported.

See below on how to configure ExcelIdentityManagementProperty.

Step #	Task

1	<p>Open the entity that you wish to map.</p>  <p>The screenshot shows the 'System Models Query' interface with the title 'TrainingModel#2.0'. The entity structure is displayed as follows:</p> <ul style="list-style-type: none"> Model TrainingModel#2.0 <ul style="list-style-type: none"> baseType: \emptyset Entity Types (4 ...) <ul style="list-style-type: none"> Company Address Training Attendee Properties (7) <ul style="list-style-type: none"> firstName secondName jobTitle emailAddress company address socialSecurityNumber Meta Data \emptyset Property Meta Data \emptyset Super Types (1)
2	<p>Collapse the property with a complex type that you would like to attach the metadata to and select the option <i>Meta Data</i>.</p>  <p>The screenshot shows the properties section of the 'TrainingModel#2.0' entity. The 'company' property is selected, and its 'Meta Data (0)' option is highlighted with a yellow background and a cursor icon pointing at it.</p>
3	<p>Click Add.</p>  <p>The screenshot shows the toolbar at the bottom of the interface. The 'Add' button is highlighted with a yellow background and a cursor icon pointing at it.</p>

4

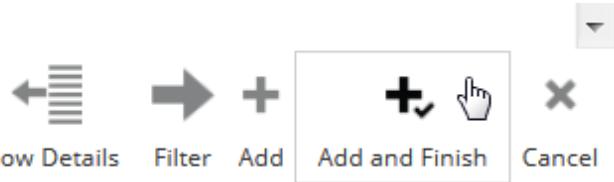
Select ExcelReferenceColumnMapping type...



Type for filtering types and values...

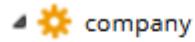
Values**Types**

AmbiguousPropertyAssignment
 AnalyzedProperty
 BidirectionalProperty
 CascadingDelete
 CollectionElementCountConstraint
 CorrelationPropertyAssignment
 DistinctPropertyAssignment
 ExcelColumnMapping
 ExcelIdentityManagementProperty
ExcelReferenceColumnMapping
 FormattingMetaData
 FractionDigitsFormatting

...and click **Add and Finish**.

5

A newly created instance of ExcelReferenceColumnMapping will be added to your property's metadata.



Meta Data (1)

Excel Column Mapping:

6

We configure the metadata by entering the appropriate values for both *Column Name* and *referenceProperty*.

Column Name refers to the column in the spreadsheet where tribefire should find the data. In this example, we wish to import the data from the column labeled *Company*.

D
Company
Braintribe
tribefire
TechSphere
tribefire
TechSphere
Braintribe
tribefire
TechSphere
Braintribe

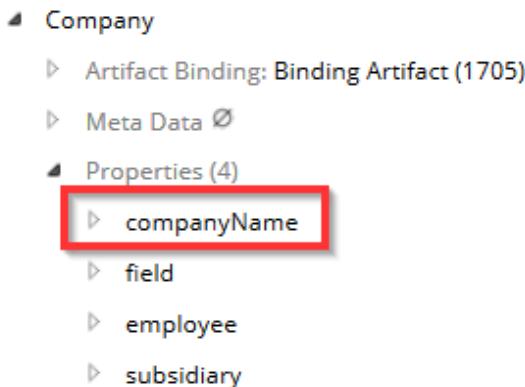
Therefore, we enter the name of this category into the *Column Name* property.

Column Name	Company
-------------	---------

7

The next step is to configure *referenceProperty*. Similar to a foreign key in a database, this property allows tribefire to find the relevant instance of the complex type's entity, so that it can be written to this property. If no instance exists, a new one will be created.

In this example the property is of the type company, and the data imported is the company's name. In the entity Company there is also a property called *companyName*.



The data contained in this property is the same as contained by the column *Company* in Excel.

companyName	field	subsidiary
Braintribe	IT	100
TechSphere	IT	101
tribefire	Software Development	102

D
Company
Braintribe
tribefire
TechSphere
tribefire
TechSphere
Braintribe
tribefire
TechSphere
Braintribe

We can use this information to create the relationship between the two properties (company and companyName).

Click the context-menu icon...



...and select **Change**



8

Search for the property that you wish to map.

In this example, the data imported is a company's name. Therefore, we find the relevant property in the entity Company which refers to the company's name, in this instance the property companyName.

Click **Finish** to select the property.



9	<p>The ExcelReferenceColumnMapping has now been configured.</p> <h3>ExcelReferenceColumnMapping's Properties</h3> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">Column Name</td><td style="width: 85%;">Company</td></tr> <tr> <td>- referenceProperty</td><td></td></tr> <tr> <td colspan="2" style="text-align: center;">companyName</td></tr> </table> <p style="text-align: center;">Click Save to persist your changes.</p> <div style="text-align: center; margin-top: 20px;">  </div>	Column Name	Company	- referenceProperty		companyName	
Column Name	Company						
- referenceProperty							
companyName							
10	<p>You should repeat these steps for each property that is of a complex type, so that every complex type property in each entity that you wish to import data to has a ExcelReferenceColumnMapping metadata attached to it.</p> <p>In this example, the entity Attendee has another property which is complex, address. Therefore, the same steps should also be taken on this property.</p>						

Excel Identity Management Property

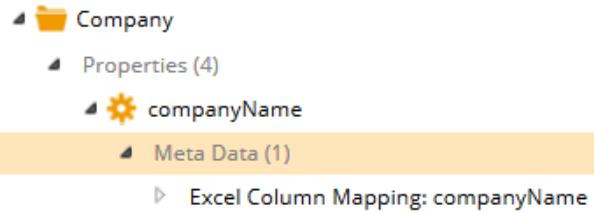
This metadata restricts the property to which it is attached, so that it is unique. A unique property is one whose values cannot be duplicated, functioning similarly to a key in a database. If you wish to import data to properties which are of complex types, this metadata is required.

In the step above the property companyName, belonging to the entity Company, was configured as the referenceProperty. This means that, there must exist an unique value in this entity. In this case it is also the property companyName. That is, no company name can appear twice in the dataset.

Step #	Task
1	<p>Open the entity that you wish to map.</p> <ul style="list-style-type: none"> <li style="margin-bottom: 5px;"> Company <li style="margin-bottom: 5px;"> Properties (4) <li style="margin-bottom: 5px;"> companyName <li style="margin-bottom: 5px;"> field <li style="margin-bottom: 5px;"> employee <li style="margin-bottom: 5px;"> subsidiary

2

Select the property that should be considered unique in this entity, collapse it and select *Meta data*. In this example, the property *companyName* should be considered unique.



3

Click **Add**



4

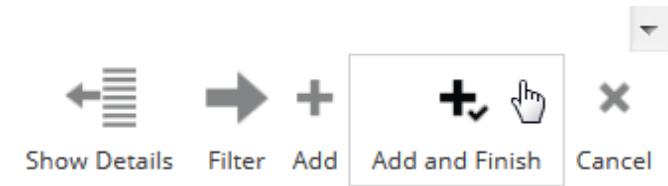
Select ExcelentityManagementProperty...



Type for filtering types and values...

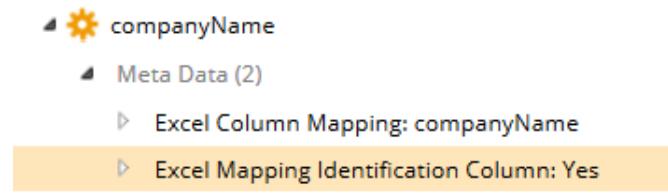
Values**Types**

AmbigiousPropertyAssignment
 AnalyzedProperty
 BidirectionalProperty
 CascadingDelete
 CollectionElementCountConstraint
 CorrelationPropertyAssignment
 DistinctPropertyAssignment
 ExcelColumnMapping
ExcelentityManagementProperty
 ExcelReferenceColumnMapping
 FormattingMetaData
 FractionDigitsFormatting
 FulltextProperty

...and click **Add and Finish**.

5

A newly created instance of ExcelentityManagementProperty will be added to your property's metadata.



6	<p>The property should automatically be configured after creation.</p> <h3>ExcelIdentityManagementProperty's Properties</h3> <p>Identification Property <input checked="" type="checkbox"/></p> <p>Click Save to persist the changes.</p> <div style="text-align: center;">     </div>
7	<p>You should identify an unique property in each entity that you wish to import data to, if possible. However, if you import to complex types, the entity to which that type belongs to must have a ExcelIdentityManagementProperty or you will receive an error and no data will be imported on that property.</p> <p>As stated above the entity type Attendee has two complex properties that will receive imported data: company and address. This step shows how to define the property company. A unique property must also be defined on the address entity. In this case the property is called addressCode, and is used to map attendees and their addresses.</p>

Creating and Executing the Import Tasks

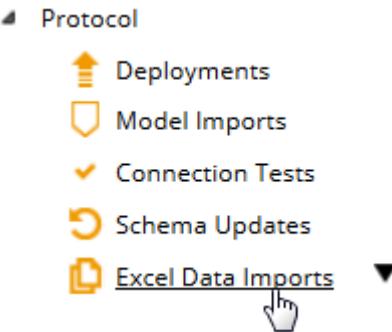
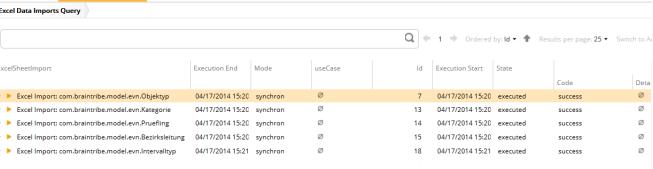
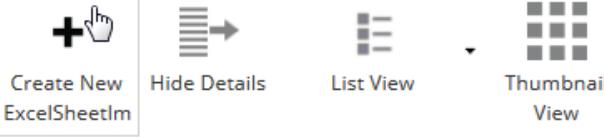
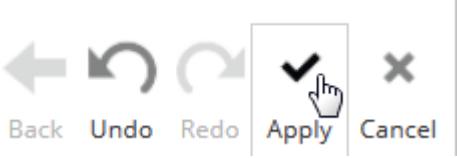
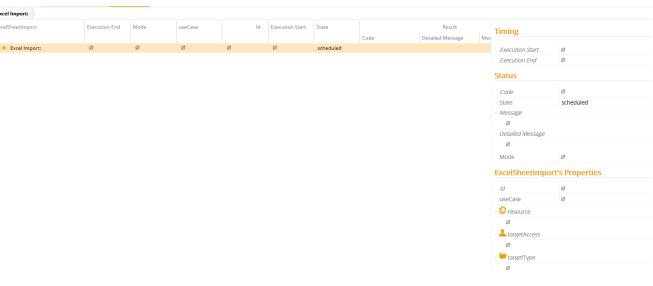
After configuring your model so that data can be imported from Excel, you must then create the tasks that will carry out the import action. You must create a task for each entity that data will be imported to. There are three properties that must be defined: resource, targetAccess and targetType.

Property	Description	Notes
Resource	This property refers to the Excel spreadsheet which contains the data.	Before you can create the import tasks, you must first upload the Excel file that will be used.
Target Access	This refers to the access which the target model belongs to.	Each model requires an access, so that tribefire can manipulate and query data. Before you can import data, you must first create an access for your model. After you have created the access, you must also deploy it .
Target Type	This property refers to which entity the data belongs to.	Each entity to which you would like to import data must have its own task configured.

Creating Import Task

Once you have uploaded your Excel spreadsheet and deployed an access, you can create the import task.

Step #	Task

1	Click Excel Data Imports. This can be found in the workbench panel. 
2	The Excel Data Imports query will be executed, displaying all existing import tasks. 
3	Deselect any highlighted items by pressing the space bar and click Create New ExcelSheetImport. 
4	Set the State to <i>scheduled</i> ...  <p>...and click Apply.</p> 
5	A new import task will be created. 

6

Click the context-menu icon for resource...

...and select **Change**.

7

Select the Excel spreadsheet that you uploaded...

Resource Query											
Resource	File Size	Height/Crop	MIME	MimeType	pageCount	resourceID	widthInCM	W	Created	Created	Name
1 - MetaDataExcel.com.brlm	1871693		0.00	7zf0036553344	application/xml	0	0	0.00	Rn_1	12/20/2013 14:36	context
2 - infoChange1.png	1547		0.00	715461x1749@760	image/png	1	0	0.00	Rn_103	10/29/2013 16:15	context
3 - EventModel2Change	23929		0.00	826401x1000@760	application/xaml+xml	0	0	0.00	Rn_104	09/23/2014 14:38	context
4 - EventModel2Change.xsd	2051		0.00	249401x1000@760	application/xsd+xml	0	0	0.00	Rn_105	09/23/2014 14:38	context
5 - evn_SNSD_PutData_Auth	303616		0.00	249401x1749@760	application/xsd+xml	0	0	0.00	Rn_110	09/25/2014 14:25	context
6 - Training2Change	8562		0.00	59f1d31e1440f	application/xsd+xml	0	0	0.00	Rn_111	04/17/2014 15:16	context
7 - BtTrainingData.xls	32259		0.00	8d4ab2f0b1d40	application/xls	0	0	0.00	Rn_112	04/17/2014 15:15	context
8 - FeedbackChange.xls	377402		0.00	5d45272080@760	application/xls	0	0	0.00	Rn_113	04/17/2014 15:15	context
9 - FeedbackChange.png	1151		0.00	1a5f53227080@760	image/png	1	0	0.00	Rn_114	10/23/2013 14:13	img
10 - ticketChange.png	1118		0.00	589327fbad402	image/png	1	0	0.00	Rn_125	10/23/2013 14:15	img

...and click **Finish**.

8

For targetAccess select the access which your model belongs to and for targetType select the entity to which data will be uploaded.

- resource
BtTrainingData.xls

- targetAccess
custom.Training (custom.Training)

- targetType
Attendee

Click **Save** to persist your changes.

9

You must create a task for every entity you wish to import data to.

Excel Data Imports Query											
ExcelSheetImport	Execution End	Mode	useCase	ID	Execution Start	State	Code	Detail			
> Excel Import: com.braintribe.model:evn:Objektg	04/17/2014 15:20	synchronous	Ø	7	04/17/2014 15:20	executed	success	Ø			
> Excel Import: com.braintribe.model:evn:Kategorie	04/17/2014 15:20	synchronous	Ø	13	04/17/2014 15:20	executed	success	Ø			
> Excel Import: com.braintribe.model:evn:Pruefung	04/17/2014 15:20	synchronous	Ø	14	04/17/2014 15:20	executed	success	Ø			
> Excel Import: com.braintribe.model:evn:Beklaerung	04/17/2014 15:20	synchronous	Ø	15	04/17/2014 15:20	executed	success	Ø			
> Excel Import: com.braintribe.model:evn:Intervalltyp	04/17/2014 15:21	synchronous	Ø	18	04/17/2014 15:21	executed	success	Ø			
> Excel Import: com.braintribe.model:evn:Person	Ø	Ø	Ø	100	Ø	scheduled	Ø				
> Excel Import: com.braintribe.model:evn:Adresse	Ø	Ø	Ø	107	Ø	scheduled	Ø				
> Excel Import: com.braintribe.model:Address	Ø	Ø	Ø	108	Ø	scheduled	Ø				
> Excel Import: braintribe.model:Company	Ø	Ø	Ø	109	Ø	scheduled	Ø				

Executing Import tasks

The last step is to execute the tasks you created in the previous step. The data will automatically be imported and saved.

Step #	Task
1	Select the import task you would like to execute. 
2	Click Run. 
3	The task will be executed. In the console, you will receive a success message. <pre>INFO : Successfully imported ExcelSheet for type: braintribe.model.Training</pre>
4	Repeat this step for each task. If you refresh the Excel Import Query, the task will be updated and will have either a success or failure Code 

Your data will now have been imported from Excel.

Control Center

Table of Contents

▼ Expand / collapse

- tribefire Control Center
 - Center Panel
 - User information/Access selection
 - Login Information
 - Access Selection menu
 - Workbench Panel
 - Quick Access Search
 - Smart Enterprise Information
 - Modeling
 - Virtualization
 - System
 - Extensions
 - Actions
 - Resources

Child Pages

▼ Expnd / collapse

- Applications
- Modeling
- Virtualization
- tribefire GME - System

tribefire Control Center

The tribefire Control Center is the main administrative view that offers you control of your models, accesses and connections. A tool that you can use to configure and administer your tribefire system, as well as viewing important logging information and other tribefire resources, the Control Center offers you complete and powerful way to use tribefire.



Due to the powerful, flexible nature of tribefire, the version of your Control Center may look different in comparison to the screenshots shown. You can alter the look and feel of your tribefire Control center by using the Cortex Workbench, available by selecting the cog icon and selecting it from the list.

The screenshot shows the tribefire Control Center interface. The left side features a navigation sidebar with sections like 'Smart Enterprise Information', 'Virtualization', 'System', 'Actions', and 'Resources'. The main area displays icons for 'Information Processes', 'Information Accesses', 'Images', 'System Accesses', 'Distributed Models', 'Custom Models', and 'Connections'.

Center Panel

In the main section are thumbnails representing the most important section of tribefire: Information Processes, Information Accesses, Images, System Accesses, Distributed Models, Custom Models and Connections.



Information Processes



Information Accesses



Images



System Accesses



Distributed Models



Custom Models



Connections

Above these thumbnails, you will see three smaller icons. The first icon, represented by a house, will display the home screen, that is the screen displayed in the overview screenshot above. The second icon, represented as two arrows, displays what changes have been made to tribefire which have yet to be persisted by the system and the last icon is the clipboard icon. This shows you what, if any, objects have been copied to the clipboard.



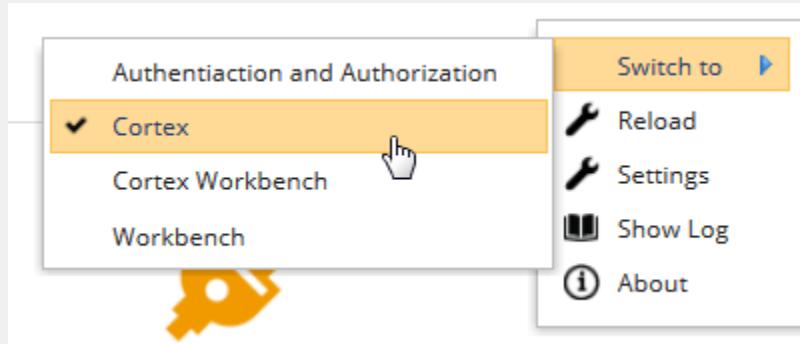
User information/Access selection

Login Information

Above the center panel, in the top right-hand corner, is the login information. This displays the current user logged into tribefire Control Center.



Access Selection menu



Clicking the cog icon will display the session settings menu. There are five options to choose from. The most important is the **Switch to** option. Selecting this will display a list of accesses that have been deployed to the tribefire system. By default there are four accesses that come as standard: Authentication and Authorization, Cortex, Cortex Workbench and Workbench.

- Authentication and Authorization - Controls the security setting for tribefire, including the ability to create roles, groups and users.
- Cortex - the access that represents the Control Center. This is the access displayed when you first log on
- Cortex Workbench - A workbench that lets you alter the look and feel of the Cortex access
- Workbench - A workbench that lets you alter the look and feel of the Cortex Workbench

In addition to **Switch to**, there are also four other options available

- Reload - Reloads the current access you are working on
- Setting - Shows the settings menu
- Show Log - Display log information about the current session of tribefire Control Center
- About - Shows system information regarding the version of tribefire Control Center currently in use.

Workbench Panel

The Workbench panel of tribefire Control Center allows you to navigate between the most **important elements of tribefire cortex**. These links allow you to create models, connections and accesses, as well as being able to monitor processes affecting the system. It is split into two main parts: Smart Enterprise Information and System. Smart Enterprise Information represents two of the three layers of the cortex paradigm, the Modeling and Virtualization layer. Whereas, the System section allows you to install new cartridges, apps or processes, monitor the different system processes, such as Deployment or Connection Test logs and view the resources this instance of tribefire has access to.



Due to the powerful nature of tribefire it is possible that the screenshots shown below do not match your instance of Control Center. It is possible to change this layout, and fully customize it, using the Workbench.

Quick Access Search

Quick Access

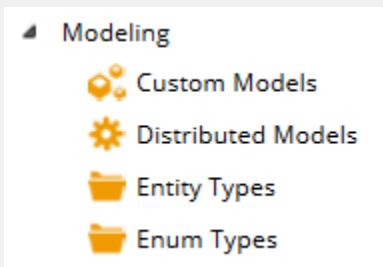
Link	Description
Quick Access Search Field	Can be used to quickly search for types or values associated with Control Center.

Smart Enterprise Information

This section is divided further into two subsections, representing two of the three layers of tribefire cortex's paradigm: the Modeling and Virtualization layers.

Modeling

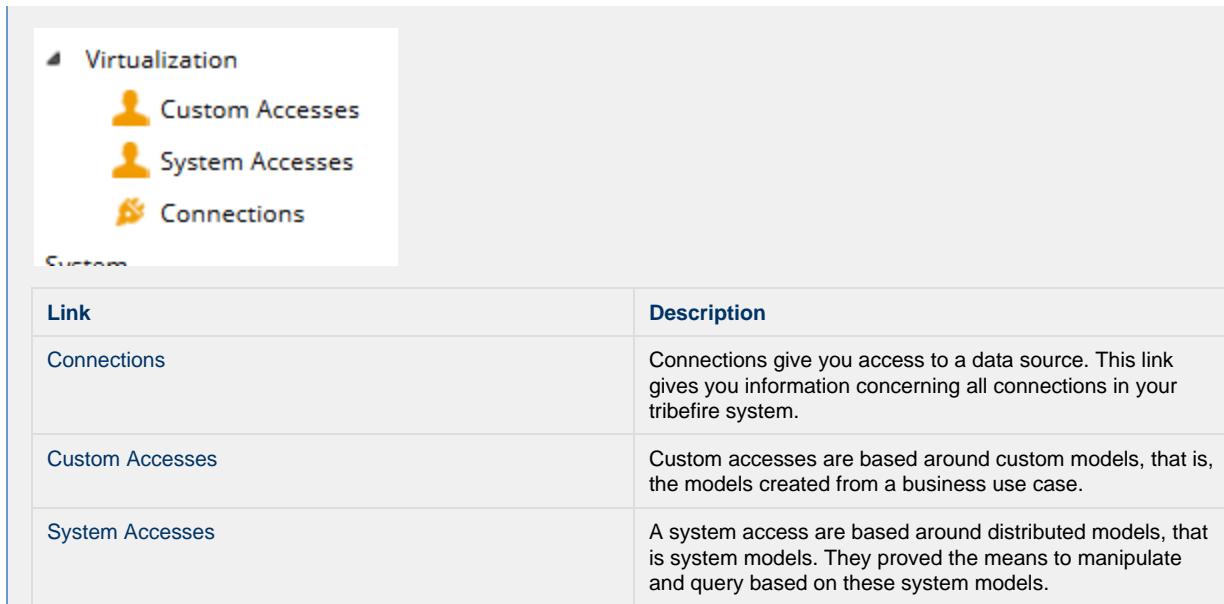
The modeling links provide access to the models that are associated with this instance of tribefire. This section represents the Modeling Layer of tribefire cortex and here you can build your own custom models based on a business user case. In addition to this, you can view and configure any integration or system models that have been created, or view different information processes.



Link	Description
Custom Models	Custom Models are models build on a business use case. They differ from distributed models in that they are used to manipulate and view data coming from a repository.
Distributed Models	tribefire cortex is build completely on models. The Distributed Models represent all models which effect tribefire Control Center, such as the DeploymentModel or WorkbenchModel.
Entity Types	Each model has a number of different entity types. This link provides you with all the types which tribefire Control Center has access to. This link includes types from all different types of models, be it system, integration, or information models.
Enum Types	Like entity types, each model can have a number of enum types. An enum (enumeration) defines a list of known values that don't change. They can be attached to a property to provide the user with a drop-down list of options to select from. An example of an enumeration would be the days of the week. We know that there are only seven days, whose names never change.

Virtualization

The virtualization link provides access to connectors and accesses and represents the Virtualization layer of tribefire cortex. The Virtualization layer is where the connections to the physical repositories are made and information gathered from them. This is done by the use of connections and accesses. The connections provide the parameters and settings which allow tribefire Control Center to access these repositories, whereas the accesses uses these connections to generate models and retrieve data from the data sources.



Virtualization

- Custom Accesses
- System Accesses
- Connections

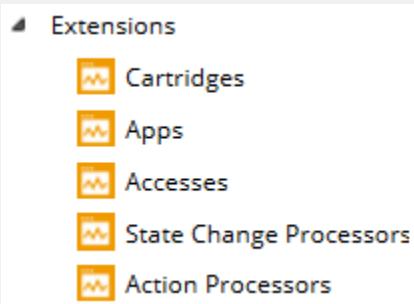
Link	Description
Connections	Connections give you access to a data source. This link gives you information concerning all connections in your tribefire system.
Custom Accesses	Custom accesses are based around custom models, that is, the models created from a business use case.
System Accesses	A system access are based around distributed models, that is system models. They proved the means to manipulate and query based on these system models.

System

Whereas the Smart Enterprise Information section deals with the manipulation of models, accesses and applications, the System section deals with tribefire's system properties. There are three main sections, which are then further subdivided: Extensions, Actions and Resources.

Extensions

This section allows you to extend your instance of tribefire by installing new cartridges, apps, processors or actions. These custom elements can be installed to tribefire to manipulate the system, and therefore your models and their data in the way that you want.

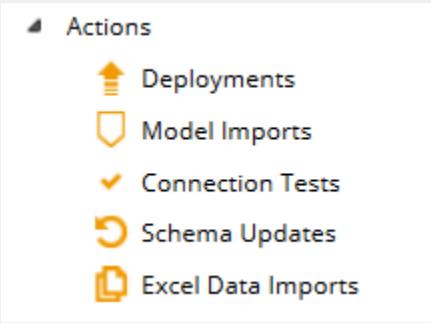


Extensions

- Cartridges
- Apps
- Accesses
- State Change Processors
- Action Processors

Actions

This section deals with logging information for tribefire, including information about connection tests, access and other types of deployments, what models have been imported and any database schema which have been updated.



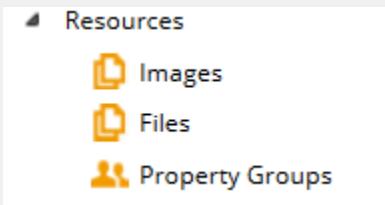
Actions

- Deployments
- Model Imports
- Connection Tests
- Schema Updates
- Excel Data Imports

Link	Description
Connection Tests	With each connector that you build, you can carry out a connection test to determine whether the parameters you entered are correct and that a connection has been established. This link will provide you with information regarding each test executed, including information regarding execution date, success, and any error messages.
Deployments	With each access or connector that you build, you must deploy it to the execution layer before you can start to use it. Each time you attempt to deploy either an access or connector, this attempt will be logged here, giving you information regarding deployment date, success, and any error messages.
Model Imports	When connecting to an external data source, for example a database, you must generate a model based on the data source's underlying schema. This link will provide you with information regarding each attempted import, giving you information regarding import data, success, and any error messages.
Schema Updates	After creating a connection to an underlying data source, the connector will then create a schema which represents the data structure of this data source, for example, tables in a database. This schema must also be updated if any changes have been made to the underlying data structure. This link provides you with information regarding the creation and update of schema, including information regarding the date of the update attempt, success, and any error messages.
Excel Data Imports	It is possible to upload data contained in an Excel spreadsheet directly to a model. You must first configure your model to receive the data and then create tasks to execute the import. This link allows you to create those tasks.

Resources

Split up into different sections, it deals with the various resources used by tribefire. The files and images links will display all the files and images available to Control Center, whereas the property groups link will display all the property groups available in Control Center.



Link	Description
Files	This link provides you with all the files that have been uploaded, and which tribefire Control Center has access to.
Property Groups	This link provides you with all the property groups which have been created in this instance of tribefire.
Images	This link provides you with all the images that have been uploaded, and which tribefire Control Center has access to.

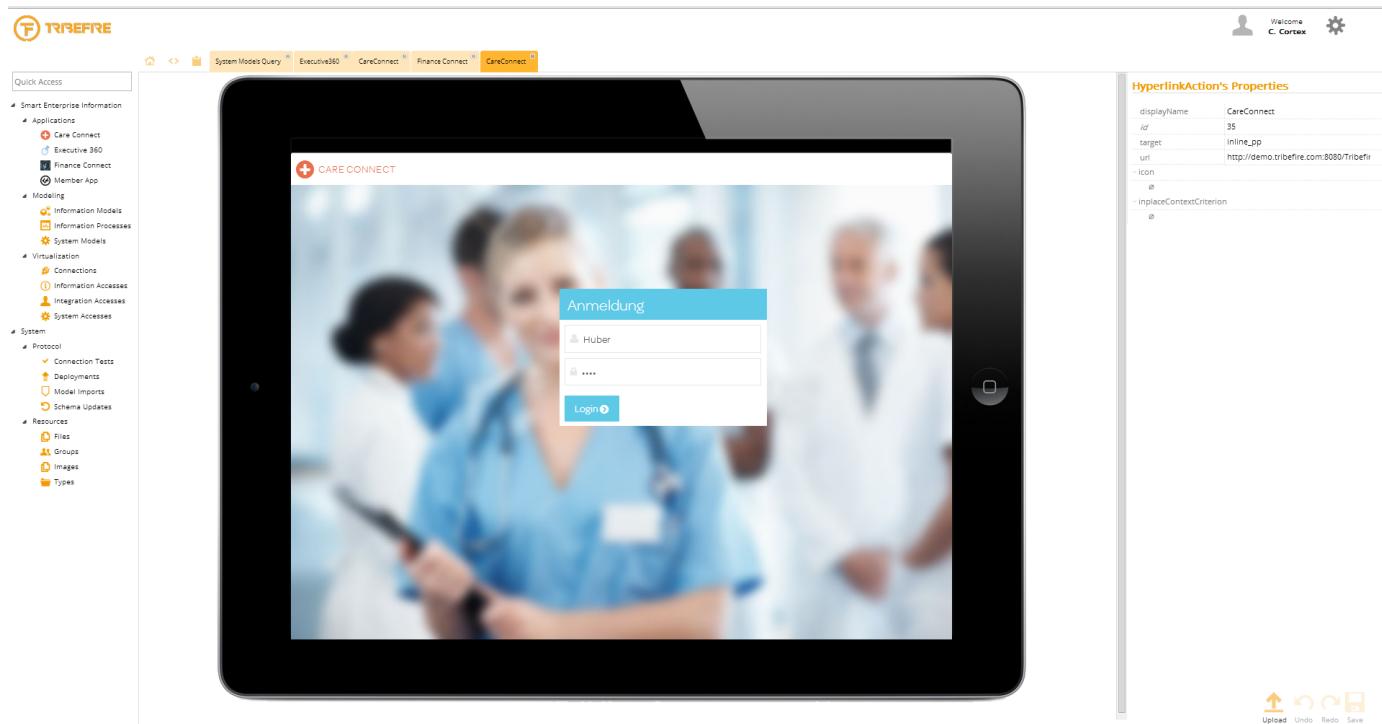
Applications

Application Layer

The application layer is where developers can build their applications that will access the data in tribefire. There are a wide array of options that developers can use to access models inside tribefire, for example web services or REST.

This section of tribefire Control Center aims to replicate that layer by providing links to different applications that make use of the models configured on your instance of tribefire. These links can be configured using the Workbench, allowing to add new links or edit existing ones.

Clicking on an application will execute this application and allow you to use it, either from within GME or externally in a new browser tab. You can also see the properties of each applications link. Including the location of the App.



Modeling

Child pages

▼ Expand / collapse

- Custom Models / Distributed Models
- Entity Types

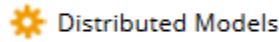
Modeling Layer

The modeling layer is the centerpiece of the tribefire platform. The easy to use and integrated graphical modeler can be used to create basic business entities like "Customer", "Company" and "Opportunity". After the creation of these entities the relationships between these entities will be modeled. This provides the foundation of the Smart Enterprise Information Model, which will now be enriched with more detailed attributes that describe the entities in more detail. tribefire supports a broad variety of entity types like string, integer, date, enumerations etc. the user can choose from.

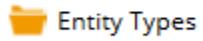
◀ Modeling



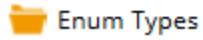
[Custom Models](#)



[Distributed Models](#)



[Entity Types](#)



[Enum Types](#)

This Modeling section of tribefire Expert Mode aims to replicate this layer by grouping together the links for Information Models, Information Processes and System Models:

- [Custom Models](#)
- [Distributed Models](#)
- [Entity Types](#)
- [Enum Types](#)

Custom Models / Distributed Models

Table of Contents

▼ Expand / collapse

- Introduction
- Assembly Panel
- Search Panel
- Properties Panel
- The buttons panel
 - Buttons when the model is selected
 - Buttons when a model is not selected

Introduction

In tribefire cortex everything is a model. And although they are the same from a technical standpoint, they are considered under different names for the sake of clarity. The two most important types of model in the tribefire Control Center are the Custom Models and the Distributed Models.

Custom Models are models that are designed to describe a business use case. They can be uploaded as a Zargo file for example, and are used at a higher level than some other models. That is, they do not concern themselves with the operations of tribefire itself and are only used to model and view data.

An example of Custom Model

The screenshot shows a search interface for 'Custom Models Query'. The results table has columns for 'Name' and 'Id'. One row is shown, corresponding to the 'SalesModel#1.0' entry in the tree view on the left. The 'Name' column contains 'com.braintribe.m' and the 'Id' column contains '12'.

Name	Id
com.braintribe.m	12

Distributed Models, as the name would suggest, are important, system models which are included by standard with every installation of tribefire. Examples include the basic deployment model, this is what Control Center is based on, or the Basic deployment workbench model, this is what Cortex Workbench is based on.

An Example of Distributed Model

The screenshot shows a search interface for 'Distributed Models Query'. The results table has columns for 'Name' and 'Id'. Multiple rows are listed, corresponding to entries in the tree view on the left. The 'Name' column contains 'com.braintribe.m' and the 'Id' column contains values ranging from 1 to 11.

Name	Id
com.braintribe.m	11
com.braintribe.m	3
com.braintribe.m	6
com.braintribe.m	5
com.braintribe.m	2
com.braintribe.m	7
com.braintribe.m	1
com.braintribe.m	4
com.braintribe.m	8
com.braintribe.m	10
com.braintribe.m	9

Assembly Panel

The results will be displayed in the Assembly Panel:

The screenshot shows a search interface for 'Custom Models Query'. At the top, there is a search bar with a magnifying glass icon and a placeholder 'Search'. Below it are navigation buttons for page 1, ordering by 'Id' (with an upward arrow), and a 'Results per page' dropdown set to 25. A 'Switch to A' link is also present. The main area displays a table with one row. The table has columns for 'Name' and 'Id'. The single row contains 'com.braintribe.m' in the Name column and '12' in the Id column. The entire row is highlighted with a yellow background. On the far left, under 'Gm Meta Model', there is a tree view with 'SalesModel#1.0' expanded.

Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

This screenshot shows the 'Basic Search' mode of the search panel. It features a search bar with a magnifying glass icon and a placeholder 'Search'. Below the search bar are navigation buttons for page 1, ordering by 'Id' (with an upward arrow), and a 'Results per page' dropdown set to 25. A 'Switch to Advanced' link is also present. The rest of the interface is identical to the 'Advanced Search' mode shown below it.

Advanced Search

This screenshot shows the 'Advanced Search' mode of the search panel. It includes a search bar with a magnifying glass icon and a checked checkbox. Below the search bar are navigation buttons for page 1, ordering by 'Id' (with an upward arrow), and a 'Results per page' dropdown set to 25. To the right of the search bar are links for 'Save As' and 'Switch to Basic'.

For more information on the Search Panel's functionality.

Properties Panel

To the right of all the models is the properties panel.

General

Id

3

- Name

com.braintribe.model:BasicDeploymentWorkbenchModel#1.0

Model's Properties

- baseType

object

- Entity Types

- DisjunctionCriterion
- Condition
- TypeMatch
- ...

- Enum Types

- ResponseCode
- Operator
- ExecutionMode
- ...

- Meta Data

∅

- simpleTypes

- long
- integer
- boolean



This displays the important information about each model, including its full name (generally, something like: com.braintribe.model:MODELNAME#VERSION_NUMBER) and its properties (including entity and enum types). Attached to this panel are also buttons which you can use to upload a file, undo or redo an action, or save any changes. You can hide or display the details panel by using the button found at the bottom of the screen.

For more information on the Model's properties.

The buttons panel

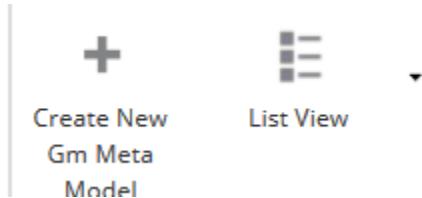
Underneath the models is the button pane. The buttons displayed depend on whether a model is selected or not. Click on the model you would like to select it. To deselect a model, hold CTRL and then click on the model.

Buttons when the model is selected



Button Name	Button Description	Notes
Open	This will display information about this model only. You will see that after opening you can return to the main System Models Query by selecting this arrow above the search pane. The information about the model opened will stay open until you open another model.	You can perform the same action as the open button by double clicking on the model you wish to view
Edit	This will display the GIMA, which will allow you to edit the properties of a specific model.	
Delete	This will delete the specific model.	After deletion the model won't be removed from the list until the search has been refreshed. Because of the way tribefire works, you will also have to save after deletion. This is because tribefire won't persist the changes in its system till saving has taken place. This allows you ultimate flexibility in terms of undo
Refresh	This refreshes the tribefire system.	
Hide Details / Show Details	This will either hide or display the properties panel, depending on the context	
Deploy Model	Deploys model to a repository.	
Auto Enrich	This will automatically enrich your model with metadata.	Metadata is information which describes and affects the behavior of the model. When enriching your model, this information will automatically be added by tribefire. One such example is the configuration of the columns name. It may be that you have a column name firstName. Enriching will add the property display metadata to your model and therefore, it will be displayed in data view as First Name.
Create Model	Allows you to create a new model based on an existing model in tribefire	
Export to Zargo	Allows you to export a model to a zargo file.	
Extract Metadata	This will extract all metadata from this model and store it locally on the tribefire host.	You can find the extracted metadata by navigating to the Files link in tribefire Control Center.
Attach metadata from extract	This allows you to add previously extracted metadata to a model. Selecting this button will display the GIMA allowing you to select the model that should be used and the metadata information that should be added	Before attaching metadata, you will have to uploaded the metadata file before you can use it.
Show/Hide Details	Allows you to select what elements of the model should be shown.	
Add to Clipboard	This will add the currently selected model to tribefire GME's clipboard	

List View	Clicking this button will display a context menu, allowing you to select from three different types of list views: Simple, Detailed Flat	
Thumbnail View	This will display each model as a thumbnail. To switch back to list view, select List View	

Buttons when a model is not selected

If there are no models selected, the following buttons should be available for selection: Create New Model, and List View / Thumbnail View. The List View / Thumbnail buttons should function in exactly the same manner as when a model is selected. Therefore, there is only one button whose functionality is different when no model is selected, Create New Model

Clicking the Create New Model button will display the create new model pane. You should enter then name of your model.

After clicking apply, the new model should appear in its own window, allowing you to add a base type to this object and any Entity Types, Enum Types, MetaData or simple Types. If you wish to create new Entity Types or Enum Types you must first create them by selecting types from the System/Resources section of GME.

Once you have completed adding a new model, you can click save to persist it in the system.

Entity Types

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - Buttons when a GmEntityType is selected
 - Buttons when no GmEntityType is selected

Introduction

The Entity Types page displays all the different entity types which are available for your instance of tribefire. This includes all types belonging to the system models, as well as any models that you have created or uploaded.

The Assembly Panel

GmEntityType	Is Abstract	Is Marked For Discard	Is Plain	Id	Type Signature
DisjunctionCriterion	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1	com.braintribe.model.generic.pr.criter
Condition	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2	com.braintribe.model.query.condition
TypeMatch	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3	com.braintribe.model.generic.pr.criter
SetAddress	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4	com.braintribe.model.generic.manipu
Join	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5	com.braintribe.model.query.join
EntityReference	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	6	com.braintribe.model.generic.value.Er

Property	Description
GmEntityType	The entity type
Is Abstract	Is this entity type abstract.
Is Marked For Discard	Is this entity type marked for discarding.
Is Plain	Is this entity type plain.
Id	The Internal, generated Id type for this entity type.
Type Signature	The type signature of the entity types refers to its name and how it is referenced within tribefire.

The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

<input type="text"/>	<input type="button" value="Search"/>	Ordered by:	Results per page:	Switch to Advanced
1				

Advanced Search

<input checked="" type="checkbox"/>	<input type="button" value="Search"/>	Save As	Switch to Basic
1			

For more information on the Search Panel's functionality.

The Properties Panel

GmEntityType's Properties

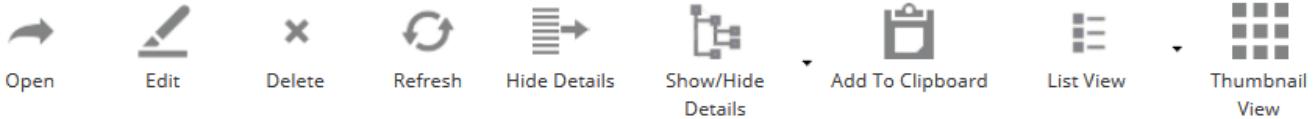
<i>Id</i>	1
- Type Signature	
com.braintribe.model.generic.pr.criteria.DisjunctionCriterion	
Is Abstract	<input type="checkbox"/>
Is Marked For Discard	∅ <input type="checkbox"/>
Is Plain	<input checked="" type="checkbox"/>
- Artifact Binding	
BindingArtifact (15289)	
- Meta Data	
∅	
- Properties	
1. criteria	
2. id	
- Super Types	
1. JunctionCriterion	

GmEntityType's properties

Property	Description
Id	The Internal, generated Id type for this entity type
GmEntityType	The entity type
Type Signature	The type signature of the Java class associated with this type
Is Abstract	Is this entity type abstract
Is Marked For Discard	Is this entity type marked for discarding
Is Plain	Is this entity type plain
ArtifactBinding	This states the dependencies that are required by the entity to function correctly
MetaData	Whether any metadata has been attached to this element.
Properties	The properties associated with this entity type
Super Types	The Super Type of this entity type.

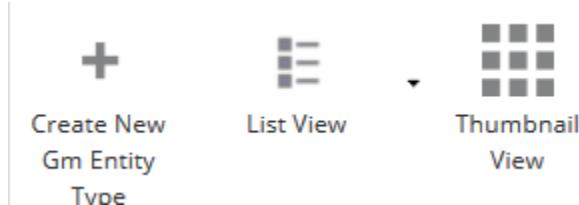
The Buttons Panel

Buttons when a GmEntityType is selected



Button Name	Description	Notes
Open	Opens the selected Entity Type	
Edit	Edits the selected Entity Type	
Delete	Deletes the selected Entity Type	The deleted entity type won't automatically be removed from the list until you have refreshed the Types Query. The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo
Refresh	Refreshes the tribefire system.	
Hide / Show Details	Hides or shows the properties panel	
Add to Clipboard	Adds the selected image resource to tribefire GME's clipboard	
List View	Allow you to select between different list views	
Thumbnail View	Allow you to display all the image resource as thumbnails	

Buttons when no GmEntityType is selected



When no GmEntityType is selected, the panel displays four buttons: Create New Gm Entity Type, List View and Thumbnail View. Apart from Create New Import Type, the other three buttons perform the same functionality as when a Gm Entity Type is selected.

Button Name	Description
Create New Gm Entity Type	Displays the new Gm Entity Type window.

Virtualization

Table of Contents

- ▼ Expand / collapse
 - Introduction

Child pages

- ▼ Expand / collapse
 - Connections
 - Custom Accesses / System Accesses
 - Simulation Mode
 - Access Types

Introduction

This layer is used to create direct connections to repositories. This is done through the use of a series of default connectors, including, but not limited to, database, sharepoint, documentum, twitter, facebook, etc. After creating a connection, the schema of such systems is imported into tribefire, known as "Rough Types". These rough types are also models and are used by the integration accesses to create entities in an automatically generated system model. Each connector has a related integration access which allows this process to work.

Virtualization

-  [Custom Accesses](#)
-  [System Accesses](#)
-  [Connections](#)

The Virtualization area of tribefire Expert Mode replicates this structure by grouping together links for Connections, Information Accesses, Integrations Accesses and System Accesses.

- Connections
- Custom Accesses
- System Accesses

Connections

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- Search Panel
- The Properties Panel
- The Buttons Panel
 - Buttons when a connector is selected
 - Buttons with no selected connections

Introduction

Connections are the link between a repository and an access. This section allows you to define different connections to different repositories. By standard tribefire has several predefined connections available, including JDBC, Salesforce, Documentum and Sharepoint. A [list of the main connectors found in tribefire can be found here](#)

The main connections panel is split into different panels:

The Assembly Panel

Depending on the view that you have selected the models will be displayed in a different way. You can change the view by selecting from three different types of list views (Simple, Detailed, Flat) or a thumbnail view. These views can be selected from the button's panel.

Connector	Deployment Sta...	Description	External Id	Hardwired	Name	Id
MsSQL.connector (MsSQL.connector)	∅	∅	MsSQL.connector		MsSQL.connector	5

Name	Description
Connector	The name of the connection
Deployment State	Describes whether the connection has been deployed or not
Description	A description of this connection
External ID	The name of this connection's external Id
Name	The name of the connector
Id	The Id of this connector.

Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

🔍
Ordered by:
Results per page:
[Switch to Advanced](#)

Advanced Search

🔍

For more information on the Search Panel's functionality.

The Properties Panel

The details panel will display different information depending on the type of connected has has been selected. It will display all the configuration information for the selected connector.

General

<i>Id</i>	6
External Id	SalesModel
Name	SalesModel
- Description	
	Ø
- Model	
	SalesModel#1.0

Status

<i>Deployment State</i>	Ø
-------------------------	---

SmartEnterpriseInformationAccess's Properties

- delegates	Ø
- Workbench Access	Ø

Simulation

<i>Simulation Mode</i>	<input checked="" type="checkbox"/>
------------------------	-------------------------------------



Attached to this panel are also buttons that allow you to upload a file, undo or redo an action and save changes to persist them in the persistence layer.

For more information on the details panel

The Buttons Panel

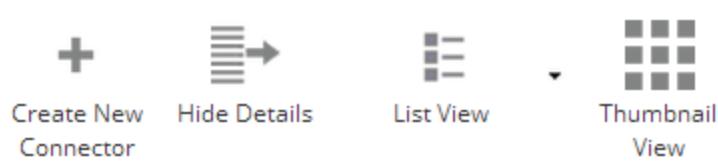
Depending on the context the buttons panel will display different buttons. The context depending on whether a connection has been selected or not.



Buttons when a connector is selected

Button Name	Description	Notes
Open	This opens the currently selected Connector in its own window	
Edit	This opens the Edit Panel, allowing you to edit the properties of the selected connection	
Delete	This button deletes the currently selected connection	<p>The deleted connector won't automatically be removed from the list until you have refreshed the Connections Query</p> <p>The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.</p>
Refresh	This refresh the connections	
Hide Details / Show Details	Depending on the context this will either show or hide the properties panel	
Update DB Schema	The DB Schema is the structure removed from the connection's repository, for example, the tables and columns found in a database. Updating DB schema is useful when the structure of the repository that your connector is attached to has been changed	
Test Connection	This will test the selected connection. Depending on the result of this test a success or failure message will be displayed at the top of the screen next to your login name	
UnDeploy	This undeploys a model. For more information on deployment, see below	
Deploy	For a connection to be valid you must deploy it first. Deployment means that this connection is live and can start to extract information from the repository	
Add To clipboard	This adds the selected connection to the clipboard	
List View	List view provides you with three different types of view: Simple, Detailed, Flat	For more information on List View
Thumbnail View	Thumbnail view displays each connection as a thumbnail.	For more information on Thumbnail View

Buttons with no selected connections



When no connection is selected then the buttons in the buttons panel will change. There are four buttons displayed, three of which function in exactly the same way as when a connection is selected.

The four buttons are: Create New Connector, Hide Details, List View and Thumbnail View

Button Name	Description	Note
Create New Connector	Opens the new connector button, allowing you to create a new connection	<p>This list will show all types of connectors available in your installation of tribefire. Depending on the connector you choose the next screen will be different.</p> <p>Selecting a type which has an Orange Plug icon will display all types of this kind</p> <p>After selecting the type of connector you wish to use, a second screen will be displayed where you can enter the specific properties required for this type of connector to function correctly.</p>

For more information on the different types of connectors

Connectors

Table of Contents

▼ Expand / collapse

- Introduction
- Connectors

Child pages

▼ Expand / collapse

- Generic Jdbc Connection
- MsSQL Connector

Introduction

There are many different connectors available for tribefire, allowing to access a number of different repositories or data sources.

Connectors

[Bloomberg Connector](#)

[Documentum Dfs Connector](#)

[Facebook Connector](#)

[Generic Jdbc Connector](#)

[Google Calander Connector](#)

[Google Mail Connector](#)

[JIRA Connector](#)

[MsSql Jdbc Connector](#)

[Oracle Jdbc Connector](#)

[SAP Connector](#)

[Salesforce Connector](#)

[SharePoint Connector](#)

[Twitter Connector](#)

Generic Jdbc Connection

Table of Contents

Expand / collapse

- Introduction
- Generic Jdbc Connection Properties
 - Status
 - GenericJdbcConnector's Properties
 - Connection
 - Account
 - General

Introduction

The Generic Jdbc Connection allows you to create a generic connection to a Jdbc database, for example MySQL. With this connection you can import schema from your database and, with an access, turn them into models for editing and configuring.

! Before attempting to build a connection and deploy it, you must make sure that the appropriate database driver has been installed to the directory:

TRIBEFIRE_INSTALLATION_DIRECTORY/host/lib

Generic Jdbc Connection Properties

Status

Deployment State **Ø**

Status

Property	Description
Deployment State	This is the deployment state of this connection. It will have no state until the first deployment. Afterwards, it can have either one of two values: Deployed or Undeployed

! Before you are able to test this connection or refresh a db schema, you must first deploy the connection. Likewise, if any changes are made to the properties of this connection, you must also undeploy and then deploy for them to take effect.

GenericJdbcConnector's Properties

- Db Schemas

Ø

GenericJdbcConnector's Properties

Property	Description

Db Schemas	These are the Schema that are returned from the jdbc database and contain the information about each table stored in the database. You do not have to configure this property yourself. Clicking the Update DB Schema button will locate this information from the jdbc database and automatically add it here.
------------	---

Connection

– Url

∅

– Driver

∅

Hibernate Dialect ∅

– Hibernate Dialect Class

∅

Connection

Property	Description
Url	<p>The url for a database connection. Generally this will contain the type of database server, the location and the database GME should access. However, each Jdbc server requires a different url.</p> <p>Click here for a list of standard Jdbc URLs</p>
Driver	<p>The driver that this connection should use to access the jdbc database.</p> <p>Click here for information regarding database drivers</p> <div style="border: 2px solid red; padding: 5px; margin-top: 10px;">  The driver that you define here must be located in the lib directory of your tribefire host. </div>
Hibernate Dialect	This property defines which SQL language the Hibernate access should generate.
Hibernate Dialect Class	This defines the class which the Hibernate dialect should use

Account

Account

User	∅
Password	∅

To access any jdbc database, you must have a account defined in your database server. When Control Center attempts to create a connection to this database, it will provide login credentials provided here. If they are valid, tribefire will gain access and create a connection to the database.

Property	Description
User	The user name of the database account
Password	The password of the database account

General

<i>Id</i>	∅
External Id	∅
Name	∅
– Description	
	∅

General

Property	Description
Id	The automatically generated Id for this connection, created when you persist it to tribefire for the first time
External Id	The name used by external programs to refer to this connector
Name	The name of this connection
Description	You can enter a description of the connection here.

Jdbc URLs**Table of Contents**

▼ Expand / collapse

- Introduction
- Drivers
- Database Information

Introduction

JDBC stands for Java Database Connectivity and provides access to several relational databases. tribefire Control Center allows you to configure different connections which can access a host of different jdbc databases. However, to do so, you must provide the Control Center with the correct database URL and driver, as each database requires different information. This documentation provides the standard syntax and default drivers for each different database.

Drivers

When defining a connector, there are two important configurations you must make. The first is to enter the correct URL that will allow tribefire to access the underlying database server, and the second is to define the correct driver that will be used when establishing a connection. The driver that you define **must** be located in the lib directory of your tribefire host. If this is not the case, you will not be able to access the database.

You can download the appropriate driver from the table below.

Database Information

Database	Driver	URL	Usage Notes
DB2	com.ibm.db2.jdbc.net.DB2Driver	jdbc:db2:hostname:port_Number/databaseName	
Derby	org.apache.derby.jdbc.EmbeddedDriver	jdbc:derby:Database_path;user=USER_NAME;password=USER_PASSWORD	 You can add other parameters to the URL for additional functionality. See here for more information
MsSQL	com.microsoft.sqlserver.jdbc.SQLServerDriver	jdbc:sqlserver://ServerName:portNumber	 For more information on the MsSQL see here  In tribefire GME there is a specific MsSQL connector and it is recommended that you use this connection when accessing a MsSQL database server.
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://ServerName:portNumber/databaseName	
MariaDB	org.mariadb.jdbc.Driver	jdbc:mysql://ServerName:portNumber/databaseName	 When connecting to a MySQL database, it is recommended that you use these details when creating a connection
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@hostname:portNumber:databaseName	

MsSQL Connector

Table of Contents

Expand / collapse

- Introduction
- MsSQL Connection Properties
 - Status
 - MsSqlJdbcConnector's properties
 - Connection
 - Account
 - General

Introduction

The MsSQL connection allows you to connect tribefire Control Center to a MsSQL database. By creating a connection it allows you to build an access which will then retrieve and store information into this underlying database, as well as being able to create searches and manipulations.

MsSQL Connection Properties

The MsSQL connection's properties are split into different sections

Status

Deployment State



Status

Property	Description
Deployment State	This is the deployment state of this connection. It will have no state until the first deployment. Afterwards, it can have either one of two values: Deployed or Undeployed

! Before you are able to test this connection or refresh a db schema, you must first deploy the connection. Likewise, if any changes are made to the properties of this connection, you must also undeploy and then deploy them to take effect.

MssqlJdbcConnector's Properties

Database	Ø
Driver	Ø
Instance	Ø
Version	Ø
– Db Schemas	
Ø	
– Properties	
Ø	

MsSqlJdbcConnector's properties

Property	Description

Database	The database in the MsSQL server that you wish this connection to access
Driver	<p>A drop-down list defining the driver that should be used to complete this connection.</p> <p>You can select from:</p> <ul style="list-style-type: none"> • MicrosoftJdbc4Driver • Jtds <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> i The recommended driver for correct functionality is MicrosoftJdbc4Driver </div> <div style="border: 2px solid red; padding: 10px; margin-top: 10px;"> ! You must ensure that you have the correct drivers in your tribefire/host/lib directory. The recommended driver is <code>sqljdbc4.jar</code>, which should be used in conjunction with the MicrosoftJdbc4Driver Option </div>
Instance	The Instance name of your MsSQL server. You can discover the name of your instance by viewing the MsSQL service. The instance name will be located in brackets next to this service
Db Schemas	These are the Schema that are returned from the MsSQL and contain the information about each table stored in the database. You do not have to configure this property yourself. Clicking the Update DB Schema button will locate this information from MsSQL and automatically add it here.
Properties	You can add any number of String properties to your connector.

Connection

Host	Ø
Port	Ø

Connection

Property	Value
Host	The address of the machine that the MsSQL server is running on.

Port	The open port through which GME can access the MsSQL server
------	---

Account

User	∅
Password	∅

Account

To access any MsSQL database, you must have a account defined in the MsSQL server. When tribefire attempts to create a connection to this database, it must provide valid login credentials.

Property	Description
User	The user name of the account
Password	The password of the account

General

<i>Id</i>	∅
<i>External Id</i>	∅
<i>Name</i>	∅
– Description	
	∅

General

Property	Description
Id	The automatically generated Id for this connection.
ExternalId	The external Id for this connector. It will be used to reference this connector.
Name	The name of this connector in Control Center.
Description	You can add a small description, describing what this connector does.

Custom Accesses / System Accesses

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- Search Panel
- The Properties Panel
- The Buttons Panel
 - Buttons when an access has been selected
 - Buttons when no Access has been selected

Introduction

An Accesses can be thought of as a bridge between a model that you have modeled and the connection that is attached to specific repository. The access is important for manipulating a model, and without it nothing can be done to the model. There are two types of access, Custom Accesses and System Accesses, and although they are the same from a technical standpoint, we use this naming convention for clarity.

The custom accesses are used for models that belong to custom models and system access are used for models that belong to distributed models. After installation, depending on the package installed, there will be no custom accesses and four system accesses, each representing one of the four distributed models included with tribefire: Workbench, Cortex Workbench, Authentication and Authorization, and Cortex.

An example of a Custom Access

Incremental Access	Deployment State	Description	External Id	Hardwired	Name	Simulated	Id
custom.TrainingModel (custom.TrainingModel)	∅	∅	custom.TrainingM	<input type="checkbox"/>	custom.TrainingM	<input type="checkbox"/>	1

An example of a System Access

Incremental Access	Deployment Sta...	Description	External Id	Hardwired	Name	Simulated	Id
Workbench (workbench)	deployed	The workbench fo... workbench	workbench	<input checked="" type="checkbox"/>	Workbench	<input type="checkbox"/>	1
Cortex Workbench (cortex.wb)	deployed	The workbench o... cortex.wb	cortex.wb	<input checked="" type="checkbox"/>	Cortex Workben...	<input type="checkbox"/>	4
Authentiaktion and Authorization (auth)	deployed	The access for us... auth	auth	<input checked="" type="checkbox"/>	Authentiaktion ar...	<input type="checkbox"/>	2
Cortex (cortex)	deployed	The cortex (master) cortex	cortex	<input checked="" type="checkbox"/>	Cortex	<input type="checkbox"/>	3

Both types of access have the same properties.

The Assembly Panel

The main panel displays all the different accesses that are currently configured in your tribefire installation. Depending on the view – which can be set by using the list view button, to access different types of list view, or the thumbnail button (to toggle between a list and thumbnail view).

Incremental Access	Deployment State	Description	External Id	Hardwired	Name	Simulated	Id
custom.TrainingModel (custom.TrainingModel)	∅	∅	custom.TrainingM	<input type="checkbox"/>	custom.TrainingM	<input type="checkbox"/>	1

Property	Description
Access	This will display both the Name and external Id of the access. Next to each access in this column you will notice a triangle icon. Clicking this will reveal editable properties for each access. After clicking on this the access will be extended and extra properties will be displayed. You will notice that some properties are marked by a cloud icon. This means that the information has not yet been loaded, and represents a key feature of tribefire: Lazy loading. This feature will help with performance as not all information has to be loaded at once.
Deployment State	Either Deployed or Undeployed. Indicates the deployment state of an access. To make use of this access and receive data from an underlying repository (through a connection) you must first make sure that the model has been deployed.
Description	This column offers an overview of a simple description of each access. This has no impact on the system itself and is used to help users understand what each integration access does.
Simulation Mode	Simulation Mode is an extremely useful feature when developing your tribefire installation. This mode allows you to test out your system before it is connected to a real repository and data. Using tribefire's own database, SMOOD, Any data and connections used will be saved
ID	The ID number of the access.
External ID	The external Id is used to refer to the access, for example when displaying the access in the cog icon next to the log in screen or when building an app that this access must be connected to
Name	The name of your access

Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

Advanced Search

For more information regarding the search panel.

The Properties Panel

The properties panel displays all the relevant properties for the currently selected access.

 Depending on the type of access that is selected, different information will be displayed in the properties panel. The following information will only detail the most common properties shared by all accesses.

an example of a SMOOD integration access

General

<i>Id</i>	3
External Id	BTModel
Name	BTModel
– Description	
∅	
– Model	
SalesModel#1.0	

Status

<i>Deployment State</i>	∅
-------------------------	---

SmoodAccess's Properties

– Workbench Access	∅
--------------------	---

Simulation

<i>Simulation Mode</i>	<input type="checkbox"/>
------------------------	--------------------------

Connection

<i>File Path</i>	∅
------------------	---

General	
Property	Description
ID	This displays the ID value for this access
External ID	This displays the external ID name for this access
Name	The name of your access
Model	This displays the model which the access describes and is connected to.

Status	
Property	Description
Deployment State	The deployment state of the access. Either deployed or undeployed

Simulation	
Property	Description
Simulation Mode	Whether this access should function in simulation mode

The Buttons Panel

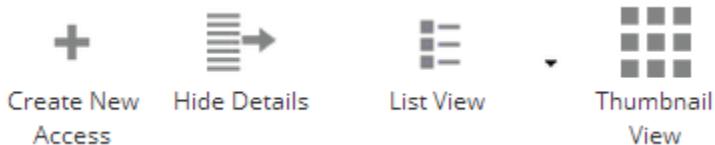
The buttons panel is a dynamic panel will offer you differing options depending on the context. The main context is whether an access has been selected or not. One further context is use of JDBC access. If one of these accesses has been selected then a new button will be displayed, *Create Model From DB*, allowing you to take information from a database and create a new model from it. This button will appear between the deploy and undeploy button.

Buttons when an access has been selected



Button Name	Description	Notes
Open	This displays the access in its own query window. You can navigate back to the main integration access query by selecting in the selection panel, displayed above the main access panel.	
Edit	Opens up the GIMA so that you can edit some configuration properties of your selected access	
Delete	Deletes the currently selected access.	The access won't disappear from the main panel until you have refreshed the Integration Access Query After deleting your access, you must also click the save button to persist this change.
Hide Details / Show Details	Hides or shows the properties panel depending on the context	

Create Workbench Access	This allows you to create a new workbench for you access easily, rather than having to create and configure a system access for yourself.	Create a new Workbench Access
Deploy	Deploys a model for use. Only after deployment can you start to use and work with your access.	When you click deploy the performing action screen will be shown, graying out the rest of the Control Center. After this action has been performed you will either receive a success or failure message If the deployment was successful, the status of the Deployment as show in both the details section and the main access panel will change to reflect its new status. Also the Traces property in the Status section will be updated to reflect the system action that has just taken place on this access
UnDeploy	This undeploys an access. This will effectively take your access off-line (at least in the sense of being able to connect to the connected data repository)	After clicking this button the performing action screen will be displayed and then once the action is completed a success or failure will be displayed. If the undeployment was successful then the status of the deployment as shown in both the details section and the main access panel will change to reflect its new status. Also, the Traces property in the status section will be updated to reflect the system action that has just taken place on this access.
Create Model From DB (only with JDBC accesses)	This allows you to use a JDBC to extra a schema from a selected database and import to create a new model in tribefire	
Add to Clipboard	Adds the selected access to the clipboard	
List View	This allows you to select between three different types of list (simple, detailed, flat)	
Thumbnail View	This displays all accesses in a thumbnail view.	

Buttons when no Access has been selected

When no access has been selected the buttons panel will display only four buttons: Create New Access, Hide Details, List View and Thumbnail view. All the buttons except for the Create New Access view retain the same functionality as before.

Button Name	Description
Create New Access	Clicking this button will display a list of standard accesses available in your installation of tribefire. After selecting the type of access that you require, the GIMA will be displayed, where you can enter the specific properties required by this access type.

After entering the required information the Access will be opened (this is the same functionality as clicking the open button on a selected access.) Here you can add extra information and configure the access to a specific model that this access describes. After the completion of this configuration you can then save your access to persist it and also deploy or create a new workbench to edit it.

Simulation Mode

Table of Contents

Expand / collapse

- Introduction
- Working in Simulation Mode
 - Turning Simulation Mode On
 - Turning Simulation Mode Off
- Usage

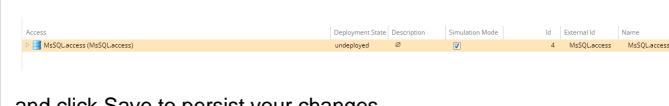
Introduction

Simulation Mode is a feature of tribefire which allows you to test the configuration of your models and accesses without the need for an actual connection to a repository. This means you can use an access or even design an app, based on these accesses and use mock data to test your system. When you are ready to create actual connections, you must only disable the simulation mode and the connection will be made.

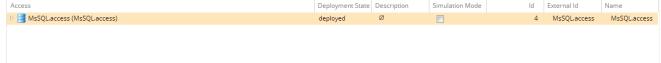
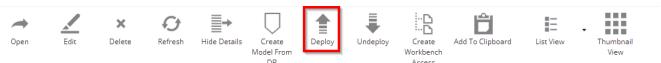
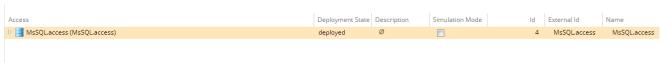
When you use your accesses in simulation mode the access will store any test data locally in Smood.

Working in Simulation Mode

Turning Simulation Mode On

Step #	Task
1	<p>If your access has already been deployed, you must first undeploy it.</p>  <p>Click the Undeploy button to do so.</p> 
2	Your access will be undeployed.
3	<p>Click the Simulation Mode option so that it becomes checked...</p>  <p>and click Save to persist your changes.</p> 
4	Click Deploy
5	Once deployed your access is now ready to be used in simulation mode.

Turning Simulation Mode Off

Step #	Task
1	<p>Once you ready to connect you access to a repository, you must turn simulation mode off. Deselect the option Simulation Mode...</p>  <p>...and click Save to persist your changes.</p>
2	<p>Click undeploy...</p>  <p>and then deploy.</p> 
3	<p>Once deployed, your access will now be live and use a repository for its data source instead of Smood, as is the case in simulation mode.</p> 

Usage

The following will show you how the correct usage of simulation mode and how it works in relation to tribefire.

Hibernate Access

In tribefire a hibernate access has been created. This access has a connection to a MsSQL database.

Access	Deployment State	Description	Simulation Mode	Id	External Id	Name
MsSQL.access (MsSQL.access)	deployed	Ø	<input checked="" type="checkbox"/>	4	MsSQL.access	MsSQL.access

Explorer Mode

In Explorer Mode, we can see that this access has been provided with data from the MsSQL database.

Person Query						
<input type="text"/> Ordered by: ID ▾ Results per page: 25 ▾ Switch to Advanced						
Person	birthplace	dateOfBirth	firstName	ID	nationality	secondName
Person (3)	Manchester	08/21/1986 00:00	Roberts	3	British	Smith
Person (4)	London	09/24/1984 00:00	Lisa	4	British	Roberston
Person (5)	Vienna	04/28/1979 01:00	Hans	5	Austrian	Heuber
Person (6)	Graz	04/10/1990 00:00	Maria	6	Austrian	Goesser
Person (7)	Detroit	03/16/1992 00:00	Peter	7	American	Stubbs

Here we can see the corresponding data in MsSQL.

	ID	firstName	secondName	dateOfBirth	birthplace	nationality
1	3	Roberts	Smith	1986-08-23	Manchester	British
2	4	Lisa	Roberston	1984-09-26	London	British
3	5	Hans	Heuber	1979-04-30	Vienna	Austrian
4	6	Maria	Goesser	1990-04-12	Graz	Austrian
5	7	Peter	Stubbs	1992-03-18	Detroit	American

Simulation Mode

The access is then placed in simulation mode, which you can do by following the steps above.

Access	Deployment State	Description	Simulation Mode	Id	External Id	Name
MsSQLaccess (MsSQLaccess)	deployed	Ø	<input checked="" type="checkbox"/>	4	MsSQLaccess	MsSQLaccess

If we look in Data View again, you will see that there is now no data shown. This is because in simulation mode, we are no longer using the connection to the data source, but instead storing and retrieving the data locally, from an instance of Smood.

Person	Order	Count	Search	Navigation	Actions
Person Query	1	25	<input type="text"/>	Ordered by: ID Results per page: 25	

We can also test this access by adding some mock data.

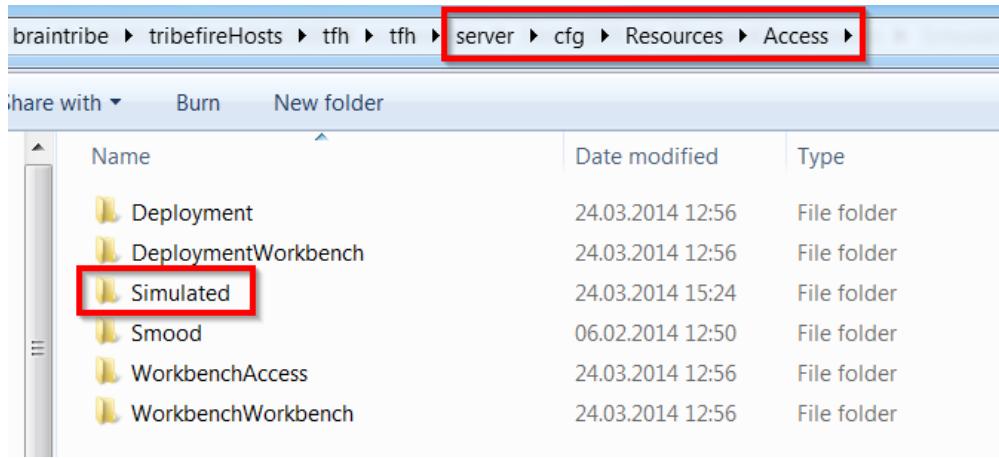
Person	birthplace	dateOfBirth	firstName	ID	nationality	secondName
Person (1)	New York	10/23/1996 15:24	Jane	1	American	Jones

The data in the database stays the same as it was before.

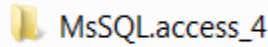
The screenshot shows a software interface with a 'Results' tab selected. Below it is a table with the following data:

	ID	firstNa...	secondNa...	dateOfBirth	birthplace	nationality
1	3	Roberts	Smith	1986-08-23	Manchester	British
2	4	Lisa	Roberston	1984-09-26	London	British
3	5	Hans	Heuber	1979-04-30	Vienna	Austrian
4	6	Maria	Goesser	1990-04-12	Graz	Austrian
5	7	Peter	Stubbs	1992-03-18	Detroit	American

Instead, the mock data is stored locally on the tribefire host. In the folder server/cfg/Resources/Access/Simulated.



Here are a list of directories, each one representing an instance of a simulated access.



In this directory you will find a XML file. If you open it, you will see that the data we just entered has been stored here.

```
<entity id="0" type="com.braintripe.$dbo.Person">
  <property name="ID">
    <integer>1</integer>
  </property>
  <property name="birthplace">
    <string>New York</string>
  </property>
  <property name="dateOfBirth">
    <date>1996-10-23T15:24:00.000+0200</date>
  </property>
  <property name="firstName">
    <string>Jane</string>
  </property>
  <property name="nationality">
    <string>American</string>
  </property>
  <property name="secondName">
    <string>Jones</string>
  </property>
```

Access Types

Table of Contents

- ▼ Expand / collapse
 - Introduction
 - Accesses

Introduction

Each model requires an access so that tribefire can make manipulations on it. Depending on the type of model, and the type of data the model should receive, you will require a different access. There are also connectors that correspond to the different types of data repositories that tribefire can use. You should also use the correct access when configuring it.

Accesses

- [Bloomberg Access](#)
- [Documentum Access](#)
- [Facebook Access](#)
- [GSC DSC Challenge Access](#)
- [Google Calendar Access](#)
- [Google Mail Access](#)
- [Hibernate Access](#)
- [JIRA Access](#)
- [SAP Access](#)
- [Salesforce Access](#)
- [SharePoint Access](#)
- [Skype Access](#)
- [Smart Access](#)
- [Smart Enterprise Information Access](#)
- [Smood Access](#)
- [Twitter Access](#)
- [Xml Access](#)

Hibernate Access

Table of Contents

▼ Expand / collapse

- Introduction
 - General
 - Simulation
 - HibernateAccess's Properties
 - Connection

Introduction

The Hibernate Access is an access which uses a database connection (either a generic Jdbc Connection or a MsSQL connection) to manipulate data from a database. This allows you to create a model from tables found in a database.

General

<i>Id</i>	∅
<i>External Id</i>	∅
<i>Name</i>	∅
– Description	
	∅
– Model	
	∅

General

Property	Description
<i>Id</i>	The automatically generated Id for this access, defined when this access is persisted for the first time.
<i>External ID</i>	The external ID of this access, it is used by external programs (such as Apps) to refer to this access.
<i>Name</i>	The name of this access.
<i>Description</i>	You can add a description of this access here.
<i>Model</i>	The model in tribefire with which this access is associated with. You can create model from the database by clicking the Create Model From DB button located at the bottom of GME. <div style="border: 1px solid orange; padding: 10px; margin-top: 10px;">  You must first deploy your access, before you can create a new model from a database. </div>

Status

Deployment State



Property	Description
Deployment State	This is the deployment state of this access. It will have no state until the first deployment. Afterwards, it can have either one of two values: Deployed or Undeployed.

Simulation

Simulation Mode



Simulation

Property	Description
Simulation Mode	Determines whether this access should function in Simulation Mode. Simulation Mode means that no connection to a database is required. This access will function as a Smood access until you turn off the Simulation Mode and connect to a database, allowing you to create and configure models and information without worrying the underlying connection.

HibernateAccess's Properties

– Workbench Access



HibernateAccess's Properties

Property	Description
Workbench Access	Defines whether this Access has a workbench access. You can create a workbench access by clicking on the Create Workbench Access button located at the bottom of Control Center.

Connection

- Connection



Connection

Property	Description
Connection	This defines a valid connection to a database. Unless you are working in Simulation Mode, you must provide a valid connection for the correct functionality of this access.

Smart Enterprise Information Access

 Work in progress!

Table of Contents

▼ Expand / collapse

- Introduction

Introduction

 To Do

This page will contain information on the proper configuration of a Smart Enterprise Information Access. At the moment this access is not yet available in tribefire Demo GME 2.0 (11/03/2014), this page will be updated when it is. The purpose of having this page at the moment is for linking/placeholding purposes.

Smood Access

Table of Contents

▼ Expand / collapse

- Introduction
- Smood Access properties
 - General
 - Status
 - SmoodAccess's Properties
 - Simulation
 - Connection

Introduction

Smood stands for Smart Memory Object Oriented Database and is Braintribe's own implementation for model persistence. Creating a new Smood Access will create a new instance of the Smood database and will persist any data within tribefire. This data is then stored locally in your tribefire host, in the form of a XML file.

Smood Access properties

The Smood Access' properties are divided into several different sections

General

<i>Id</i>	10
<i>External Id</i>	SmoodAccess
<i>Name</i>	SmoodAccess
- Description	
∅	
- Model	
∅	

General

Property	Description
<i>Id</i>	The automatically generated Id for this Smood Access.
<i>External Id</i>	The reference used by external apps to connect to this access. <div style="border: 2px solid red; padding: 5px; margin-top: 10px;">  This property is mandatory. You will receive an error message if you try to save a Smood access without an external id. </div>
<i>Name</i>	The name of this access.
<i>Description</i>	A property field which allows you to add a description of what this access is used for.

Model	The Model which this access is connected to.
<p> Although this property is not mandatory, that is you can save an access without a model, you must attach a valid model to this property before the access can be deployed.</p>	

Status

Deployment State | 

Status

Property	Description
Deployment State	<p>Whether this access has been deployed or not. Deploying an access means making a connection to an underlying repository.</p> <p> When you first create an access, it will have no deployment state. This value will change after you deploy a model.</p>

SmoodAccess's Properties

- Workbench Access



SmoodAccess's Properties

Property	Description
Workbench Access	A workbench access allows you to customize your Smood Access with queries, actions, hyperlinks and so on. This property defines what workbench access (if any) this Smood Access should use.

Simulation

Simulation Mode | 

Simulation

Property	Description
Simulation Mode	Whether this Smood Access should function in simulation mode or not.

Connection

File Path



Connection

Property	Description
File Path	This defines the actual location of the data.xml file that stores this Smood Access' data.  This will automatically be configured upon deployment, so you do not have to configure this property yourself.

tribefire GME - System

Table of Contents

▼ Expand / collapse

- Introduction
- Extensions
- Actions
- Resources

Child pages

▼ Expand / collapse

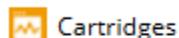
- Connection Tests
- Deployments
- Model Imports
- Schema Updates
- Files
- Property Groups
- Images

Introduction

Along with the key features of the tribefire system—based upon the three layers approach, Applications, Modeling and Virtualizations—Control Center also provides important logging and resource management, as well as being to able to extend tribefire by using cartridges and processors. This includes the ability to view log information regarding connection tests, deployment attempts and so on. In addition to this, you can also view file and image resources available in your instance of tribefire, along with the available entity types and groups, and install cartridges or processors to effect changes on tribefire, and therefore your models, based on your requirements. These fully customizable elements can be installed simply through the addition of custom WAR files into your tribefire installation and then configuring them in Control Center.

The individual links are grouped under three sub-headings: Extensions, Actions and Resources.

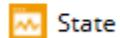
Extensions



Cartridges



Apps

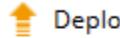


State Change Processors



Action Processors

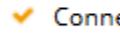
Actions



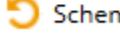
Deployments



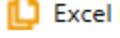
Model Imports



Connection Tests



Schema Updates



Excel Data Imports

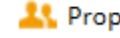
Resources



Images



Files



Property Groups

Extensions

- Cartridges
- Apps
- State Change Processors
- Action Processors

Actions

- Deployments
- Model Imports
- Connection Tests
- Schema Updates
- Excel Data Imports

Resources

- Images
- Files
- Property Groups

Connection Tests

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - Buttons when a test connection is selected
 - Buttons when no test connection is selected

Introduction

The Connection Tests query displays all JDBC Connector tests that have been attempted in your installation of tribefire and is divided into four panels: Assembly Panel, Search Panel, Buttons Panel and Properties Panel.

i You can manually test a JDBC connection by navigating to the Connections Query, displayed in the Virtualization section in the left-hand panel, selecting a JDBC connection and then clicking the Test Connection button. The resulting log from this connection attempt will be displayed here.

The Assembly Panel

TestJdbcConnector	Execution End	Mode	Id	Execution Start	State	Result		
						Code	Detailed Message	Message
▷ ✓ scheduled	∅	synchron	17	∅	scheduled			

Property	Description
TestJdbcConnector	Displays the state and, if any exists, the result of the connection test.
Execution End	When the execution of this test connection ended.
Mode	Either synchron or asynchron.
ID	The mode of the connection test attempt.
Execution Start	The generated ID for this connection test attempt.
Code	When the execution of this connection test was started.
Detailed Message	The result of the test connection.
Message	A more detailed message on the success or failure of the connection test.

The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

The screenshot shows a search interface with a search bar at the top containing a magnifying glass icon. To the right of the search bar are navigation buttons for page 1 (left, right), sorting options ('Ordered by: Id' with up and down arrows), and results per page ('Results per page: 25'). There is also a link to 'Switch to Advanced'.

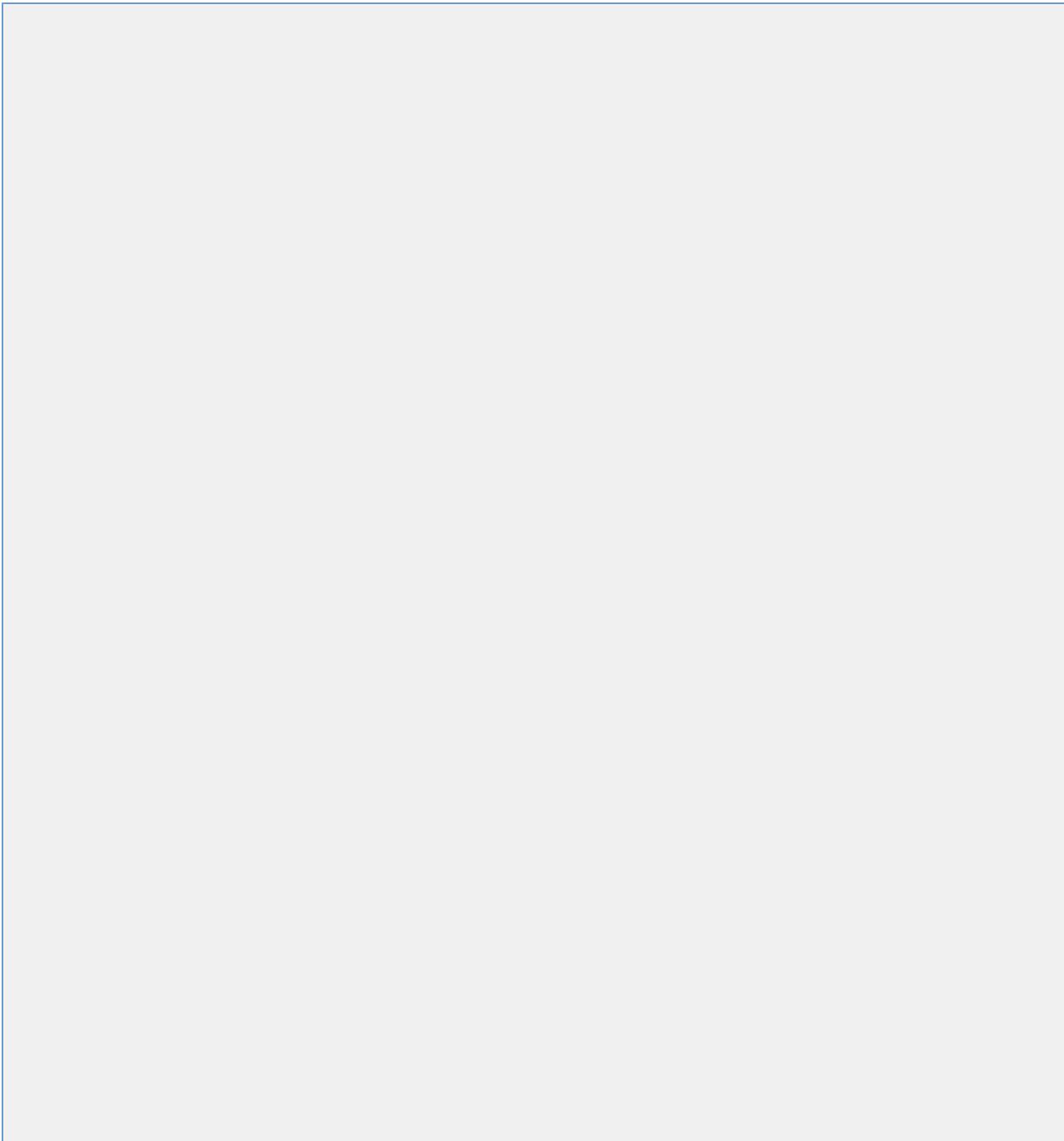
Advanced Search

The screenshot shows an advanced search interface with a search bar at the top containing a checked checkbox icon. To the right of the search bar are navigation buttons for page 1 (left, right), a 'Save As' button, and a 'Switch to Basic' link.

For more information on the Search Panel's functionality.

The Properties Panel

The details panel displays the relevant properties for each instance of a test connection.



Timing

Execution Start \emptyset

Execution End \emptyset

Status

Code \emptyset

State scheduled

– Message

\emptyset

– Detailed Message

\emptyset

Mode synchron

General

Id 17

– Connection

SampleMSSqlConnector (SampleMsSqlConnector)



Timing	
Property	Description
Execution Start	When the execution of this test connection began.
Execution End	When the execution of this test connection ended.

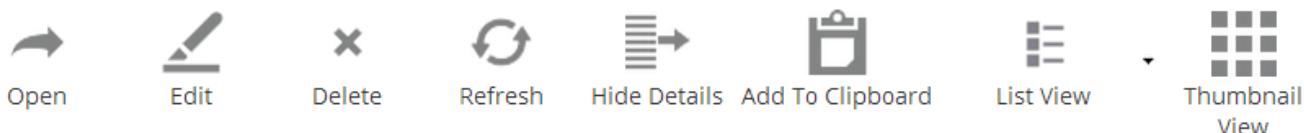
Status	
Property	Description
Code	The result of this test connection. Either Success or Failure.
State	The state of this test connection, it has four possible values: <ul style="list-style-type: none"> Triggered - The test connection has been triggered. (that is, started) Scheduled - The test connected is scheduled Executing - The test connection is currently under way Executed - The test connection has been completed
Message	A short message detailing the test connection attempt.
Detailed Message	A more detailed message detailing the success or failure of the test connection attempt.
Mode	What mode this test connection uses. Either Synchron or Asynchron.

General	
Property	Description
ID	The generated, internal ID for the selected test connection
Connection	The connection that should be tested.

The Buttons Panel

The buttons panel is a dynamic panel will offer you differing options depending on the context.

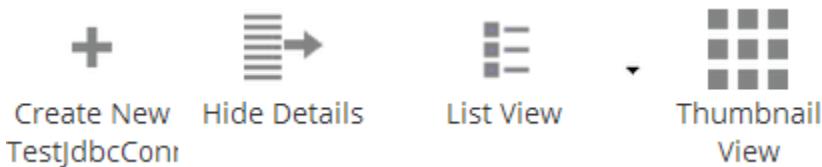
Buttons when a test connection is selected



For more information on the details panel

Button Name	Description	Notes
Open	Opens the selected test connection in a new view for editing.	
Edit	Edits the selected test connection.	

Delete	Deletes the selected test connection.	The deleted test connection won't automatically be removed from the list until you have refreshed the Connection Tests Query The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.
Refresh	Refreshes the tribefire system.	
Hide / Show Details	Hides or shows the properties panel	
Add to Clipboard	Adds the selected test connection to tribefire Control Center's clipboard	
List View	Allow you to select between different list views.	
Thumbnail View	Allow you to display all the connection tests as thumbnails.	

Buttons when no test connection is selected

When no connection test object has been selected, the panel will display four buttons: Create New Test JDBC Connection, Hide Details, List View and Thumbnail View. Apart from the Create New Test JDBC connection, all the other buttons perform the same functionality as when a connection test object has been selected.

Button Name	Description
Create New Test JDBC Connection	This opens up the new test connection window and allows you to configure a new test connection.

Deployments

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel

Introduction

The deployment Query tracks the logging information regarding all deployments attempted by the system and is divided into four panels: The Assembly panel, the search panel, the properties panel and the buttons panel.

The Assembly Panel

Deploy	Execution End	Mode	ID	Execution Start	State	Code	Detailed Message	Message
▷ executed success	02/14/2014 14:03	synchron	5	02/14/2014 14:03	executed	success	∅	Successfully executed
▷ executed failure	02/14/2014 14:45	synchron	9	02/14/2014 14:45	executed	failure	Deployment action fail	deploy action failed.
▷ executed failure	02/14/2014 14:46	synchron	10	02/14/2014 14:46	executed	failure	Deployment action fail	deploy action failed.
▷ executed failure	02/17/2014 09:28	synchron	11	02/17/2014 09:28	executed	failure	Deployment action fail	deploy action failed.
▷ executed success	02/17/2014 14:52	synchron	12	02/17/2014 14:52	executed	success	∅	Successfully executed
▷ executed success	02/17/2014 14:52	synchron	14	02/17/2014 14:52	executed	success	∅	Successfully executed
▷ executed failure	02/18/2014 11:13	synchron	15	02/18/2014 11:13	executed	failure	Deployment action fail	deploy action failed.
▷ executed failure	02/18/2014 11:14	synchron	16	02/18/2014 11:14	executed	failure	Deployment action fail	deploy action failed.
▷ executed success	02/18/2014 15:21	synchron	19	02/18/2014 15:21	executed	success	∅	Successfully executed

The main panel will display all the recorded deployment attempts stored in tribefire, along with specific important properties associated with each deployment attempt.

Property	Description
Deploy	The state and code of the deployment attempt.
Execution End	When the execution of this attempt ended.
Mode	Either Synchron or Asyncron.
ID	The generated internal ID of the deployment attempt.
Execution Start	When the execution of this attempt began.
State	The state of this deployment attempt: <ul style="list-style-type: none"> • Scheduled - The deployment attempt is scheduled • Executing - The deployment attempt is currently under way • Executed - The deployment attempt has been completed
Code	Whether this attempt succeeded or failed.
Detailed Message	A detailed message about the success or failure of this attempt.
Message	A short message about this deployment attempt.

The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

The screenshot shows a search interface with a header bar containing a search input field, a magnifying glass icon, page navigation (1), sorting options (Ordered by: Id,升序↑,降序↓), a results per page dropdown (Results per page: 25), and links to 'Switch to Advanced' and 'Switch to Basic'. Below this is a section titled 'Advanced Search' with a checked checkbox and similar navigation controls. A note below the search bar states: 'For more information on the Search Panel's functionality.' A large blue rectangular box covers the majority of the page content area.

Advanced Search

For more information on the Search Panel's functionality.

The Properties Panel

The properties panel displays the relevant properties for each instance of a deployment attempt.

Timing

Execution Start	02/14/2014 14:03
Execution End	02/14/2014 14:03

Status

Code	success
State	executed
- Message	
Successfully executed Action.	
- Detailed Message	
Ø	
Mode	synchron

General

Id	5
- Deployable	
SampleAccess (SampleAccess)	



Timing

Property	Description
Execution Start	When the execution of this deployment attempt started.
Execution End	When the execution of this deployment attempt completed.

Status

Property	Description
Code	Whether this deployment attempt succeeded or failed.
State	The state of this deployment attempt: <ul style="list-style-type: none"> • Triggered - The deployment attempt has been triggered. (that is, started) • Scheduled - The deployment attempt is scheduled • Executing - The deployment attempt is currently under way • Executed - The deployment attempt has been completed
Message	A brief message detailing the deployment attempt.
Detailed Message	A detailed message about the success or failure of this deployment attempt.
Mode	Either Synchron or Asynchron.

General

Property	Description
ID	The generated internal ID of this deployment attempt.
Deployable	The object that this deployment attempt is associated with. That is, what object was deployed.

Model Imports

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - Buttons when a Model Import Object is selected
 - Buttons when no Model Import object is selected

Introduction

The Model Imports Query tracks all the logging information regarding the importing of models from a database and is divided into four panels: the assembly panel, the search panel, the properties panel and the buttons panel.

The Assembly Panel

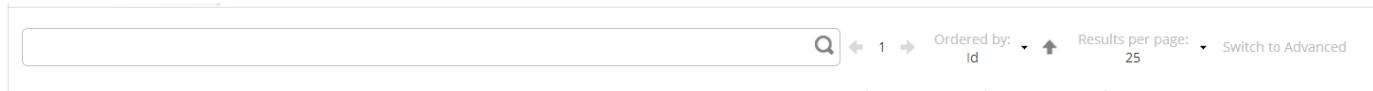
ImportTypeFromDbSchema	Execution End	Mode	Id	Execution Start	State	Code	Detailed Message	Message
▷ executed failure	02/18/2014 17:05	synchron	22	02/18/2014 17:05	executed	failure	java.lang.NullPointerException	Error while executing /
▷ scheduled	∅	synchron	23	∅	scheduled			

Property	Description
ImportTypeFromDbSchema	The status and code of model import attempt.
Execution End	When the execution of this model import attempt completed.
Mode	Either Synchron or Asynchron.
Execution Start	When the execution of this model import began.
State	The state of this model import attempt: <ul style="list-style-type: none"> • Triggered - The model import attempt has been triggered • Scheduled - The model import attempt is scheduled • Executing - The model import attempt is currently under way • Executed - The model import attempt has been completed
Code	Whether this model import attempt succeeded or failed.
Detailed Message	A detailed message about the success or failure of this model import attempt.
Message	A short message about the model import attempt.

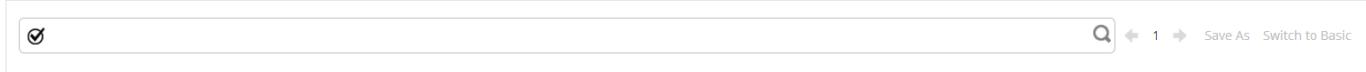
The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

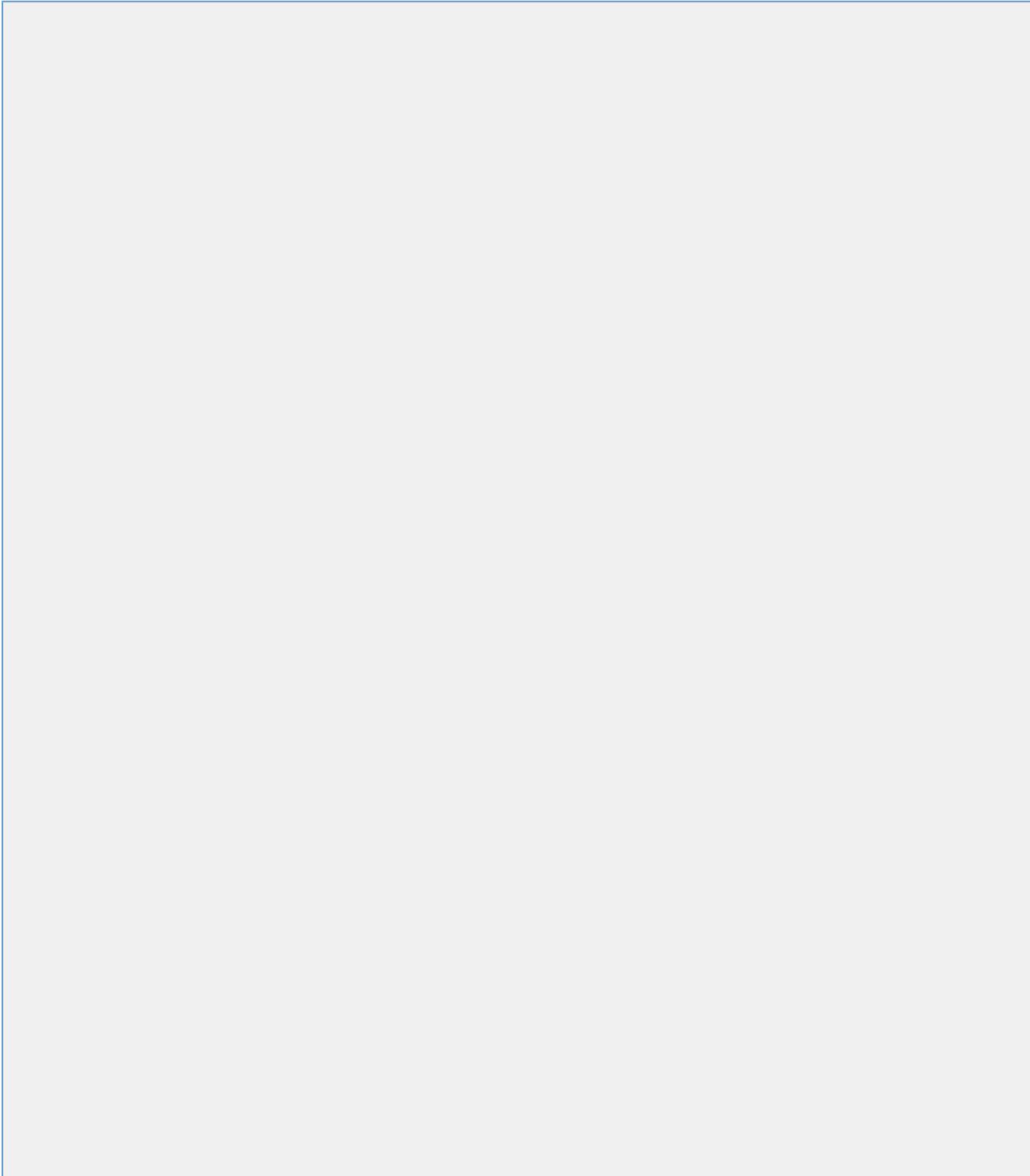


Advanced Search



For more information on the Search Panel's functionality.

The Properties Panel



Timing

Execution Start	02/18/2014 17:05
Execution End	02/18/2014 17:05

Status

Code	failure
State	executed

- Message

Error while executing Action

- Detailed Message

```
java.lang.NullPointerException at  
com.braintribe.model.processing.deployment.action.ImportType...  
at  
com.braintribe.model.processing.sp.action.ActionStateChangePro...  
at  
com.braintribe.model.processing.sp.invocation.AbstractSpProces...  
at  
com.braintribe.model.processing.sp.aspect.StateProcessingAspect...  
at  
com.braintribe.model.processing.sp.aspect.StateProcessingAspect...  
at  
com.braintribe.model.access.impl.aop.context.impl.AroundConte...  
at  
com.braintribe.model.access.impl.aop.AopAccess.withAop(AopAcc...  
at  
com.braintribe.model.access.impl.aop.AopAccess.applyManipula...  
at
```



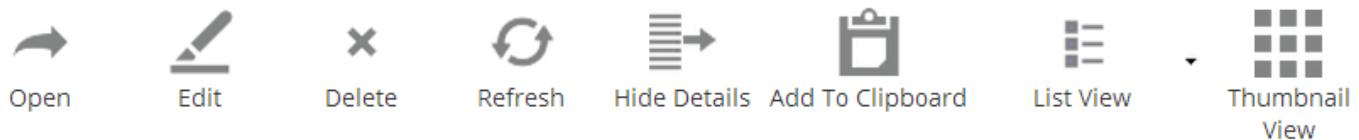
Timing	
Property	Description
Execution Start	When the execution of the model import began.
Execution End	When the execution of the model import completed.

Status	
Property	Description
Code	Whether this model import attempt succeeded or failed.
State	The state of this model import attempt: <ul style="list-style-type: none"> • Triggered - The model import attempt has been triggered • Scheduled - The model import attempt is scheduled • Executing - The model import attempt is currently under way • Executed - The model import attempt has been completed
Message	A short message about the model import attempt.
Detailed Message	A detailed message about the success or failure of the model import attempt.

The Buttons Panel

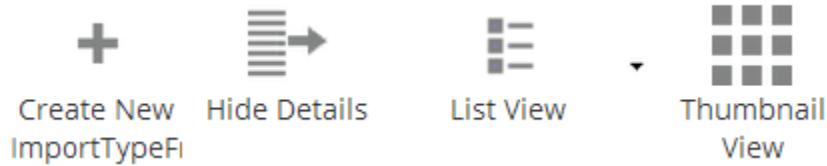
The buttons panel is a dynamic panel will offer you differing options depending on the context.

Buttons when a Model Import Object is selected



Button Name	Description	Notes
Open	Opens the selected Model Import object in its own window for editing.	
Edit	Opens the Model Import object GIMA.	
Delete	Deletes the Model Import object.	The deleted test connection won't automatically be removed from the list until you have refreshed the Connection Tests Query. The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.
Refresh	Refreshes the tribefire system.	
Hide/Show Details	Hides or shows the properties panel.	

Add to clipboard	Adds the selected Model import object to tribefire GME's clipboard.	
List View	Allows you to select from three different list views	
Thumbnail View	Allows you to display the Model Import objects in the thumbnail view	

Buttons when no Model Import object is selected

When no Model Import Object is selected, the panel displays four buttons: Create New Import Type, Hide Details, List View and Thumbnail View. Apart from Create New Import Type, the other three buttons perform the same functionality as when a Model Import Object is selected.

Button Name	Description
Create New Import Type	Opens up a new Model Import display, allowing you to create a new Model Import attempt

Schema Updates

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - When a DB Schema Update Object is selected
 - When no DB Schema Update Object is selected

Introduction

A schema is the informational structure that is gathered from a connection. It is used to create a prototype which will then be used by an access to map the data to a user model, creating entities and Enums from this information. When this schema is changed in a repository, a Schema Update must take place so that the model can reconfigure the changes in tribefire. The Schema Updates Query displays all logging information regarding this.

The Assembly Panel

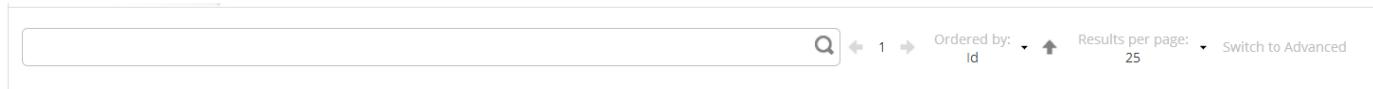
UpdateDbSchema	Execution E...	Mode	Id	Execution Start	State	Result
▶ ⚡ scheduled	∅	synchron	24	∅	scheduled	Code Detailed Message Message

Property	Description
UpdateDbSchema	Main property in display panel, shows the State and Code for each Update DB Schema attempt.
Execution Ends	When this Update DB Schema attempt ended.
Mode	Either Synchron or Asynchron.
ID	The generated, internal ID for this UPdate DB Schema attempt.
Execution Start	When this Update DB Schema attempt began.
State	The state of this Update DB Schema attempt: <ul style="list-style-type: none"> • Triggered - The Update DB Schema attempt has been triggered • Scheduled - The Update DB Schema attempt is scheduled • Executing - The Update DB Schema attempt is currently under way • Executed - The Update DB Schema attempt attempt has been completed
Code	Whether this Update DB Schema attempt succeeded or failed.
Detailed Message	A detailed message about the success or failure of the Update DB Schema attempt.
Message	A message about the Update DB Schema attempt.

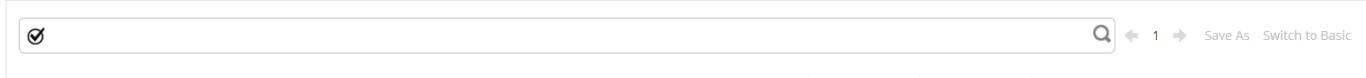
The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

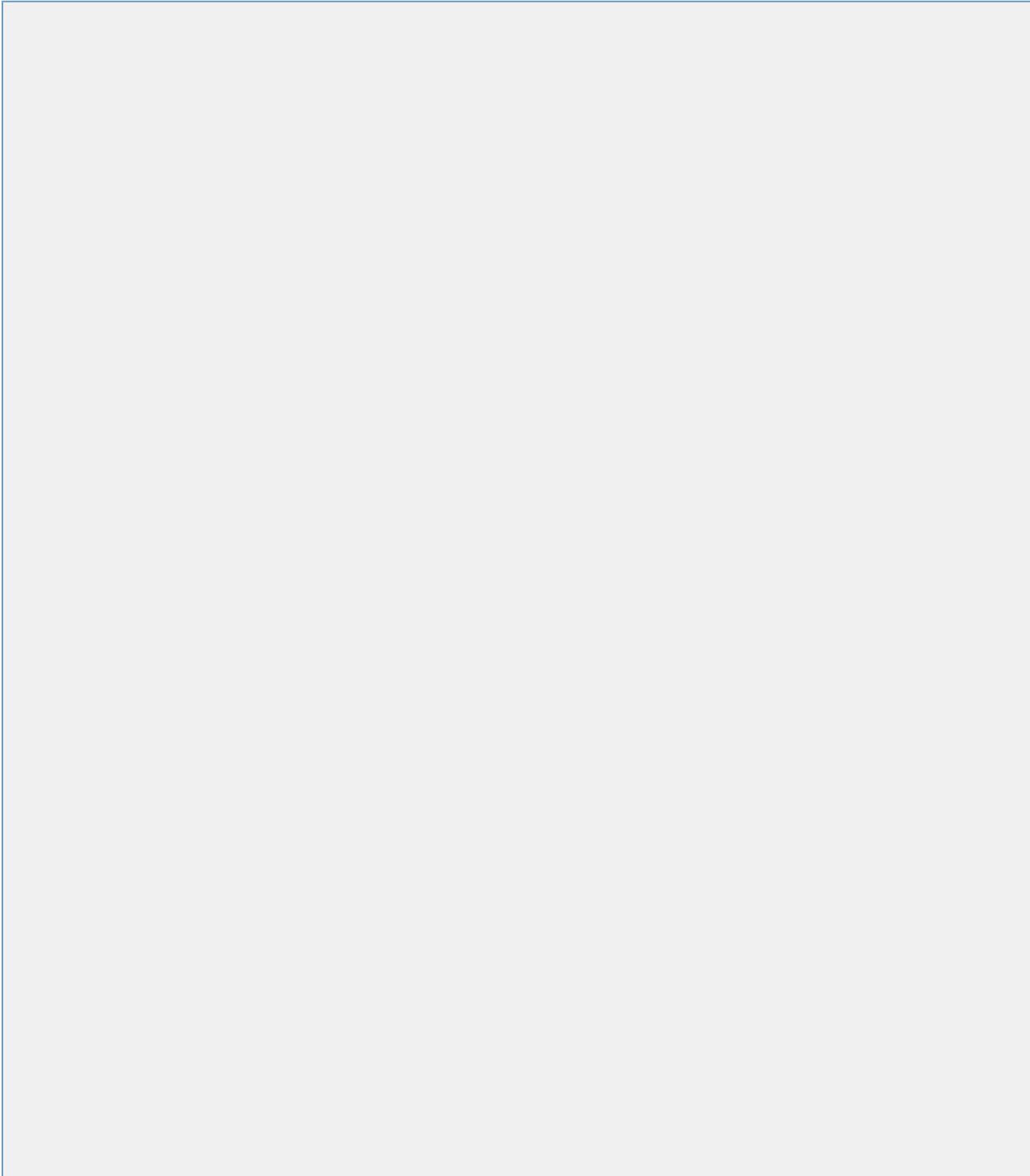


Advanced Search



For more information on the Search Panel's functionality.

The Properties Panel



Timing

Execution Start	∅
Execution End	∅

Status

Code	∅
State	scheduled
- Message	∅
- Detailed Message	∅
Mode	synchron

General

Id	24
- Connection	∅

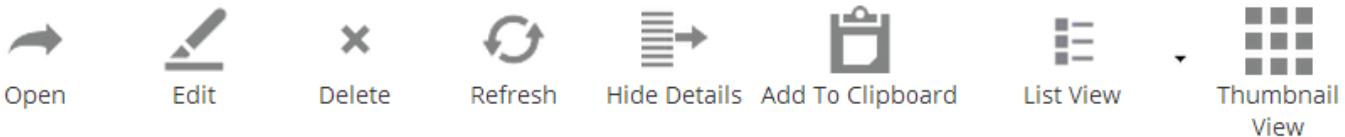


Timing	
Property	Description
Execution Start	When the execution of the Update DB Schema attempt began.
Execution End	When the execution of the Update DB Schema attempt completed.

Status	
Property	Description
Code	Whether this Update DB Schema attempt succeeded or failed.
State	<p>The state of this model import attempt:</p> <ul style="list-style-type: none"> • Triggered - The Update DB Schema attempt has been triggered • Scheduled - The Update DB Schema attempt is scheduled • Executing - The Update DB Schema attempt is currently under way • Executed - The Update DB Schema attempt attempt has been completed
Message	A short message about the Update DB Schema attempt.
Detailed Message	A detailed message about the success or failure of the Update DB Schema attempt.

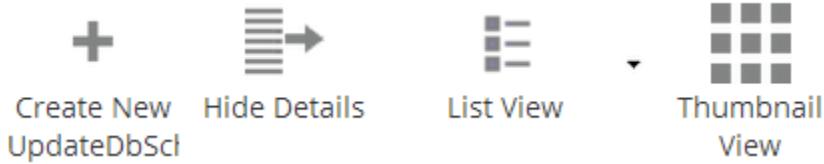
The Buttons Panel

When a DB Schema Update Object is selected



Button Name	Description	Notes
Open	Opens the selected Update DB Schema object in its own window for editing.	
Edit	Opens the Update DB Schema object edit window.	
Delete	Deletes the Update DB Schema object.	<p>The deleted Update DB Schema won't automatically be removed from the list until you have refreshed the Update DB Schema Query.</p> <p>The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.</p>

Refresh	Refreshes the tribefire system.	
Hide/Show Details	Hides or shows the properties panel.	
Add to clipboard	Adds the selected Update DB Schema object to tribefire GME's clipboard.	
List View	Allows you to select from three different list views.	
Thumbnail View	Allows you to display the Update DB Schema objects in the thumbnail view.	

When no DB Schema Update Object is selected

When no DB Schema Update object is selected, the panel displays four buttons: Create New UpdateDBSchema, Hide Details, List View and Thumbnail View. Apart from Create New Import Type, the other three buttons perform the same functionality as when a Model Import Object is selected.

Button Name	Description
Create New UpdateDBSchema	Opens up a new UpdateDBSchema display, allowing you to create a new UpdateDBSchema attempt

Files

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - When a raw resource is selected
 - Buttons when no raw resource is selected

Introduction

The Files sections shows you all the resources that have been uploaded to tribefire and are available in the Control Center. A file is uploaded and are known as RawResources in the tribefire environment.

Files is divided into four sections: Assembly Panel, Properties Panel, Search Panel and Button Panel.

The Assembly Panel

RawResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resolver URI	Width In Cm	Id	Info		
									Created	Creator	Name
MetaDataExtract-com.braintribe	1831693	0.00	7cd5803d5	application/xml	0	Ø	0.00	RN_1	12/20/2013 16:36	cortex	MetaDataExtract-com.l
SalesModel-1.0.zargo	56429	0.00	e2171d6ee	application/zip	0	Ø	0.00	RN_104	02/14/2014 14:01	cortex	SalesModel-1.0.zargo
BtTrainingModel-1.0.zargo	8388	0.00	6b11112b9	application/zip	0	Ø	0.00	RN_106	02/14/2014 14:02	cortex	BtTrainingModel-1.0.z
StevesModel-1.0.zargo	17476	0.00	634175a20	application/zip	0	Ø	0.00	RN_108	02/14/2014 14:34	cortex	StevesModel-1.0.zargo

Property	Description
RawResource	The name of the raw resource.
File Size	The size of the raw resource.
Height In Cm	Height of the raw resource(in Centimeters).
Md5	The Md5 encoding value. In tribefire this is referred to as the Context ID.
Mime Type	The raw resource's type (for example, Zip or XML).
Page Count	If Document, the amount of pages in the raw resource.
Resolver URI	The resolver URI of this raw resource, it is a string of characters used to identify a web resource.
Width in Cm	Width of the raw resource(in Centimeters).
ID	The internal, generated ID for this raw resource.
Created	When this raw resource was imported.
Creator	The name of the logged in user who imported this raw resource.
Name	The name of the imported raw resource.

The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

	<input style="width: 15px; height: 15px; border: none; margin-right: 5px;" type="text"/> Q ← 1 → Ordered by: ↑ ↓ Results per page: 25 Switch to Advanced
--	---

Advanced Search



1 Save As Switch to Basic

For more information on the Search Panel's functionality.

The Properties Panel

RawResource's Properties

<i>/d</i>	RN_106
File Size	8388
Height In Cm	0.00
Md5	6b11112b93b6b6703c08cf247b084333
Mime Type	application/zip
Page Count	0
Resolver URI	Ø
Width In Cm	0.00

– Resource Source
StaticSource (RS_106)

– Security Policy
Ø

– Tags
Ø

Info

Created	02/14/2014 14:02
Creator	cortex
Name	BtTrainingModel-1.0.zargo

Upload Undo Redo Save

RawResource's Properties

Property	Description
ID	The internal, generated ID for this raw resource.
File Size	The size of the raw resource.
Md5	The Md5 encoding value. In tribefire this is referred to as the Context ID.
Mime Type	The raw resource's type (for example, Zip or XML).
Page Count	If Document, the amount of pages in the raw resource.
Resolver URI	The resolver URI of this raw resource, it is a string of characters used to identify a web resource.
Width in Cm	Width of the raw resource(in Centimeters)
Resource Source	What type of resource is this. For example, a Static Source.
Security Policy	The security policy, if any, that is associated with this raw resource.
Tags	A tag, if any associated with this raw resource.

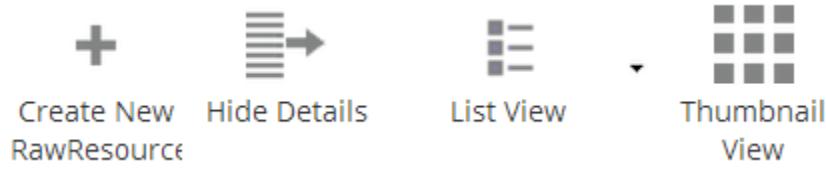
Info

Property	Description
Created	When this raw resource was imported.
Creator	The name of the logged in user who imported this raw resource.
Name	The name of the file this raw resource represents.

The Buttons Panel**When a raw resource is selected**

Button Name	Description	Notes
Open	Opens the selected raw resource in a new view for editing.	
Edit	Edits the selected raw resource.	
Delete	Deletes the selected raw resource.	<p>The deleted raw resource won't automatically be removed from the list until you have refreshed the Files Query.</p> <p>The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.</p>

Refresh	Refreshes the tribefire system.	
Hide / Show Details	Hides or shows the properties panel.	
Download Resource	Downloads the selected raw resource for viewing.	Depending on the type of file the raw resource represents, this file will either be displayed in a browser or downloaded to your browser's download directory.
Import Model From Zargo	Import the selected Zargo raw resource as a model. This model will then be available in the Custom Models or Distributed Models section, depending on what type of model has been generated	Only a valid Zargo model can be uploaded. Attempted to create a model from any other type of file will result in an error message.
Add to Clipboard	Adds the selected raw resource to tribefire GME's clipboard.	
List View	Allow you to select between different list views.	
Thumbnail View	Allow you to display all the raw resource as thumbnails.	

Buttons when no raw resource is selected

When no raw resource is selected, the panel displays four buttons: Create New RawResource, Hide Details, List View and Thumbnail View. Apart from Create New Import Type, the other three buttons perform the same functionality as when a RawResource is selected.

Button Name	Description	Notes
Create New RawResource	Displays the new raw resource window, allowing you to choose between a PdfResource or a RawResource	<p>Creating a new raw resource doesn't automatically upload and associate a file with this raw resource.</p> <p>Click for instructions on how to upload a file</p>

Uploading Files

Table of Contents

Expand / collapse

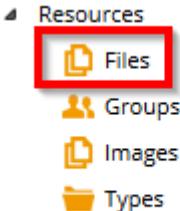
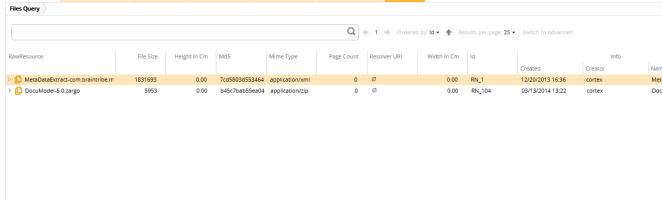
- Introduction
- Uploading a new File

Introduction

This guide will show you how to upload a file to tribefire Control Center.

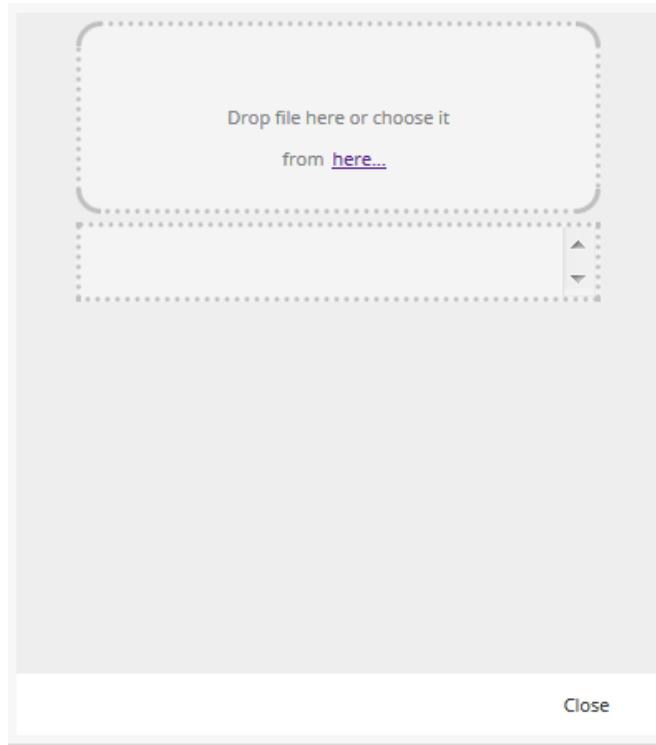
Although this guide uses the Files link, it is possible to upload a file from any located that has the properties panel displayed. Each instance of the property panel has the Upload button available.

Uploading a new File

Step #	Task
1	<p>Click the File link in the left-hand menu panel</p> 
2	<p>The File Query will be executed, showing all files in the tribefire system.</p> 
3	<p>Click the Upload button in the details panel.</p> 

4

The Upload New File window will appear. You can either drag the file you wish to upload on to it, or click on the [here...](#) link and browse to the file.



5

Once you have select the file it will be uploaded and, if successful, you will receive a success message.



Refresh the Files Query by clicking on the magnifying glass and you will see the newly uploaded file displayed.

File Query										
RawResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Resizer URL	Width In Cm	ID	Created	Creator
> MetaDataVerdict.com.braintribe.m	1831653	0.00	7cf58030553464	application/x-vml	0	Ø	0.00	RN_1	12/20/2013 16:36	corex
> MetaModel-0.0.xargo	5953	0.00	b45c7a0554a04	application/x-zip	0	Ø	0.00	RN_104	03/13/2014 13:22	corex
samplePDF.pdf	29309	0.00	a944933d5d70e9	application/pdf	0	Ø	0.00	RN_105	03/14/2014 09:43	corex

Downloading Files

Table of Contents

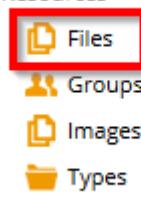
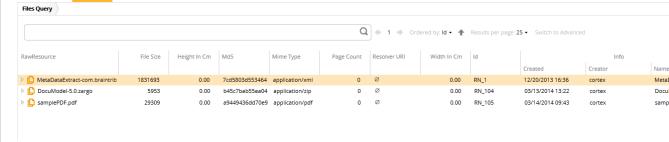
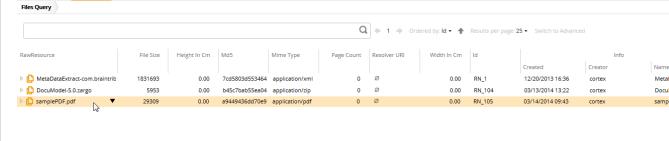
Expand / collapse

- Introduction
- Downloading Files

Introduction

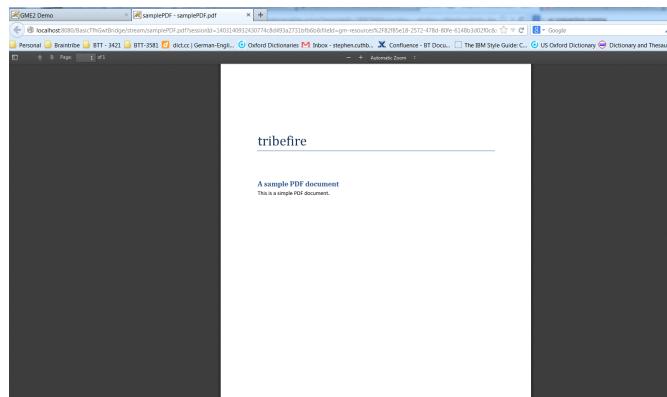
This guide will show you how to download a resource from tribefire Control Center to your computer.

Downloading Files

Step #	Task
1	<p>Click on the Files link from the left-hand menu.</p> 
2	<p>The Files Query will be executed, displaying all files in the tribefire system.</p> 
3	<p>Select the File you would like to download.</p> 
4	<p>Click the Download Resource button.</p> 

5

Depending on the type of resource you wish to download, it will either be displayed in a new tab in your browser or you will be asked to save it to your default downloads folder.



Property Groups

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - Buttons panel when property group is selected
 - Buttons when no property group is selected

Introduction

The groups section of tribefire GME allows you to create custom groups, which, by using the [appropriate metadata](#), allows you to group properties of an entity into customized sections, when displayed in the Properties Panel.

Once you have created a group, you can use it multiple times across multiple entities.

The Assembly Panel

PropertyGroup	Name	Id
▷ Account	Account	1
▷ Connection	Connection	2
▷ General	General	3
▷ Status	Status	6
▷ Timing	Timing	7
▷ Simulation	Simulation	8

Property	Description
Property Group	The property group object.
Name	The name of the property group.
Id	The internal, generated Id of this property group.

The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

Advanced Search

For more information on the Search Panel's functionality.

The Properties Panel

PropertyGroup's Properties

<i>Id</i>	1
Name	Account

- Display Info

PropertyDisplayInfo (1831)

PropertyGroup's Properties

Property	Description
Id	The internal, generated Id of the property group.
Name	The name of the property group.
DisplayInfo	The Display Info type this property group is associated with.

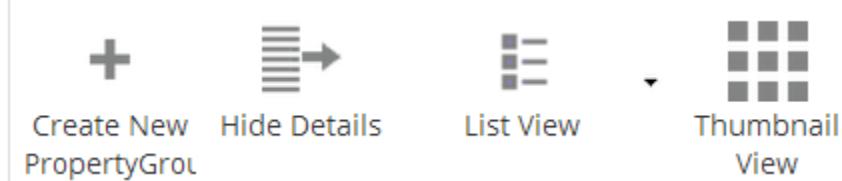
The Buttons Panel

Buttons panel when property group is selected



Button Name	Description	Notes
Open	Opens the selected property group in a new view for editing.	
Edit	Edits the selected property group.	
Delete	Deletes the selected property group.	The deleted property group won't automatically be removed from the list until you have refreshed the Group Query. The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.
Refresh		
Hide / Show Details	Hides or shows the properties panel.	
Add to Clipboard	Adds the selected property group to GME's clipboard.	
List View	Allow you to select between different list views.	

Thumbnail View	Allow you to display all the property group as thumbnails.	
----------------	--	--

Buttons when no property group is selected

When no property group is selected, the panel displays four buttons: Create New PropertyGroup, Hide Details, List View and Thumbnail View. Apart from Create New Import Type, the other three buttons perform the same functionality as when a PropertyGroup is selected.

Button Name	Description
Create New PropertyGroup	Displays the GIMA, allowing you to create a new property group

Images

Table of Contents

▼ Expand / collapse

- Introduction
- The Assembly Panel
- The Search Panel
- The Properties Panel
- The Buttons Panel
 - When an image resource is selected
 - Buttons when no image resource is selected

Introduction

This Images query displays all the images that tribefire Control Center has access to. They are known as RasterImageResources and can be used as the basis for icons in tribefire, for example.

The Assembly Panel

RasterImageResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Pixel Height	Pixel Width	Resolver URI	Width In Cm	Id	Created	Creator	Info
													Name
infoOrange16.png	1547	0.00	715eb1	image/png	1	16	16	Ø	0.00	RN_103	10/29/2013	cortex	infoOrange16.png
fileStackOrangeSr	1151	0.00	1a65a3	image/png	1	16	16	Ø	0.00	RN_34	10/23/2013	csp	fileStackOrangeSmall.j
folderOrange.png	1118	0.00	b89383	image/png	1	16	16	Ø	0.00	RN_35	10/23/2013	csp	folderOrange.png

Property	Description
RasterImageResource	The Image Resource.
File Size	The size of the image resource.
Height In Cm	The height of the image resource in centimeters.
Md5	The Md5 encode value. In tribefire, referred to as the content Id.
Mime Type	The type of image the image resource is.
Page Count	How many pages in this image resource.
Pixel Height	The height of the image resource in pixels.
Pixel Width	The width of the image resource in pixels.
Width in Cm	The width of the image resource in centimeters.
Id	The internal, generated Id of this image resource.
Created	When this image resource was created.
Creator	The name of the logged in user who created this image resource.
Name	The name of the file associated with this image resource.

The Search Panel

Above these models is a search panel. This allows you to filter and search for a specific model. There are two search modes available: Simple and Advanced. You can switch between them by selecting *Switch to Advanced* for the advanced search and *Switch to Basic* for the simple search.

Basic Search

🔍
Ordered by:
Id ↑
Results per page:
25
Switch to Advanced

Advanced Search

For more information on the Search Panel's functionality.

The Properties Panel

RasterImageResource's Properties	
<i>Id</i>	RN_103
File Size	1547
Height In Cm	0.00
Md5	715eb1a7f179d86ac1d54bdea0d35bb7
Mime Type	image/png
Page Count	1
Pixel Height	16
Pixel Width	16
Resolver URI	Ø
Width In Cm	0.00
– Resource Source	
StaticSource (RS_103)	
– Security Policy	
Ø	
– Tags	
Ø	
Info	
Created	10/29/2013 18:15
Creator	cortex
Name	infoOrange16.png
 Upload  Undo  Redo  Save	

RaswerImageResource's Properties

Property	Description
ID	The internal, generated ID for this image resource.
File Size	The size of the image resource.
Height In Cm	The height of the image resource in centimeters.
Md5	The Md5 encode value. In tribefire, referred to as the content Id.
Mime Type	The type of image the image resource is.
Page Count	How many pages in this image resource.
Pixel Height	The height of the image resource in pixels.
Pixel Width	The width of the image resource in pixels.
Width in Cm	The width of the image resource in centimeters.
Resource Source	The resource source type for this image resource.
Security Policy	A security policy, if any, associated with this image resource.
Tags	Tags, if any, associated with image resource.

Info

Property	Description
Created	The date this image resource was created.
Creator	The name of the logged in user who created this image resource.
Name	The name of the image associated with this image resource.

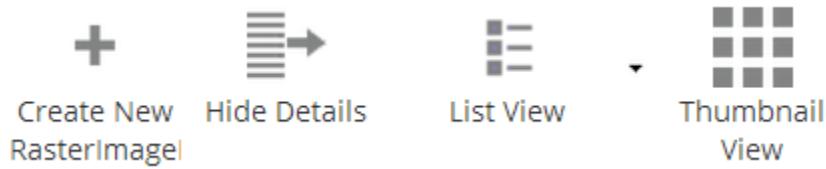
The Buttons Panel

When an image resource is selected



Button Name	Description	Notes
Open	Opens the selected image resource in a new view for editing.	
Edit	Edits the selected image resource.	

Delete	Deletes the selected image resource.	The deleted image resource won't automatically be removed from the list until you have refreshed the Images Query. The deletion won't be made permanent until you have saved changes. This is because any changes made to the system don't have an immediate effect on tribefire. Saving causes these changes to be persisted in tribefire, thus allowing for maximal usage of undo redo.
Refresh	Refreshes the tribefire system	
Hide / Show Details	Hides or shows the properties panel	
Download Resource	Downloads the selected image resource for viewing.	
Add to Clipboard	Adds the selected image resource to tribefire Control Center's clipboard.	
List View	Allow you to select between different list views.	
Thumbnail View	Allow you to display all the image resource as thumbnails.	

Buttons when no image resource is selected

When no image resource is selected, the panel displays four buttons: Create New RasterImageResource, Hide Details, List View and Thumbnail View. Apart from Create New Import Type, the other three buttons perform the same functionality as when a image resource is selected.

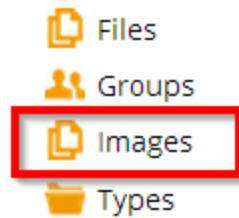
Button Name	Description	Notes
Create New RasterImageResource	Displays the new GIMA allowing you to create a new RasterImageResource.	Creating a new RasterImageResource doesn't automatically upload and associate a file with this raw resource. Click for instructions on how to upload an image.

Uploading a New Image

Uploading a New Image

Click on Images in the left-hand menu.

Resources



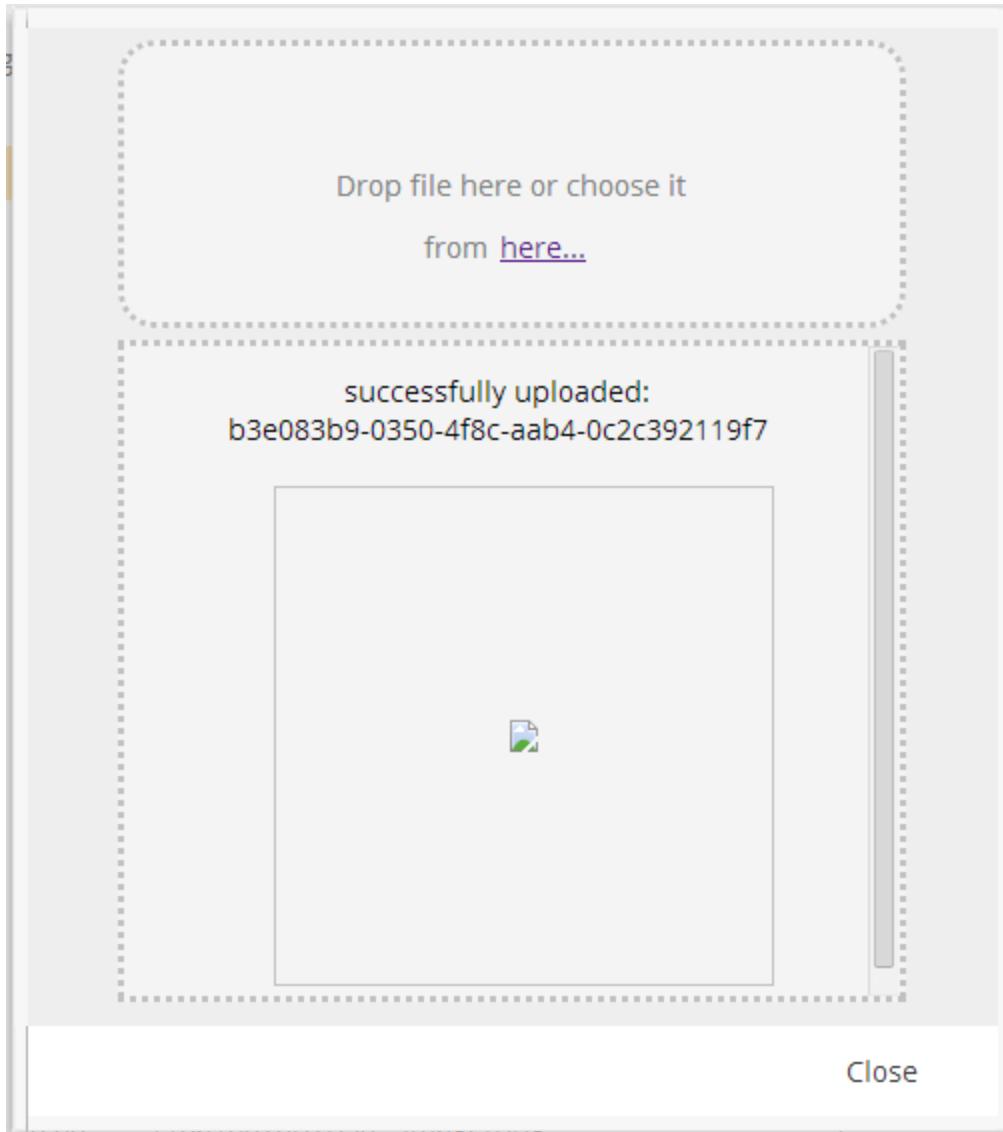
i Although in this documentation we will use the Image Query to upload a image, you can import a file from any of the Detail Panels displayed in tribefire. At the bottom of every Detail Panel are a series of buttons, including Upload. Clicking this button will display the Upload window, which you can use to upload a file, image or Zargo file.

This will display all the images that in tribefire Control Center.

RasterImageResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Pixel Height
infoOrange16.png	1547	0.00	715eb1a7f179d81	image/png	1	16
fileStackOrangeSmall.png	1151	0.00	1a65a322706863	image/png	1	16
folderOrange.png	1118	0.00	b89383fcab34d2	image/png	1	16
settingsOrange.png	1293	0.00	adcb334e0eb2e5	image/png	1	16
usersOrangeSmall.png	1384	0.00	4acfe57f3c26760	image/png	1	16
plugOrangeSmall.png	1494	0.00	601031fafaf74d82	image/png	1	16
...

In the Details Panel are four buttons: Upload, Undo, Redo and Save. Click the Save button to display the Upload Window.





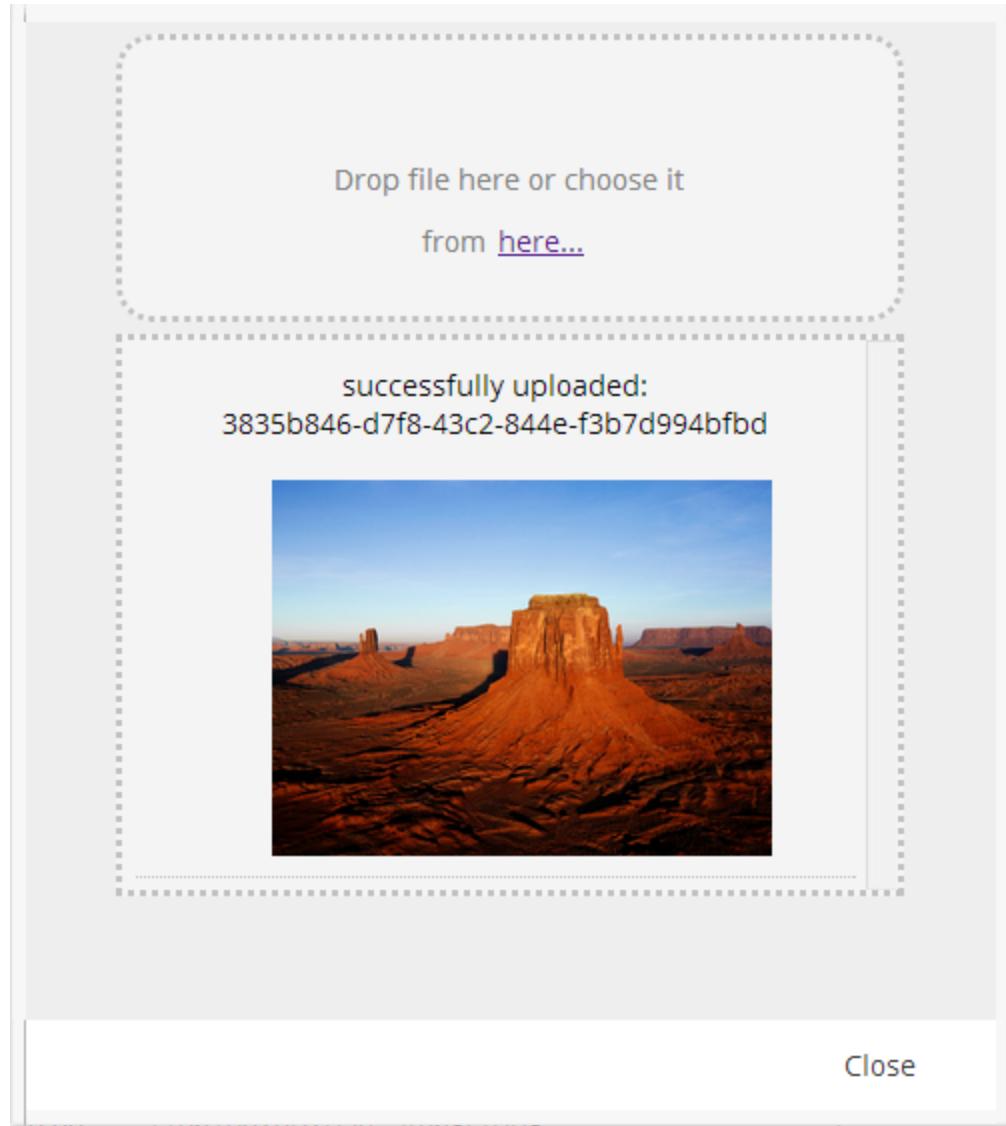
If any pictures have already been uploaded they will be displayed here also, along with their picture.

You have two options when uploading a new image; you can either drag and drop a file from somewhere in your directory onto the upload window or you can click the [here...](#) link and browse for the image you require. Select a image that you wish to upload.



The image used in this documentation is called SampleImage.jpeg

Once the picture has been uploaded, you will receive a success message and the picture, along with its content ID will be displayed.



Click close to return to Control Center. Clicking on the magnifying glass will refresh the search and display your newly uploaded image.

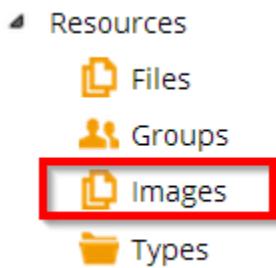
Images Query								
RasterImageResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Pixel Height	Pixel Width	
infoOrange16.png	1547	0.00	715eb1a7f179d81	image/png	1	16	16	
SampleImage.jpg	347852	0.00	bcbe6518774e1d	image/jpeg	1	854	1280	
fileStackOrangeSmall.png	1151	0.00	1a65a322706863	image/png	1	16	16	
folderOrange.png	1118	0.00	b89383fcab34d2	image/png	1	16	16	
settingsOrange.png	1293	0.00	adcb334e0eb2e5	image/png	1	16	16	

Viewing an Image

Viewing an Image

You can also view any image that is accessible in tribefire Control Center.

Click the Images link in the left-hand menu.



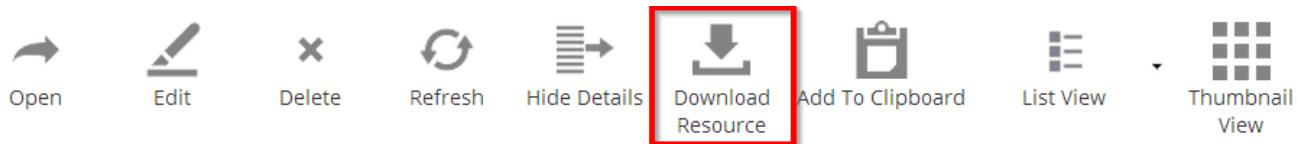
This will execute an Images Query and display all images available for this Access.

Images Query								
	File Size	Height In Cm	Md5	Mime Type	Page Count	Pixel Height	Pixel Width	
▶ infoOrange16.png	1547	0.00	715eb1a7f179d81	image/png	1	16	16	
▶ SampleImage.jpg	347852	0.00	bcb6518774e1d	image/jpeg	1	854	1280	
▶ fileStackOrangeSmall.png	1151	0.00	1a65a322706863	image/png	1	16	16	
▶ folderOrange.png	1118	0.00	b89383fcab34d2:	image/png	1	16	16	
▶ settingsOrange.png	1293	0.00	adcb334e0eb2e5	image/png	1	16	16	
▶ - - - - -	1000	0.00	1a65a322706863	image/png	1	16	16	

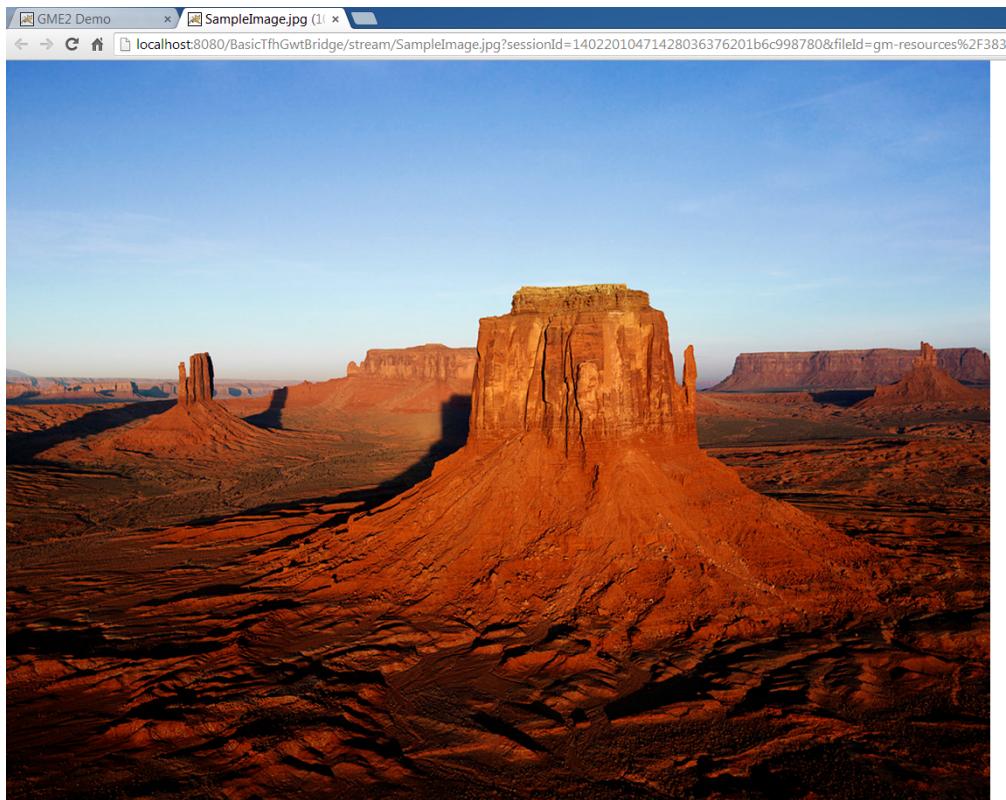
Select the image you would like to view.

RasterImageResource	File Size	Height In Cm	Md5	Mime Type	Page Count	Pixel Height	Pixel Width
▶ infoOrange16.png	1547	0.00	715eb1a7f179d81	image/png	1	16	16
▶ SampleImage.jpg	347852	0.00	bcb6518774e1d	image/jpeg	1	854	1280
▶ fileStackOrangeSmall.png	1151	0.00	1a65a322706863	image/png	1	16	16

At the bottom of the tribefire GME's display is the buttons panel. Click on the button Download Resource.



The image will then be displayed in a new tab, where you can view it or save it to a local directory.



tribefire GME Control Center - Cortex Workbench

Table of Contents

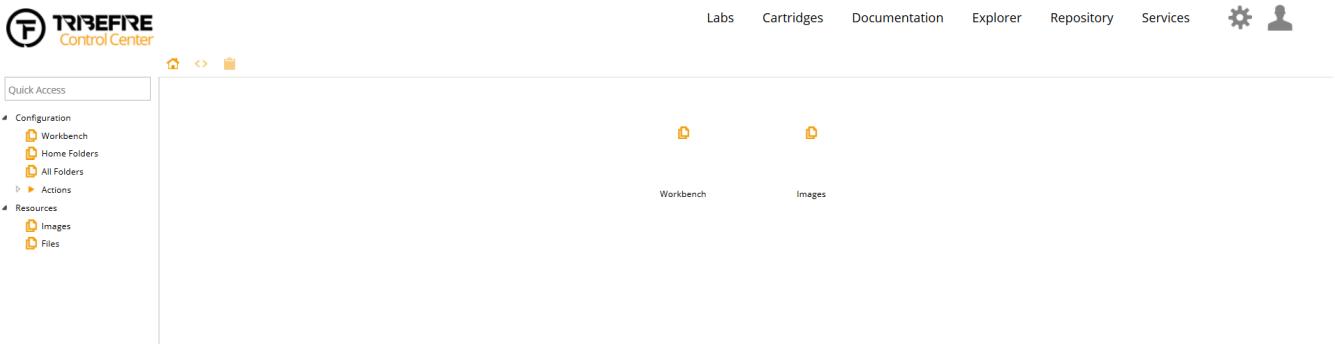
▼ Expand / collapse

- Cortex Workbench- Overview
 - The Central Panel
 - The Workbench Panel
 - Quick Access Search
 - Configuration
 - Resources
 - Cortex Workbench- User information/Access selection
 - Login Information
 - Access Selection menu

Cortex Workbench- Overview

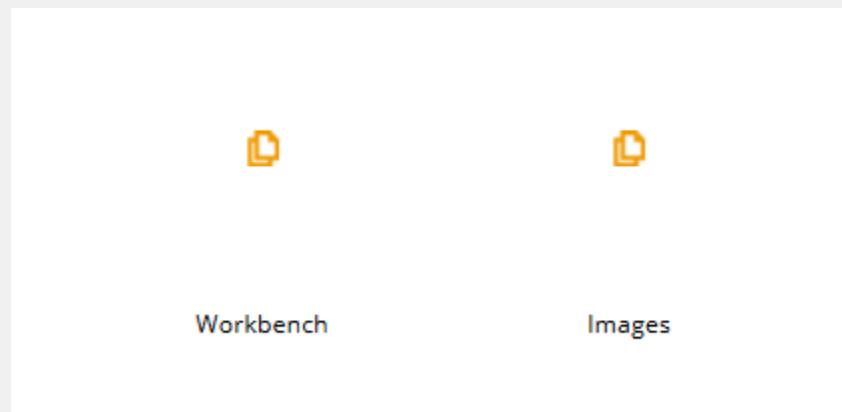
The Cortex Workbench allows you to alter the look and feel of tribefire Control Center. Along with being able to alter the links displayed in the Control Center's workbench panel, you can also create queries and custom actions.

The Cortex Workbench is divided into two sections: Configuration and Resources.



The Central Panel

The central panel consists of three main buttons: Workbench and Images. Workbench is used to configure the setup of the workbench panel in Control Center and is done through the use of folders and queries.



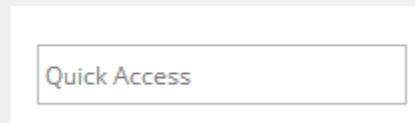
Link	Description
Workbench	This action will open up the workbench configuration hierarchy. Starting with the root folder, that is the top level, it defines the different folders that should be displayed on Control Center's workbench panel. This is done through the creation of folders and their content, generally a query of some sort.
Image	The image thumbnail will execute the image query and displays all images available for the Cortex Workbench.

The Workbench Panel

The panel to the left hand side of the screen is called the Workbench Panel, and displays all relevant links that are used in the Cortex Workbench to configure the Control Center. They are divided into two main sections—configuration and resources, with each link further subdivided.

Above the links is a search panel, labeled Quick Access. This can be used to quickly search for entity types and values associated with the Cortex Workbench.

Quick Access Search



Link	Description
Quick Access Search Field	Can be used to quickly search for types or values associated with the Cortex Workbench.

Configuration

The configuration section is divided into four sections—Workbench, Homes Folders, All Folders, and Actions—and are used to configure the look and feel of tribefire Control Center, as well as recording actions that will create functionality in tribefire.

Configuration

-  [Workbench](#)
-  [Home Folders](#)
-  [All Folders](#)
- ▶ [Actions](#)

Link	Description
Workbench	The main configuration link in Cortex Workbench, it is used to create folders that will appear in the Control Center's workbench. It starts with a root folder, which represents the top layer of the hierarchy, and has further child folders that are then displayed in Control Center. An example of this would be Smart Enterprise Information and System, both of which can then be further subdivided.
Home Folders	This is the folders which are displayed on the home screen in tribefire Control Center.
All Folders	This displays a list of all folders that have been created in the Cortex Workbench, making no distinction between whether they are workbench folders or home folders.
Actions	Actions are operations in tribefire that provide some functionality, an example of an action is the Import Model from Zargo operation. There are two types of actions listed, Global and Context Sensitive; A global action can take place anywhere, whereas a context sensitive action must only occur during a specific situation.

Resources

The resources section shows you all the different resources that are available in the Cortex Workbench, either as a file (RawResource) or as an image (RasterImageResource)

Resources

-  [Images](#)
-  [Files](#)

Link	Description
Images	This link provides you with all the images that have been uploaded, and which tribefire Control Center has access to.
Files	This link provides you with all the files that have been uploaded, and which the Cortex Workbench has access to.

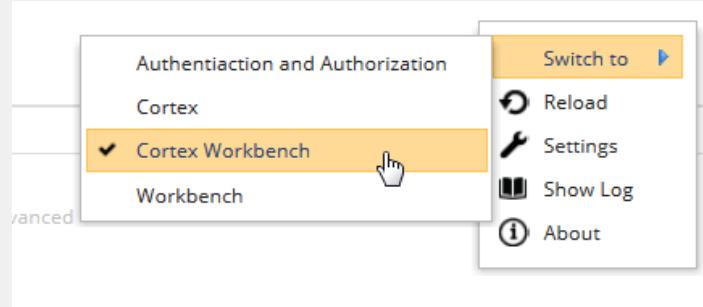
Cortex Workbench- User information/Access selection

Login Information

Above the center panel, in the top right-hand corner, is the login information. This displays the current user logged into tribefire Control Center's Cortex Workbench.



Access Selection menu



Clicking the cog icon will display the session settings menu. There are five options to choose from. The most important is the **Switch to** option. Selecting this will display a list of accesses that have been deployed to the tribefire system. By default there are four accesses that come as standard: Authentication and Authorization, Cortex, Cortex Workbench and Workbench.

- Authentication and Authorization - Controls the security setting for tribefire, including the ability to create roles, groups and users.
- Cortex - the access that represents the Control Center. This is the access displayed when you first log on
- Cortex Workbench - A workbench that lets you alter the look and feel of the Cortex access
- Workbench - A workbench that lets you alter the look and feel of the Cortex Workbench

In addition to **Switch to**, there are also four other options available

- Reload - Reloads the current access you are working on
- Settings - Shows the settings menu
- Show Log - Displays log information about the current session of tribefire Control Center
- About - Shows system information regarding the version of tribefire Control Center currently in use.

Hyperlink Action

Hyperlink Action

The Hyperlink Action allows you to define and create hyperlinks within your tribefire access. These actions function in the same way as other actions, except that they will link to a website (which can be a normal website, or, for example, a link to an app which your model is based on).

HyperlinkAction's Properties

displayName	∅
<i>id</i>	∅
target	∅
url	∅
- icon	
	∅
- inplaceContextCriterion	
	∅

HyperlinkAction Properties

Property	Description	Accepted Value
displayName	The name which will be displayed when this action is executed.	Any valid String
<i>id</i>	The automatically generated id given to this object when it is persisted.	NA
target	How this Hyperlink should be displayed, for example in a tab in tribefire or in a new window.	a valid target element inline - Opens link in tribefire GME without any details panel _new - Opens the link in a new tab
url	The url of that this Hyperlink action should open on execution.	A valid url
icon	An icon that is placed next to the hyperlink action.	A correctly configured icon object
inplaceContextCriterion	A defined context where this Hyperlink action should appear.	A correctly configured criterion object

Queries

Queries

There are two main elements that you can query, either an Entity or a Property. Both of these elements have their own query types and can be configured separately. Below the properties of each query type are explained.

Entity Query

EntityQuery's Properties

<i>Entity Id</i>	∅
<i>Entity Type Signature</i>	∅
– Froms	∅
– Ignore Privileged Roles	∅
– Ordering	∅
– queryContext	∅
– Restriction	∅
– Traversing Criterion	∅

Property	Description	Accepted Value
Entity ID	The automatically generated ID.	NA
Entity Type Signature	The Entity Type that will be queried.	A valid entity type name
Froms	Where the entities should be found.	A valid From object
Ignore Privileged Roles	Whether this query should ignore privileged roles or not.	
Ordering	How the returned results will be sorted.	A Valid Ordering Object: Simple Ordering Cascaded Ordering
queryContext	The queryContext associated with this Query.	A valid queryContext Object
Traversing Criterion	How the query will be traversed.	A valid traversing criterion

Property Query

PropertyQuery's Properties

Entity Id 11

Property Name \emptyset

– Entity Reference

\emptyset

– Ignore Privileged Roles

\emptyset

– Ordering

\emptyset

– Restriction

\emptyset

– Traversing Criterion

\emptyset

Property	Description	Accepted Value
Entity Id	The automatically generated ID for this property query	NA
Property Name	The Name of the property to be searched	A valid property
Entity Reference	A reference to a persistent entity	A valid persistentEntityReference object
Ignore Priviledge Roles	Whether this query should ignore privileged roles or not.	
Ordering	How the results of this query should be displayed	A valid ordering object
Restiction	A restriction on the query that refines the results	A valid restriction object
Traversing Criterion	How the query should traverse the data when finding the results	A valid traversing criterion

Ordering

Table of Contents

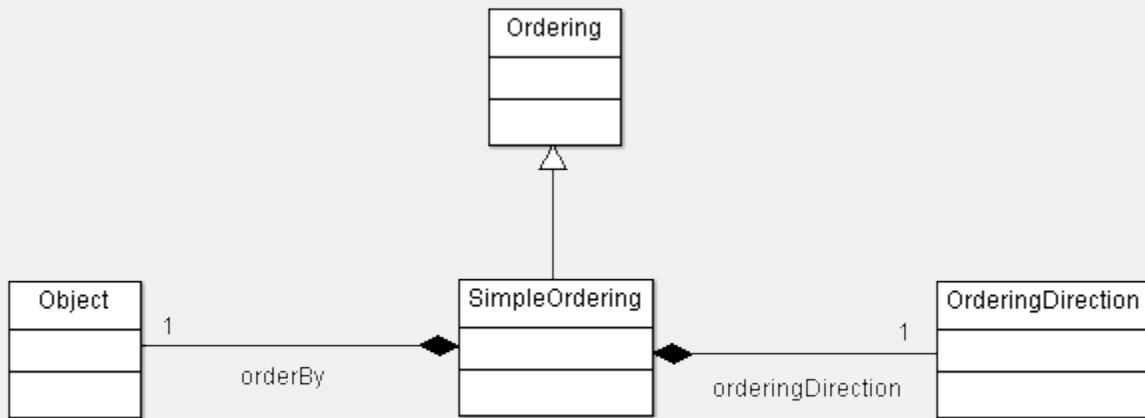
- ▼ Expand / collapse
 - Introduction
 - Simple Ordering
 - Example of Simple Ordering
 - Cascade Ordering
 - Example of Cascaded Ordering

Introduction

This page explains the system of ordering in tribefire GME. There are two types of ordering Simple Ordering and Cascade Ordering.

Simple Ordering

a uml modeling of the Simple Ordering object



A representation of the SimpleOrdering object in tribefire GME

SimpleOrdering ()

SimpleOrdering's Properties

Direction	∅
Order By	∅

← ↶ ↷ ✓ ✗
 Back Undo Redo Apply Cancel

Simple Ordering will order your search results on only one property. You can use either a string, a Property Operand or a Query Function to define what that field should be. You can also decide the ordering direction (i.e. ascending or descending) with which the ordering should be done.

Property	Description	Accepted Value
----------	-------------	----------------

(Ordering) Direction	The direction which the ordered value should take	Ascending: A-Z Descending: Z-A
Order By	On what property the ordering should take place.	String - The name of the property A valid Property Operand A valid Query Function.

Example of Simple Ordering

TemplateQueryAction

- TemplateQueryAction (1)
 - icon: Ø
 - inplaceContextCriterion: Ø
 - template: **Template**
 - metaData Ø
 - prototype: **EntityQuery (1)**
 - script: Ø

EntityQuery's Properties

Entity Id	1
Entity Type Signature	braintribe.model.Attendee
- Froms	Ø
- Ignore Privileged Roles	Ø
- Ordering	SimpleOrdering (3)
- queryContext	Ø
- Restriction	Restriction (13)
- Traversing Criterion	Ø

In this example we will use a TemplateQueryAction for the base object of our ordered query. See the TemplateQueryAction page for details on how to create a new Template Query Action.

As the name would suggest, the TemplateQueryAction uses a template on which this action is based. This template must have a EntityQuery object attached to it as a prototype. This EntityQuery is of the type Query and is used for defining the Query itself. The main property of interest here is the Ordering property of the EntityQuery. This relates to the diagram shown earlier.

After creating an `TemplateQueryAction`, a template for this action and then an `EntityType` on which it is based. You can now add the `Ordering` object.

EntityQuery's Properties

<code>Entity Id</code>	16
<code>Entity Type Signature</code>	Ø
- Froms	Ø
- Ignore Privileged Roles	Ø
- Ordering	Ø
- queryContext	Ø
- Restriction	Ø
- Traversing Criterion	Ø

Double Click on the `Ordering` property so that the create new ordering window appears..

The screenshot shows the `Ordering Query` interface. At the top, there are icons for Home, Filter, Compare, and a folder, followed by the title `Ordering Query`. Below the title is a search bar with the placeholder text `Type for filtering types and values...`. Under the search bar, there is a section titled `Values` containing three items: `CascadedOrdering (2) - CascadedOrdering`, `SimpleOrdering (3) - SimpleOrdering`, and `SimpleOrdering (4) - SimpleOrdering`. Below the values is a section titled `Types` containing three items: `CascadedOrdering`, `Ordering`, and `SimpleOrdering`. The `SimpleOrdering` item is highlighted with a yellow background and has a cursor icon pointing at it.

..select the `SimpleOrdering` object from types and a new `SimpleOrdering` type will be added to tribefire.



- Ordering

SimpleOrdering (17)

double click this new object to open it for editing:

SimpleOrdering	Direction	Entity Id
SimpleOrdering (17)	Ø	17

SimpleOrdering's Properties	
Entity Id	17
Direction	Ø
Order By	Ø

There are only two properties that we must configure so your queries are correctly ordered: Direction and Order By. You can use either a String, a Property Operand or a Query Function.

In this example we will add a String called secondName. This property is attached to the model with the type signature braintribe.model.Attendee. So the query will order all attendees by their second names.

SimpleOrdering	Direction	Entity Id
SimpleOrdering (17)	ascending	17

SimpleOrdering's Properties	
Entity Id	17
Direction	ascending
Order By	secondName

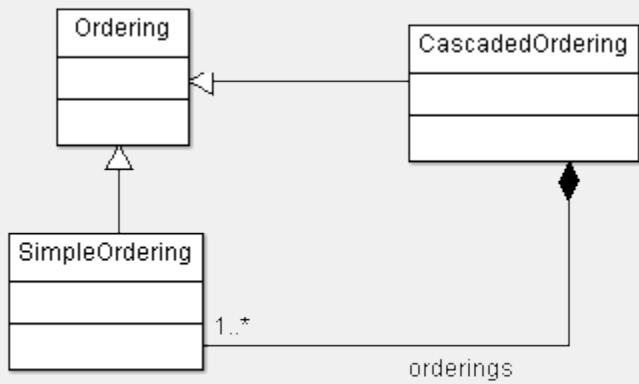
Click Save to persist your changes. Your search will now be ordered according to the SimpleOrdering object you defined.

Attendee	emailAddress	firstName	id	jobTitle	secondName
Attendee (5)	maryAnn@email.com	Mary	5	Support	Ann
Attendee (10)	LucyCooper@msn.com	Lucy	10	Developer	Cooper
Attendee (12)	OliviaFog@Imanc.com	Olivia	12	Technical Writer	Fog
Attendee (6)	ThomasHardy@er...	Thomas	6	Technical Writer	Hardy
Attendee (11)	DavidSilva@msn.com	David	11	Support	Silva
Attendee (2)	johnsmith@email.com	John	2	Programmer	Smith
Attendee (14)	JohnSmithson@er...	John	14	Technical Writer	Smithson
Attendee (7)	TonyStark@Iman...	Tony	7	Engineer	Stark
Attendee (15)	Zachary@email.com	Ben	15	Technical Writer	Zachary

Cascade Ordering

Cascade ordering allows you to define a series of simple orderings which will order a series of columns, according to the order of simple Ordering you have configured.

an UML representation of the Cascading Ordering object



CascadedOrdering object as represented in tribefire GME

CascadedOrdering's Properties

<i>Entity Id</i>	∅
- Orderings	∅

The CascadedOrdering object is a rather simple one. It has only one editable property: Orderings. This is associated with the type SimpleOrdering, allowing you to add as many SimpleOrdering instances to the CascadedOrdering object.

Property	Description	Accepted Value
Entity Id	The automatic, internally generated Id for this instance of CascadedOrdering.	NA
Orderings	A list of SimpleOrdering, the order of which determines the order of the columns to be sorted.	Any number of valid SimpleOrderings

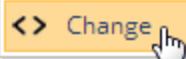
Example of CascadedOrdering

After creating an TempalteQueryAction, a template for this action and then an EntityType on which it is based. You can now add the Ordering object.

Hover over the Ordering property till the context-menu icon appears, click it and then Change to display the new ordering window:

EntityQuery's Properties

<i>EntityId</i>	24
<i>Entity Type Signature</i>	∅
– Froms	∅
– Ignore Privileged Roles	∅
– Ordering	∅
– queryContext	∅
– Restriction	∅
– Traversing Criterion	∅

 Change

Changes the given property value

Select the CascadeOrdering from the types section



Type for filtering types and values...

Values

- [SimpleOrdering \(17\) - SimpleOrdering](#)
- [SimpleOrdering \(19\) - SimpleOrdering](#)
- [SimpleOrdering \(20\) - SimpleOrdering](#)
- [SimpleOrdering \(3\) - SimpleOrdering](#)
- [SimpleOrdering \(4\) - SimpleOrdering](#)

Types

- [CascadedOrdering](#)
- [Ordering](#)
- [SimpleOrdering](#)

A new cascading property will be created. Double click on it to open it for editing.

– Ordering

[CascadedOrdering \(25\)](#)

– queryContext

Ordering

Then hover over the Orderings property so that the context-menu icon appears, click it and then select open.

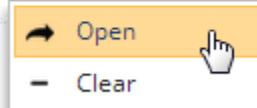
CascadedOrdering's Properties

EntityId

25

– Orderings

∅



click the add button to add new a list of new simple ordering objects. You can either create new ones or use previously configured ones:

A screenshot of a software interface showing a toolbar with icons for home, search, and file operations. Below the toolbar, a tab labeled "SimpleOrdering Query" is selected. A large text input field is present below the tabs.

Values[SimpleOrdering \(17\) - SimpleOrdering](#)[SimpleOrdering \(19\) - SimpleOrdering](#)[SimpleOrdering \(26\) - SimpleOrdering](#)[SimpleOrdering \(27\) - SimpleOrdering](#)[SimpleOrdering \(4\) - SimpleOrdering](#)**Types**[SimpleOrdering](#)

Select the objects you would like and then click Finish. Navigate back to the CascadedOrdering property and you should see a list of SimpleOrdering objects that have been added to this property.

When you execute your query it will now order your query results according to the different SimpleOrdering objects you have selected. In the example below, the first SimpleOrdering object orders the jobTitle field and then the second orders the secondName field.

Attendee	emailAddress	firstName	id	jobTitle	secondName
Attendee (10)	LucyCooper@msn.com	Lucy	10	Developer	Cooper
Attendee (7)	TonyStark@iman.com	Tony	7	Engineer	Stark
Attendee (2)	johnsmith@email.com	John	2	Programmer	Smith
Attendee (5)	maryAnn@email.com	Mary	5	Support	Ann
Attendee (11)	DavidSilva@msn.com	David	11	Support	Silva
Attendee (12)	OliviaFog@iman.com	Olivia	12	Technical Writer	Fog
Attendee (6)	ThomasHardy@email.com	Thomas	6	Technical Writer	Hardy
Attendee (14)	JohnSmithson@er.com	John	14	Technical Writer	Smithson
Attendee (15)	Zachary@email.com	Ben	15	Technical Writer	Zachary

Restrictions

Table of Contents

- ▼ Expand / collapse
 - Introduction

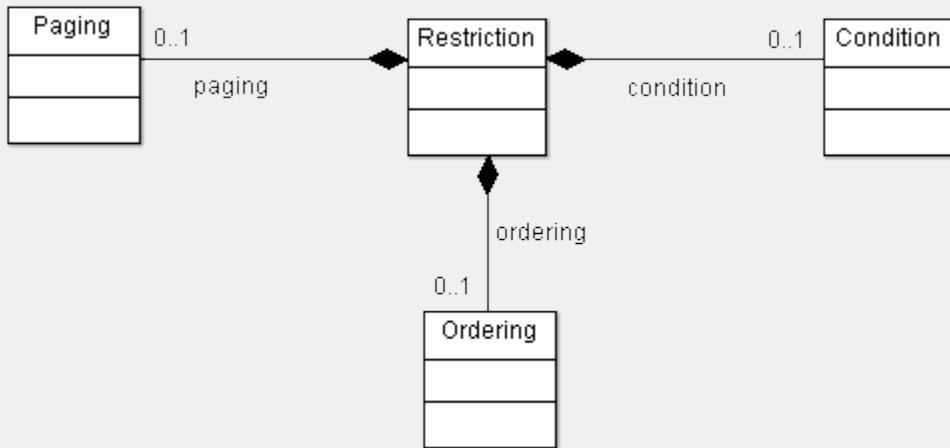
Child pages

- ▼ Expand / collapse
 - Paging
 - Conditions

Introduction

A simple query action, or indeed any query, will return all the results that are found for the entity type signature defined. If we want to manipulate the query so that it returns only specific objects we must use the **Restriction** object.

*a UML representation of the **Restriction** object.*



The restriction object as represented by tribefire.

Restriction's Properties

<i>Entity Id</i>	97
------------------	----

– Condition

∅

– Paging

∅

The restriction object is part of a query object (either an Entity Query, a Property Query or a Select Query.) along with its other parameters, such as **ordering** or traversing criteria and has two properties which can be defined. Both of these properties are in themselves objects.

Property	Description	Accepted Value
----------	-------------	----------------

Condition	This is an object which defines how the query will be restricted	Any valid Condition objects. Some examples are Negation , Abstract Junctions and Value Comparisons .
Paging	The paging property defines how many and from which index the results will be displayed	This accepts only a validly defined Paging Object .

Paging

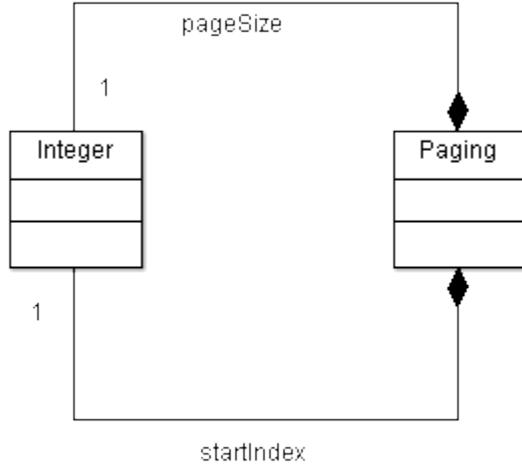
Table of Contents

- ▼ Expand / collapse
 - Introduction
 - Example of Usage

Introduction

The Paging Object is used to define the paging properties of the Restriction object. This object defines how the results of a query should be displayed in tribefire. Its purpose is to restrict the number of results displayed, which is very useful when working with a large data set, or if your working environment is not terribly efficient.

a UML representation of paging object.



the paging object as represented in tribefire.

Paging's Properties

<i>Entity Id</i>	100
Page Size	0
Start Index	0

Property	Description	Accepted Value
Page Size	This defines how many results should be returned at the one time.	Any valid integer. The integer given relates to the number of results returned
Start Index	Every object saved in tribefire is given its own index. The property defines at which index the results should start.	Any valid integer. The integer given will be the index of the first result returned.

Example of Usage

Paging's Properties

Entity Id	64
Page Size	5
Start Index	5

The two properties of the Paging Object have been defined with the integer 5 in both property fields. This means that when the query that this Paging object belongs to is executed, there will be only five results returned and the first results will be of the index 5.

executed query with Paging property set.

Attendee	Email Address	First Name	Job Title	Second Name	Id
▶ Attendee (32)	PeterKramer@en...	Peter	Web-Developer	Kramer	32
▶ Attendee (11)	DavidSilva@msn...	David	Support	Silva	11
▶ Attendee (20)	VictoriaRussell@...	Victoria	Technical Writer	Russell	20
▶ Attendee (22)	PaulCarrol@cpo...	Paul	Technical Writer	Carrol	22
▶ Attendee (6)	ThomasHardy@e...	Thomas	Technical Writer	Hardy	6

The screenshot above shows the results returned with the Paging Object defined. The query has returned five results (as defined by the Page Size property) and begins with the first index of five (as defined by the Start Index property). Because there was no entry with the Index of 5, the next available index is the one the query returned (with the index of 6).

Conditions

Table of Contents

▼ Expand / collapse

- Introduction

Child pages

▼ Expand / collapse

- Abstract Junction
- Negation
- Value Comparison
- Full Text Comparison

Introduction

The Conditions property of the Restriction Object can be defined from a variety of different conditions. There is no Condition Object, rather this property is defined by selecting an object that belongs to the Condition collection.

Restriction's Properties

Entity Id

13

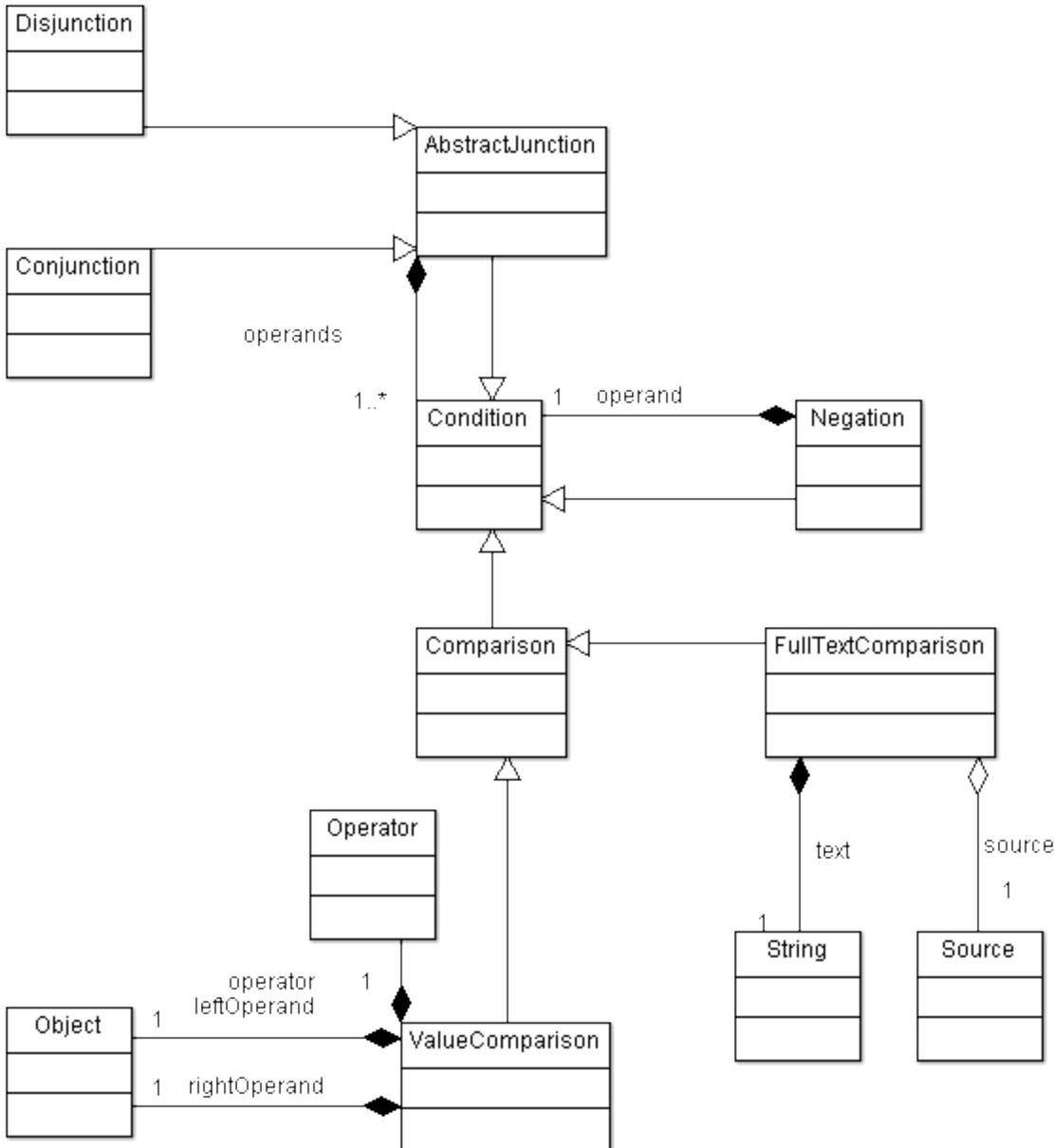
– Condition

∅

– Paging

∅

a UML representation of the Condition Object and some of the main Objects that it can be defined with.



the different conditions available in tribefire GME.

Types

AbstractJunction

Comparison

Condition

Conjunction

Disjunction

EmptySet

FullPermutation

FulltextComparison

NegatedValueComparison

Negation

SequentiallyReducedConjunction

ValueComparison



The two objects (AbstractJunction and Condition) that are displayed in italics are abstract, meaning that they can't be used themselves.
Selecting either of these will show all Objects that are of this type.

Abstract Junction**Table of Contents**

- ▼ Expand / collapse
 - Introduction

Child pages

- ▼ Expand / collapse
 - Conjunction
 - Disjunction

Introduction

AbstractJunction is an abstract object which is used to define two types of Junctions. They are Conjunction and Disjunction. A junction is used to combine two different operands so that you can build queries which restrict the results based on more than one condition. The conjunction object is the equivalent of using the operator AND in a SQL query, whereas Disjunction is the equivalent of the OR operator.

**Types of AbstractJunction**

- Conjunction



- Disjunction



Conjunction

Table of Contents

- ▼ Expand / collapse
 - Introduction
 - Adding a New Conjunction
 - Example of Usage
 - Example Results

Introduction

The Conjunction Object is of the type AbstractJunction. This object functions as an AND operator, allowing you to define several different operators, all of which must be true for the results to be returned.



The Conjunction Object has only one property that can be defined, Operands. You can use this property to add a list of Operands to a query.

Conjunction's Properties

Entity Id 106

- Operands

∅

You can add as many Operands as you like. The query will test each of these conditions against a dataset and only returns objects that match all of the operands defined.

Adding a New Conjunction

After adding a new Restriction, you can add the Conjunction to the Condition property. To do this mouse hover over the condition property until the context-menu icon shows.

Restriction's Properties

Entity Id 105

- Condition



∅

- Paging

∅

Click on it and then select the Change option. This will display a list of Condition that you can add to the restriction property. Select the Conjunction entry from the Types section to create a new Conjunction, or using the search panel for a previously defined Conjunction. Remember, if you select a existing Conjunction and then change its properties, all other Queries that use this object will also be affected.

The screenshot shows the GME (Graphical Model Editor) interface. At the top, there is a toolbar with icons for Home, Filter, Compare, Copy, Paste, and a search bar labeled "Condition Query". Below the toolbar is a search input field containing the placeholder text "Type for filtering types and values...".

The main content area is divided into sections:

- Values**: A list of items:
 - NegatedValueComparison (74) - NegatedValueComparison
 - Negation (43) - Negation
 - ValueComparison (50) - ValueComparison
 - ValueComparison (79) - ValueComparison
 - ValueComparison (83) - ValueComparison
- Types**: A list of items:
 - AbstractJunction*
 - Comparison
 - Condition*
 - Conjunction** (highlighted with a yellow background and a cursor icon)
 - Disjunction
 - EmptySet
 - FullPermutation
 - FulltextComparison
 - NegatedValueComparison
 - Negation
 - SequentiallyReducedConjunction
 - ValueComparison

You Conjunction will then be added to the Restriction's properties. If it is a new instance, you will have to save it by clicking the Save button for it to be persisted to tribefire GME and also to receive an ID.

Restriction's Properties

<i>Entity Id</i>	105
– Condition	
Conjunction ()	
– Paging	
∅	

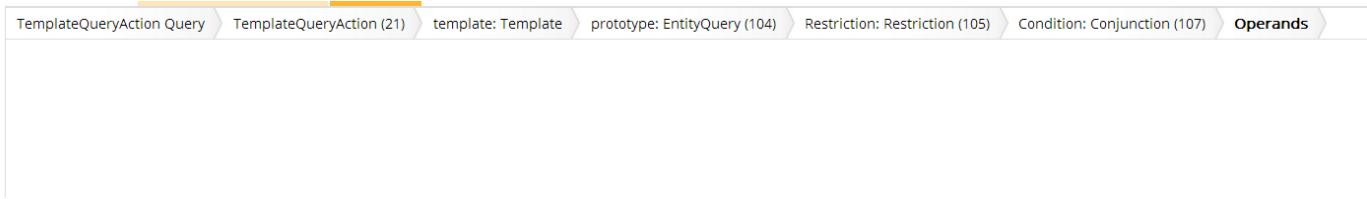
Double Click on the Conjunction to add new Operands.

Using the mouse, hover over the operands property until the context-menu icon appears and then select open.

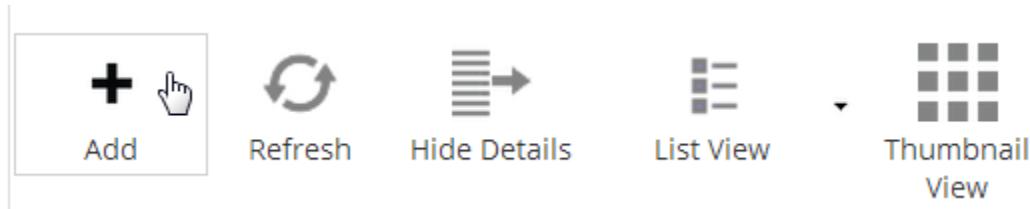
Conjunction's Properties

<i>Entity Id</i>	49
– Operands	
1. ValueComparison (50) 2. ValueComparison (52)	
 Navigates to the selected property	

The Operands Window will appear



at the bottom there is the buttons panel. Click the add button to display the create new operands window.



This allows you to select from a list the different types of Operands that this query will use as part of its conjunction.

Condition Query

Values

- NegatedValueComparison (74) - NegatedValueComparison
- Negation (43) - Negation
- SequentiallyReducedConjunction (78) - SequentiallyReducedConjunction
- ValueComparison (52) - ValueComparison
- ValueComparison (83) - ValueComparison

Types

- AbstractJunction*
- Comparison
- Condition*
- Conjunction
- Disjunction
- EmptySet
- FullPermutation
- FulltextComparison
- NegatedValueComparison
- Negation
- SequentiallyReducedConjunction
- ValueComparison

Select the type you would like and the click finish. It will now be displayed in Operands window. You can repeat this step for as many Operands as you wish.

TemplateQueryAction Query TemplateQueryAction (21) template: Template prototype: EntityQuery (104) Restriction: Restriction (105) Condition: Conjunction (107) **Operands**

▷ ValueComparison (108)

Example of Usage

Conjunction's Properties

Entity Id 49

- Operands

1. ValueComparison (50)
2. ValueComparison (52)

The screenshot above shows an example of a Conjunction object with two Operands of the type ValueComparison defined. If we were to open the two Value Comparison properties they would show the two values this query should match for results to be returned

ValueComparison's Properties

<i>Entity Id</i>	50
Operator	like
Right Operand	Technical Writer
- leftOperand	
PropertyOperand (51)	

Description - Value Comparison 50

This ValueComparision uses a PropertyOperand which defines the property field as jobTitle. That is, on which property should this query compare a value. The right-hand side is of the value *Technical Writer*. This means that the first validation is to return all entries who have the jobTitle of Technical Writer.

When written in SQL this Value Comparison would look like this

```
WHERE jobTitle = 'Technical Writer'
```

ValueComparison's Properties

<i>Entity Id</i>	52
Operator	like
Right Operand	*cpo*
- leftOperand	
PropertyOperand (53)	

Value Comparison 52

The second value comparison also has a PropertyOperand. This property operand defines the property field as emailAdress. The right side of the operand uses two wildcard characters (*) to search for the words cpo in the email address. This will find all occurrences of cpo in an email address.

Written in SQL this Value Comparison would like like:

```
WHERE emailAddress = '%cpo%'
```

The example above will filter all results, returning only those who have the job of Technical Writer and the value cpo is contained in their email address

Example Results

Here is a Simple Action Query showing all the results without any restrictions.

Attendee	Email Address	First Name	Job Title	Second Name	Id
▷ Attendee (2)	johnsmith@email.com	John	Programmer	Smith	2
▷ Attendee (5)	maryAnn@email.com	Mary	Support	Ann	5
▷ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy	6
▷ Attendee (7)	TonyStark@lman.com	Tony	Engineer	Stark	7
▷ Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper	10
▷ Attendee (11)	DavidSilva@msn.com	David	Support	Silva	11
▷ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog	12
▷ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson	14
▷ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary	15
▷ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth	16
▷ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream	17
▷ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass	18
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell	20
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman	21
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol	22

When this Query has been properly configured it will return the following:

Attendee	Email Address	First Name	Job Title	Second Name	Id
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell	20
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol	22
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman	21

Disjunction**Table of Contents**

▼ Expand / collapse

- Introduction
- Adding a New Disjunction
- Example of Usage
 - Example Results

Introduction

The Disjunction Object is of the type AbstractJunction. This object functions as an OR operator, allowing you to define several different operators, any one of which must be true for the results to be returned.



Disjunction Objects have only one property which can be defined: Operands. This is used to build up a list of operands which will be used to filter results when executing the query.

Disjunction's Properties*Entity Id*

109

– Operands

∅

You can add as many Operands as you like. The query will test each of these conditions against a dataset and return any objects that match one of the operands defined.

[Adding a New Disjunction](#)

After adding a new Restriction, you can then add the Disjunction to the condition property. To do this mouse hover over the condition property until the context-menu icon shows.

Restriction's Properties*Entity Id*

105

– Condition



∅

– Paging

∅

Click on the icon and then click Change from the resulting menu. This will display the new Condition window.

The screenshot shows a user interface for a 'Condition Query'. At the top, there are several icons: a house, a magnifying glass, a double arrow, a folder, and a document. To the right of these is a yellow button labeled 'Condition Query'. Below this is a search bar containing the placeholder text 'Type for filtering types and values...'. Underneath the search bar, the word 'Values' is displayed in orange. A list of query types follows:

- NegatedValueComparison (74) - NegatedValueComparison
- Negation (43) - Negation
- SequentiallyReducedConjunction (78) - SequentiallyReducedConjunction
- ValueComparison (52) - ValueComparison
- ValueComparison (83) - ValueComparison

Below this, the word 'Types' is displayed in orange. A list of type names follows:

- AbstractJunction*
- Comparison
- Condition*
- Conjunction
- Disjunction**
- EmptySet
- FullPermutation
- FulltextComparison
- NegatedValueComparison
- Negation
- SequentiallyReducedConjunction
- ValueComparison

Select the Disjunction entry to create a new Disjunction property or use the search bar to filter for a preexisting Disjunction property. Remember that if you select a preexisting Disjunction and then change it, all other queries using this Disjunction will also be affected.

The selected Disjunction will be added to the condition property. If you have created a new Disjunction you will have to click the save button, located at the bottom of the properties panel, to persist it in tribefire. During saving, this new Disjunction will also receive an ID.

Restriction's Properties

<i>Entity Id</i>	105
– Condition	
Disjunction ()	
– Paging	
∅	

Double Click on the Disjunction object to add new Operands.

Using the mouse, hover over the operands property until the context-menu icon appears and then select open.

Disjunction's Properties

<i>Entity Id</i>	110
– Operands	
∅	

A context menu is displayed with the following options:

- Open (highlighted with a yellow background)
- Clear

This will open the Operands object. You can add new Operands by clicking the Add button located at the bottom of the screen. This will display the Selection Constellation, which allows you to choose from a list of different operands.

Values

[NegatedValueComparison \(74\)](#) - NegatedValueComparison

[Negation \(43\)](#) - Negation

[SequentiallyReducedConjunction \(78\)](#) - SequentiallyReducedConjunction

[ValueComparison \(52\)](#) - ValueComparison

[ValueComparison \(83\)](#) - ValueComparison

Types

[*AbstractJunction*](#)

[Comparison](#)

[*Condition*](#)

[Conjunction](#)

[Disjunction](#)

[EmptySet](#)

[FullPermutation](#)

[FulltextComparison](#)

[NegatedValueComparison](#)

[Negation](#)

[SequentiallyReducedConjunction](#)

[ValueComparison](#)

Select a new Operand and click finish. It will then be added to the Operands object.

ValueComparison (111)

You can repeat this step as often as you wish. Each new operand will be listed in the Operands property.
Example of Usage

Disjunction's Properties

Entity Id 56

– Operands

1. ValueComparison (57)
2. ValueComparison (60)

The screenshot above shows an example of a Disjunction object with two Operands of the type [ValueComparison](#) defined. When executing this query, the dataset will be validated against this two properties. If it matches either one of them, the result will be returned.

ValueComparison's Properties

<i>Entity Id</i>	57
Operator	like
Right Operand	Technical Writer
– leftOperand	
PropertyOperand (58)	

Description - ValueComparison 57

This ValueComparison uses a Property Operand which defines the property field as jobTitle. That is, on which property should this query compare a value. The right-hand side of this comparison is a String value *Technical Writer*. This means that if the query finds any data with the value Technical Writer in the jobTitle field this result will be returned.

When written in SQL this Value Comparison would look like this

```
Where jobTitle = 'Technical Writer'
```

ValueComparison's Properties

<i>Entity Id</i>	60
Operator	like
Right Operand	Document Writer
– leftOperand	
PropertyOperand (61)	

Description - ValueComparison 60

This ValueComparison uses a Property Operand which defines the property field as jobTitle. That is, on which property should this query compare a value. The right-hand side of this comparison is a String value *Document Writer*. This means that if the query finds any data with the value Document Writer in the jobTitle field this result will be returned.

When written in SQL this Value Comparison would look like this

```
Where jobTitle = 'Document Writer'
```

The outcome of this Query will be all entries which have either the job title Technical Writer or Document Writer

Example Results

Here is a Simple Action Query showing all the results without any restrictions.

Attendee	Email Address	First Name	Job Title	Second Name		Id
▷ Attendee (2)	johnsmith@email.com	John	Programmer	Smith		2
▷ Attendee (5)	maryAnn@email.com	Mary	Support	Ann		5
▷ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy		6
▷ Attendee (7)	TonyStark@lman.com	Tony	Engineer	Stark		7
▷ Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper		10
▷ Attendee (11)	DavidSilva@msn.com	David	Support	Silva		11
▷ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog		12
▷ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson		14
▷ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary		15
▷ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth		16
▷ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream		17
▷ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass		18
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell		20
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman		21
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol		22

When this Disjunction Query has been properly configured it will return the following:

Attendee	Email Address	First Name	Job Title	Second Name		Id
▷ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass		18
▷ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth		16
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell		20
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol		22
▷ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy		6
▷ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream		17
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman		21
▷ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog		12
▷ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson		14
▷ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary		15

Negation

Table of Contents

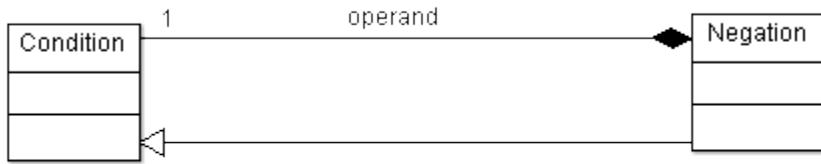
Expand / collapse

- Introduction
- Adding Negation Object
- Usage Example
 - Example Results

Introduction

The Negation object is a simple object which will negate the operand assigned to it. That means that if the operand evaluates the results to true, the negation will return false, and vice versa. So if you have a query that has an operand to return all results matching a specific value, the negation object will carry out the same search but will ignore all entries with this value and return the rest.

The Negation Object has only one property which can be defined, Operand. It is attached to the Conditions property of Restriction.
a UML representation of the Negation Object.



The Negation object as represented in tribefire.

Negation's Properties

Entity Id

112

- Operand

∅

To configure the Negation object, we add the Negation object to the Condition property of Restriction and then add an Operand that should be negated.

Adding Negation Object

At the Condition property of a Restriction object, mouse hover so that the context-menu is shown.

Restriction's Properties

Entity Id

105

- Condition



Ø

- Paging

Ø

Click on the Icon and then Click Change to display the add new Condition window.

The screenshot shows a software interface titled "Condition Query". At the top, there is a toolbar with icons for Home, Filter, Compare, and Copy/Paste, followed by a search bar containing the placeholder text "Type for filtering types and values...". Below the search bar, there are two main sections: "Values" and "Types".

Values

- NegatedValueComparison (74) - NegatedValueComparison
- Negation (112) - Negation
- Negation (43) - Negation
- SequentiallyReducedConjunction (78) - SequentiallyReducedConjunction
- ValueComparison (83) - ValueComparison

Types

- AbstractJunction*
- Comparison
- Condition*
- Conjunction
- Disjunction
- EmptySet
- FullPermutation
- FulltextComparison
- NegatedValueComparison
- Negation**
- SequentiallyReducedConjunction
- ValueComparison

Select the Negation entity and click Finish. The Negation object will be added to the Condition property of the Restriction object.

Restriction's Properties

<i>Entity Id</i>	105
- Condition	
Negation (113)	
- Paging	
Ø	

Click on the Negation entity to open its properties.

The screenshot shows a navigation path with the following steps:

- TemplateQueryAction Query
- TemplateQueryAction (21)
- template: Template
- prototype: EntityQuery (104)
- Restriction: Restriction (105)
- Condition: Negation (113)**

On the right side, there is a panel titled "Negation's Properties" containing the following information:

<i>Entity Id</i>	113
- Operand	Ø

To add a new Operand, move hover over the Operand property until the context-menu icon appears. Click on the icon to display the context menu.

Negation's Properties

<i>Entity Id</i>	113
- Operand	
Ø	↔ Change 

Select Change to display the Selection Constellation.

The screenshot shows a user interface for a 'Condition Query'. At the top, there are several icons: a house, a magnifying glass, a double-headed arrow, a folder, and a document. To the right of these is a button labeled 'Condition Query'. Below this is a search bar with the placeholder text 'Type for filtering types and values...'. Under the search bar, there are two main sections: 'Values' and 'Types'. The 'Values' section contains a list of entities: NegatedValueComparison (74) - NegatedValueComparison, Negation (112) - Negation, Negation (43) - Negation, SequentiallyReducedConjunction (78) - SequentiallyReducedConjunction, and ValueComparison (83) - ValueComparison. The 'Types' section contains a list of entity types: AbstractJunction, Comparison, Condition, Conjunction, Disjunction, EmptySet, FullPermutation, FulltextComparison, NegatedValueComparison, Negation, SequentiallyReducedConjunction, and ValueComparison.

Values

NegatedValueComparison (74) - NegatedValueComparison
Negation (112) - Negation
Negation (43) - Negation
SequentiallyReducedConjunction (78) - SequentiallyReducedConjunction
ValueComparison (83) - ValueComparison

Types

AbstractJunction
Comparison
Condition
Conjunction
Disjunction
EmptySet
FullPermutation
FulltextComparison
NegatedValueComparison
Negation
SequentiallyReducedConjunction
ValueComparison

Select the Operand you would like to be negated and click Finish. The Operand object will then be added to the Negation object.

Negation's Properties

Entity Id 113

- Operand

ValueComparison (114)

Usage Example

Negation's Properties

<i>Entity Id</i>	43
------------------	----

- Operand

ValueComparison (44)

The screenshot above shows a configured Negation object, using [ValueComparison](#) as its Operand.

ValueComparison's Properties

<i>Entity Id</i>	44
------------------	----

Operator	like
----------	------

Right Operand	Technical Writer
---------------	------------------

- leftOperand

PropertyOperand (45)

Description - Value Comparison 44

This ValueComparison uses a PropertyOperand to define the left-hand side Operand. This defines on which property the value comparison should take place. In this example the Property Operand defines the property jobTitle. The right-hand side of the value comparison is a String Technical Writer. The operator is like, which is used to compare strings.

Written in SQL this ValueComparison would look like

```
WHERE jobTitle = 'Technical Writer'
```

The outcome of this query will display all entries that do not have the jobTitle Technical Writer
Example Results

Here is a Simple Query Action showing all results without any filters

Attendee	Email Address	First Name	Job Title	Second Name	Id
▷ Attendee (2)	johnsmith@email.com	John	Programmer	Smith	2
▷ Attendee (5)	maryAnn@email.com	Mary	Support	Ann	5
▷ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy	6
▷ Attendee (7)	TonyStark@lman.com	Tony	Engineer	Stark	7
▷ Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper	10
▷ Attendee (11)	DavidSilva@msn.com	David	Support	Silva	11
▷ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog	12
▷ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson	14
▷ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary	15
▷ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth	16
▷ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream	17
▷ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass	18
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell	20
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman	21
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol	22

Here is the result of the Negation Query as defined above. That is all entries where the jobTitle is not Technical Writer

Attendee	Email Address	First Name	Job Title	Second Name	Id
▷ Attendee (28)	JuliaDocherty@email.com	Julia	Project Developer	Docherty	28
▷ Attendee (31)	RobertLovelock@email.com	Robert	Programmer	Lovelock	31
▷ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass	18
▷ Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper	10
▷ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth	16
▷ Attendee (32)	PeterKramer@email.com	Peter	Web-Developer	Kramer	32
▷ Attendee (11)	DavidSilva@msn.com	David	Support	Silva	11
▷ Attendee (5)	maryAnn@email.com	Mary	Support	Ann	5
▷ Attendee (26)	KevinNolan@email.com	Kevin	Trainer	Nolan	26
▷ Attendee (25)	RobertSmith@email.com	Robert	IT Support	Smith	25
▷ Attendee (23)	DouglasStevens@email.com	Douglas	Developer	Stevens	23
▷ Attendee (24)	PaulineJones@email.com	Pauline	Administrator	Jones	24
▷ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream	17
▷ Attendee (30)	LisaAnn@email.com	Lisa	Database Manager	Ann	30
▷ Attendee (2)	johnsmith@email.com	John	Programmer	Smith	2
▷ Attendee (27)	JackFarmer@email.com	Jack	Trainer	Farmer	27
▷ Attendee (29)	MartinAmos@email.com	Martin	Project Manager	Amos	29
▷ Attendee (7)	TonyStark@lman.com	Tony	Engineer	Stark	7

Value Comparison

Table of Contents

Expand / collapse

- Introduction
- Value Comparison's Properties
- Adding New Value Comparison
- Configuring ValueComparison
- Usage Example
- Example Results

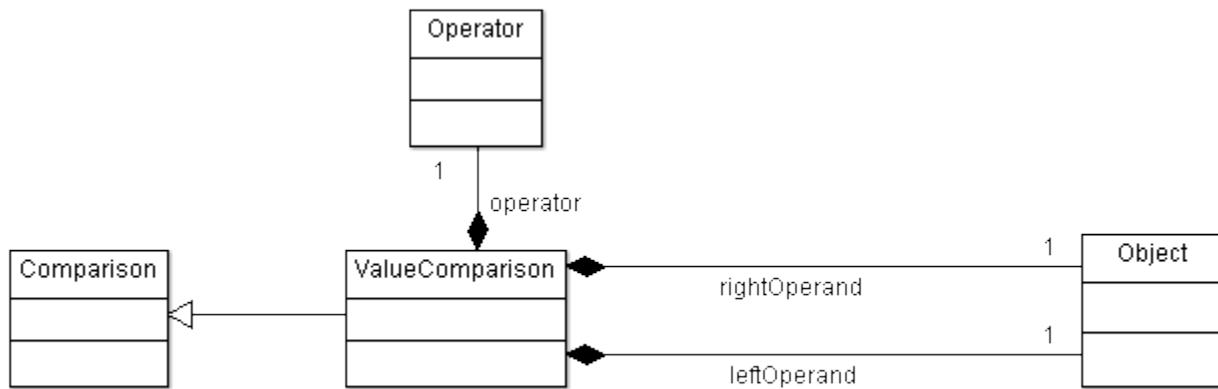
Child pages

Expand / collapse

- PropertyOperand

Introduction

The Value Comparison object compares a property to a value and returns all entries that are validated by this comparison. The ValueComparison is of the type Comparison and has three properties which can be defined: Operator, Left Operand and Right Operand.
a UML representation of the valueComparison object.



the valueComparison object as represented in tribefire.

ValueComparison's Properties

Entity Id	115
Operator	Ø
Right Operand	Ø
- leftOperand	Ø

Value Comparison's Properties

Property	Description	Accepted Value
Operator	The Operator that will be used to judge the two Operand values	A valid operator as chosen from a drop down list. <i>see below for a list of all operators available in tribefire.</i>
Right Operand	The Operand that sits on the right-hand side of a value comparision	A valid PropertyOperand, QueryFunction, simple Object or collection
Left Operand	The Operand that sits on the left-hand side of a value comparision	A valid PropertyOperand defining a property from a dataset

Value	String	Description
contains	contains	left contains right. This means that the left operand must be of a collection type. The right-hand side would be an element that is contained in this collection type.
in	in	Left operand element is in right operand collection. Right must be a collection that could contain an element like the value defined by the left operand.
equal	=	Left operand is equal to right operand . Both values must be of a comparable type. If you wish to compare strings, use the like operator not equal.
notEqual	!=	Left operand does not equal Right operand . Returns values that do not equal the value defined by the Right Operand.
greater	>	Left operand is greater than right operand.
greater or Equal	>=	Left operand is either greater or equal to right operand.
less	<	Left operand is less than right operand.
less or Equal	<=	Left operand is less or equal to right operand.
like	like	Used to compare Strings and has a similar functionality as equal. Right operand can also contain wildcards (*) and (?).
ilike	ilike	The same as like but the value of right operand is case insensitive.

 The two operands and the operator are used to build a query statement.

The syntax is:

```
leftOperand Operator rightOperand
```

An simple example

[▼ Click here to expand...](#)

```
jobTitle like Technical Writer
```

Adding New Value Comparison

To add a new Value Comparison mouse hover the Condition property of the Restrictions object so that the context-menu icon appears. Click on the icon to display the context-menu and then select Change.

Restriction's Properties

The screenshot shows the 'Restriction's Properties' dialog. The 'Entity Id' field is set to '10'. The 'Condition' property is selected, and a context-menu icon with the 'Change' option highlighted is shown. A tooltip for the 'Change' option is visible, stating 'Changes the given property value'.

The Selection Constellation will be displayed. Select the entity ValueComparison and click Finish.

The screenshot shows a software interface with a top navigation bar containing icons for home, search, comparison, folder, and a refresh symbol, followed by the title "Condition Query". Below the title is a search bar with the placeholder text "Type for filtering types and values...".

Values

- NegatedValueComparison (74) - NegatedValueComparison
- Negation (112) - Negation
- Negation (43) - Negation
- SequentiallyReducedConjunction (78) - SequentiallyReducedConjunction
- ValueComparison (83) - ValueComparison

Types

- AbstractJunction*
- Comparison
- Condition*
- Conjunction
- Disjunction
- EmptySet
- FullPermutation
- FulltextComparison
- NegatedValueComparison
- Negation
- SequentiallyReducedConjunction
- ValueComparison**

The new Value Comparison will be added to the Condition property of the Restriction object.

Restriction's Properties

<i>Entity Id</i>	10
– Condition	
ValueComparison (11)	
– Paging	
Ø	

Configuring ValueComparison

A new ValueComparison object has three values that can be defined: Left Operand, Right Operand and Operator. See [above](#) for more information on what each property does.

ValueComparison's Properties

<i>Entity Id</i>	115
Operator	Ø
Right Operand	Ø
– leftOperand	
Ø	

The property Operator is defined by selecting the entry you want from the drop down menu. See [the table above](#) for an explanation of what each Operator does.

ValueComparison's Properties

<i>Entity Id</i>	11
Operator	
Right Operand	equal
- leftOperand	notEqual
PropertyOperand (12)	like ilike greater greaterOrEqual less lessOrEqual in contains

To configure the operands, click on the valueComparison and then select the expand icon to display the objects properties.

The screenshot shows a software interface with a navigation bar at the top. Below the navigation bar, there is a list of objects. The first object in the list is "ValueComparison (0)". The second object is "ValueComparison (115)", which is currently selected, indicated by a yellow background. To the right of the list, there are two columns: "Operator" and "Entity Id". Under "Operator", the value "like" is listed. Under "Entity Id", the value "115" is listed. A small expand/collapse icon is located between the "Operator" and "Entity Id" columns.

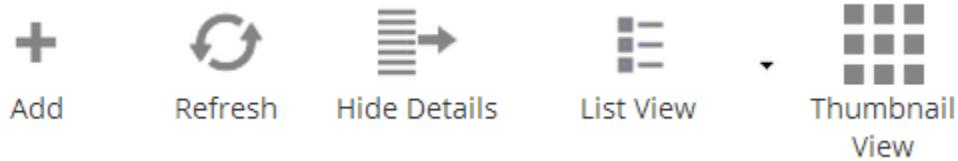
The valueComparison's details will then be shown.

The screenshot shows the same software interface as the previous one, but the "ValueComparison (115)" object is now expanded. The expanded view shows the "Operator" column with the value "like" and the "Entity Id" column with the value "115". Below these columns, there is additional information: "Left Operand: Ø" and "Right Operand: Ø". The expand/collapse icon is now located to the left of the "ValueComparison (115)" entry.

Select the leftOperand...

The screenshot shows a software interface with a tree view on the left. A node named "ValueComparison" is expanded, revealing two child nodes: "ValueComparison (115)" and "leftOperand: Ø". Below these, there is another "Right Operand: Ø".

...and then click the **Add** button



The Selection Constellation will be displayed. Select either a preexisting property operand, if there are any available, or click the PropertyOperand value located under the heading types to create a new one. Click finish to add the property operand to the ValueComparison.

The screenshot shows the "Selection Constellation" window. At the top, there are icons for Home, Filter, Compare, and Properties, followed by a search bar containing "PropertyOperand Query". Below the search bar, there are two sections: "Values" and "Types".

- Values:** A list of "PropertyOperand" items with their counts:
 - PropertyOperand (15) - PropertyOperand
 - PropertyOperand (22) - PropertyOperand
 - PropertyOperand (31) - PropertyOperand
 - PropertyOperand (35) - PropertyOperand
 - PropertyOperand (80) - PropertyOperand
- Types:** A list of "PropertyOperand" items, with the first item highlighted in orange:
 - PropertyOperand

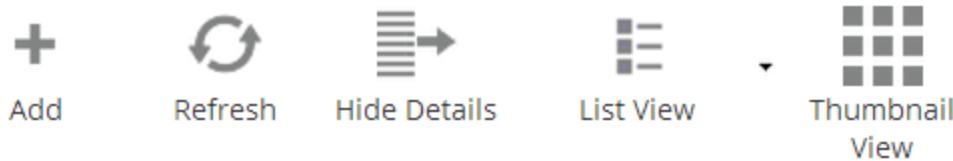
The new PropertyOperand will be added to the LeftOperand property. You can edit its properties in the details panel.

The screenshot shows the 'ValueComparison' properties dialog. In the 'Left Operand' section, there is a tree view with 'PropertyOperand (116)' selected. The 'Right Operand' section is currently empty, indicated by a placeholder 'Ø'. To the right, a panel titled 'PropertyOperand's Properties' contains a table with one row: Entity Id 116 and Property Name Ø. A red box highlights this panel.

You configure the Right Operand in the same manner. Select the Right Operand property...

The screenshot shows the 'ValueComparison' properties dialog. In the 'Right Operand' section, there is a tree view with 'PropertyOperand (116)' selected. The 'Left Operand' section is currently empty, indicated by a placeholder 'Ø'. The 'Operator' and 'Entity Id' fields are set to 'like' and 115 respectively.

...and click the **Add** button.



This will display the add new Property window. You can select either a simple type, a collection, a Property Operand or a Query Function from this window. If you wish to add a simple type, use the Search panel to write its value and then select it.

The screenshot shows the 'PropertyOperand Query' dialog. The search term 'Technical Writer' has been entered in the search bar. Below the search bar, the results are listed under the heading 'Values'. One result, 'string - Technical Writer', is highlighted with a yellow background and a pencil icon, indicating it is selected.

Double Click on the value or select it and click Finish to add it to the ValueComparison properties.

The screenshot shows the 'ValueComparison' properties dialog. In the 'Right Operand' section, the previously empty tree view now shows 'Technical Writer' selected. The 'Left Operand' section remains the same, with 'PropertyOperand (116)' selected. The 'Operator' and 'Entity Id' fields are still set to 'like' and 115 respectively.

After finishing, click the **Save** button located in the details panel to persist these changes.

Usage Example

Restriction's Properties

<i>Entity Id</i>	10
– Condition	
ValueComparison (11)	
– Paging	
Ø	

The screenshot above shows a configured ValueComparison Query. The Value Comparison object has been configured with a valid Operator, Left Operand and Right Operand.

ValueComparison's Properties

<i>Entity Id</i>	11
Operator	like
Right Operand	Technical Writer
– leftOperand	
PropertyOperand (12)	

Description - Value Comparison 11

This ValueComparision uses a PropertyOperand which defines the property field as jobTitle. That is, on which property should this query compare a value. The right-hand side is of the value *Technical Writer*. This means that the first validation is to return all entries who have the jobTitle of Technical Writer.

The outcome of this query will be all entries with the value Technical Writer in the jobTitle property.

Example Results

Here is a Simple Query Action showing all entries.

Attendee	Email Address	First Name	Job Title	Second Name	Id
▷ Attendee (2)	johnsmith@email.com	John	Programmer	Smith	2
▷ Attendee (5)	maryAnn@email.com	Mary	Support	Ann	5
▷ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy	6
▷ Attendee (7)	TonyStark@lman.com	Tony	Engineer	Stark	7
▷ Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper	10
▷ Attendee (11)	DavidSilva@msn.com	David	Support	Silva	11
▷ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog	12
▷ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson	14
▷ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary	15
▷ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth	16
▷ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream	17
▷ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass	18
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell	20
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman	21
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol	22

Here are the results with the properly configured ValueComparison value.

Attendee	Email Address	First Name	Job Title	Second Name	Id
▷ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol	22
▷ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog	12
▷ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy	6
▷ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell	20
▷ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson	14
▷ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman	21
▷ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary	15

PropertyOperand**Table of Contents**

- ▼ Expand / collapse
 - Introduction

Introduction

The Property Operand is used when querying values to define on which property any value should be compared to. The PropertyOperand is a simple object with only two properties that require configuring: Property Name and Source. However, when using this in conjunction with an entity query, you only need to define the property name, as the source has already been defined in the entity query.

representation of PropertyOperand as displayed in tribefire.

PropertyOperand's Properties

Entity Id \emptyset

Property Name \emptyset

- source

\emptyset

Property	Description	Accepted Value
Entity Id	The automatically generated ID given to this object after persisted in tribefire	NA
Property Name	The name of the property in a dataset. This is what the Operand will evaluate.	Any valid property name that corresponds to a property name of a dataset queried.
source	An source object that describes from which table (population) in a dataset this property operand should use	Any validly defined source object (either Join or From)

Full Text Comparison

Table of Contents

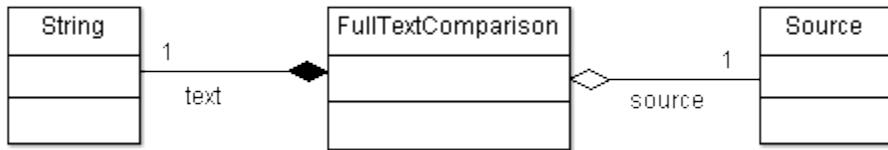
Expand / collapse

- Introduction
- Full Text Comparison Properties
- Adding New Full Text Comparison
- Usage Example
 - Usage Results

Introduction

The FulltextComparison object is a simple text search on a dataset. It will search for any occurrences of the string entered and return all entries which contain this value, regardless of which property the string occurs in. There are only two properties which need to be configured for the Full Text Comparison to function: Text and Source.

a UML representation of the *FullTextComparison* object.



The *FullTextComparison* as represented in tribefire.

FulltextComparison's Properties

Entity Id	∅
Text	∅
- Source	∅

Full Text Comparison Properties

Property	Description	Value
Entity Id	The automatically generated ID given to the object when it is persisted in tribefire	NA
Text	The text that will be search for.	Any valid String
Source	A valid source object (either from or join) on which the text will be search	Any validly configured source object

⚠ When using the *FullTextComparison* in conjunction with an Entity Query you do not have to configure the *Source* property, as this will have already been defined in the main query's properties. You must only enter a valid String to use this object.

Adding New Full Text Comparison

To add a new Full Text Comparison, mouse hover over the Condition property of the Restrictions object so that the context-menu icon appears. Click on the icon to display the context-menu and then select change.

Restriction's Properties

Entity Id	10
- Condition	 Change
- Paging	Changes the given property value

The Selection Constellation will appear. Select the entry FullTextComparison and then click Finish to add the object to the Restriction object.



Type for filtering types and values...

Values

- Disjunction (110) - Disjunction
- NegatedValueComparison (74) - NegatedValueComparison
- ValueComparison (115) - ValueComparison
- ValueComparison (50) - ValueComparison
- ValueComparison (59) - ValueComparison

Types

- AbstractJunction*
- Comparison
- Condition*
- Conjunction
- Disjunction
- EmptySet
- FullPermutation
- FulltextComparison
- NegatedValueComparison
- Negation
- SequentiallyReducedConjunction
- ValueComparison

The new FulltextComparison object will now be defined in the Condition property.

Restriction's Properties

<i>Entity Id</i>	66
– Condition	
FulltextComparison (67)	
– Paging	

∅

Click on the FulltextComparison object to open it for editing.

The screenshot shows a navigation path at the top: TemplateQueryAction Query > TemplateQueryAction (13) > template: Template > prototype: EntityQuery (65) > Restriction: Restriction (66) > Condition: FulltextComparison (67). Below this, a table displays the properties of the selected FulltextComparison object (Entity Id: 67). To the right, a panel titled "FulltextComparison's Properties" shows the configuration details for this specific object.

FulltextComparison	Text	Entity Id
FulltextComparison (67)	∅	67

FulltextComparison's Properties	
<i>Entity Id</i>	67
Text	∅
– Source	
∅	

You can now define in the Text property what String you would like to query.

Usage Example

Restriction's Properties

<i>Entity Id</i>	66
– Condition	
FulltextComparison (67)	
– Paging	
∅	

The above screenshot shows a properly configured FulltextComparison object. Clicking on it will show its configured values.

The screenshot shows the SAP Fiori Query Action configuration interface. A 'FulltextComparison' object is selected, with its properties listed on the right: Entity Id (67), Text ('support'), and Source (empty set). The navigation bar at the top shows the path: TemplateQueryAction Query > TemplateQueryAction (13) > template: Template > prototype: EntityQuery (65) > Restriction: Restriction (66) > Condition: FulltextComparison (67).

The Text property has been defined with the String support. The outcome of this query will be all results where the text value support appears, regardless of where and in which property it appears.

Usage Results

Here is a Simple Query Action showing all entries

Attendee	Email Address	First Name	Job Title	Second Name	Id
▶ Attendee (2)	johnsmith@email.com	John	Programmer	Smith	2
▶ Attendee (5)	maryAnn@email.com	Mary	Support	Ann	5
▶ Attendee (6)	ThomasHardy@email.com	Thomas	Technical Writer	Hardy	6
▶ Attendee (7)	TonyStark@lman.com	Tony	Engineer	Stark	7
▶ Attendee (10)	LucyCooper@msn.com	Lucy	Developer	Cooper	10
▶ Attendee (11)	DavidSilva@msn.com	David	Support	Silva	11
▶ Attendee (12)	OliviaFog@lman.com	Olivia	Technical Writer	Fog	12
▶ Attendee (14)	JohnSmithson@email.com	John	Technical Writer	Smithson	14
▶ Attendee (15)	Zachary@email.com	Ben	Technical Writer	Zachary	15
▶ Attendee (16)	RogerSmyth@cpo.com	Roger	Document Writer	Smyth	16
▶ Attendee (17)	PeterDownstream@cpo.com	Peter	Document Writer	Downstream	17
▶ Attendee (18)	TinaGlass@cpo.com	Tina	Document Writer	Glass	18
▶ Attendee (20)	VictoriaRussell@cpo.com	Victoria	Technical Writer	Russell	20
▶ Attendee (21)	ThomasWeinman@cpo.com	Thomas	Technical Writer	Weinman	21
▶ Attendee (22)	PaulCarrol@cpo.com	Paul	Technical Writer	Carrol	22

Here are the results with the properly configured Full Text Comparison object

Attendee	Email Address	First Name	Job Title	Second Name	Id
▶ Attendee (5)	maryAnn@email.com	Mary	Support	Ann	5
▶ Attendee (25)	RobertSmith@email.com	Robert	IT Support	Smith	25
▶ Attendee (34)	RogerWalters@support.com	Roger	Developer	Walters	34
▶ Attendee (11)	DavidSilva@msn.com	David	Support	Silva	11

You can see that searching for the value *support* will display all occurrences of this word. All attendees which have the word support in their job title are displayed, as well attendee 34 who has the word support in his email address.

Import Automatic Folders

Table of Contents

▼ Expand / collapse

- Import Automatic Folders
- Usage Example

Import Automatic Folders

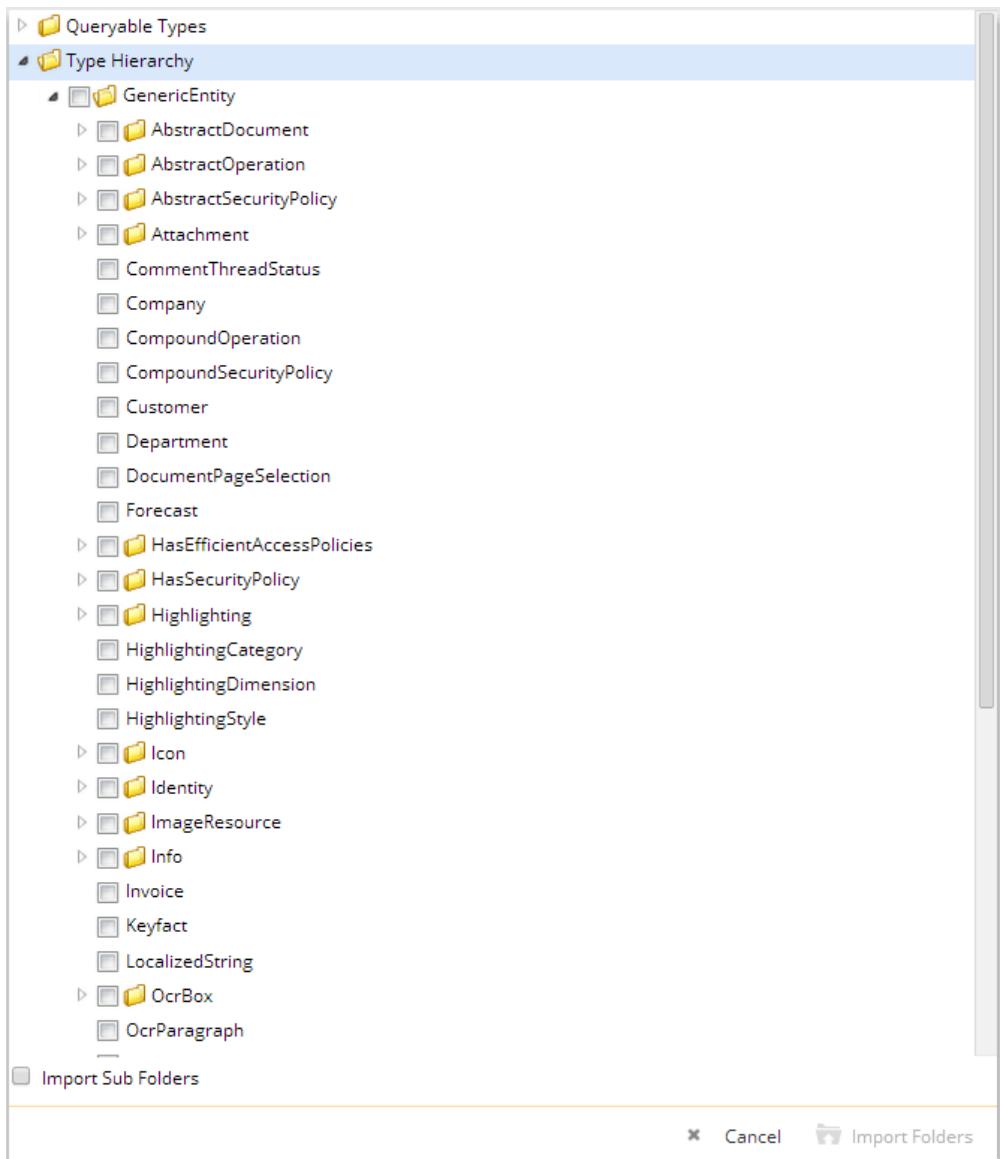
In addition to building queries using a Workbench configured for your access, you can also create new queries directly in your Explorer Mode by using the Import Automatic Folders option. This allows you to select between different entity types, defined by the model with which your access is configured. This will then create a series of Simple Query Actions and corresponding folders, which will then be automatically displayed in your Explorer Mode. If you wish to edit these folders further, you can do so by using your access' Workbench.

 Before you can make use of the Import Automatic Folders option, you must first create a standard folder by using a Workbench access configured for your access. If you do this first, you will not be able to right-click on the menu panel to obtain the Import Automatic Folders option

You can access the Import Automatic Folders option by right-clicking on the left-hand panel of your access's Data View. The Import Automatic Folder option will appear, which you can then select to begin importing folders.



After clicking the Import Automatic Folders option, a GIMA window will appear, allowing to choose between the different entity types which have been defined by the model this access is configured for.



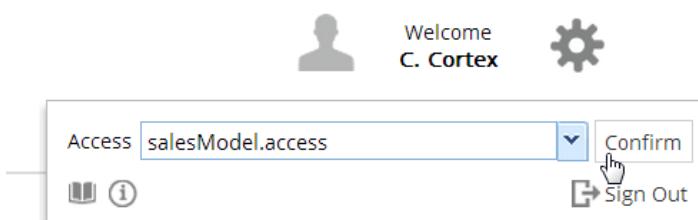
To select the entity types, you can check the box of the ones you wish to import and then click the Import Folders button. The selected entity types will then be imported.

Import Sub Folders

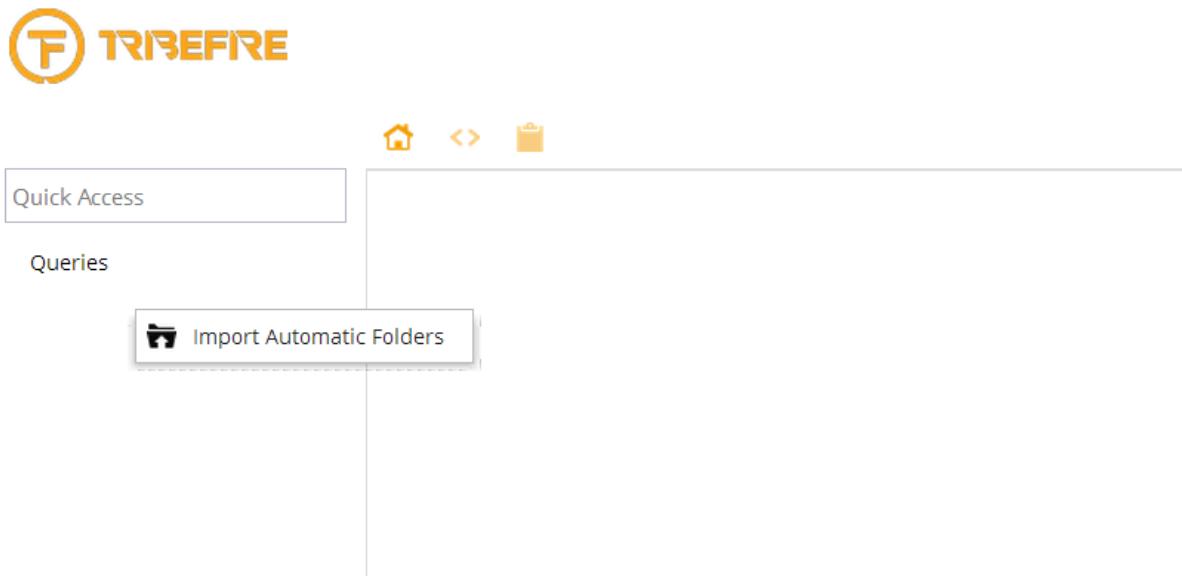
In addition to this, you can also select the box Import Sub Folders, and this will import an entity type plus any sub-types belonging to it. Select this box and then any entity types with sub-types with then select the Import Folders button.

Usage Example

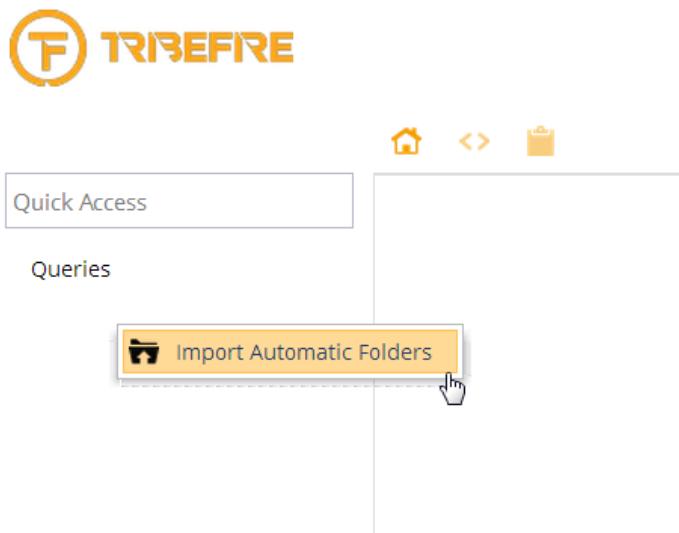
In tribefire Expert Mode select the access that you would like to use to import folders automatically and then click the Confirm button



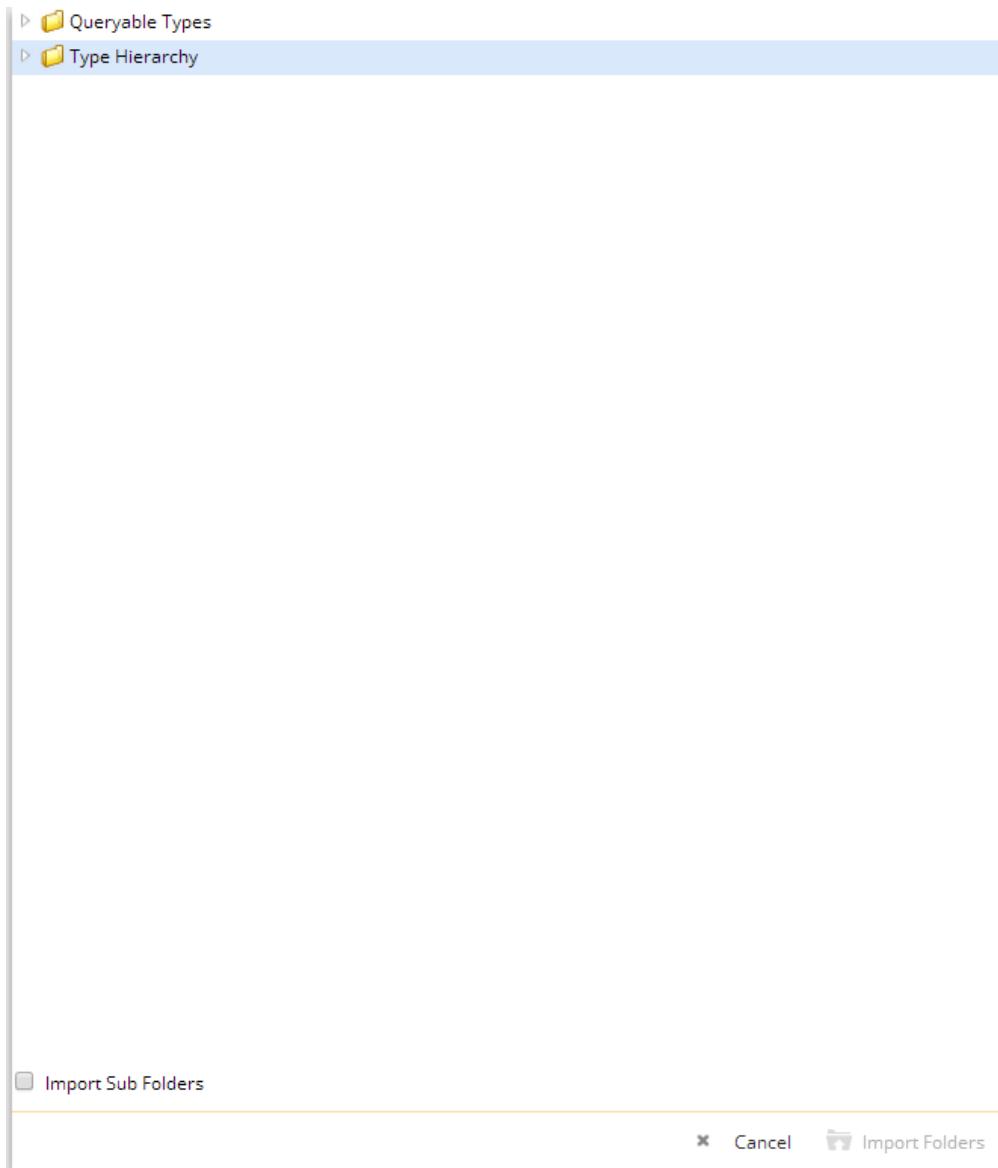
The Data View for this access will then be opened. Right-click on the left-hand menu panel to display the Import Automatic Folders option.



Click to option to select it.



A GIMA window will appear, allowing you to select different entity types.



Double-click on the option Type Hierarchy to collapse it. This will display a list of all the possible entity types which can be imported as a folder. Select two or three different entity types. You can also select the option *Import Sub Folders* and then an entity type which has further sub-types. The folders which have further sub-types are indicated by the yellow folder icon next to them. Once you have finished selecting your entity types, click Import Folders.

Queryable Types

Type Hierarchy

- GenericEntity
 - AbstractDocument
 - AbstractOperation
 - AbstractSecurityPolicy
 - Attachment
 - CommentThreadStatus
 - Company
 - CompoundOperation
 - CompoundSecurityPolicy
 - Customer
 - Department
 - DocumentPageSelection
 - Forecast
 - HasEfficientAccessPolicies
 - HasSecurityPolicy
 - Highlighting
 - HighlightingCategory
 - HighlightingDimension
 - HighlightingStyle
 - Icon
 - Identity
 - ImageResource
 - Info
 - Invoice
 - Keyfact
 - LocalizedString
 - OcrBox
 - OcrParagraph
- Import Sub Folders

Cancel Import Folders

Your entity types will then be imported and displayed on the left-hand panel of the Data View. Because we selected to import some sub folders, the hierarchy for this entity type has been kept.

Quick Access

- Customer
- Forecast
- Info
 - DocumentInfo
 - PageInfo
 - ResourceInfo

If you were to click on one of the links, the appropriate query will be executed.

The screenshot shows the TRIBEFRE software interface. At the top, there is a navigation bar with icons for Home, Back, Forward, and a search field labeled "Customer Query". Below the navigation bar is a "Quick Access" panel containing links for "Customer", "Forecast", and "Info" sections, along with "DocumentInfo", "PageInfo", "ResourceInfo", and "Queries" sub-links under the "Info" section. The main content area is titled "Customer Query" and displays a table with one row. The table has columns for "Customer" and "Customer (1)". The "Customer (1)" cell is highlighted with a yellow background.

Standard Components

Child pages

▼ Expand / collapse

- List View/Thumbnail View
- Localization
- Metadata
- Search Panel
- tribefire Types

Components

In the tribefire Control Center or Explorer there are standard components that are used in different contexts but all have a similar behavior. This section of the documentation will describe these different components, their functionality and how to use them.

List View/Thumbnail View

Table of Contents
<ul style="list-style-type: none"> ▼ Expand / collapse <ul style="list-style-type: none"> • Introduction <ul style="list-style-type: none"> • List view <ul style="list-style-type: none"> • Simple View • Detailed View • Flat View • Thumbnail View
Child pages
<ul style="list-style-type: none"> ▼ Expand / collapse

Introduction

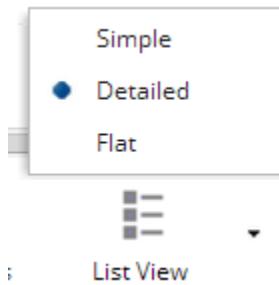
There are several ways to display information in tribefire's assembly panel, either from one of several list view or also as thumbnails view.

At the bottom of the assembly panel there are a series of buttons which change, depending on the context that tribefire has. However, every button panel includes the option to change your view.

List View Button	Thumbnail Button
 List View	 Thumbnail View

List view

The list view button allows you to select between three different types of list view: Simple, Detailed and Flat. Click on the list view button, so that a context-menu will appear with the three selectable options. You can then select the type of view you would like.



Simple View

The simple view displays only the main name of the results, with no extra details included.

Access	
BTModel (BTModel)	
SalesModel (SalesModel)	
SampleAccess (SampleAccess)	

example of simple view

However, it is still possible to select an object, and using the collapse icon  to view and edit further properties.

Access

 BTModel (BTModel)
 <i>SalesModel (SalesModel)</i>
delegates \emptyset
 Model: SalesModel#1.0
 Workbench Access: \emptyset
 SampleAccess (SampleAccess)

Detailed View

The detailed view shows a series of columns displaying the most important information about the selected object.



Depending on the context of the selected object, the columns will display different information.

Access	Deployment State	Description	Simulation Mode	Id	External Id	Name
 BTModel (BTModel)	\emptyset	\emptyset	<input type="checkbox"/>	3	BTModel	BTModel
 <i>SalesModel (SalesModel)</i>	\emptyset	\emptyset	<input checked="" type="checkbox"/>	6	SalesModel	SalesModel
 SampleAccess (SampleAccess)	deployed	\emptyset	<input type="checkbox"/>	1	SampleAccess	SampleAccess

However, it is still possible to select an object, and using the collapse icon  to view and edit further properties.

Access	Deployment St...	Description	Simulation Mode	Id	External Id	Name
 BTModel (BTModel)	\emptyset	\emptyset	<input type="checkbox"/>	3	BTModel	BTModel
 <i>SalesModel (SalesModel)</i>						
delegates \emptyset			<input checked="" type="checkbox"/>	6	SalesModel	SalesModel
 Model: SalesModel#1.0						
 Workbench Access: \emptyset						
 SampleAccess (SampleAccess)	deployed	\emptyset	<input type="checkbox"/>	1	SampleAccess	SampleAccess

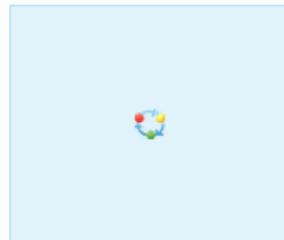
Flat View

The flat view will display various properties about each model, however, it is not possible to expand any selected objects to view and edit them.

Deployment State	Description	Simulation Mode	Id	External Id	Name
\emptyset	\emptyset	<input type="checkbox"/>	3	BTModel	BTModel
\emptyset	\emptyset	<input checked="" type="checkbox"/>	6	SalesModel	SalesModel
deployed	\emptyset	<input type="checkbox"/>	1	SampleAccess	SampleAccess

Thumbnail View

The thumbnail view will display all objects returned by a query as a series of thumbnails. No information about each object will be displayed in the main results panel. However, the details panel can be displayed and the important properties for the object selected will be displayed there.



BTModel (BTModel)



SalesModel (SalesModel)



SampleAccess
(SampleAccess)



List View

button will display the results in list view.

Selecting the

Localization

Table of Contents

- ▼ Expand / collapse
 - Localization
 - Introduction
 - Localization Panel
 - Usage
 - Defining property type
 - Accessing Localization Panel
 - Adding New Localized Values
 - Removing Localized Value
 - Clear All Localizations

Localization

Introduction

Localization is the process of adapting a product for different languages, allowing you to define values for different languages that you wish to use. In tribefire, localization is handled by the entity type `LocalizedString`. To use localization, the property type **must** be **defined** as `LocalizedString`, and **not** a **simple type** with `String` as the type signature.

GmEntityType's Properties

<i>Id</i>	142
– Type Signature	
	com.braintribe.model.generic.i18n.LocalizedString
Is Abstract	<input type="checkbox"/>
Is Marked For Discard	<input type="checkbox"/> <input checked="" type="checkbox"/>
Is Plain	<input checked="" type="checkbox"/>
– Artifact Binding	
	BindingArtifact (15860)
– Meta Data	
– Properties	
1. id	
2. localizedValues	
– propertyMetaData	
	∅
– Super Types	
1. GenericEntity	

The `LocalizedString` entity has a property called `localizedValues`, which is a map which contains the international code for a language as its key and the localized string as its value.

GmProperty's Properties

<i>Id</i>	110
Is Id	<input type="checkbox"/>
Is Overlay	<input type="checkbox"/>
Name	localizedValues

- Entity Type

LocalizedString

- Meta Data

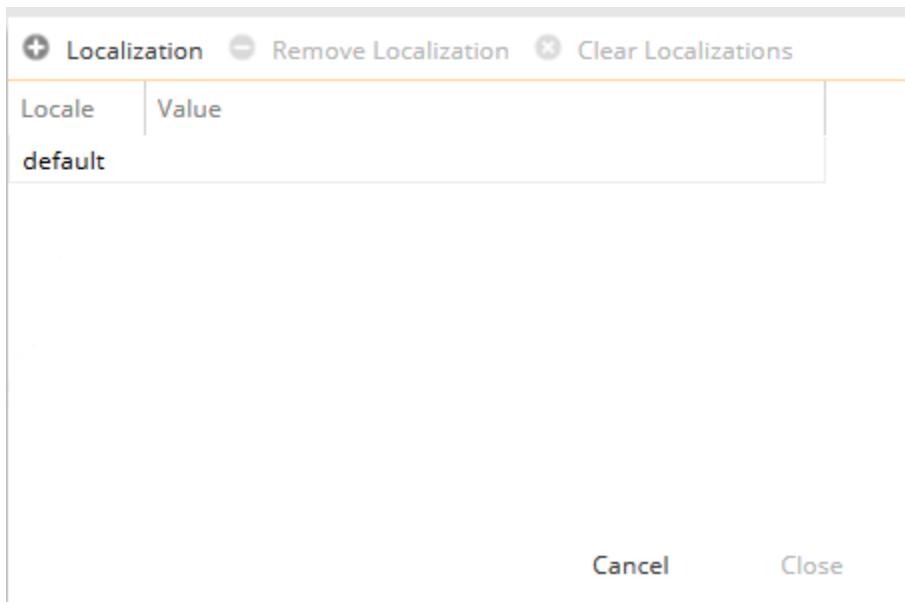
- PropertyDisplayInfo (36)

- Type

map<string,string>

Localization Panel

You can localize your string by using the **Localization panel**. This panel has two columns: **Locale** and **Value**. The Locale represents the key in the map and accepts the **internationally recognized language codes**, such as en for English or de for German. The value represents the **word in the target language**.



Usage

Defining property type

Any property that you wish to have a localized string must have the correct type defined, LocalizedString.

GmProperty's Properties

<i>Id</i>	5151
Is Id	<input type="checkbox"/>
Is Overlay	<input type="checkbox"/>
Name	field

- Entity Type

Organization

- Meta Data

- PropertyDisplayInfo (7405)

- Type

LocalizedString

Accessing Localization Panel

You can then click the ▼ icon of the property to be localized to display the context menu.



The context menu will display the Localization option.



Select it to open the Localization panel.

The screenshot shows a localization dialog box with the following interface:

- Localization** button (highlighted with a red box)
- Remove Localization** button
- Clear Localizations** button
- Locale** column: `default`
- Value** column: An empty input field.
- Cancel** and **Close** buttons at the bottom.

The default value for each localized string will be used if a language selected is not included in the localized list of values or if it is not recognized.

Adding New Localized Values

You can add a new language and value by clicking the **Localization** button.

The screenshot shows a localization dialog box with the following interface:

- Localization** button (highlighted with a red box)
- Remove Localization** button
- Clear Localizations** button
- Locale** column: `default`
- Value** column: Placeholder text `Default Value`.

You can add as many localizations as you wish.

The screenshot shows a localization dialog box with the following data:

Locale	Value
<code>default</code>	Default value
<code>en</code>	English value
<code>de</code>	German value
<code>it</code>	Italian value

Removing Localized Value

You can remove a localization from the list by selecting it and clicking the **Remove Localization** button.

Localization	Remove Localization	Clear Localizations
Locale	Value	
default	Default value	
en	English value	
de	German value	
it	Italian value	

The value will be removed.

Localization	Remove Localization	Clear Localizations
Locale	Value	
default	Default value	
en	English value	
it	Italian value	

Clear All Localizations

Click the **Clear Localizations** button to clear all localizations.

Metadata

Table of Contents

✓ Expand / collapse

- Metadata

Child pages

✓ Expand / collapse

- General Metadata Properties
- Model Metadata
- Entity Type Metadata
- Property Metadata
- Enum Type Metadata

Metadata

The definition of metadata is data about data. In the context of tribefire this is the process of using this data to enrich a model, entity type, property or enumeration (enum). This data is broadly split into two different categories: Constraints and Display. Constraints act to restrict or change a model in some way, whereas Display functions on how the model is displayed in Explorer Mode.

This process is defined by the Metadata model, which is a base class that is then used to define the different type related metadata models (that is, the ModelMetadata model, EntityTypeMetadata model and so on). Each type also has specific models that relate to either constraints (EntityConstraint model and so on) or display information (EntityDisplayInfo model and so on).

Each metadata property is then added to the relevant part of the model by using tribefire Expert Mode. This section of the documentation is split into these relevant parts:

- Model Metadata
- Entity Type Metadata
- Property Metadata
- Enum Metadata

General Metadata Properties

Table of Contents

- ▼ Expand / collapse
 - General Metadata Properties

General Metadata Properties

In addition to properties which affect the individual functionality of each metadata, there are also general properties common to all metadata.

General

<i>Id</i>	7095
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

Property	Description
Id	An automatically generated ID, configured when you first save an instance of a metadata.
Conflict Priority	A decimal number between 0 and 1. It is used by tribefire to decide which metadata should be used if there is more than one instance of the same metadata. The higher the number the more likely it will be used.
Inheritance	If the element which the metadata is attached has any sub-types, you can use this property to define whether the metadata should be inherited.
Name	You can use this property to give a name to the instance of the metadata. <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">  Some metadata use this property as part of its functionality. For example, the displayInfo metadata will use the value given here as labels for that model, entity or property. </div>
Description	You can use this property to give a description to this metadata.
Selector	The selector property is used to add a constraint on the functionality of the metadata, through the use of a discriminator, a role or use case . There are several different types of discriminators available. You can also use a NegationSelector to negate the selection constraints of a metadata's action, or use either a conjunction (AND) or a disjunction (OR) to determine two or more constraints on a selector.
Origin	This property allows you to define the original source of the metadata. That is, where was this metadata created. It could be from the Control Center, either manually or through auto enriching, through an import by using the Attach Metadata or created by using an external tool.

Important

If an entity has a metadata defined which is also defined in a super type, the metadata defined on the entity will be used. However, setting this field on the metadata in the super type will mean that the conflict property between these two same metadata will be used to decided which should be used.

Conflict Priority

Table of Contents

▼ Expand / collapse

- Conflict Priority
- Usage

Conflict Priority

The conflict priority property of a metadata is used by tribefire to decide between conflicting metadata. For example, if there are two metadata that are both of the same type, the conflict priority will be used to determine which one will be used. It will also be used to decide between a metadata defined locally (that is, define directly on the type which it belongs) or metadata inherited from a super type. However, in this circumstance the locally defined property will always be used unless the **important** boolean has been set to true in the super type where the metadata has been inherited from.

The conflict priority property is of the type **double**. The higher the number the more likely that this metadata will be selected, with the metadata with the highest value being selected by tribefire. The number defined should be a decimal number ranging from between 0 and 1. The lowest value being 0 and 1 the highest.

Usage

If you have a model, entity or property that has more than one metadata that is either the same, or will cause a conflict decision to be made in tribefire, you can assign a decimal value to each metadata.

GmEntityType

Customer

▷ Artifact Binding: [BindingArtifact \(14932\)](#)

Meta Data (2)

▷ [SelectiveInformation \(7402\)](#)

▷ [SelectiveInformation \(7403\)](#)

▷ [Properties \(12\)](#)

▷ [propertyMetaData Ø](#)

▷ [Super Types \(1\)](#)

Each metadata value should be between 0 and 1, expressed as a decimal number.

SelectiveInformation's Properties

important	<input type="checkbox"/>
Template	<code> \${name}</code>
- origin	
	Ø

General

<i>Id</i>	7402
Conflict Priority	1.00
Inheritance	enabled
Name	Ø
- Description	
	Ø
- Selector	
	Ø

SelectiveInformation's Properties

important	<input type="checkbox"/>
Template	<code> \${description}</code>
- origin	
	Ø

General

<i>Id</i>	7403
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	
	Ø
- Selector	
	Ø

In the screenshots above the two SelectiveInformation metadata will be in conflict when displayed in Explorer Mode. However, the metadata with the conflict priority of the value 1.0 (the one with the template `${name}`) will be determined the winner due to its higher value.

The screenshot shows the 'Customer Query' interface. On the left, there is a tree view with nodes for 'Customer' and three collapsed entries: 'Acme Corp.', 'CHOAM', and 'MomCorp'. On the right, there is a table with four columns: 'Customer Descr...', 'Name', and 'Id'. The table has three rows:

Customer Descr...	Name	Id
Looney Tunes	Acme Corp.	1
Dune	CHOAM	2
Futurama	MomCorp	3

The row for 'Acme Corp.' is highlighted in yellow, indicating it is the active or selected entry. The other two rows are greyed out.

Important**Table of Contents**

▼ Expand / collapse

- Important
- Usage

Important

The important field is used to determine how tribefire should choose between metadata defined on an entity or property and the same metadata inherited from a supertype. When this field is set to false any metadata defined locally, that is, the metadata defined on the entity or property, will automatically be chosen. However, if you decide that you wish the conflict priority decision to be made, you set the important field to true.

A conflict priority will then decide between the two values of the metadata and choose the one with the higher value.

You must set the Important field to true in the super type metadata that will be inherited.

Usage

In the example below, there is an entity called Address. This has a super type called Standard Identifiable. Both have the metadata EntityTypeDisplayInfo.

Address																	
 Address <ul style="list-style-type: none"> ▷ Artifact Binding: BindingArtifact (14897) ◀ Meta Data (1) <ul style="list-style-type: none"> ▷ EntityTypeDisplayInfo (7112) ▷ Properties (5) ▷ propertyMetaData Ø ▷ Super Types (1) 	General <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td><i>Id</i></td> <td>7112</td> </tr> <tr> <td>Conflict Priority</td> <td>0.00</td> </tr> <tr> <td>Inheritance</td> <td>enabled</td> </tr> <tr> <td>Name</td> <td>Address</td> </tr> <tr> <td colspan="2">– Description</td> </tr> <tr> <td colspan="2">Ø</td> </tr> <tr> <td colspan="2">– Selector</td> </tr> <tr> <td colspan="2">Ø</td> </tr> </table>	<i>Id</i>	7112	Conflict Priority	0.00	Inheritance	enabled	Name	Address	– Description		Ø		– Selector		Ø	
<i>Id</i>	7112																
Conflict Priority	0.00																
Inheritance	enabled																
Name	Address																
– Description																	
Ø																	
– Selector																	
Ø																	
StandardIdentifiable																	

The screenshot shows the Braintribe interface with the following details:

- Left Panel (Tree View):**
 - StandardIdentifiable
 - Artifact Binding: BindingArtifact (14900)
 - Meta Data (1)
 - EntityTypeDisplayInfo (7120) (highlighted in orange)
 - Properties (1)
 - propertyMetaData Ø
 - Super Types (1)
- Right Panel (General Tab):**

General

<i>Id</i>	7120
Conflict Priority	0.50
Inheritance	enabled
Name	Inherited Metadata

 - Description: Ø
 - Selector: Ø
- Bottom Section (EntityTypeDisplayInfo's Properties):**

EntityTypeDisplayInfo's Properties

Font Color	Ø
important	<input checked="" type="checkbox"/>

 - origin: Ø

In the example shown above, the super type's metadata has the important field set to false. This means that the value of the metadata in the sub type (Address) will automatically be used, even though the value of the super type's Conflict Priority is higher.

The screenshot shows the Braintribe Address Query interface with the following details:

- Top Bar:** Address Query
- Table:**

Address	City	Country
Address (2)	Vienna	Austria
Address (6)	Ø	Österreich

If we were to set the importance field in the metadata of the super type to true. The Conflict Priority values will now be used to decide which metadata should be used. In this case, the super type has a higher value (0.5 being greater than 0.0) and will be used.

General

<i>Id</i>	7120
Conflict Priority	0.50
Inheritance	enabled
Name	Inherited Metadata
– Description	Ø
– Selector	Ø

EntityTypeDisplayInfo's Properties

Font Color	Ø
important	<input checked="" type="checkbox"/>
– origin	Ø

The label for this metadata is *Inherited Metadata* and will be used instead of the locally defined metadata.

The screenshot shows a software interface titled "Inherited Metadata Query". The title bar has icons for home, back, forward, and close, with "Inherited Metadata Query" highlighted. Below the title bar, there is a toolbar with a magnifying glass icon. The main area is a table with the following data:

Inherited Metadata	City	Country
Inherited Metadata (2)	Vienna	Austria
Inherited Metadata (6)	Ø	Österreich

Inheritance

Table of Contents

▼ Expand / collapse

- Inheritance
- Usage
 - Enabled
 - Disabled

Inheritance

The inheritance field allows you to define whether the metadata **should be inherited** with the property in a subtype. There are two possible values: **enabled** or **disabled**. If the value of this property is set to enabled, the metadata's functionality will be inherited along with the entity, property, or enum type and used in the subtype. However, if the value of this property is set to disabled, then the metadata will not be inherited and will have no effect on the sub type's inherited entity, property or enum.

The **default value** for inheritance when creating a new instance of a metadata is **enabled**.

Usage

In the example below there is a entity type called Organization, which has various properties, including headquartersLocation. A subtype of Organization is called Company, which inherits this property. headquartersLocation has a PropertyDisplayInfo metadata with the value Headquarters Location attached to it.

Enabled

If you set the value of inheritance to **enabled**, the metadata's functionality will be inherited along with the entity, property or enum type to which it belongs.

PropertyDisplayInfo's Properties

Font Color	∅
important	█
– origin	
	∅

General

<i>Id</i>	7406
Conflict Priority	0.00
Inheritance	enabled
Name	Headquarters Location
– Description	
	∅
– Selector	
	∅

The functionality of the metadata will also be inherited.

Company	Appointments	Description	Field	Headquarters Location	Name	Id
Company (1)	Ø	A software file with the courage of being different	Ø	Vienna	BrainTribe	1

Disabled

If you set the value of inheritance to **disabled**, the metadata's functionality will **not** be inherited along with the entity, property or enum type to which it belongs.

PropertyDisplayInfo's Properties

Font Color	Ø
important	Ø
- origin	Ø
	Ø

General

Id	7406
Conflict Priority	0.00
Inheritance	disabled
Name	Headquarters Location
- Description	Ø

- Selector

Ø

The functionality of the metadata will not be inherited.

Company	Appointments	Description	Field	headquartersLocation	Name	Id
Company (1)	Ø	A software file with the courage of being different	Ø	Ø	BrainTribe	1

Selectors

Table of Contents

▼ Expand / collapse

- MetadataSelectors
 - Introduction
 - Configuration
 - Simple Property Discriminators
 - Use Case Selector
 - Role Selector
 - Logical Selectors

MetadataSelectors

Introduction

The different selectors in tribefire allow you to define constrictions on the functionality of metadata. This means that after properly configuring the selector, the metadata will only function when that condition is met.

The base class for all selectors is the `MetaDataSelector`. The selector property found in all metadata is also of the type `MetaDataSelector`. You configure your metadata by adding a new, or preexisting, instance of a `MetaDataSelector` to the `Selector` property.

Configuration

General

<i>Id</i>	7399
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

There are four subtypes of `MetadataSelectors` available for use in tribefire: `Simple Property Discriminators`, `Use Case Selector`, `Role Selector` and `Logical Selectors`. To add an instance of one of these subtypes to your metadata, you simply select it from a list, either a preexisting instance (listed under the heading `Values` in the screenshot below) or create a new instance (listed under the heading `Types` in the screenshot below).

Values

[NegationSelector \(5\)](#) - NegationSelector
[RoleSelector \(4\)](#) - RoleSelector
[RoleSelector \(9\)](#) - RoleSelector
[UseCaseSelector \(10\)](#) - UseCaseSelector
[UseCaseSelector \(6\)](#) - UseCaseSelector

Types

[BooleanPropertyDiscriminator](#)
[ConjunctionSelector](#)
[DatePropertyDiscriminator](#)
[DisjunctionSelector](#)
[IntegerPropertyDiscriminator](#)
JunctionSelector
[LongDiscriminatorValue](#)
MetaDataSelector
[NegationSelector](#)
[NullPropertyDiscriminator](#)
[RoleSelector](#)
SimplePropertyDiscriminator
[StringPropertyDiscriminator](#)
[StringRegexPropertyDiscriminator](#)
[UseCaseSelector](#)

Simple Property Discriminators

There are several different types of property discriminators, each one related to its associated simple type. For example, to use this selector on the simple type String, you would use the StringPropertyDiscriminator, or if you wish to match a to a regular expression, the StringRegexpPropertyDiscriminator. There is also a BooleanPropertyDiscriminator, DatePropertyDiscriminator, IntegerPropertyDiscriminator and so on.

Use Case Selector

The UseCaseSelector is used to determine on which component (that is, the area of the screen) in tribefire the constraint should take place. There is only one type of UseCaseSelector and contains only one property: Use Case. You enter the use case on which you would like this constraint to act. A list of the different use cases and what areas they effect can be found at the UseCaseSelector page.

UseCaseSelector

Role Selector

The role selector is used to define metadata behavior depending on roles defined in Authentication and Authorization. Each user can be assigned roles or to a group, which is made up of different roles. Using the RoleSelector means that the metadata will only take effect if the current user has been assigned the role defined.

RoleSelector

Logical Selectors

The logical selectors do not place constraints on the metadata themselves. Rather, they change the behavior of the other MetadataSelectors. There are two main types of logical selector: NegationSelector and JunctionSelector, which has two further subtypes associated with it DisjunctionSelector and ConjunctionSelector. The JunctionSelector can not be used directly.

The NegationSelector negates the behavior of a MetadataSelector, meaning that instead of the metadata operating when the selector's condition

is met, it will work in all other case apart from the defined condition. Logically this means, *the metadata should not take effect if the selector's condition is met.*

The DisjunctionSelector and ConjunctionSelector have a similar functionality. Both allow you to build a list of different MetadataSelectors for a metadata, instead of only being able to define one. The only difference between the two is their logical function. The DisjunctionSelector functions as an **OR** operator, meaning that if one of the conditions is met then the metadata will take effect. Whereas, the ConjunctionSelector functions as an **AND** operator. This means that all MetadataSelector conditions must be met before the metadata will take effect.

[NegationSelector](#)

[DisjunctionSelector](#)

[ConjunctionSelector](#)

Conjunction Selector

Table of Contents

- ▼ Expand / collapse
 - ConjunctionSelector
 - Usage

ConjunctionSelector

The conjunction selector allows you to determine **two or more** cases that must all be **true** for this metadata to take **effect**. It has only one property which can be configured: Operands. Using this property, you build a list of metadata selectors that **all** must be true before the action of the metadata function. This is the equivalent of an **AND** operator.

You can add as many selectors to the Operand property as you like.

Usage

Define the selector property with an instance of the ConjunctionSelector.

PropertyDisplayInfo's Properties

Font Color	<input type="button" value="∅"/>
important	<input checked="" type="checkbox"/>
– origin	<input type="button" value="∅"/>

General

<i>Id</i>	7348
Conflict Priority	0.00
Inheritance	enabled
Name	Customer Description
– Description	<input type="button" value="∅"/>

– Selector
ConjunctionSelector (8)

Double click the newly created instance to open it up for editing.

The screenshot shows the Entity Explorer interface with the following details:

- Path: Information Models Query > SalesModel#1.0 > Entity Types > PropertyDisplayInfo (7348) > Selector: ConjunctionSelector (8)
- Selected item: ConjunctionSelector (8)
- Properties pane (right side):
 - ConjunctionSelector's Properties**
 - Id*: 8
 - Operands

Here you can add as many MetadataSelectors as you wish.

Information Models Query SalesModel#1.0 Entity Types PropertyDisplayInfo (7348) Selector: ConjunctionSelector (8)

ConjunctionSelector	Id	ConjunctionSelector's Properties
ConjunctionSelector (8)	8	<ul style="list-style-type: none"> - Operands 1. UseCaseSelector (10) 2. RoleSelector (9)

You define each selector as you would normally.

UseCaseSelector's Properties		RoleSelector's Properties	
<i>Id</i>	10	<i>Id</i>	9
Use Case	assembly	- Roles	<ul style="list-style-type: none"> • john.smith.role.a

In the screenshots above, two selectors have been chosen: a UseCaseSelector and a RoleSelector. This means that both these conditions must be met before the metadata will take affect.

The screenshot shows a 'Customer Query' interface. On the left, there is a sidebar with icons for home, search, and a user profile. The main area has tabs for 'Customer Query' and 'Customer'. A search bar at the top right contains the text 'Customer Description' with a red box around it. Below the search bar is a table with columns: Customer, Customer Description, Name, and Id. The table contains four rows labeled 'Customer (1)' through 'Customer (4)'. To the right of the table is a panel titled 'Customer's Properties' showing three properties: Id (1), description (Looney Tunes), and Name (Acme Corp.).

Disjunction Selector

Table of Contents

- ▼ Expand / collapse
 - DisjunctionSelector
 - Usage

DisjunctionSelector

The disjunction selector allows you to define two or **more** MetadataSelectors where the **metadata** should **function**. Unlike the ConjunctionSelector, where all selector conditions must be met, the DisjunctionSelector functions as an **OR** operator. This means that if any **one** of the **conditions** are **met**, the metadata will function according to the behavior defined by that selector.

You can add as many selectors to the Operand property as you like.

Usage

Define the selector property with an instance of the DisjunctionSelector

SelectiveInformation's Properties

important	<input type="checkbox"/>
Template	\${name}
- origin	
	Ø

General

<i>Id</i>	7402
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	Ø
- Selector	DisjunctionSelector (14)

Double click on the newly created instance to open it up for editing.

The screenshot shows the 'SelectiveInformation' entity properties page. The 'DisjunctionSelector' property is expanded, revealing its value: 'DisjunctionSelector (14)'. This value is highlighted with a red rectangle. The top navigation bar includes tabs for 'Information Models Query', 'SalesModel#1.0', 'Entity Types', 'SelectiveInformation (7402)', and 'Selector: DisjunctionSelector (14)'. The right side of the screen displays the 'DisjunctionSelector's Properties' panel, which lists the 'Id' as 14 and the 'Operands' as an empty set (Ø).

You can add as many MetadataSelectors as you wish.

The screenshot shows the 'Information Models Query' interface with the path: SalesModel#1.0 > Entity Types > SelectiveInformation (7402) > Selector: DisjunctionSelector (14). The 'DisjunctionSelector (14)' node is selected, showing its properties: Id (14), Operands (2), and Roles (16). The 'DisjunctionSelector's Properties' panel shows the same information.

You define each selector as normal

RoleSelector's Properties

The screenshot shows the 'Information Models Query' interface with the path: SalesModel#1.0 > Entity Types > SelectiveInformation (7402) > Selector: RoleSelector (16). The 'RoleSelector (16)' node is selected, showing its properties: Id (16) and Roles (1). The 'RoleSelector's Properties' panel shows the same information.

RoleSelector's Properties

The screenshot shows the 'Information Models Query' interface with the path: SalesModel#1.0 > Entity Types > SelectiveInformation (7402) > Selector: RoleSelector (15). The 'RoleSelector (15)' node is selected, showing its properties: Id (15) and Roles (1). The 'RoleSelector's Properties' panel shows the same information.

This means that if any of the conditions are met, that is, if one of the selectors are true, the metadata function according to that condition. In the screenshots above, two RoleSelectors have been defined. This means, that if the user has either the role `john.smith.role.a` OR `robert.taylor.role.a` then the metadata will take effect.

The user John Smith has the role `john.smith.role.a` defined, so the metadata will be displayed in this case

The screenshot shows the 'Customer Query' interface with the path: Customer > Customer. The user 'John Smith' is logged in, as shown in the top right corner. The left sidebar shows the 'Customer' entity with items: Acme Corp., CHOAM, MomCorp, and Rich Industries. The main table lists customers with columns: Customer Description, Name, and Id. The table shows four rows: Looney Tunes (Acme Corp., Id 1), Dune (CHOAM, Id 2), Futurama (MomCorp, Id 3), and Richie Rich (Rich Industries, Id 4). The 'Customer's Properties' panel on the right shows the properties for the first customer: Id (1), Customer Description (Looney Tunes), and Name (Acme Corp.).

The user Robert Taylor has the role `robert.taylor.role.a` defined, so the metadata will be displayed in this case.

The screenshot shows the 'Customer Query' interface with the path: Customer > Customer. The user 'Robert Taylor' is logged in, as shown in the top right corner. The left sidebar shows the 'Customer' entity with items: Acme Corp., CHOAM, MomCorp, and Rich Industries. The main table lists customers with columns: Customer Description, Name, and Id. The table shows four rows: Looney Tunes (Acme Corp., Id 1), Dune (CHOAM, Id 2), Futurama (MomCorp, Id 3), and Richie Rich (Rich Industries, Id 4). The 'Customer's Properties' panel on the right shows the properties for the first customer: Id (1), description (Looney Tunes), and Name (Acme Corp.).

However, the user Cortex has neither the role `john.smith.role.a` nor `robert.taylor.role.a` assigned, and therefore, the metadata will not be displayed in this case.

Customer Query

Customer Query

Customer

Customer (1)

Customer (2)

Customer (3)

Customer (4)

Customer Descr... Name Id

Customer Descr...	Name	Id
Looney Tunes	Acme Corp.	1
Dune	CHOAM	2
Futurama	MomCorp	3
Richie Rich	Rich Industries	4

Customer's Properties

Id	1
description	Looney Tunes
Name	Acme Corp.

Negation Selector

Table of Contents

- ▼ Expand / collapse
 - NegationSelector
 - Usage

NegationSelector

The Negation Selector is of the type metadata selector and allows you to negate the action of a metadata depending on the configuration of its property: Operand. The operand accepts a further metadata selector, either a discriminator, a use case selector, or role selector. The logic defined by the negation selector is *do not use this metadata when this is the case*.

Usage

The property selector is defined with an instance of NegationSelector.

PropertyDisplayInfo's Properties

Font Color	∅
important	□
- origin	
	∅

General

<i>Id</i>	7348
Conflict Priority	0.00
Inheritance	enabled
Name	Customer Description
- Description	
	∅
- Selector	NegationSelector (5)

Clicking on the NegationSelector instance will open it up, allowing you to define its parameters. In this case there is only one definable property: **Operand**

Information Models Query / SalesModel#1.0 / Entity Types / PropertyDisplayInfo (7348) / Selector: NegationSelector (5)					
NegationSelector	Id 5				
▷ NegationSelector (5)	NegationSelector's Properties				
	<table border="1"> <tr> <td><i>Id</i></td> <td>5</td> </tr> <tr> <td>- Operand</td> <td>∅</td> </tr> </table>	<i>Id</i>	5	- Operand	∅
<i>Id</i>	5				
- Operand	∅				

Double click the Operand property to chose a metadata selector, this can be either a discriminator, a use case or a role selector.

Values

- NegationSelector (5) - NegationSelector
- RoleSelector (2) - RoleSelector
- RoleSelector (3) - RoleSelector
- RoleSelector (4) - RoleSelector
- UseCaseSelector (6) - UseCaseSelector

Types

- BooleanPropertyDiscriminator
- ConjunctionSelector
- DatePropertyDiscriminator
- DisjunctionSelector
- IntegerPropertyDiscriminator
- JunctionSelector
- LongDiscriminatorValue
- MetaDataSelector
- NegationSelector
- NullPropertyDiscriminator
- RoleSelector
- SimplePropertyDiscriminator
- StringPropertyDiscriminator
- StringRegexPropertyDiscriminator
- UseCaseSelector

Warning: You can also choose a further NegationSelector instance. However, since this would negate a negation it would have no effect at all in tribefire.

Define the selector that you have chosen as normal and click **save** to persist your changes. In this instance a UseCaseSelector with the use case property has been selected. This means that the metadata will have no effect on the properties panel, but should function like normal in the other components.

Information Models Query	SalesModel#1.0	Entity Types	PropertyDisplayInfo (7348)	Selector: NegationSelector (5)	Operand: UseCaseSelector ()				
NegationSelector	↳ NegationSelector (5)				<table border="1"> <tr> <td>Id</td> <td>5</td> </tr> <tr> <td>Operand: UseCaseSelector (7)</td> <td></td> </tr> </table>	Id	5	Operand: UseCaseSelector (7)	
Id	5								
Operand: UseCaseSelector (7)									
UseCaseSelector's Properties									
<table border="1"> <tr> <td>/Id</td> <td>7</td> </tr> <tr> <td>Use Case</td> <td>property</td> </tr> </table>						/Id	7	Use Case	property
/Id	7								
Use Case	property								

The negation selector has been properly defined. Its functionality can be seen in Explorer Mode. As can be seen in the screenshot below, the metadata continues to function in the assembly panel but not in the properties panel.

Customer Query

Customer Description			
	Name	Id	
Customer (1)	Looney Tunes	Acme Corp.	1
Dune	CHOAM	2	
Futurama	MomCorp	3	
Richie Rich	Rich Industries	4	
Hitchhiker's Guide	Sirius Cybernetic	5	
Monty Python	Very Big Corp. of	6	
Soylent Green	Soylent Corp.	7	

Customer (1) is highlighted with a red box around the "Customer Description" header.

Customer's Properties

Id	description
1	Looney Tunes
	Name
	- Departments
	Ø
	- Documents

The "description" column header is highlighted with a red box.

Role Selector

Table of Contents

- ▼ Expand / collapse
 - RoleSelector
 - Usage

RoleSelector

The RoleSelector allows you to define which metadata should be used depending a specific role. Roles are created and assigned to users using Authentication and Authorization. Once you have created a set of users and roles, you can then use RoleSelector to define the metadata's behavior.

Usage

To configure the role selector for tribefire Control Center, you add a new instance of RoleSelector...

The screenshot shows the tribefire Control Center interface with the following details:

- Top Bar:** Icons for Home, Find, Refresh, and a folder labeled "MetaDataSelector Query".
- Search Bar:** A text input field with placeholder text "Type for filtering types and values...".
- Values Section:** A list of objects:
 - RoleSelector (2) - RoleSelector
 - RoleSelector (3) - RoleSelector
 - UseCaseSelector (1) - UseCaseSelector
- Types Section:** A list of selector types:
 - BooleanPropertyDiscriminator
 - ConjunctionSelector
 - DatePropertyDiscriminator
 - DisjunctionSelector
 - IntegerPropertyDiscriminator
 - JunctionSelector
 - LongDiscriminatorValue
 - MetaDataAdapter
 - NegationSelector
 - NullPropertyDiscriminator
 - RoleSelector** (highlighted with a yellow background)
 - SimplePropertyDiscriminator
 - StringPropertyDiscriminator
 - StringRegexPropertyDiscriminator
 - UseCaseSelector

...to the Selector property of your metadata. Click on the RoleSelector instance to open it

SelectiveInformation's Properties

important	<input type="checkbox"/>
Template	\${name}
- origin	
	Ø

General

<i>Id</i>	7399
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	
	Ø
- Selector	
	RoleSelector ()

Collapse the RoleSelector by clicking the collapse icon and then select the property **Roles**. Click Add

RoleSelector

- RoleSelector (4)
- Roles Ø

	Id
	4

Add **Refresh** **Hide Details** **List View** **Thumbnail View**

Enter a role which matches one of the roles defined in Authentication and Authorization...

MetaDataSelector Query

john.smith.role.a

Values

string - john.smith.role.a

Role's Properties

<i>Id</i>	john.smith.role.a
Description	Ø
Localized Name	Ø
Name	john.smith.role.a

...and click Add and Finish.

Show Details Add **Add and Finish** Cancel

This means that this metadata will only be used by users who have the assigned role.

Welcome
John Smith

Company Query

Company Query

Results per page: 25 ▾ Switch to Advanced

Company	Appointments	Description	Name	Id	Company's Properties						
BrainTribe	∅	∅	BrainTribe	1	<table border="1"> <tr> <td><i>Id</i></td> <td>1</td> </tr> <tr> <td>Appointments</td> <td>∅</td> </tr> <tr> <td>Description</td> <td>∅</td> </tr> </table>	<i>Id</i>	1	Appointments	∅	Description	∅
<i>Id</i>	1										
Appointments	∅										
Description	∅										

Otherwise, the metadata will not be used.

Welcome
Cortex

Company Query

Company Query

Results per page: 25 ▾ Switch to Advanced

Company	Appointments	Description	Name	Id	Company's Properties		
Company (1)	∅	∅	BrainTribe	1	<table border="1"> <tr> <td><i>Id</i></td> <td>1</td> </tr> </table>	<i>Id</i>	1
<i>Id</i>	1						

Use Case Selector

Table of Contents

Expand / collapse

- UseCaseSelector
 - Gima
 - Selection
 - Thumbnail Panel
 - Assembly Panel
 - Property Panel
 - QuickAccess
 - AutomaticQueries

UseCaseSelector

The UsecaseSelector allows you to define where an action or event should take place in tribefire. There are several options available which effect the functionality of a particular component. You configure the UseCaseSelector by entering the name of the component you wish to be effected. There are seven components that make up the construction of tribefire GME: [Gima](#), [Selection](#), [Thumbnail](#), [Assembly](#), [Property](#), [QuickAccess](#) and [AutomaticQueries](#).

Gima

The [Gima](#) component, which stands for Guided Instantiation and Manipulation Assistant, is the UI which is displayed when you which to create a new instance of an entity, or also when you edit that instance. It allows you to define or change the entities properties.

The screenshot shows the Gima interface with the title "Attendee 0". Below it, under "Attendee's Properties", there is a table with four rows:

emailAddress	Ø
firstName	Ø
jobTitle	Ø
secondName	Ø

To configure the UseCaseSelector for this component you enter **gima** in to the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	4
Use Case	gima

Selection

The selection component, also known as the Selection Constellation, is the UI that is used when adding new or existing elements to properties. It allows you to select a existing instance or create a new one based on an entity type. The selection constellation also allows you different views that can be used.

You can choose between the different views by clicking on one of the buttons displayed at the top of the Selection Constellation UI.



Icon	Description
	Displays the home screen in Selection Constellation.

	Display the Quick Access component in Selection Constellation, allowing you to choose between an existing instance of an entity or creation of a new one based on an entity type.
	Displays the changes screen in the Selection Constellation. This shows all changes that have been made to tribefire which have yet to be persisted.
	Displays the Clipboard screen in the Selection Constellation.
	Displays the Workbench in the Selection Constellation.
EntityType Query	Displays a query based on the type required. That is, it displays all entities of a particular type required for selection.

The Selection Constellation with Quick Access view selected.

The screenshot shows the Selection Constellation interface with the 'Quick Access' view selected. The top navigation bar includes icons for Home, Quick Access (highlighted in orange), Changes, Clipboard, and Workbench. A search bar below the navigation bar contains the placeholder 'Type for filtering types and values...'. Below the search bar, there are two sections: 'Values' and 'Types'. The 'Values' section lists 'Attendee (2) - Attendee' and 'Attendee (6) - Attendee'. The 'Types' section lists 'Attendee'.

The Selection Constellation with Query view selected.

The screenshot shows the Selection Constellation interface with the 'Query' view selected. The top navigation bar includes icons for Home, Quick Access, Changes, Clipboard, and Workbench. A search bar below the navigation bar contains the placeholder 'Type for filtering types and values...'. Below the search bar, there is a table titled 'Attendee Query' showing a list of attendees. The table has columns: Attendee, emailAddress, firstName, id, jobTitle, and secondName. Two rows are visible: 'Attendee (2)' with email john.smith@email.com, first name John, id 2, job title Technical Writer, and second name Smith; and 'Attendee (6)' with email Lisa.jones@email.com, first name Lisa, id 6, job title Developer, and second name Jones.

To configure the UseCaseSelector for the Selection Constellation, you enter **selection** into the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	4
Use Case	selection

Thumbnail Panel

The thumbnail UI refers to the sections of tribefire which displays icons as opposed to details after selecting thumbnail view.

To configure the UseCaseSelector for the thumbnail panel, you enter **thumbnail** into the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	6
Use Case	thumbnail

Assembly Panel

The assembly panel refers to the main section of the tribefire. This is where the results of any query executed are displayed.

The screenshot shows the Assembly Panel interface. At the top, there is a search bar with a magnifying glass icon and a dropdown menu labeled "EntityType Query". Below the search bar are navigation icons for back, forward, and search, along with a "Results per page: 25" dropdown and a "Switch to Advanced" link. The main area of the panel displays the message "No items to display". At the bottom, there is a toolbar with four buttons: "Create New" (plus sign), "Hide Details" (arrow pointing right), "List View" (grid icon), and "Thumbnail View" (grid icon).

To configure the UseCaseSelector for the Assembly panel, you enter **assembly** into the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	8
Use Case	assembly

Property Panel

The property panel refers to the panel, displayed at the right-hand side of tribefire GME, which shows all of the currently selected entity. You can display or hide this panel by selecting the Hide/Show Details button found at the bottom of tribefire. The property panel will display different information depending on the entity type. You can also define Groups and assign properties to them to order how the properties are displayed in the property panel.

GmEntityType's Properties

<i>Id</i>	392
– Type Signature	
com.braintribe.model.meta.GmProperty	
Is Abstract	<input type="checkbox"/>
Is Marked For Discard	<input type="checkbox"/> <input checked="" type="checkbox"/>
Is Plain	<input checked="" type="checkbox"/>
– Artifact Binding	
BindingArtifact (15944)	
– Meta Data	
• EntityIcon (1942)	
– Properties	
1. entityType	
2. id	
3. isId	
4. ...	
– propertyMetaData	
<input type="checkbox"/>	
– Super Types	
1. GenericEntity	

To configure the UseCaseSelector for the Property panel, you enter **property** into the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	8
Use Case	property

QuickAccess

The QuickAccess component refers to the panel which is displayed at the left-hand side of the screen. It displays the individual instances of entity and the entity types of the model.



Values

[Address \(4\) - Address](#)
[Attendee \(6\) - Attendee](#)
[Company \(5\) - Company](#)
[Training \(3\) - Training](#)
[Training \(8\) - Training](#)

Types

[Address](#)
[Attendee](#)
[Company](#)
[EntityType](#)
[StandardIdentifiable](#)
[Training](#)

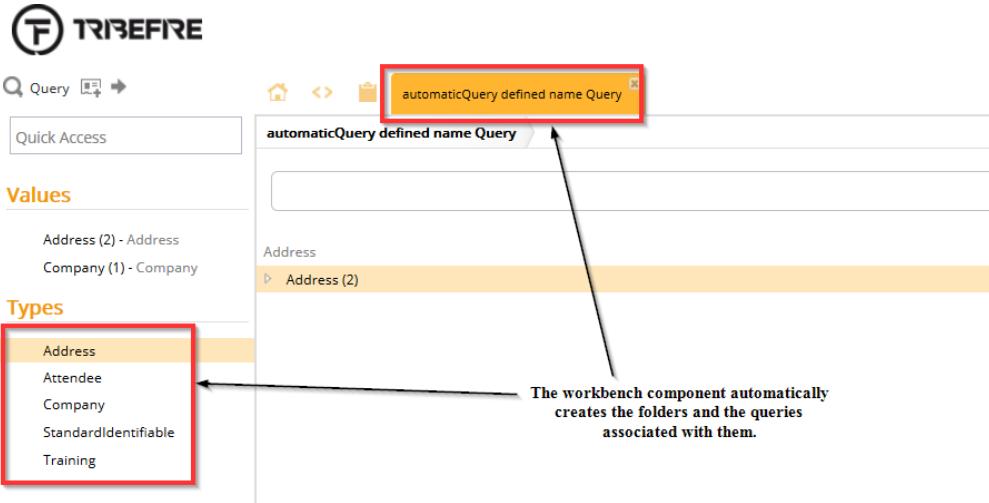
To configure the UseCaseSelector for the QuickAccess component, you enter **quickAccess** into the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	8
Use Case	<code>quickAccess</code>

AutomaticQueries

The automaticQueries use case refers to the Workbench component. It is used to control how the Workbench component should function if it has not yet been configured. That is, if no workbench has been deployed, or a deployed Workbench has no folders defined. If this is the case, tribefire will automatically create folders, which are then displayed in the left-hand panel, and their associated queries. Using the automaticQueries use case, you can define certain metadata on the entities that will be displayed, such as display info, querying allowed and so on.



To configure the UseCaseSelector for the AutomaticQueries component, you enter automaticQueries into the Use Case property.

UseCaseSelector's Properties

<i>Id</i>	8
Use Case	automaticQueries

Model Metadata

Child pages

▼ Expand / collapse

- Model Display Info
- Model Icon

Model Metadata

Each model can be enriched with various metadata properties that can affect the behavior or display of the model. There are eight metadata properties relating to the model.

ModellIcon

ModelDisplayInfo

Model Display Info

Metadata Property Name

ModelDisplayInfo

Description

This metadata allows you to configure how the model should be **labelled**. By using the **Name** property of this metadata, you can determine how the model's name will be displayed.

Usage

You can enter a value into the **Name** property of this metadata. This value will then be used when labelling a model.

ModelDisplayInfo's Properties

Font Color	<input type="button" value="∅"/>
- origin	
	<input type="button" value="∅"/>

General

<i>Id</i>	<input type="button" value="∅"/>
Conflict Priority	0.00
Inheritance	enabled
Name	<input type="button" value="∅"/>
- Description	
	<input type="button" value="∅"/>
- Selector	
	<input type="button" value="∅"/>

Model Icon

Metadata Property Name

Modellcon

Description

This metadata allows you to configure a custom icon for your model.

Usage

To add an icon to a model, you attached an image to the **Icon** property.

Modelcon's Properties

-  icon

∅

- origin

∅

General

<i>Id</i>	7099
Conflict Priority	0.00
Inheritance	enabled
Name	∅
- Description	
	∅
- Selector	
	∅

Entity Type Metadata

Child pages
▼ Expand / collapse
<ul style="list-style-type: none"> • Database Mapping • Default Sort • Entity Compound Viewing 2 • Entity Condensation • Entity Emphasis • Entity Icon • Entity Instantiation Disabled • Entity Priority • Entity Simplification • Entity Type Display Info • Entity Visibility • Entry Deletion • Group Priority • Querying Allowed • Selective Information

Entity Type Metadata

Each entity type can be enriched with metadata which, depending on the metadata property that is configured, can either affect the behavior or the display of the entity type. You can either manually configure individual metadata or use the auto enriching function to automatically to create it.

<i>Compound Unique Key Constraint</i>	Database Mapping	Default Sort	Entity Compound Viewing	Entity Compound Viewing 2
Entity Condensation	Entity Deletion	Entity Emphasis	Entity Icon	Entity Instantiation Disabled
<i>Entity Priority</i>	Entity Simplification	<i>Entity Single Instance</i>	Entity Type Display Info	Entity Visibility
<i>Free Entity Instantiation</i>	Group Priority	Querying Allowed	Selective Information	

Database Mapping

Metadata Property Name

Database Mapping

Description

The Database Mapping metadata is used to **map** an entity type to a **database** table. The access, which is used by this specific model, will define the connection to the database. Although you can create and configure this metadata property, it is created automatically when you create a model from a database schema.

You can configure this metadata property by entering the exact table name found in the database in the **Table Name** property of the metadata. You must also ensure that you have a valid connection defined in your access, so that tribefire is able to connect to the database and query its tables.

Usage

the name of the table found the database must be entered to the Table Name of the Database Mapping metadata property.

General

<i>Id</i>	7551
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

Database Mapping's Properties

Table Name	Ø
– origin	
Ø	
– Xml Snippet	
Ø	

Default Sort

Metadata Property Name

DefaultSort

Description

You can use this metadata to configure which **column** should be **sorted by default**. Without this metadata, the column that is sorted by default is the ID column. However, you can configure Default Sort so that a different column is sorted. To configure this metadata you must chose which **property** you wish to use and whether the sorting order should be **ascending** or **descending**. You can do this by adding the property to the Property field and setting the Direction property to either ascending or descending.

Usage

You can configure this metadata by defining which property should be used for default sort and which direction the results should be sorted in.

In the example below, the property being used is birthDate and the direction ascending (that is, from oldest to youngest)

General

<i>Id</i>	7102
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

DefaultSort's Properties

Direction	ascending
important	<input checked="" type="checkbox"/>
– origin	
Ø	
–  Property	
birthDate	

When displayed in Explorer Mode, you will see that the *Ordered by* option is set to the default sort value, and that the results have been sorted from oldest date to newest.

Person Query

Person	birthDate	description	facebook	firstName	googlePlus	id	lastName	position
▶ Person (5)	05/03/1955 00:00	More than 30 years	Craig.Orman	Craig	Craig.Orman	5	Orman	CSO
▶ Person (19)	03/24/1962 00:00	Ø	Clark.Mild	Clark	Clark.Mild	19	Mild	Marketing Analyst
▶ Person (3)	10/03/1963 00:00	An engineering gr...	Tony.Stark	Anthony	Tony.Stark	3	Stark	CEO
▶ Person (4)	05/05/1965 00:00	Ø	Ryan.Bolt	Ryan	Ryan.Bolt	4	Bolt	Senior Analyst
▶ Person (11)	02/07/1969 00:00	Efficient Administ...	Mich.Silva	Michael	Mich.Silva	1	Silva	CMO
▶ Person (33)	06/20/1969 00:00	Jorge Luz has ove...	Jorge.Luz	Jorge	Jorge.Luz	33	Luz	Decision Maker
▶ Person (31)	11/22/1969 00:00	Nathan Mint is th...	Nate.Mint	Nathan	Nate.Mint	31	Mint	CIO
▶ Person (10)	01/04/1970 00:00	Ø	Maurice.Smith	Maurice	Maurice.Smith	10	Smith	Sales Manager
▶ Person (16)	01/12/1970 00:00	Ø	John.White	John	John.White	16	White	Direct Marketing
▶ Person (21)	01/17/1970 00:00	Ø	David.Claude	David	David.Claude	21	Claude	Marketing Manager
▶ Person (8)	05/02/1970 00:00	Ø	Martin.Cruz	Martin	Martin.Cruz	8	Cruz	Finance Manager
▶ Person (17)	05/04/1970 00:00	Ø	Cloe.Morales	Cloe	Cloe.Morales	17	Morales	Marketing Analyst
▶ Person (18)	05/10/1970 00:00	Ø	Luana.Gold	Luana	Luana.Gold	18	Gold	Direct Marketing
▶ Person (11)	05/01/1972 00:00	Ø	Mark.Gross	Mark	Mark.Gross	11	Gross	Media Director
▶ Person (28)	10/17/1972 00:00	I am a Sales Direc...	Richard.Roy	Richard	Richard.Roy	28	Roy	Sales Director
▶ Person (7)	12/04/1972 00:00	Ø	John.Altman	John	John.Altman	7	Altman	Conflicts Attorney
▶ Person (24)	05/12/1973 00:00	Ø	George.Kent	George	George.Kent	24	Kent	Sales Account Ex...
▶ Person (30)	09/12/1974 00:00	Prior to helping fo...	Lidya.Oliver	Lidya	Lidya.Oliver	30	Oliver	Head of Projects
▶ Person (2)	05/03/1975 00:00	With more than 10...	Suzy.Laura	Suzy	Suzy.Laura	2	Laura	CFO
▶ Person (32)	05/08/1975 00:00	A recognized profe...	Victor.Red	Victor	Victor.Red	32	Red	Purchasing Mana...
▶ Person (6)	01/08/1977 00:00	Ø	Laura.Star	Laura	Laura.Star	6	Star	Advertising Mana...
▶ Person (12)	03/09/1977 00:00	Ø	Peter.McRed	Peter	Peter.McRed	12	McRed	Pre-Sales Consult...
▶ Person (22)	05/02/1978 00:00	Ø	Lorraine.Plant	Lorraine	Lorraine.Plant	22	Plant	Department Man...
		-						

Entity Compound Viewing 2

Metadata Property Name

EntityCompoundViewing2

Description

This metadata allows you to configure an entity so that any types which are in themselves entities can be combined in Explorer Mode. This means that if the property is of another entity type, you can use this metadata to display properties from this second entity in the results of the main entity. For example if you have a property called person, which is of an entity type Person, in a Company entity, you can then display the Person entities properties when displaying the results of Company.

To configure this metadata, you must use a PropertyPath entity to define the link between the two properties. This is done by creating a PropertyPath entity in the Property Path field of EntityCompoundViewing2. The PropertyPath entity has a property called Properties. Here you must add two properties. The first is the property from the main entity, which is where the compounded entities will be displayed. The next step is to add a property belonging to the second entity, that is, the entity will additionally be displayed in the main entity.

You must create a new instance of EntityCompoundViewing2 for each property that you wish to display.

The Prevent Group boolean is used to configure whether the added properties should be grouped in a section named after the compounded property, or whether they should be displayed alongside the other properties in the property panel.

Usage

You can configure this metadata by creating a new EntityCompoundViewing2 instance and adding a new PropertyPath instance in the Property Path field.

General

<i>Id</i>	7106
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

EntityCompoundViewing2's Properties

important	<input checked="" type="checkbox"/>
Prevent Group	<input checked="" type="checkbox"/>
– origin	Ø
– Property Path	PropertyPath (12)

Then you add two properties to PropertyPath Properties field. The first property belongs to the entity to which this entity is attached. It should be a complex type, or a type which is another entity. The second property should belong to this second entity.

In the example below there is an entity called Company. Company has a property salesDocument, which is of the type SalesDocument. SalesDocument has various properties, including one called title. This has been added to PropertyPath.

PropertyPath's Properties

Id 12

- Properties

1. salesDocument
2. title

You must create a EntityCompoundViewing2 instance for every property you wish to display. In this example, another two properties have been added, meaning that in Explorer mode, three extra properties will be displayed.

In the screenshot below, you can see that company now also displays three properties from SalesDocument.

The screenshot shows the Entity Explorer interface. A Company record is selected, displaying its properties: id (1), name (BrainTribe), appointments (empty), and description (empty). To the right of the main properties, a secondary panel titled "salesDocument" is expanded, showing its own properties: description (empty), preview (empty), and title (Valuation - Any Madder).

Prevent Group

When you compound properties for viewing in an entity, they are displayed in the assembly panel and the properties panel. You can decide how they are **displayed** in the **properties panel** by setting the value of Prevent Group to either **true** or **false**. If set to false, all compounded properties will be grouped together, under a section named after the compounded property. If set to false, all properties will be displayed in only one section, regardless of which entity the property belongs to.

Each individual instance of EntityCompoundViewing2 must be configured. If you have three instances and only configure one property with Prevent Group to true, the other two will still be grouped.

False

If you set each instance of the EntityCompoundViewing2's Prevent Group property to false, each property will be grouped together.

salesDocument

description	∅
title	Valuation - Any Madder
– preview	
	valuation_common_V2.jpg

Company's Properties

appointments	∅
description	∅
<i>id</i>	1
name	BrainTribe
– customers	
•	
• Customer (13)	
• Customer (2)	
• ...	
– documents	
∅	

True

If you set each instance of the EntityCompoundViewing2' Prevent Group property to true, these properties will not be grouped, instead displayed together will all other properties.

Company's Properties

appointments	∅
description	∅
description	∅
<i>id</i>	1
name	BrainTribe
title	Valuation - Any Madder
- customers	
	<ul style="list-style-type: none">• Customer (23)• Customer (1)• ...
- documents	
	∅
- keyfacts	
	<ul style="list-style-type: none">• Keyfact (21)• Keyfact (24)• Keyfact (22)• ...
- preview	
	valuation_common_V2.jpg
- profilePicture	
	btlogo_black.png

Entity Condensation

Metadata Property Name

EntityCondensation

Description

This metadata allows you to configure how a property within an entity should behave when collapsed. This refers specifically to properties which are complex types, as these are the only properties that can have further properties. Configuring this metadata allows you to only display the top level of the property. There are three modes of functionality available: Optional, Auto and Forced.

Function	Description
Optional	In explorer mode this allows you to choose between modes, done through buttons displayed in tribefire.
Auto	The condensation is done automatically.
Forced	This forces this metadata to condense the entity to show only the top layer.

You can configure this metadata by choosing one method of functionality from a drop-down list and defining on which property this metadata should act.



You must refresh your browser after saving changes before they will take effect.

Usage

There are two options that you need to configure for this metadata to function correctly. You must choose the mode of operation from the drop-down list and also select which property this metadata should function on.

General

<i>Id</i>	∅
Conflict Priority	0.00
Inheritance	enabled
Name	
- Description	∅
- Selector	∅

EntityCondensation's Properties

Condensation Mode	<input type="button" value="optional"/> <input type="button" value="auto"/> <input type="button" value="forced"/>
important	optional
- origin	auto
∅	forced
- ⚙ Property	∅

Optional

When this metadata functions in optional mode, two buttons will appear in Explorer Mode, allowing you to choose between different modes of functionality.

In the example below, optional mode has been selected and will function on a property called delegates, which belongs to an entity called Person.

EntityCondensation's Properties

Condensation Mode	<input checked="" type="checkbox"/> optional
important	<input type="checkbox"/>
- origin	∅
-  Property	
	delegates

There are two buttons displayed Show Details Here and Show/Hide Details. This allows you to decide between whether the person instance should show only the top level or also the delegates also.

Show Everything

Show Only delegates

Show Details Here
Show/Hide Details

Details Hidden

Person	birthDate	description	facebook	firstName	googlePlus	id	lastName	position	skype
Person (5)	05/03/1955 00:00	More than 30 years	Craig.Orman	Craig	Craig.Orman	5	Orman	CSO	Craig.Orma
Person (19)	03/24/1962 00:00	∅	Clark.Mild	Clark	Clark.Mild	19	Mild	Marketing Analyst	Clark.Mild

Details Show

Person (5)	05/03/1955 00:00	More than 30 years	Craig.Orman	Craig	Craig.Orman	5	Orman	CSO	Craig.Orma
▶ delegates (9) ▶ image: 4.png									

Forced

After selecting this mode and then the property that it should function on, Explorer Mode will only display the top layer of this hierarchy.

In the example below, the mode forced has been selected and will function on the property delegates, which belongs to a property called entity.

EntityCondensation's Properties

Condensation Mode **forced**

important

- origin



- **Property**

delegates

Only the top layer of the hierarchy will be display in Explorer Mode.

▲ Person (5)	05/03/1955 00:00	More than 30 years	Craig.Orman	Craig	Craig.Orman	5	Orman	CSO	Craig.Orma
▷ Person (23)	01/01/1982 00:00	∅	Suzana.Turner	Suzana	Suzana.Turner	23	Turner	Pre-Sales Consult	Suzana.Tur
▷ Person (24)	05/12/1973 00:00	∅	George.Kent	George	George.Kent	24	Kent	Sales Account Exe	George.Ker
▷ Person (10)	01/04/1970 00:00	∅	Maurice.Smith	Maurice	Maurice.Smith	10	Smith	Sales Manager	Maurice.Sn
▷ Person (22)	05/02/1978 00:00	∅	Lorraine.Plant	Lorraine	Lorraine.Plant	22	Plant	Department Man	Lorraine.Pla
▷ Person (25)	02/28/1990 00:00	∅	Paul.Wayne	Paul	Paul.Wayne	25	Wayne	Pre-Sales Consult	Paul.Wayne
▷ Person (26)	08/04/1986 00:00	∅	Alex.Souza	Alex	Alex.Souza	26	Souza	Regional Sales	Alex.Souza
▷ Person (27)	03/11/1987 00:00	∅	Josh.Cross	Josh	Josh.Cross	27	Cross	Assistant Sales M	Josh.Cross
▷ Person (15)	03/22/1989 00:00	∅	Marcio.Ocho	Marcio	Marcio.Ocho	15	Ocho	Pre-Sales Consult	Marcio.Och
▷ Person (12)	03/09/1977 00:00	∅	Peter.McRed	Peter	Peter.McRed	12	McRed	Pre-Sales Consult	Peter.McRe

Entity Emphasis

Metadata Property Name

EntryEmphasis

Description

This metadata property allows you to configure the emphasis of each instance when displayed in Explorer Mode. That state can be configured by the setting of the boolean value to either true or false, which can be done via the **Emphasized** property of the metadata. If emphasized each instance will be displayed in bold.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the entity type is emphasized. If the checkbox is not checked, instances of this entity type will be displayed as normal text.

Usage

Checked

General

<i>Id</i>	7323
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

EntityEmphasis's Properties

<i>Emphasized</i>	<input checked="" type="checkbox"/>
– origin	Ø
–	Ø

instances of the entity type will be displayed emphasized, that is displayed in bold.

		Attendee	emailAddress	First Name	id	jobTitle	secondName
»	Attendee (1)	DavidWilliams@g...	David	1	Technical Writer	Williams	
»	Attendee (2)	DavidWilliams2@o...	David	2	Developer	Williams	
»	Attendee (5)	RobertSmith@em...	Robert	5	Developer	Smith	
»	Attendee (8)	LisaJohnson@bra...	Lisa	8	Project Manager	Johnson	
»	Attendee (9)	LouisStephens@t...	Louis	9	Project Support	Stephens	
»	Attendee (11)	RogerWilson@br...	Roger	11	IT Support	Wilson	
»	Attendee (12)	MaryWaterhouse...	Mary	12	Administration	Waterhouse	
»	Attendee (13)	DavidShaw@bra...	David	13	IT Support	Shaw	

Unchecked

General

<i>Id</i>	7323
Conflict Priority	0.00
Inheritance	enabled
Name	Ø

– Description

Ø

– Selector

Ø

EntityEmphasis's Properties

Emphasized	<input type="checkbox"/>
------------	--------------------------

– origin

Ø

instances will not be emphasized, instead displayed in normal text.

Attendee Query

Attendee	emailAddress	First Name	id	jobTitle	secondName
↳ Attendee (1)	DavidWilliams@g...	David	1	Technical Writer	Williams
↳ Attendee (2)	DavidWilliams2@...	David	2	Developer	Williams
↳ Attendee (5)	RobertSmith@em...	Robert	5	Developer	Smith
↳ Attendee (8)	LisaJohnson@bra...	Lisa	8	Project Manager	Johnson
↳ Attendee (9)	LouisStephens@t...	Louis	9	Project Support	Stephens
↳ Attendee (11)	RogerWilson@br...	Roger	11	IT Support	Wilson
↳ Attendee (12)	MaryWaterhouse...	Mary	12	Administration	Waterhouse

Entity Icon

Metadata Property Name

EntityIcon

Description

This metadata property allows you to add an icon next to each instance of the entity type. The icon can be attached by adding an image to the **Icon** property of the metadata. You must first [upload the image to tribefire](#) before you can use it as an icon. Once the image has been uploaded, you can then attach by double clicking on the Icon property.

Usage

an icon is added to the Icon property

General

Id	7324
Conflict Priority	0.00
Inheritance	disabled
Name	Ø
– Description	Ø
– Selector	Ø

EntityIcon's Properties

– Icon

Boss.png

– origin

Ø

each instance of the entity type will then have an icon

Attendee Query

Attendee	emailAddress	First Name
▷ Attendee (1)	DavidWilliams@g	David
▷ Attendee (2)	DavidWilliams2@	David
▷ Attendee (5)	RobertSmith@em	Robert
▷ Attendee (8)	LisaJohnson@bra	Lisa
▷ Attendee (9)	LouisStephens@t	Louis
▷ Attendee (11)	RogerWillson@br	Roger
▷ Attendee (12)	MaryWaterhousei	Mary

Entity Instantiation Disabled

Metadata Property Name

EntityInstantiationDisabled

Description

This metadata property allows you to configure the instantiation permissions of this entity type. This state can be configured by setting the boolean value to either true or false, which can be done via the **Instantiation Disabled** property of the metadata. If the box is checked, then this instantiation of entity type will be disabled, that is, you won't be able to create a new instance of this entity.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that instantiation is disabled. If the checkbox is not checked, this entity can be instanced.

There is a further property which can be configured in this metadata: Included In Entity Type. This allows you to decide whether the option to create a new instance of this entity should be available from the selection list. That is the window which appears when selecting a value for a property. It is generally divided into two types: Instances and Type. Normally, you can choose either a preexisting instance or select the type to create a new one. Checking the **Included In Entity Type** box means that the option to create a new instance will no longer be available.

 After configuring this metadata, you must restart your tribefire host for the changes to take effect.

Usage

There are two main configurable properties for this metadata: Instantiation Disabled and Included In Entity Type. Check the Instantiation Disabled box to disable instantiation of this Entity.

General

<i>Id</i>	7101
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

EntityInstantiationDisabled's Properties

important	<input checked="" type="checkbox"/>
Included In Entity Type	<input checked="" type="checkbox"/>
Instantiation Disabled	<input checked="" type="checkbox"/>
Unsatisfied Message	Ø
– origin	Ø

Entity Priority

Metadata Property Name

EntityPriority

Description

This metadata property allows you to configure the **priority** of the entity it is configured for. What this means is that you can configure different entity types using this metadata, meaning that you have an order of importance between the different entities within a model. However, you use this metadata to define this ordering.

To configure this metadata you can enter a decimal value between 0 and 1. The higher the value the higher the priority.

Usage

You attach a EntityPriority to each of the entities you wish to prioritize. If there are entities within a model that do not have this metadata set they will default to 0. To set a priority for an entity, you enter a value between 0 and 1 in the **Priority** field.

General

<i>Id</i>	∅
Conflict Priority	0.00
Inheritance	enabled
Name	∅
– Description	∅
– Selector	∅

EntityPriority's Properties

important	<input checked="" type="checkbox"/>
Priority	0.00
– origin	∅

Entity Simplification

Metadata Property Name

EntitySimplification

Description

This metadata property allows you to configure how the properties of the entity type will be **handled**. The metadata property refers specifically to complex types, that is **properties** which are themselves **entity types**. This state can be configured by setting the boolean value to either true or false, which can be done via the **Simplify** property of the metadata. If the box is checked, any properties of this entity which are of a complex type will be simplified. This means that they will be displayed as a simple type in Expert Mode.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that properties which are of a complex type belonging to the entity type will be simplified. If the checkbox is not checked, no then the complex property types remain complex.

Usage

Checked

General

<i>/d</i>	7326
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

EntitySimplification's Properties

Simplify	<input checked="" type="checkbox"/>
– origin	Ø

any properties which are of a complex type will be handled as a simple type

Attendee	company	emailAddress	First Name	<i>/d</i>	jobTitle	secondName
► Attendee (1)	Company (7)	DavidWilliams@g	David	1	Technical Writer	Williams
► Attendee (2)	Ø	DavidWilliams2@o	David	2	Developer	Williams
► Attendee (5)	Ø	RobertSmith@em	Robert	5	Developer	Smith
► Attendee (8)	Ø	LisaJohnson@bra	Lisa	8	Project Manager	Johnson
► Attendee (9)	Ø	LouisStephens@t	Louis	9	Project Support	Stephens
► Attendee (11)	Ø	RogerWillson@br	Roger	11	IT Support	Wilson
► Attendee (12)	Ø	MaryWaterhouse@	Mary	12	Administration	Waterhouse
► Attendee (13)	Ø	DavidShaw@brai	David	13	IT Support	Shaw

Attendee's Properties	
company	Company (7)
emailAddress	DavidWilliams@gmail.com
First Name	David
<i>/d</i>	1
jobTitle	Technical Writer
secondName	Williams
– address	Ø

Unchecked

General

<i>Id</i>	7326
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

EntitySimplification's Properties

Simplify	<input type="checkbox"/>
– origin	
Ø	

any properties which are of a complex type will be handled as a complex type

Attendee	emailAddress	First Name	<i>id</i>	jobTitle	secondName
► Attendee (1)	DavidWilliams@gmail.com	David	1	Technical Writer	Williams
► Attendee (2)	DavidWilliams2@gmail.com	David	2	Developer	Williams
► Attendee (5)	RobertSmith@email.com	Robert	5	Developer	Smith
► Attendee (8)	LisaJohnson@braintribe.com	Lisa	8	Project Manager	Johnson
► Attendee (9)	LouisStephens@braintribe.co	Louis	9	Project Support	Stephens
► Attendee (11)	RogerWillson@braintribe.com	Roger	11	IT Support	Wilson
► Attendee (12)	MaryWaterhouse@braintribe.c	Mary	12	Administration	Waterhouse
► Attendee (13) ▼	DavidShaw@braintribe.com	David	13	IT Support	Shaw

Attendee's Properties

emailAddress	DavidWilliams@gmail.com
First Name	David
<i>id</i>	1
jobTitle	Technical Writer
secondName	Williams
– address	Ø
– company	Company (7)

Entity Type Display Info

Metadata Property Name

EntityTypeDisplayInfo

Description

This metadata property allows you to configure how each instance of the entity type will be displayed. The value of the **Name** property of the metadata determines how each instance will be **labeled** in Expert Mode. This value can be any valid string.

Usage

No configuration

If you attach this metadata property to an entity type but do not configure it, the entity type will be labeled according to its name.

General

<i>Id</i>	∅
Conflict Priority	0.00
Inheritance	enabled
Name	∅
– Description	∅
– Selector	∅

EntityTypeDisplayInfo's Properties

Font Color	∅
– origin	∅

In Expert Mode the name of the entity type will be used as the label for each instance.

Attendee Query

Q ← 1 → Order

Attendee
▷ Attendee (1)
▷ Attendee (2)
▷ Attendee (5)
▷ Attendee (8)
▷ Attendee (9)

Configured

You can use the **Name** property of the metadata to determine how each instance of the entity type will be labeled.

General

<i>Id</i>	7329
Conflict Priority	0.00
Inheritance	enabled
Name	Attending
– Description	
∅	
– Selector	
∅	

EntityTypeDisplayInfo's Properties

Font Color	∅
– origin	
∅	

The value of the **Name** property will be used as the label for each entity type in Expert Mode.

Attending

- ▷ Attending (1)
- ▷ Attending (2)
- ▷ Attending (5)
- ▷ Attending (8)
- ▷ Attending (9)

Entity Visibility

Metadata Property Name

EntityVisibility

Description

This metadata property allows you to configure the visibility of this entity type. This state can be configured by setting the boolean value to either true or false, which can be done via the **Visible** property of the metadata. If the box is checked, then this entity type will be hidden and not available for searching in Expert Mode.

 This use of this metadata property does not preclude the option of configuring queries in Workbench for this entity type. They will still be available for use in Expert Mode and it is only the entity type which will be hidden in Expert Mode, not any queries based upon it.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the entity type is invisible. If the checkbox is not checked, this entity type will be visible.

 After configuring this metadata property, you must ensure that you refresh your browser after saving your settings. Otherwise, this metadata property will have no effect.

Usage

Checked

General

<i>/d</i>	7331
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
	Ø
– Selector	
	Ø

EntityVisibility's Properties

Visible	<input checked="" type="checkbox"/>
– origin	

When checked this entity type will be available and visible in Data View. In this example an entity type called Address has been configured with this metadata property.

a	
---	--

Values

- [Attendee \(15\) - Attendee](#)
- [Attendee \(5\) - Attendee](#)
- [Attendee \(8\) - Attendee](#)
- [Attendee \(9\) - Attendee](#)
- [Training \(3\) - Training](#)

Types

- | |
|----------------|
| Address |
|----------------|
- [Attendee](#)
 - [Company](#)
 - [StandardIdentifiable](#)
 - [Training](#)

Unchecked

General

<i>Id</i>	7331
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

EntityVisibility's Properties

<i>Visible</i>	<input type="checkbox"/>
– origin	Ø

If the *Visible* property of this metadata is not checked, the entity type will not be visible in Data View. In this example an entity type called Address has been configured with this metadata property. As shown in the screenshot below, Address is not visible.

a

— —

Values

Attendee (15) - Attendee

Attendee (5) - Attendee

Attendee (8) - Attendee

Attendee (9) - Attendee

Training (3) - Training

Types

Attendee

Company

StandardIdentifiable

Training

Entry Deletion

Metadata Property Name

EntryDeletion

Description

This metadata property allows you to configure the deletable state of this entity type. This state can be configured by setting the boolean value to either true or false, which can be done via the **Deletable** property of the metadata. If the box is not checked, then no instances of this entity type can be deleted.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the entity type is deletable. If the checkbox is not checked, no instances of this entity type can be deleted.

Usage

Checked

General

<i>/d</i>	7322
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

EntityDeletion's Properties

Deletable	<input checked="" type="checkbox"/>
Unsatisfied Message	Ø
– origin	Ø

You are allowed to delete instances.

The screenshot shows a list of attendees in a table. The first item, "Attendee (1)", is highlighted with a yellow background. A context menu is open over this item, with the "Delete" option highlighted and a cursor pointing at it. The menu also includes options like Open, Edit, Refresh, Hide Details, Add To Clipboard, List View, and Thumbnail View.

	emailAddress	First Name	id	jobTitle	secondName
▼ DavidWilliams@g...	David	1	Technical Writer	Williams	
▼ DavidWilliams2@...	David	2	Developer	Williams	
▼ RobertSmith@em...	Robert	5	Developer	Smith	
▼ LisaJohnson@bra...	Lisa	8	Project Manager	Johnson	
▼ LouisStephens@t...	Louis	9	Project Support	Stephens	
▼ RogerWilson@br...	Roger	11	IT Support	Wilson	
▼ MaryWaterhouse@...	Mary	12	Administration	Waterhouse	
▼ DavidShaw@bra...	David	13	IT Support	Shaw	

Unchecked

General

<i>Id</i>	7322
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

EntityDeletion's Properties

Deletable	<input checked="" type="checkbox"/>
Unsatisfied Message	Ø
– origin	
Ø	

You will not be allowed to delete instances.

Attendee Query

Attendee	emailAddress	First Name	id	Job Title	secondName
Attendee (1)	DavidWilliams@g	David	1	Technical Writer	Williams
Attendee (2)	DavidWilliams2@o	David	2	Developer	Williams
Attendee (5)	RobertSmith@em	Robert	5	Developer	Smith
Attendee (8)	Lisajohnson@bra	Lisa	8	Project Manager	Johnson
Attendee (9)	LouisStephens@t	Louis	9	Project Support	Stephens
Attendee (11)	RogerWilson@br	Roger	11	IT Support	Wilson
Attendee (12)	MaryWaterhouse@	Mary	12	Administration	Waterhouse
Attendee (13)	DavidShaw@brail	David	13	IT Support	Shaw

Attendee (2) context menu:

- Open
- Edit
- Refresh
- Hide Details
- Add To Clipboard
- List View
- Thumbnail View

Group Priority

Metadata Property Name

GroupPriority

Description

This metadata allows you to **order** any **groups** that are displayed in an instance of this entity. The first step when using this metadata is to assign the entity's properties into different groups, using the **Group Assignment** metadata. This will then display all properties in their logical groups. You can then use this metadata to **assign** a different priority to each **group**, therefore ordering each according to the assigned value. The value should be a decimal number between 0 and 1, where 1 will display a group at the top of a list and 0 at the bottom. The different groups will be ordered according to this value, **highest to lowest**. To properly use this metadata, each group that is used within an entity, should each be assigned a Group Priority metadata. For example, if an entity's properties were to be assigned between three groups, you would have to assign a Group Priority metadata to each of these three groups.

Usage

After assigning the entity's properties into different groups, you must then configure a Group Priority for each of these groups.

In the example below, there are three different groups used in the entity Person: General, Social Media and Profile Picture.

GroupPriority's Properties

important	<input type="checkbox"/>
Priority	0.40
– Group	
Profile Picture	
– origin	∅

Each group is then assigned a different value. The table below shows that General should be ordered first, as it has the highest priority value, then Social Media, and Profile Picture.

Group	Priority
General	1
Social Media	0.5
Profile Picture	0.4

This ordering is then reflected in Expert Mode.

General

birthDate	04/07/1992 11:55
description	Ø
firstName	John
<i>id</i>	1
lastName	Smith
position	Technical Writer

- delegates

Ø

Social Media

facebook	johnsmith
googlePlus	+johnsmith
skype	john.smith
twitter	@johnsmith

Profile Picture

- image

Ø

Querying Allowed

Metadata Property Name

QueryingAllowed

Description

This metadata property allows you to configure whether this entity type can be queried or not. This state can be configured by setting the boolean value to either true or false, which can be done via the **Allowed** property of the metadata. If the box is not checked, then this entity type cannot be queried.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the entity type can be queried. If the checkbox is not checked, the entity cannot be queried.

Usage

After attaching this metadata to an entity, you can use the **Allowed** box to define its state.

QueryingAllowed's Properties

Allowed	<input checked="" type="checkbox"/>
important	<input checked="" type="checkbox"/>
- origin	
	∅

General

<i>Id</i>	7086
Conflict Priority	0.00
Inheritance	enabled
Name	∅
- Description	∅
- Selector	∅

If you do not check the **Allowed** box, this entity can not be queried. For example, in Explorer Mode, if you click on the entity type which you have attached this metadata to, it will not execute a query.

In the example below, the entity Person has been configured so that it cannot be queried. You can search for the entity type, however, when clicking on the entity Person, no query will be executed.

person

Values

[Keyfact \(14\) - Keyfact](#)

[Keyfact \(15\) - Keyfact](#)

[Keyfact \(18\) - Keyfact](#)

[Keyfact \(20\) - Keyfact](#)

[Person \(3\) - Person](#)

Types

Person



Selective Information

Metadata Property Name

SelectiveInformation

Description

This metadata allows you to configure what is **displayed** in each instance of an entity type in Expert Mode. Without any configuration, the name of the entity type will be displayed along with the instance's ID in brackets. You can **configure** this metadata property by entering a string sequence in the **Template** field. This string sequence will then form the basis for what is displayed at each instance of the entity.

Syntax

You can configure the Selective Information metadata by following the correct syntax. If you wish to display a property you enter the following

```
 ${ PROPERTY_NAME }
```

This will display the value of that defined property for each instance

You can also enter any valid string, as well as being able to mix and match

```
A STRING VALUE = ${ PROPERTY_NAME }
A STRING VALUE = ${ PROPERTY_NAME }, ANOTHER STRING ${ ANOTHER_PROPERTY }
```



The value of the property can be null, however, the property itself must exist. If you reference a property that is not part of the entity type, or misspell the property name, a runtime error will occur in tribefire.

Usage

Once you have attached a Selective Information metadata to your entity, you can then enter a value, by following the syntax from above, into the Template field.

SelectiveInformation's Properties

Template	Name : \${name}
- origin	
Ø	

General

<i>Id</i>	7548
Conflict Priority	1.00
Inheritance	enabled
Name	Ø
- Description	
Ø	
- Selector	
Ø	

This will be displayed in Data View, instead of the Instance name and ID brackets.

Customer	Description	Name	Id
▶ Name : Acme Corp.	Looney Tunes	Acme Corp.	1
▶ Name : CHOAM	Dune	CHOAM	2
▶ Name : MomCorp	Futurama	MomCorp	3
▶ Name : Rich Industries	Richie Rich	Rich Industries	4
▶ Name : Sirius Cybernetics Corp.	Hitchhiker's Guide	Sirius Cybernetics	5
▶ Name : Very Big Corp. of America	Monty Python	Very Big Corp. of /	6
▶ Name : Soylent Corp.	Soylent Green	Soylent Corp.	7

Property Metadata

Child pages
<p>▼ Expand / collapse</p> <ul style="list-style-type: none"> • Bidirectional Property • Collection Element Count Constraint • Excel Column Mapping • Excel Identity Management Property • Excel Reference Column Mapping • Group Assignment • Mandatory Property • On Change • Password Property • Property Bulleting • Property Display Info • Property Editable • Property Empty String • Property Icon • Property Mapping • Property Priority • Property Shares Entities • Property Simplification • Property Visibility • Range Boundary • String Regexp Constraint • Unique Key Constraint

Property Metadata

Each property can be enriched with metadata which, depending on the metadata property that is configured, can either affect the behavior or the display of the property. You can either manually configure individual metadata or use the auto enriching function to automatically create it.

Bidirectional Property	<i>Cascading Delete</i>	Collection Element Count Constraint	Excel Column Mapping	Excel Identity Management Property
Excel Reference Column Mapping	<i>Formatting Metadata</i>	<i>Fraction Digits Formatting</i>	Group Assignment	Mandatory Property
<i>Map As List Property</i>	<i>Max Length</i>	On Change	Password Property	Property Bulleting
<i>Property Create Entities</i>	<i>Property Display Info</i>	Property Editable	<i>Property Emphasis</i>	Property Empty String
Property Icon	Property Mapping	Property Priority	Property Shares Entities	Property Simplification
Property Visibility	Range Boundary	String Regexp Constraint	<i>Total Digits Formatting</i>	Unique Key Constraint
<i>Whitespace Formatting</i>				

Bidirectional Property

Metadata Property Name

BidirectionalProperty

Description

This metadata allows you to create a **reciprocal** relationship between **two properties**, meaning that when one property is defined with a value, the **linked** property will be **filled** with the **corresponding** value. This allows you to create relationships between two different properties, either within the same entity type or across different entities. This means that when you define a value for one property, the linked property will also be updated. Basically, this metadata tells tribefire that when the configured property has been given a value, this value should also be added to the linked property.

An example of this would be a relationship between Employees and Company. If you correctly configure an Employee and define the company they work for, the corresponding employees property in Company will also be automatically updated.



You must configure this metadata on each property that you wish to be affected. In the above example, you must configure both the company property of Employee and the employee property in Company with the Bidirectional metadata. If you don't it will only function in one direction.

Usage

To configure this metadata, you define the **Linked Property** field with the appropriate property that should be linked. This means that any value defined in this property will also be defined in the linked property.

General

<i>Id</i>	7099
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

BidirectionalProperty's Properties

important	<input checked="" type="checkbox"/>
– Linked Property	Ø
– origin	Ø

In the example below, there are two entity types: Attendee and Company. The Attendee entity has various properties, including company. In the Company entity there is a corresponding property Employee. The company property of Attendee is configured with the Bidirectional metadata and the linked property Employee is given.

BidirectionalProperty's Properties

important	<input type="checkbox"/>
-  Linked Property	
employee	
- origin	
	∅

An Attendee is created and a company assigned.

Attendee's Properties

emailAddress	john.smith@braintribe.com
firstName	John
<i>id</i>	3
jobTitle	Technical Writer
secondName	Smith
- address	
	∅
- company	
	Company (1)

This Company's employee property is automatically updated with the Attendee.

Company's Properties

companyName	Braintribe
field	IT
<i>id</i>	1
- employee	
	• Attendee (3)
- subsidiary	
	∅



If you wish this relationship to work in the other direction, you must also defined the employee property of Company with a corresponding Bidirectional property.

Collection Element Count Constraint

Metadata Property Name

CollectionElementCountConstraint

Description

This metadata property allows you to place a **constraint** on the amount of elements a property that is of a collection type can contain. There are two definable variables which you can use to limit your collection: **Min Count** and **Max Count**. Min Count allows you to define the minimum amount of elements, whereas the Max Count configures the maximum amount allowed in this properties collections. Both variables should be configured with a valid integer.

Usage

After adding this metadata to a property, you can set the Min and Max Count values.

General

<i>/d</i>	7312
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

CollectionElementCountConstraint's Properties

Max Count	3
Min Count	1
Unsatisfied Message	
– origin	Ø

In the example above, the minimum amount of elements in a collection must be one and the maximum amount allowed is three.

Customer's Properties

description	Ø
<i>id</i>	Ø
name	Braintribe
– departments	Ø
– documents	Ø
– forecasts	Ø
– invoices	Ø
– keyfacts	Ø
– logo	Ø
– notes	Ø
– opportunities	Ø
– salesDocuments	Ø

As a result, if you attempt to save a property with less than one or more than three, you will receive a validation error.

Customer **Customer** (New)

The following condition is not met at the property **opportunities**:

the element count of opportunities must be greater or equal to 1 and lesser or equal to 3

Excel Column Mapping

Metadata Property Name

ExcelColumnMapping

Description

You can import data from an Excel spreadsheet by defining columns in a worksheet that are related to properties in tribefire. You associate the properties with the columns in Excel by defining metadata on each property that data should be imported to. There are two types of metadata that are used to map properties and columns: ExcelColumnMapping and ExcelReferenceColumnMapping.

The ExcelColumnMapping metadata is used to map simple types. To configure this metadata you define the Column Name property with the name of the corresponding column in Excel.

Usage

This metadata only has one property that you needs configuration.

ExcelColumnMapping's Properties

Column Name

The value given should correspond to the name of the column in your Excel spreadsheet.

companyName	field	subsidiary
Braintribe	IT	100
TechSphere	IT	101
tribefire	Software Development	102

In the screen shot above, there are three columns. They correspond to the properties found in a Company entity.

companyName
 Meta Data (1)
 ▶ Excel Column Mapping: companyName

For the property companyName (which belongs to the Company entity) we attach the ExcelColumnMapping metadata and insert the value companyName (the name define in the Excel spreadsheet).

General

<i>Id</i>	7126
Conflict Priority	0.00
important	<input type="checkbox"/>
Inheritance	enabled
Name	Ø
- Description	Ø
- origin	Ø
- Selector	Ø

ExcelColumnMapping's Properties

Column Name	companyName
-------------	-------------

Excel Identity Management Property

Metadata Property Name

ExcelIdentityManagementProperty

Description

You can import data from an Excel spreadsheet by defining columns in a worksheet that are related to properties in tribefire. The Excel Identity Management Property allows you to identify a unique element in each entity that data will be imported to. This functions similarly to a key in a database or the [Unique Key Constraint](#) and when configured, the property to which this metadata has been attached cannot hold duplicate values.

When importing data to simple property types, this metadata is not required. However, it is required when importing data to complex types. If a property is of a complex type (that is, the type is itself an entity type) then the entity which it represents must have a ExcelIdentityManagementProperty defined. This is so that tribefire can use it to identify the correct instance of the element it represents.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is presented by a checkbox, which when checked means that the property is unique. If the checkbox is not checked, the property is not unique.

For example, if you have a entity with a property called company, which is a complex type representing an entity Company, you must configure this metadata on the Company entity. This allows tribefire to either assign the correct company instance to the property or, if there is no existing instance representing the company defined, it will create a new one and attach it to the company property.



If you import data and there are two duplicate values that represent the property attached with this metadata, the last of these values will be used. The other two will not be stored.

Usage

Configure

The metadata only has one property which requires configuration: Identification Property. This is a boolean value that can be set to either true or false. If you check this box, it sets the value to true, and means that this property will only import unique values. However, if you uncheck this option, the metadata functionality will be turned off and tribefire will be able to import duplicate values.

ExcelIdentityManagementProperty's Properties

Identification Property

Unique Values

The ExcelIdentityManagementProperty functions similar to a unique key in a database. When this metadata is assigned to a property, it cannot hold duplicate values.

In the example below, the entity Attendee has a property socialSecurityNumber which has been given this metadata.

GmProperty's Properties

<i>Id</i>	4251
<i>Is Id</i>	<input type="checkbox"/>
<i>isOverlay</i>	<input type="checkbox"/>
<i>Name</i>	<code>socialSecurityNumber</code>

- Entity Type

Attendee

- Meta Data

- Excel Column Mapping: Social Security Number
- Excel Mapping Identification Column: Yes

- Type

`string`

The data that will be imported has two social security numbers which are the same.

First Name	Last Name	Address	Company	Email Address	Job Title	Social Security Number
Robert	Jones	103	Braintripe	Robert.Jones@braintripe.com	Developer	665-5665-AT
Lisa	Steiner	104	tribefire	Lisa.Steiner@tribefire.com	Project Manager	998-9999-DE
David	Weller	105	TechSphere	David.Weller@TechSphere.com	Technical Writer	489-9887-TO
Maria	Collins	106	tribefire	Maria.Collins@tribefire.com	Accountant	339-4916-DE
Paula	Hill	107	TechSphere	Paula.Hill@TechSphere.com	Developer	789-8973-RT
James	Osborne	108	Braintripe	James.Osborne@braintripe.com	Administrator	263-6631-AT
Stephen	MacGarry	109	tribefire	Stephen.MacGarry@tribefire.com	Developer	554-5668-PR
June	Bauer	110	TechSphere	June.Bauer@TechSphere.com	Project Manager	467-8250-RT
Penelope	O'Donnell	111	Braintripe	Penelope.O'Donnell@braintripe.com	Technical Writer	654-9876-AT
Peter	Hardy		Braintripe	Peter.Hardy@braintripe.com	Project Manager	467-8250-RT

In this scenario, only the last value will be imported.

Attendee	emailAddress	firstName	id	jobTitle	secondName	socialSecurityN...
▶ Attendee (7)	Robert.Jones@braintripe.com	Robert	7	Developer	Jones	665-5665-AT
▶ Attendee (8)	Lisa.Steiner@tribefire.com	Penelope	8	Technical Writer	O'Donnell	654-9876-AT
▶ Attendee (9)	Paula.Hill@TechSphere.com	Paula	9	Developer	Hill	789-8973-RT
▶ Attendee (10)	Lisa.Steiner@tribefire.com	Lisa	10	Project Manager	Steiner	998-9999-DE
▶ Attendee (12)	Maria.Collins@tribefire.com	Maria	12	Accountant	Collins	339-4916-DE
▶ Attendee (13)	James.Osborne@braintripe.com	James	13	Administrator	Osborne	263-6631-AT
▶ Attendee (14)	Peter.Hardy@braintripe.com	Peter	14	Project Manager	Hardy	467-8250-RT
▶ Attendee (16)	Stephen.MacGarry@tribefire.com	Stephen	16	Developer	MacGarry	554-5668-PR
▶ Attendee (17)	David.Weller@TechSphere.com	David	17	Technical Writer	Weller	489-9887-TO

Reference for Complex Types

In addition to this metadata functioning like a unique key, it is also used in conjunction with the metadata ExcelReferenceColumnMapping to import data related to complex types. If there is a property in your entity type which is complex, then you require a ExcelIdentityManagementProperty configured on the entity type the property represents. This allows tribefire to identify and match the data to an instance of this type and define that property. If an instance doesn't exist then a new one will be created for the property.

In the example below, the entity type Attendee has the property company.

Attendee

- Properties (7)
 - firstName
 - secondName
 - jobTitle
 - emailAddress
 - company**
 - Meta Data (1)
 - address
 - socialSecurityNumber

This property is complex and represents the entity type Company.

GmProperty's Properties

<i>Id</i>	4278
Is Id	<input type="checkbox"/>
isOverlay	<input type="checkbox"/>
Name	company

- Entity Type

- Attendee
- Meta Data
 - Excel Column Mapping: Company
- Type
 - Company

Therefore, we must attach an instance of the ExcelIdentityManagementProperty metadata to a unique value in the entity type Company. In this case, the companyName will be assigned the metadata.

companyName

- Meta Data (2)
 - Excel Column Mapping: companyName
 - Excel Mapping Identification Column: Yes**

A column company has been defined in the worksheet that will be imported.

First Name	Last Name	Address	Company	Email Address	Job Title	Social Security Number
Robert	Jones	103	Braintripe	Robert.Jones@braintripe.com	Developer	665-5665-AT
Lisa	Steiner	104	tribefire	Lisa.Steiner@tribefire.com	Project Manager	998-9999-DE
David	Weller	105	TechSphere	David.Weller@TechSphere.com	Technical Writer	489-9887-TO
Maria	Collins	106	tribefire	Maria.Collins@tribefire.com	Accountant	339-4916-DE
Paula	Hill	107	TechSphere	Paula.Hill@TechSphere.com	Developer	789-8973-RT
James	Osborne	108	Braintripe	James.Osborne@braintripe.com	Administrator	263-6631-AT
Stephen	MacGarry	109	tribefire	Stephen.MacGarry@tribefire.com	Developer	554-5668-PR
June	Bauer	110	TechSphere	June.Bauer@TechSphere.com	Project Manager	467-8250-RT
Penelope	O'Donnell	111	Braintripe	Penelope.Odonnel@braintripe.com	Technical Writer	654-9876-AT
Peter	Hardy		Braintripe	Peter.Hardy@braintripe.com	Project Manager	467-8250-RT

There is a second Company worksheet that contains three unique company names.

companyName	field	subsidiary
Braintripe	IT	100
TechSphere	IT	101
tribefire	Software Development	102

Through the use of this metadata and the ExcelReferenceColumnMapping this data will be linked when imported.

Attendee's Properties

emailAddress	Lisa.Steiner@tribefire.com
firstName	Lisa
<i>id</i>	10
jobTitle	Project Manager
secondName	Steiner
socialSecurityNumber	998-9999-DE
- address	
Address (25)	
- company	
Company (11)	

Company's Properties

companyName	tribefire
field	Software Development
<i>id</i>	11
– employee	
∅	
– subsidiary	
Address (28)	

Without this metadata, you will receive an error message on the console when attempting an import. This will not stop tribefire importing other data, however, this property will remain null.

```
WARN : No identifcation value found for entity: Company[@938103173,companyName="TechSphere",employee=null,field=null,id=null,subsidiary=null]. Ignore.  
WARN : No identifcation value found for entity: Company[@8363516795,companyName="Braintribe",employee=null,field=null,id=null,subsidiary=null]. Ignore.  
WARN : No identifcation value found for entity: Company[@8961469724,companyName="TechSphere",employee=null,field=null,id=null,subsidiary=null]. Ignore.  
WARN : No identifcation value found for entity: Company[@8623388365,companyName="Braintribe",employee=null,field=null,id=null,subsidiary=null]. Ignore.
```

Excel Reference Column Mapping

Metadata Property Name

ExcelReferenceColumnMapping

Description

You can import data from an Excel spreadsheet by defining columns in a worksheet that are related to properties in tribefire. You associate the properties with the columns in Excel by defining metadata on each property that should data should be imported to. There are two types of metadata that are used to map properties and columns: [ExcelColumnMapping](#) and [ExcelReferenceColumnMapping](#).

The ExcelReferenceColumnMapping metadata is used to map complex types. To configure this property, you enter the name of the corresponding column in Excel and also select the unique property belonging to the entity type it relates to. That is, if the property is of a complex type, it will, therefore, relate to another entity type. This related entity type must have a unique value, as defined by the [ExcelIdentityManagementProperty](#) metadata, which should also be defined in this metadata.

This metadata is attached to the property that the data will be imported to, whereas the [ExcelIdentityManagementProperty](#) is attached to a unique property of the entity that it is related to.

For example, an entity called Attendee has a property called company. This is of a complex type and refers to another entity called Company. When configuring company (the property belonging to Attendee) you enter the corresponding column name defined in Excel and also select a property belonging to the Company entity which is unique, in this example companyName.

Usage

This metadata is used in conjunction with the [ExcelIdentityManagementProperty](#) metadata to import data related to complex types. There are two properties which require configuration: Column Name and referenceProperty. The value of Column Name should correspond to the column in Excel from where the data will be received; the value of referenceProperty should refer to a property of the related entity type to which it refers.

ExcelReferenceColumnMapping's Properties

Column Name	<input type="text" value="∅"/>
-  referenceProperty	<input type="text" value="∅"/>

In the example below, there is an entity called Attendee. It has a property called company and describes which company each instance of an Attendee works for. It is a complex type and refers to another entity called Company.

GmProperty's Properties

<i>Id</i>	4278
<i>Is Id</i>	<input type="checkbox"/>
<i>isOverlay</i>	<input type="checkbox"/>
<i>Name</i>	company
-  Entity Type	Attendee
- Meta Data	
-  Type	Company

The data that is configured in the Excel spreadsheet refers to the company name and is defined by the column named Company.

First Name	Last Name	Address	Company	Email Address	Job Title	Social Security Number
Robert	Jones	103	Braintripe	Robert.Jones@braintripe.com	Developer	665-5665-AT
Lisa	Steiner	104	tribefire	Lisa.Steiner@tribefire.com	Project Manager	998-9999-DE
David	Weller	105	TechSphere	David.Weller@TechSphere.com	Technical Writer	489-9887-TO
Maria	Collins	106	tribefire	Maria.Collins@tribefire.com	Accountant	339-4916-DE
Paula	Hill	107	TechSphere	Paula.Hill@TechSphere.com	Developer	789-8973-RT
James	Osborne	108	Braintripe	James.Osborne@braintripe.com	Administrator	263-6631-AT
Stephen	MacGarry	109	tribefire	Stephen.MacGarry@tribefire.com	Developer	554-5668-PR
June	Bauer	110	TechSphere	June.Bauer@TechSphere.com	Project Manager	467-8250-RT
Penelope	O'Donnell	111	Braintripe	Penelope.Odonnel@braintripe.com	Technical Writer	654-9876-AT
Peter	Hardy		Braintripe	Peter.Hardy@braintripe.com	Project Manager	467-8250-RT

Therefore, we configure this metadata with its two properties.

ExcelReferenceColumnMapping's Properties

```

Column Name          Company
- ⚡ referenceProperty
    companyName

```

Company because that is what the column in Excel is called and comanyName because this is the property in the entity Company which matches this data.

```

Company
  Properties (4)
    ⚡ companyName
    ⚡ field
    ⚡ employee
    ⚡ subsidiary

```

This property is also defined as the unique identifier, by configuring the `ExcelIdentityManagementProperty` here. Without this property, the `ExcelReferenceColumnMapping` metadata will not function and no data will be imported.

```

WARN : No identification value found for entity: Company[0938103173,companyName="TechSphere",employee=null,field=null,id=null,subsidiary=null]. Ignore.
WARN : No identification value found for entity: Company[0363516795,companyName="Braintripe",employee=null,field=null,id=null,subsidiary=null]. Ignore.
WARN : No identifcation value found for entity: Company[0961469724,companyName="TechSphere",employee=null,field=null,id=null,subsidiary=null]. Ignore.
WARN : No identifcation value found for entity: Company[0623388365,companyName="Braintripe",employee=null,field=null,id=null,subsidiary=null]. Ignore.

```

However, when both these metadata have been correctly configured, the data will be imported and an correct instance of Company will be defined for the Attendee.

Attendee's Properties

emailAddress	Lisa.Steiner@tribefire.com
firstName	Lisa
<i>id</i>	10
jobTitle	Project Manager
secondName	Steiner
socialSecurityNumber	998-9999-DE

– address

Address (25)

– company

Company (11)

Company's Properties

companyName	tribefire
field	Software Development
<i>id</i>	11

– employee

∅

– subsidiary

Address (28)

Group Assignment

Metadata Property Name

GroupAssignment

Description

This metadata property allows you to assign different **properties** to a **group**, therefore grouping together different properties. They will then be **displayed** in these groups when creating a new **instance** of the entity and also in the **property panel**. To configure this metadata, you first must create **groups** that you wish to use. These groups are not fixed to any property and therefore can be reused in different use cases. For example, a default group called **General** can be used in several different entities. Once you have created your groups then you can use this metadata to assign your properties to it. You must attach this metadata to every property in a specific entity that you wish to be grouped.



The groups, when displayed, will not be ordered. If you wish to order these groups, you must use the Group Priority metadata. This metadata is configured at the entity level. Thus is, you assign this metadata to the entity in which the properties have been grouped.

Usage

Once you have created the groups that will be used, you can use this metadata to assign a property to this group. In the screenshot below the birthDate has been assigned to the **General** group. You must repeat this for all properties that you wish to group.

In the example below, there will be three groups used: General, Social Media and Profile Picture. All properties in the entity Person will be assigned to one of these three groups.

GroupAssignment's Properties

important	<input type="checkbox"/>
- origin	
∅	
- Property Group	
General	

General

<i>Id</i>	7084
Conflict Priority	1.00
Inheritance	enabled
Name	∅
- Description	
∅	
- Selector	
∅	

After assigning all the properties into their logical groups, they will be displayed in Expert Mode under these different groups.

Either when creating a new instance of the entity...

Person 0**General**

birthDate	<input type="text"/>	<input type="button" value="..."/>
description	∅	
firstName	∅	
lastName	∅	
position	∅	

Social Media

facebook	∅
googlePlus	∅
skype	∅
twitter	∅

...or in the properties panel.

General

birthDate	04/07/1992 11:55
description	∅
firstName	John
<i>id</i>	1
lastName	Smith
position	Technical Writer
- delegates	
	∅

Social Media

facebook	johnsmith
googlePlus	+johnsmith
skype	john.smith
twitter	@johnsmith

Profile Picture

- image	
	∅

Mandatory Property

Metadata Description Name

MandatoryProperty

Description

This metadata property allows you to configure whether this property is mandatory or not. This means that if this metadata is configured and the value set to true, then this property must have a value before it can be saved. This property can be configured by using the **Mandatory** property

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that this property must have a value before it can be saved. If the checkbox is not checked, then this property does not require any value and can be saved with a null value.

Usage

When attached, use the Mandatory property to configured the mandatory nature of this property

MandatoryProperty's Properties

Mandatory	<input checked="" type="checkbox"/>
Unsatisfied Message	
– origin	Ø
	Ø

General

<i>Id</i>	7314
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

When creating an new instance, any mandatory properties will be displayed in bold.

The screenshot shows the 'Attendee' entity properties form. At the top, there is a header 'Attendee (0)'. Below it, a section titled 'Attendee's Properties' contains four fields: 'emailAddress', 'First Name', 'jobTitle', and 'secondName'. The 'jobTitle' field is bolded, indicating it is a mandatory field. All other fields have a placeholder 'Ø'.

emailAddress	
First Name	Ø
jobTitle	Ø
secondName	Ø

If you try to save an entity instance without first entering data into all mandatory fields, you will receive a validation error.

Attendee **Attendee** (id=8)

The following condition is not met at the property **jobTitle**:

'jobTitle' must not be empty

On Change

Metadata Property Name

On Change

Description

The On Change metadata allows you to set custom actions that affect another part of tribefire when this property has been changed. You can configure it by three different methods, either through a Cartridge installed on your instance of tribefire, by a beanshell or javascript.

To configure On Change you must attach a processor to the Processor field.

On Change

– Processor

∅

There are two types of processors which can be used: Cartridge State Change Processor or Scripted State Change Processor. As described above, the Cartridge State Change Processor is used when you wish to define the change through a customized cartridge and the Scripted State Change Processor is used when you want to define the change by either a beanshell or a javascript. If you are using a Cartridge you must first install it on the system. However, when using the Scripted State Change Processor, you can enter the script directly into tribefire Control Center.

After attaching a processor, you must then add the specific object that this processor will use:

Cartridge State Change Processor

Standard Identifiable Entity

Id

∅

Deployable

Cartridge State Change Processor's Properties

– genericDependencies

∅

Cartridge Deployable

– Cartridge

∅

You then attach a Cartridge object to the Cartridge field.

Cartridge's Properties

host

∅

Cartridge

Path

∅

Port

∅

Protocol

∅

This is used to define the specific cartridge located in tribefire container.

Or a Scripted State Change Processor:

Standard Identifiable Entity

Id

∅

Scripted State Change Processor's Properties

– genericDependencies

∅

Scripted Processor

– Script

∅

Deployable

You then add a JavaScript...

Scripted Processor

– Script

Javascript ()

Deployable

...or Beanshell element to the Script field.

Scripted Processor

- Script

Beanshell()

Deployable

Deployment State	Ø
Description	Ø
External Id	Ø
Hardwired	<input checked="" type="checkbox"/>
Name	Ø
- Traces	
	Ø

Both these objects are the same, only differing in the language they accept.

Standard Identifiable

<i>Id</i>	Ø
Source	Ø

You must always start the script with the following command:

```
getcontext()
```

There are two main methods you call from the context element:

```
getEntity() or getSession()
```

Password Property

Metadata Property Name

PasswordField

Description

This metadata property allows you to configure how a property should behave. This state can be configured by setting the boolean value to either true or false, which can be done via the **Password** property of the metadata. If the box is checked, then the property will behave like a password field and mask out the property's value.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the property will act like a property. If the checkbox is not checked, the property will behave like a normal property and its values can be seen.

Usage

When attached to a property, you can use to Password field to set the state of this metadata.

PasswordField's Properties

Password



- origin



General

Id

7544

Conflict Priority

0.00

Inheritance

enabled

Name



- Description



- Selector



If checked, then the property will act like a password field and mask all values

Attendee	Attended	AttendeePassword	Date Of Birth	Email Address	First Name	Job Title	Second Name
▶ Attendee (2)	<input checked="" type="checkbox"/>	*****	09/10/1985 10:35	johnsmith@email.com	John	Programmer	Smith
▶ Attendee (5)	<input checked="" type="checkbox"/>	****	∅	maryAnn@email.com	Mary	Support	Ann
▶ Attendee (6)	<input checked="" type="checkbox"/>	****	∅	ThomasHardy@email.com	Thomas	Technical Writer	Hardy
▶ Attendee (7)	<input checked="" type="checkbox"/>	∅	∅	TonyStark@iman.com	Tony	Engineer	Stark

Property Bulleting

Metadata Property Name

PropertyBulleting

Description

This metadata property allows you to configure whether properties of a list type are display with bullet points. This means that if this metadata is configured and the value set to true, then the values of the properties in a list will be displayed with numbers next to each item. This property can be configured by using the **Bulleting** property

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the elements of this property's list will be displayed with numbers. If the checkbox is not checked, then the elements of this property's list will be displayed as normal.

Usage

Unchecked

If you attach the metadata to a property but don't check the Bulleting box, the elements of this property will be displayed normally.

In the screenshot below, the metadata has been attached to the subfolders property of the Folder entity type.

PropertyBulleting's Properties

Bulleting	<input type="checkbox"/>
important	<input type="checkbox"/>
- origin	
Ø	

General

<i>Id</i>	7096
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	
Ø	
- Selector	
Ø	

That means the elements of the subfolder property will be displayed normally.

Modeling

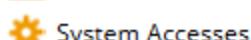


System Models

Virtualization



Integration Accesses



Checked

However, if you check the Bulleting box, then each element of the properties list will be displayed with a number next to it.

In the screenshots below the property subfolder of the entity type Folder has been attached with this metadata.

PropertyBulleting's Properties

Bulleting	<input checked="" type="checkbox"/>
important	<input type="checkbox"/>
- origin	
	Ø

Each element of the subfolder list will then be displayed with a number

- ▲ Smart Enterprise Information
 - ▲ 1. Modeling
 - 1. Information Models
 - 2. System Models
 - ▲ 2. Virtualization
 - 1. Connections
 - 2. Integration Accesses
 - 3. System Accesses

Property Display Info

Metadata Property Name

PropertyDisplayInfo

Description

This metadata property allows you to configure the name of a property. The value of the **Name** property of the metadata determines how the property will be **displayed** in Expert Mode. This value can be any valid string.

Usage

If you have attached this metadata property but not configured the Name field, the property will be displayed according to its name

Attendee	emailAddress	firstName	id	jobTitle	secondName
▶ Attendee (1)	DavidWilliams@gmail.com	David	1	Technical Writer	Williams
▶ Attendee (2)	DavidWilliams2@gmail.com	David	2	Developer	Williams
▶ Attendee (5)	RobertSmith@email.com	Robert	5	Developer	Smith

However, if you enter a value into the name property

PropertyDisplayInfo's Properties

Font Color

– origin

∅

General

*/*Id 7309

Conflict Priority 0.00

Inheritance enabled

Name First Name

– Description

∅

– Selector

∅

This value will be used instead.

Attendee	emailAddress	First Name	id	jobTitle	secondName
▶ Attendee (1)	DavidWilliams@gmail.com	David	1	Technical Writer	Williams
▶ Attendee (2)	DavidWilliams2@gmail.com	David	2	Developer	Williams
▶ Attendee (5)	RobertSmith@email.com	Robert	5	Developer	Smith

Property Editable

Metadata Property Name

PropertyEditable

Description

This metadata property allows you to configure whether this property can be edited or not. This state can be configured by setting the boolean value to either true or false, which can be done via the **Editable** property of the metadata. If the box is checked, then the value of this property can be edited.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the entity type is editable. If the checkbox is not checked, the property can not be edited.

Usage

PropertyEditable's Properties

Editable



- origin



General

<i>Id</i>	7110
-----------	------

Conflict Priority	0.00
-------------------	------

Inheritance	enabled
-------------	---------

Name	editKey
------	---------

- Description

Makes this property editable

- Selector



Property Empty String

Metadata Property Name

PropertyEmptyString

Description

This metadata property allows you to configure whether you can automatically empty a string value. This state can be configured by setting the boolean value to either true or false, which can be done via the **Emptyable** property of the metadata. If the box is checked, the string value of the property can be emptied.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that a string value of this property can be emptied. If the checkbox is not checked, the string value of this property can not be emptied.

Usage

To make it possible to empty a string value of a property, check the Emptyable check-box

PropertyEmptyString's Properties

Emptyable	<input checked="" type="checkbox"/>
Unsatisfied Message	
- origin	
Ø	

General

/d	7306
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	
Ø	
- Selector	
Ø	

You can right-click on a property and the Set empty text option will be displayed.

Attendee	emailAddress	firstName	id	jobTitle	secondName
Attendee (1)	DavidWilliams@g...	David	1	Technical Writer	Williams
Attendee (2)	DavidWilliams2@...	David	2	Developer	Williams
Attendee (5)	RobertSmith@em...	Robert	5	Developer	Smith

A context menu is open over the 'Attendee' column, showing options: 'Clear Value' and 'Set empty text'. The 'Set empty text' option is highlighted with a red box.

Property Icon

Metadata Property Name

PropertyIcon

Description

This metadata property allows you to add an icon next to a property. The icon can be attached by adding an image to the **Icon** property of the metadata. You must first [upload the image to tribefire](#) before you can use it as an icon. Once the image has been uploaded, you can then attach it by double clicking on the Icon property.

Usage

After uploading an icon, you can attach it to the Icon property. This icon will be displayed next to your property in Explorer Mode.

PropertyIcon's Properties

important	
useEntityIcon	
- _ _ Icon	∅
- origin	∅

General

<i>Id</i>	7118
Conflict Priority	0.00
Inheritance	enabled
Name	∅
- Description	∅
- Selector	∅

Property Mapping

Metadata Property Name

PropertyMapping

Description

The PropertyMapping metadata is used to **map** a property to a **column** in a **database** table. The access, which is used by this specific model, will define the connection to the database. Although you can create and configure this metadata property, it is created automatically when you create a model from a database schema.

You can configure this metadata property by entering the exact column name found in the database in the **Column Name** property of the metadata. You must also ensure that you have a valid connection defined in your access, so that tribefire is able to connect to the database and query its tables.

Usage

The column name found in the database is entered into the Column Name property of your metadata.

PropertyMapping's Properties

Auto Assignable	<input type="checkbox"/> <input checked="" type="checkbox"/>
Column Name	<input type="checkbox"/>
Xml	<input type="checkbox"/>
– origin	
–	<input type="checkbox"/>

General

/d	<input type="checkbox"/>
Conflict Priority	0.00
Inheritance	enabled
Name	<input type="checkbox"/>
– Description	
–	<input type="checkbox"/>
– Selector	
–	<input type="checkbox"/>

Property Priority

Metadata Property Name

PropertyPriority

Description

This metadata property allows you to configure the **priority** of the property. What this means is that you can configure different properties of an entity type using this metadata, resulting in a **different ordering** of these properties in Explorer Mode, both when displaying the columns and when creating a new instance of the entity type. Normally, the properties are displayed in the order in which they were created. However, you can use this metadata to change this ordering.

To configure this metadata you can enter a decimal value between 0 and 1. The higher the value the higher the priority and it will be displayed nearer to the top of a new instantiation. Using this metadata on a series of properties of an entity type, therefore, allows you to determine their ordering in Explorer Mode.

Usage

An example of the Property Priority metadata used is found below. In this example there is an entity type called Person. The columns are ordered in Explorer Mode as follows:

Person	ID	Birthplace	Date Of Birth	First Name	Nationality	Second Name
Person (3)	3	Manchester	08/21/1986 00:00	Roberts	British	Smith
Person (4)	4	London	09/24/1984 00:00	Lisa	British	Roberston
Person (5)	5	Vienna	04/25/1979 01:00	Hans	Austrian	Heuber
Person (6)	6	Graz	04/10/1990 00:00	Maria	Austrian	Goesser
Person (7)	7	Detroit	03/16/1992 00:00	Peter	American	Stubbs
Person (8)	8	Sydney	10/15/1982 01:00	Alfred	Australian	Jones
Person (9)	9	Athens	08/16/1982 00:00	Johanas	Greek	Pardo

You can then attach a Property Priority to each property of the entity type Person. The following Property Priority is configured for ID.

PropertyPriority's Properties

Priority	1.00
– origin	
Ø	

General

/d	7561
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

Each property has a Property Priority attached. See the table below for the values used to configure each.

Property Name	Priority Value
ID	1.0

secondName	0.9
firstName	0.8
dateOfBirth	0.7
nationality	0.6
birthplace	0.5

The columns will then be ordered using these priorities in Explorer Mode. As you can see the higher the value, the higher the priority of the property. This means that properties with high values will be displayed to the left and then the following properties will be ordered afterwards, depending on the Priority value.

Person	ID	Second Name	First Name	Date Of Birth	Nationality	Birthplace
Person (3)	3	Smith	Roberts	08/21/1986 00:00	British	Manchester
Person (4)	4	Roberston	Lisa	09/24/1984 00:00	British	London
Person (5)	5	Heuber	Hans	04/25/1979 01:00	Austrian	Vienna
Person (6)	6	Goesser	Maria	04/10/1990 00:00	Austrian	Graz
Person (7)	7	Stubbs	Peter	03/16/1992 00:00	American	Detroit
Person (8)	8	Jones	Alfred	10/15/1982 01:00	Australian	Sydney
Person (9)	9	Pardo	Johanas	08/16/1982 00:00	Greek	Athens

Property Shares Entities

Metadata Property Name

PropertySharesEntities

Description

This metadata property allows you to configure whether you can select a existing entity for this property or only allow a new one to be created . This state can be configured by setting the boolean value to either true or false, which can be done via the **Shares** property of the metadata. If the box is not checked, then you can select preexisting entities.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that a preexisting entity type can be used. If the checkbox is not checked, only new instances can be created.

Usage

Leaving the **Shares** box unchecked, means that only a new instance of an entity type can be created for a property.

PropertySharesEntities's Properties

important	<input checked="" type="checkbox"/>
Shares	<input checked="" type="checkbox"/>
Unsatisfied Message	Ø
- origin	

Ø

General

<i>Id</i>	7117
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	

Ø

- Selector

Ø

In Explorer Mode, you the add button will not appear, only Open, which will only give you the option to create a new instance.

Customer

- ▶ Customer (1)
 - ▶ departments Ø
 - ▶ documents (0)
 - ▶ forecasts Ø
 - ▶ invoices Ø
 - ▶ keyfacts Ø
 - ▶ logo: Ø
 - ▶ notes Ø
 - ▶ opportunities (1)
 - ▶ salesDocuments Ø
- ▶ Customer (2)
- ▶ Customer (3)
- ▶ Customer (4)
- ▶ Customer (5)
- ▶ Customer (6)
- ▶ Customer (7)
- ▶ Customer (8)
- ▶ Customer (9)
- ▶ Customer (10)
- ▶ Customer (11)
- ▶ Customer (12)
- ▶ Customer (13)
- ▶ Customer (14)
- ▶ Customer (15)

description	id	name
Looney Tunes	1	Acme Corp.
Dune	2	CHOAM
Futurama	3	MomCorp
Richie Rich	4	Rich Industries
Hitchhiker's Guid	5	Sirius Cybernetic
Monty Python	6	Very Big Corp. of
Soylent Green	7	Soylent Corp.
Zork	8	Frobozz Magic C
Batman	9	Wayne Enterpris
Bladerunner	10	Tyrell Corp.
Austin Powers	11	Virtucon
Lil' Orphan Annie	12	Warbucks Indust
Resident Evil	13	Umbrella Corp.
The Simpsons	14	Globex
Charlie...Choc. Fa	15	Wonka Industrie

List View
Thumbnail View

Property Simplification

Metadata Property Name

PropertySimplification

Description

This metadata property allows you to configure how the property will be **handled**. The metadata property refers specifically to complex types. This state can be configured by setting the boolean value to either true or false, which can be done via the **Simplify** property of the metadata. If the box is checked, any properties of this entity which are of a complex type will be simplified. This means that they will be displayed as a simple type in Explorer Mode.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that properties which are of a complex type will be simplified. If the checkbox is not checked, no then the complex property types remain complex.

Usage

Unchecked

If you attached this metadata to your property and leave the Simplify field unchecked...

PropertySimplification's Properties

Simplify	<input type="checkbox"/>
– origin	
Ø	

General

<i>Id</i>	7303
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

...then the property will be handled as a complex type.

Company's Properties

companyName	Braintrib
field	IT
<i>id</i>	4
– employee	
<ul style="list-style-type: none"> • Attendee (2) • Attendee (1) 	
– subsidiary	
∅	

Checked

If you attached this metadata to your property and you check the Simplify field...

PropertySimplification's Properties

Simplify	<input checked="" type="checkbox"/>
– origin	
∅	

General

<i>Id</i>	7303
Conflict Priority	0.00
Inheritance	enabled
Name	∅
– Description	
∅	
– Selector	
∅	

...the property will be simplified

Company's Properties

companyName	Braintribes
employee	[braintribes.model.Attendee_gm@aa09, braintribes.mode
field	IT
<i>id</i>	4
– subsidiary	
	Ø

Property Visibility

Metadata Property Name

PropertyVisibility

Description

This metadata property allows you to configure the visibility of this property. This state can be configured by setting the boolean value to either true or false, which can be done via the **Visible** property of the metadata. If the box is checked, then this property will be hidden in Explorer Mode.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is represented by a checkbox, which when checked means that the entity type is invisible. If the checkbox is not checked, this entity type will be visible.

Usage

Checked

If you attach the metadata to your property and check the Visible field...

PropertyVisibility's Properties

Visible	<input checked="" type="checkbox"/>
– origin	
	Ø

General

/d	7299
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
	Ø
– Selector	
	Ø

...then the property will be visible

Training 0	
Training's Properties	
endDate	
startDate	Ø
title	Ø
topic	Ø

Unchecked

If you attach the metadata to your property and do not check the Visible field...

PropertyVisibility's Properties

Visible	<input type="checkbox"/>
- origin	
Ø	

General

/d	7299
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
- Description	
Ø	
- Selector	
Ø	

...then the property will not be visible in Explorer Mode.

Training 0

Training's Properties

endDate	<input type="text"/>	<input type="button" value="..."/>
startDate	Ø	
topic	Ø	

Range Boundary

Metadata Property Name

RangeBoundary

Description

This metadata property allows you to place a **constraint** on a **number**. You can either set the boundary as a maximum or a minimum. This is done by setting the **boundary value** and then, if you wish to set the boundary as a minimum value, checking the minimal check-box. This means that if you try to save a number in a property that is outside this boundary, you will receive a validation error. For example, if you set the range boundary with the value **10**, you will not be able to enter and save a value **higher** than 10. The functionality of minimal is the exactly the same except that you will not be able to save a value **lower** than 10.

Usage

To configure this metadata, enter the value you wish as your boundary in the **boundary** property. This means that you will not be able to save a number higher than this value. If you wish this value to be the minimal value, below which no value can be saved, then check the minimal check box.

RangeBoundary's Properties

boundary	∅
exclusive	<input type="checkbox"/>
important	<input type="checkbox"/>
minimal	<input type="checkbox"/>
– origin	
	∅

General

<i>Id</i>	∅
Conflict Priority	0.00
Inheritance	enabled
Name	∅
– Description	
	∅
– Selector	
	∅

String Regexp Constraint

Metadata Property Name

StringRegexpConstraint

Description

This metadata allows you to configure a **regular expression** on the property, allowing you to apply **constraints** on data entered. You can enter regular expression using the **Expression** field and any string entered into the property will be validated against it. If the result of this validation is false, that is, doesn't match the regular expression, you will not be allowed to enter this string. In addition to this you can add a Message that should be displayed if the string fails to match the regular expression, which you can configure by using the **Unsatisfied Message** field.

Usage

Once you attach this metadata to a property you can add a regular expression in the Expression field, along with a Unsatisfied Message.

StringRegexpConstraint's Properties

Expression	\S+@\S+
Unsatisfied Message	not a valid email address
– origin	
Ø	

General

Id	7301
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	
Ø	
– Selector	
Ø	

The property will use this expression to validate data entered into the property. If this data is not validated, the Unsatisfied Message will be displayed.

The screenshot shows the 'Attendee Query' interface with the following details:

- Attendee Query** tab is active.
- Attendee (5)** is selected in the sidebar.
- Attendee** section in the sidebar:
 - Attendee (1)
 - Attendee (2)
 - Attendee (5)** (highlighted in orange)
 - address Ø
 - company: Ø
- Attendee** table in the main area:

	emailAddress	firstName	id	jobTitle	secondName
DavidWilliams@g...	David	1	Technical Writer	Williams	
DavidWilliams2@...	David	2	Developer	Williams	
NotAnEmail[...]	Robert	5	Developer	Smith	
- Results per page: 25** dropdown.
- Switch to Advanced** link.
- An error message **not a valid email address** is displayed next to the row for Robert Smith.

Unique Key Constraint

Metadata Property Name

UniqueKeyConstraint

Description

This metadata property allows you to configure whether this property should contain unique values. That is, if this property is unique, no two instances of this property can have the same value. This state can be configured by setting the boolean value to either true or false, which can be done via the **Unique** property of the metadata. If the box is checked, then no two instances of this property can have the same value.

There are only two accepted values for this metadata property, either **true** or **false**. This choice is presented by a checkbox, which when checked means that the property is unique. If the checkbox is not checked, the property is not unique.

Usage

If you have attached this metadata to your property and checked the unique box...

UniqueKeyConstraint's Properties

Unique	<input checked="" type="checkbox"/>
Unsatisfied Message	Ø
– origin	Ø

General

/d	7298
Conflict Priority	0.00
Inheritance	enabled
Name	Ø
– Description	Ø
– Selector	Ø

...and you try to enter a value into a property which already exists

Attendee Query					
<input type="text"/> <input type="button" value="Search"/> ← 1 → Ordered by: id ▾ ↑ Results per page: 2					
Attendee	emailAddress	firstName	id	jobTitle	secondName
Attendee (1)	DavidWilliams@gmail.com	David	1	Technical Writer	Williams
Attendee (2)	DavidWilliams@gmail.com	David	2	Developer	Williams

you will receive a validation error.

Attendee **Attendee** (id=2)

The following condition is not met at the property **emailAddress**:

emailAddress: 'DavidWilliams@gmail.com' already exists

Enum Type Metadata

Child pages

▼ Expand / collapse

Enum Type Metadata

Each enum type can be enriched with metadata which, depending on the metadata property that is configured, can either affect the behavior or the display of the enum type. You can either manually configure individual metadata or use the auto enriching function to automatically to create it.

<i>Enum Type Display Info</i>	<i>Enum Type Metadata</i>	<i>Value Display Info</i>	<i>Value Metadata</i>
-------------------------------	---------------------------	---------------------------	-----------------------

Search Panel

Table of Contents

Expand / collapse

- Search Panel
- Basic Mode
 - Usage
- Advanced Mode
 - Numbers
 - Usage
 - Syntax
 - Wild Cards
 - Junctions
 - Ordering

Search Panel

The search panel is displayed at the top of the Assembly Panel. It is used to further filter results shown in the Assembly panel, normally after a Query has been executed. There are two modes of operation: Basic or Advanced. Whereas basic mode allows you to enter a value that will be searched, advanced mode allows you to use a SQL-like language to further refine and control your search.

[Switch to Advanced](#)

[Switch to Basic](#)

You can switch between the two modes of operation by clicking the

or

buttons.



You can execute the search by clicking the magnifying glass .

Basic Mode

Basic mode allows you to enter a value into the text field and tribefire will search for all occurrences of this value. The search is not restricted to a specific property but instead searches every property found within the query.

Option	Description
6	Navigate between the result pages.
Ordered by: Id ▾	Select which column the search results should be ordered by. You can change the ordering column by clicking on this button and then selecting the property of your choice.
	Ordering direction the arrow pointing upwards means the results will be sorted in ascending order (that is, from low to high) and the arrow pointing down means the results will be sorted in descending order (that is, from high to low).
Results per page: 5 ▾	Displays how many results should be displayed on one page. The choices are: 5, 10, 15, 20, 25, 50, 100 and 200.
Switch to Advanced	This switches the search method to advanced mode .

Usage

You can enter a value into the text field and tribefire will find all occurrences of this value in the results displayed.

value		1	Ordered by: Id	Results per:
GmEntityType				
EntityReference	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	6
Today	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	36
TodayMax	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	38
PreliminaryEntityReference	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	54
MapEntryDescriptor	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	55
PersistentEntityReference	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	72

Advanced Mode

The advanced mode allows you to further refine your search results by building a search query. You can query specific properties for values, using specific operators, as well being able to an AND or OR junction. As you enter your search function, tribefire will display possible values in a drop-down list. The search panel also dynamically validates your search result, allowing you to see if what you have entered is correct or not.

		1	Save As	Switch to Basic
--	--	---	---------	---------------------------------

The search works like a normal function where the two operands are balanced between an operator. You start by entering the property you would like to search on, then select the operator (=, >, like and so on) before entering a value that you should be searched for. If you are searching on a string, you must enter the value in quotation marks (""). It is also possible to enter a regular expression that will be matched on.

Operator	Description
=	Equals. This compares two numbers and if they are the same then the result will be displayed.
!=	Does not Equal. This compares two numbers and if they are not the same the result will be displayed.
like	<p>This compares two strings and if they match the results will be displayed.</p> <p>You can also define regular expressions instead a proper string.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> Any value that you enter must be contained within quotation marks, regardless if it is a whole string or a regular expression. </div>
>	Greater than. Compares two numbers, if the number found is greater than the number provided in the search it will be displayed.
>=	Greater than or Equals. Combines the > and the = operators. This means that if the number compared is greater than or equal to the number provided in the search, it will be displayed.
<	Less than. Compares two numbers and if the number found is less than the number provided in the search it will be displayed.
<=	Less then or Equals. Combines the < and the = operators. Compares two numbers and if the number found is less than or equal to the number provided in the search, it will be displayed.

Numbers

When using comparing numbers, you must specific which type of number it is.

Number type	Added Value	Example
Integer	When entering an integer, you do not have to add any additional identifiers.	394

Double	If the number you wish to compare is of the type double, you must add a D at the end of the number	100.00D
Long	If the number you wish to compare is of the type long, you must add a L at the end of the number	30000L

Usage

The advanced search mode has the following syntax:

Syntax

Left Operand Operator Right Operand

The left operand represents the property on which you wish to search, while the right operand represents that value that should be matched. The operator is how the search value should be compared.

- importance > 5

This translates as display all instances whose property **importance** (which is of the type Integer) has a value higher than 5.

Strings use the *like* operator, and the value of the right operand must be placed in quotation marks.

- companyName like "Braintribe"

This translates as display all instances whose property **companyName**(which is of the type String) has the value Braintribe.



While entering your search, it will be dynamically validated, represented by either a or a icon. The tick represents a valid function, whereas a exclamation mark represents an invalid function. If your function is not correct, you will not be able to search.

As you type, tribefire will display a drop-down list, displaying suggestions based on the entered value.

The screenshot shows a search interface with a dropdown menu. The input field contains 'pr'. Below it, a list of suggestions includes 'predictedAmount' and 'probability'. At the bottom of the interface, there are various filters and a search bar.

You can select an item from the list and it will automatically be added, or you can continue to type and ignore the suggestions. The list is context sensitive, meaning that it will display suggestions based on the part of the function you are entering, so that after entering a property, as shown in the screen above, it will then display an operator and so on.

Wild Cards

If you wish to match values that are of the type String, you can either enter a full string value, a whole word or phrase, or you can enter the value using [regular expressions](#), for example, S* would return all strings of a property that begin with the letter S.

The screenshot shows a list of opportunities. A search query 'name like "TF*"' is entered in the search bar. The results show several opportunities, with the first one ('Opportunity (28)') highlighted. The columns include Opportunity, bookedDate, closeDate, description, id, importance, name, opportunityOff..., and predictedAm... . The highlighted row shows details for 'TF - Franchise'.

Junctions

You can search with more than one function by using a junction. A junction allows you to combine more than one function together to create more flexible, or refined, searches. There are two different types of junctions: Conjunction (AND) and Disjunction (OR).

- Conjunction, which means AND, allows you to further refine the search, so that every function that you use must be true for the result to be display.

- Disjunction, which means OR, allows you to write several functions, so that if one of them is found to be true, that result will be displayed.

You use a junction by writing out the first function and then typing either AND or OR before writing another function.

Operand Operator Operand AND Operand Operator Operand

name like "TF*" AND importance > 5 🔍

Opportunity	bookedDate	closeDate	description	id	importance	name	opportunityOff...	predictedAm...
↳ Opportunity (32)	05/01/2012 14:51	04/01/2012 14:51	TF - Chuck Tore	32	9	TF - Chuck Tore	∅	∅
↳ Opportunity (33)	02/01/2012 11:12	02/01/2012 11:12	TF - Narcissus	33	7	TF - Narcissus	∅	∅
↳ Opportunity (17)	07/25/2012 00:01	06/25/2012 00:01	TF - Red Fox	17	6	TF - Red Fox	LICENSE	∅

Operand Operator Operand OR Operand Operator Operand

importance = 6 OR importance = 8 🔍

Opportunity	bookedDate	closeDate	description	id	importance	name	opportunityOff...	predictedAm...
↳ Opportunity (17)	07/25/2012 00:01	06/25/2012 00:01	TF - Red Fox	17	6	TF - Red Fox	LICENSE	∅
↳ Opportunity (13)	∅	11/18/2014 15:09	TF - Valley Seasor	13	8	TF - Valley Seasor	LICENSE	1349400.00

Ordering

You can also order your search on any property, either ascending or descending, by using the command ORDER BY followed by the property name on which the ordering should take place, and finally which direction. If you do not define a direction, it will automatically default to ascending (that is, lowest to highest).

To select the ordering direction, you enter **asc** (ascending) or **desc** (descending).

order by Property Name

importance > 9 order by name 🔍

Opportunity	bookedDate	closeDate	description	id	importance	name	opportunityOff...	predictedAm...
↳ Opportunity (5)	∅	11/22/2013 14:52	∅	5	10	Big data TRAININ	LICENSE	1211000.00
↳ Opportunity (12)	∅	12/10/2014 15:07	TF - Invest Footba	12	10	TF - Invest Footba	LICENSE	1582950.00
↳ Opportunity (35)	∅	12/21/0014 00:08	TF - Pariah Aminc	35	20	TF - Pariah Aminc	LICENSE	3269700.00

order by Property Name asc

importance > 9 order by name **asc** 🔍

Opportunity	bookedDate	closeDate	description	id	importance	name	opportunityOff...	predictedAm...
↳ Opportunity (5)	∅	11/22/2013 14:52	∅	5	10	Big data TRAININ	LICENSE	1211000.00
↳ Opportunity (12)	∅	12/10/2014 15:07	TF - Invest Footba	12	10	TF - Invest Footba	LICENSE	1582950.00
↳ Opportunity (35)	∅	12/21/0014 00:08	TF - Pariah Aminc	35	20	TF - Pariah Aminc	LICENSE	3269700.00

order by Property Name desc

⌚ importance > 9 order by name desc

Opportunity	bookedDate	closeDate	description	id	importance	name	opportunityOff...	predictedAm...
▶ Opportunity (35)	∅	12/21/2014 00:08	TF - Pariah Aminc	35	20	TF - Pariah Aminc	LICENSE	3269700.00
▶ Opportunity (12)	∅	12/10/2014 15:07	TF - Invest Footba	12	10	TF - Invest Footba	LICENSE	1582950.00
▶ Opportunity (5)	∅	11/22/2013 14:52	∅	5	10	Big data TRAININ	LICENSE	1211000.00

tribefire Types

Child pages
<p>▼ Expand / collapse</p> <ul style="list-style-type: none">• BindingArtifact• GmSimpleType• EntityType• EnumType• Model• Property

Introduction

Each model is built with a series of entity types, which themselves have a series of properties. There are many different entity types in tribefire. This documentation aims to give you an overview of the most important types and a description of their properties.

BindingArtifact

Binding Artifact

The type BindingArtifact is used to control the organization of dependencies of the classes that other entities are generated from. If an entity type (and therefore, the class that it has been generated from) has dependencies in a repository, you can use the BindingArtifact to define it. This functions as, and its values map to, the XML found in a POM file.

BindingArtifact's Properties

<i>Id</i>	Ø
Artifact Id	Ø
Group Id	Ø
Version	Ø

Property	Description
ID	An ID that is automatically generated by tribefire the first time you save this entity instance.
Artifact Id	The name of the artifact that should be used.
Group Id	The location of the artifact in a repository.
Version	The version of the artifact that should be used.

GmSimpleType

GmSimpleType

A simple type is a type which do not reference any other elements. They can only hold a simple piece of data, such as Strings, Integers, Booleans, Doubles and so on. If you wish to declare a single type for use in tribefire, you must create and configure the GmSimpleType. There is only one property which is required when defining GmSimpleType: Type Signature. You should write which type you wish to use here. For example, enter string into the Type Signature field if you wish to configure GmSimpleType as a string.

GmSimpleType's Properties

<i>Id</i>	Ø
Type Signature	Ø

Property	Description
ID	An ID that is automatically generated by tribefire the first time you save this entity instance.
Type Signature	The named reference to a particular type that this GmSimpleType should use. See below for a list of possible values.

You can enter any of the following values into Type Signature, depending on the nature of the simple type you wish to use.

Type	Description
Object	An undefined simple variable, it is the base type for all other simple types
String	A piece of text
float	a single-precision 32-bit IEEE 754 floating point
decimal	A base ten decimal number
integer	a number with the minimum value of -2 ³¹ and a maximum of 2 ³¹ -1
long	a number with the minimum value of -2 ⁶³ and a maximum of 2 ⁶³ -1
double	a double-precision 64-bit IEEE 754 floating point
date	a date value. Using this type in tribefire will allow you to use a calendar to select a date
boolean	A type with only two possible values: true or false.

EntityType

Table of Contents

- ▼ Expand / collapse
 - Entity Type Properties
 - GmEntityType's Properties

Entity Type Properties

An entity type has several different types which can be defined and are all labeled under the heading GmEntityType's Properties.

 All properties that are displayed in bold are mandatory. You will not be able to save any changes to a model without first entering information in these fields.

GmEntityType's Properties

Id

∅

– Type Signature

∅

Is Abstract

Is Marked For Discard

∅

Is Plain

∅

– Artifact Binding

∅

– Meta Data

∅

– Properties

∅

– Super Types

∅

GmEntityType's Properties

Property	Type	Description
ID	NA	The internal ID of an entity type, it is automatically generated when you first save the entity type.
Type Signature	String	The type signature of the entity type is used within tribefire to reference this type.
Is Abstract	Boolean	If the value is set to true, then this entity type can not be used directly in a model, and should be used as a super type for another entity. This has the same function as an abstract class in Java.

Is Marked For Discard	Boolean	When exporting a model, or deploying it, and you do not wish this entity to be exported or deployed with it, setting this boolean value to true will filter this entity out of the process.
Is Plain	Boolean	Defines whether this entity type when generated from a java class is a normal class or an interface.
Artifact Binding	BindingArtifact	The binding artifact allows you to control the dependencies of this entity type. That is, when exporting or deploying, a model out of tribefire, for example to a repository, you need to define any dependencies this entity type has, you can map them here. The binding artifact functions similarly to the function of a pom file. Where you must set the Artifact ID (What is the name of the Artifact) and the group ID (where in the repository directory structure this artifact is found). Then you set which version of the artifact should be used.
Meta Data	There are various metadata types that can be used.	This allows you to define various metadata which will enrich your entity type with information.
Properties	GMPropertyType	Each entity type has various properties defined which help to describe the entity type. An example of this could be a name or ID. There are three types of properties, either simple, complex or collection (map or a list).
Super Types	GMEEntityType	Super types refer to the concept of inheritance. An entity which defines a super type will be related to this type by a parent (super type) child (sub type) relationship. An example of this type of relationship would be Employee/Person. Where the Person would be the super type of Employee. The employee would inherit all the characteristics of the person type.

EnumType

Table of Contents

- ▼ Expand / collapse
 - [EnumType](#)

EnumType

An enumeration is a collection of fixed values which are complete, that is no values can be added or removed from this set. This refers to the enumerations when deployed, it is possible, of course to configure your enumerations to add or remove values from it by using the tribefire Control Center. Enumerations are defined through the use of the GmEnumType and all its properties are configured under this heading.

 All properties that are displayed in bold are mandatory. You will not be able to save any changes to a model without first entering information in these fields.

GmEnumType's Properties

<i>Id</i>	∅
Is Marked For Discard	∅ <input checked="" type="checkbox"/>
Type Signature	∅
– Artifact Binding	
	∅
– Constants	
	∅
– Meta Data	
	∅
– Values	
	∅

Property	Type	Description
ID	NA	The internal ID of an entity type, it is automatically generated when you first save the entity type.
Is Marked For Discard	Boolean	When exporting a model, or deploying it, and you do not wish this entity to be exported or deployed with it, setting this boolean value to true will filter this entity out of the process.
Type Signature	String	The type signature is used to refer to the enum type from within tribefire. For example, when querying or for manipulations.

Artifact Binding	BindingArtifact	The binding artifact allows you to control the dependencies of this entity type. That is, when exporting or deploying, a model out of tribefire, for example to a repository, you need to define any dependencies this entity type has, you can map them here. The binding artifact functions similarly to the function of a pom file. Where you must set the Artifact ID (What is the name of the Artifact) and the group ID (where in the repository directory structure this artifact is found). Then you set which version of the artifact should be used.
Constants	GmEnumConstant	A constant which represents a value (see below) of the enum type.
Meta Data	There are various metadata types which you can be defined	This allows you to define various metadata which will enrich your enum type with information.
Values	String	The values that are contained in the enum type set.

Model

- ▼ Expand / collapse
 - Model
 - General Properties
 - Model's Properties

Model

The model is at the heart of tribefire, everything you see is based on a model of one sort or another. Each model has various standard properties, which are defined here.

 All properties that are displayed in bold are mandatory. You will not be able to save any changes to a model without first entering information in these fields.

General

Id

∅

- Name

∅

Model's Properties

- baseType

∅

- Entity Types

∅

- Enum Types

∅

- Meta Data

∅

- simpleTypes

∅

General Properties

Property	Type	Description
ID	NA	The internal ID of this model, it is automatically generated when you first save your model.
Name	String	The name of the model. Generally, it will contain the type signature of the model, followed by a colon and then the main name of the model.

Model's Properties

Property	Type	Description
baseType	GmBaseType	This property describes the base type of this model. That is, what type this model should be inherited from.
Entity Type	GmEntityType	This defines which entity types belong to this model.
Enum Types	GmEnumType	This defines which enum types belong to this model
Meta Data	There are various metadata types available	This allows you to define various metadata which will enrich your model with information
simple Types	GmSimpleType	Allows you to define various simple types which belong to this model.

Property

Table of Contents

- ▼ Expand / collapse
 - Property

Property

Each entity type has a number of properties which help to describe it. These properties can range from objects such as name or ID to something more complex. However, they are all based on the type GmProperty and share properties themselves. A property can be either a simple type - String, Integer, Boolean and so on , a complex type, that is, a type which is itself another entity, or a collection, such as a list or a map.

All property are configured under the heading GmProperty's Properties

 All properties that are displayed in bold are mandatory. You will not be able to save any changes to a model without first entering information in these fields.

GmProperty's Properties

<i>Id</i>	∅
Is Id	<input type="checkbox"/>
isOverlay	<input type="checkbox"/>
Name	∅
– Entity Type	
∅	
– Meta Data	
∅	
– Type	
∅	

Property	Type	Description
ID	NA	The internal ID of the property, it is automatically generated when you first save the property.
Is Id	Boolean	Determines whether this property should function as an ID of this entity type. <div style="border: 1px solid #f0e68c; padding: 5px; margin-left: 20px;"> This field relates to the property itself, not the ID.</div>
Is Overlay	Boolean	Whether this property which belongs to the entity type has been inherited from a supertype or not. If set to true this is the case. If set to false the property is generated out of its entity type.
Name	String	The name of this property

Entity Type	GmEntityType	This defines to which entity type this property belongs.
Meta Data	There are various metadata types that can be used.	This allows you to define various metadata which will enrich your property with information.
Type	There are several types that can be defined here. The most common are GmSimpleType (either String, Integer, Boolean), GmEntityType, GmEnumType or type which belongs to the GmCollectionTypes	This defines which this type the property should be.

tribefire Cartridges



tribefire Core Models

tribefire Core Models

For getting into the very details of tribefire's core models please visit following sub-pages:

- Manipulation Model
- Process Model
- The MetaModel aka a model that models models
- The Query Model

Manipulation Model

Table of Contents

Expand / collapse

- [Introduction](#)
- [The Manipulation Model](#)
 - [Manipulation](#)
 - [Atomic Manipulation](#)
 - [CompoundManipulation](#)
 - [PropertyManipulation](#)
 - [AbsentingManipulation](#)
 - [CollectionManipulation](#)
 - [ChangeValueManipulation](#)
 - [Creation / deletion manipulations](#)

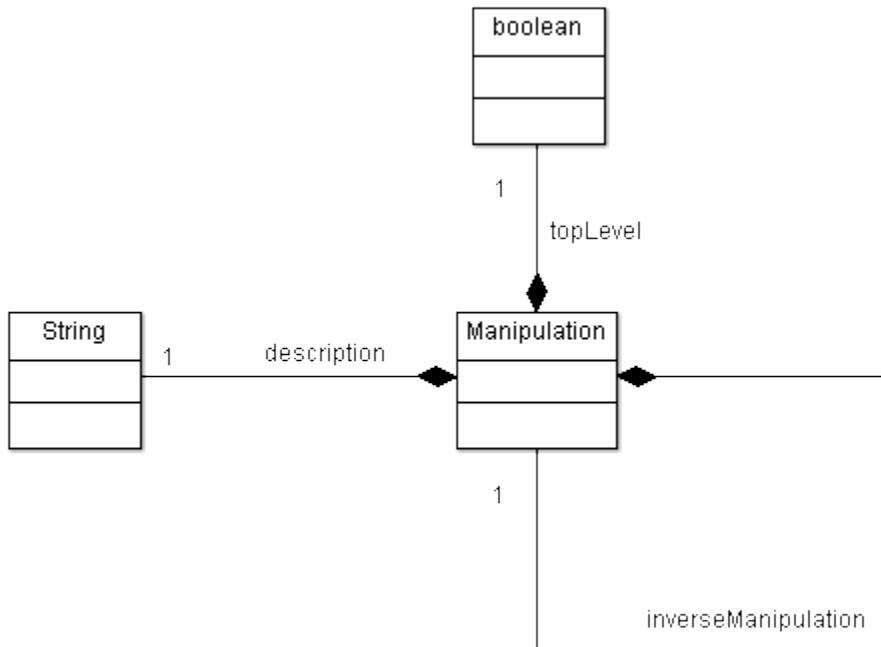
Introduction

The manipulation model is one of the most central models in the GM environment. It is basically able to reflect any modification of a GE entity.

The Manipulation Model

Manipulation

The Manipulation is the basic entity in the model. It simple has a description, and declares an inverse manipulation.



Name	Type	Description
topLevel	boolean	true if it's a toplevel
description	String	a descriptive text.

inverseManipulation	Manipulation	the inverse manipulation, i.e. the manipulation that would undo this manipulation.
---------------------	--------------	--

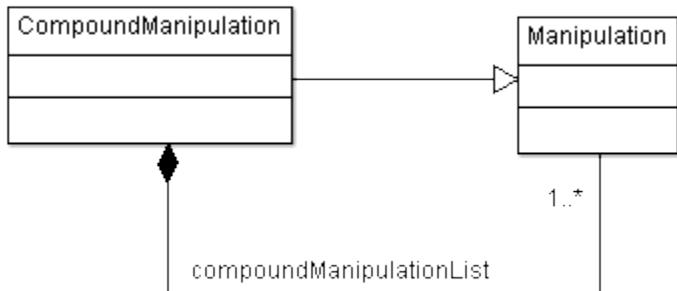
Atomic Manipulation

An atomic manipulation is a single manipulation standing by itself.



CompoundManipulation

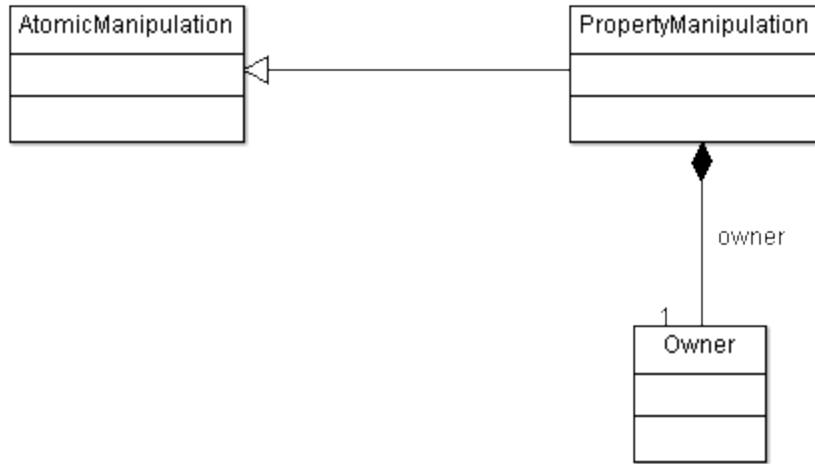
A compound manipulation contains several other manipulations.



Name	Type	Description
compoundManipulationList	List<Manipulation>	The list of manipulations contained in the compound manipulations.

PropertyManipulation

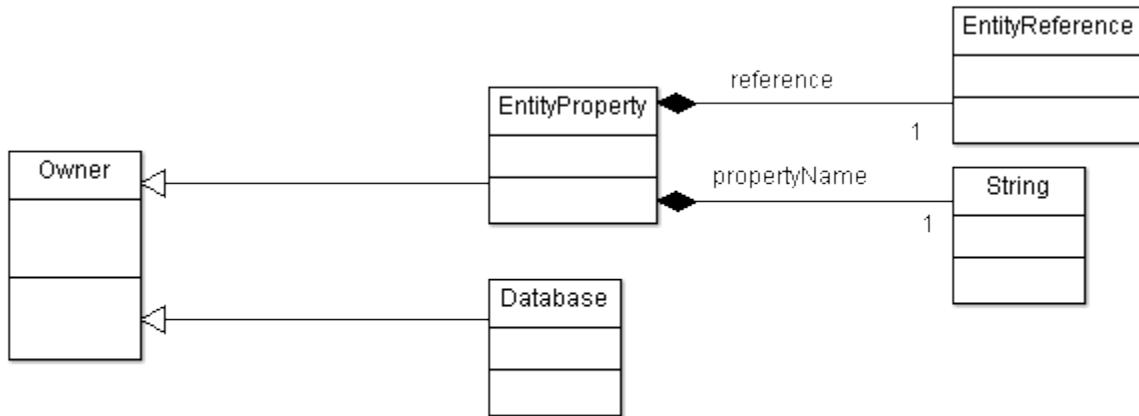
A PropertyManipulation is a manipulation that acts on a property. Obviously, it has an owner, i.e. the entity which is the container for the property.



Name	Type	Description
owner	Owner	the owner of the property

Owners

Every property has an owner, therefore, a property manipulation must also declare its owner. An owner can be either refer to an entity's property or to a database.



Subtype	Description
Database	deprecated
EntityProperty	An owner that represents an entity property.

Database

This entry is deprecated as it won't be used anyhow. So please ignore it.

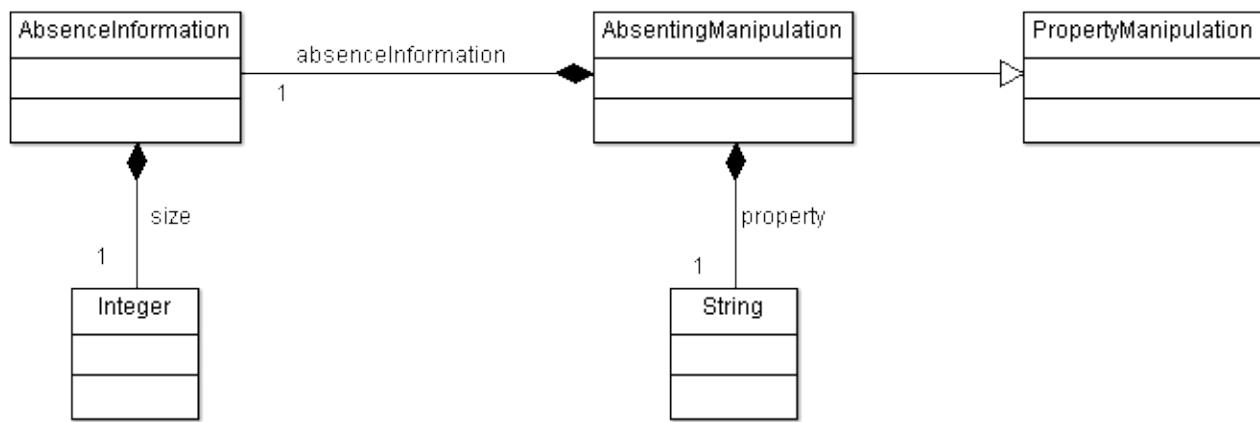
EntityProperty

An EntityProperty connects the manipulation to a property of an instance, i.e. a GenericEntity.

Name	Type	Description
reference	EntityReference	A reference to the instance that is the container for the property.
propertyName	String	The name of the property.

AbsentingManipulation

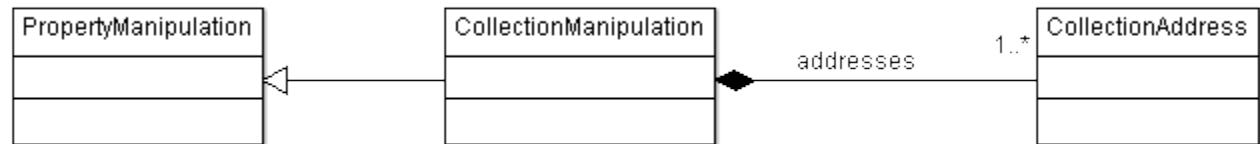
An absenting manipulation adds an AbsenceInformation to a property.



Name	Type	Description
property	String	The property the manipulation applies to.
absenceInformation	AbsenceInformation	The absence information that is to be added to the property.

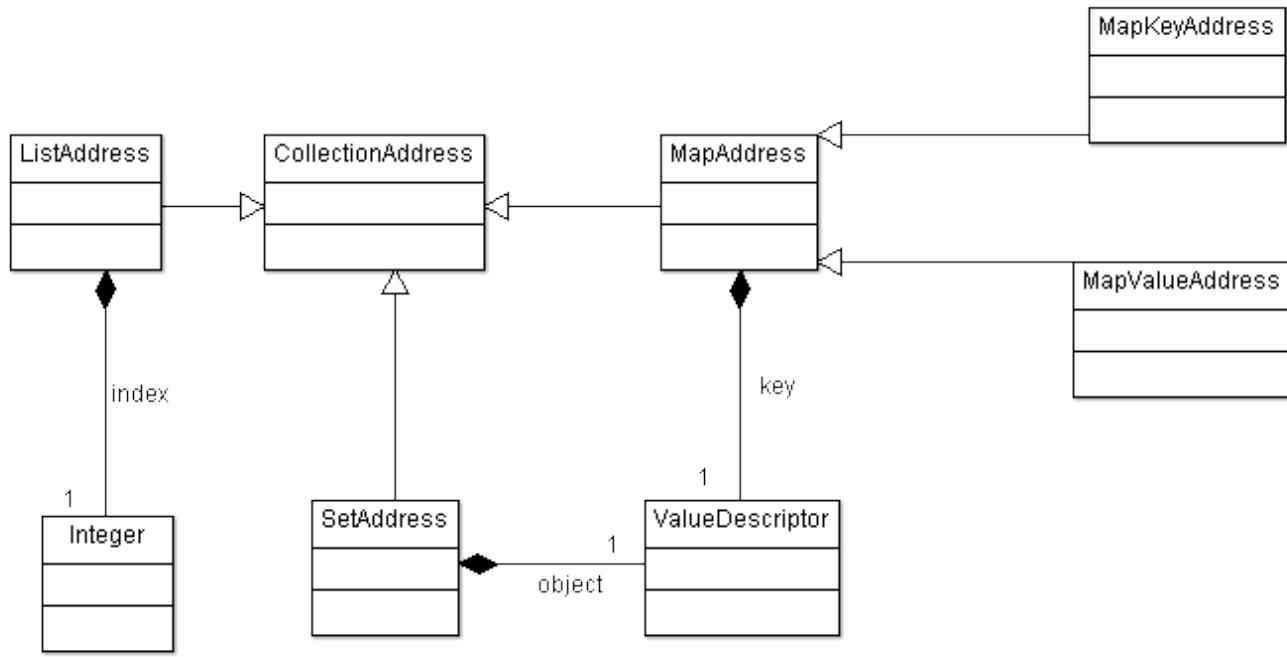
CollectionManipulation

A CollectionManipulation is a PropertyManipulation that acts on a property that's a collection.



CollectionAddresses

A property of type Collection obviously needs more addressing data than an atomic property. CollectionAddress address (pun intended) that issue.



ListAddress

A list address addresses its value via the position in the list.

Name	Type	Description
index	int	Index in list.

SetAddress

A set address addresses its value directly by referencing it via a descriptor.

Name	Type	Description
object	ValueDescriptor	A direct reference to the object via a descriptor.

MapAddress

Name	Type	Description
key	ValueDescriptor	Being a base class, the actual semantics of the member key is defined by the sub types, see below

MapKeyAddress

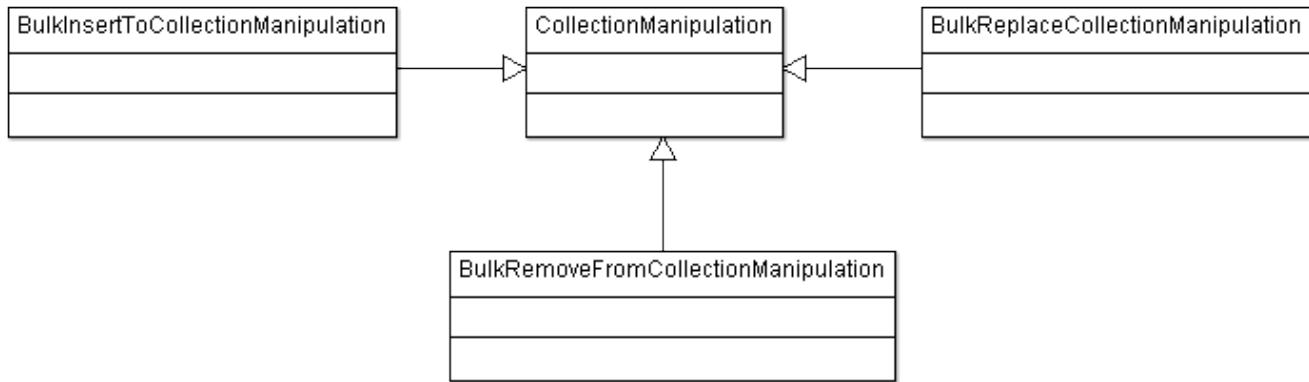
A map key address addresses its value in a map via the key, i.e. it interprets the member "key" as key.

MapViewAddress

A map value address address its value in a map directly by its value, i.e. it interpretes the member "key" directly as value.

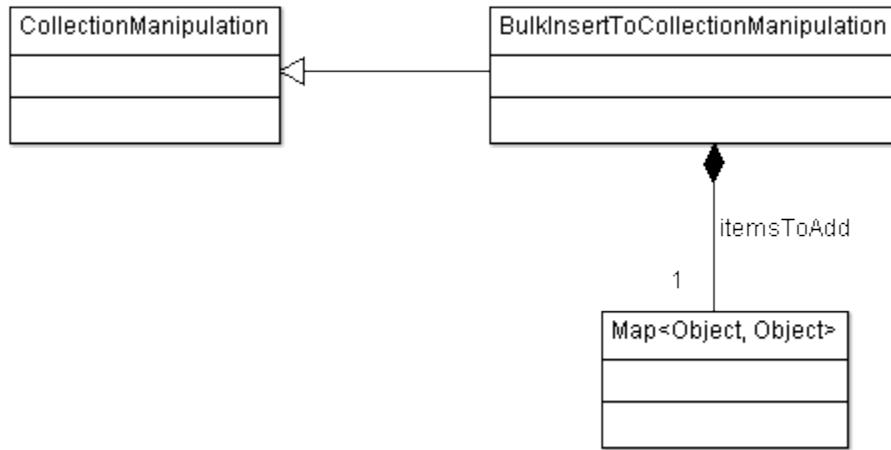
Bulk collection manipulations

BulkCollectionManipulations do exactly what they're named after: they do everything in series, i.e. process bulks of manipulations.



BulkInsertToCollectionManipulation

The bulk insertion to collection manipulation handles exactly what it is named after.



Name	Type	Description
itemsToAdd	Map<Object, Object>	A map that contains the objects that should be inserted. If the target property's a map, all elements are to be used. If it's a onedimensional collection, then the keys should be used.

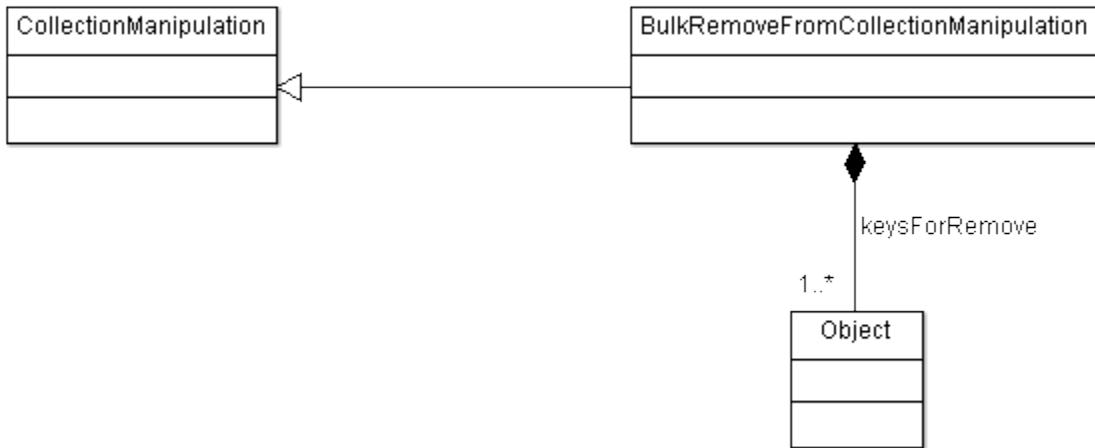
Parameter description

Property target type	Parameter composition
List	Map<Index, Value>
Map	Map<Key; Value>

Set	Map<Value, Value>
-----	-------------------

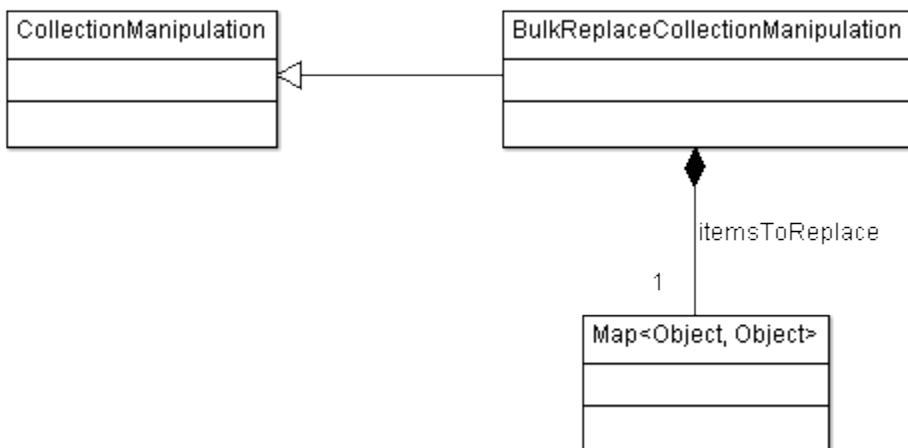
BulkRemoveFromCollectionManipulation

The manipulations is intended for the bulk removal of objects from a collection.



Name	Type	Description
keysForRemove	Set<Object>	A set with the keys or the value descriptors that are to be removed from the map or onedimensional collection respectively.

Property target type	Parameter composition
List	Set<Index>
Map	Set<Key>
Set	Set<Value>

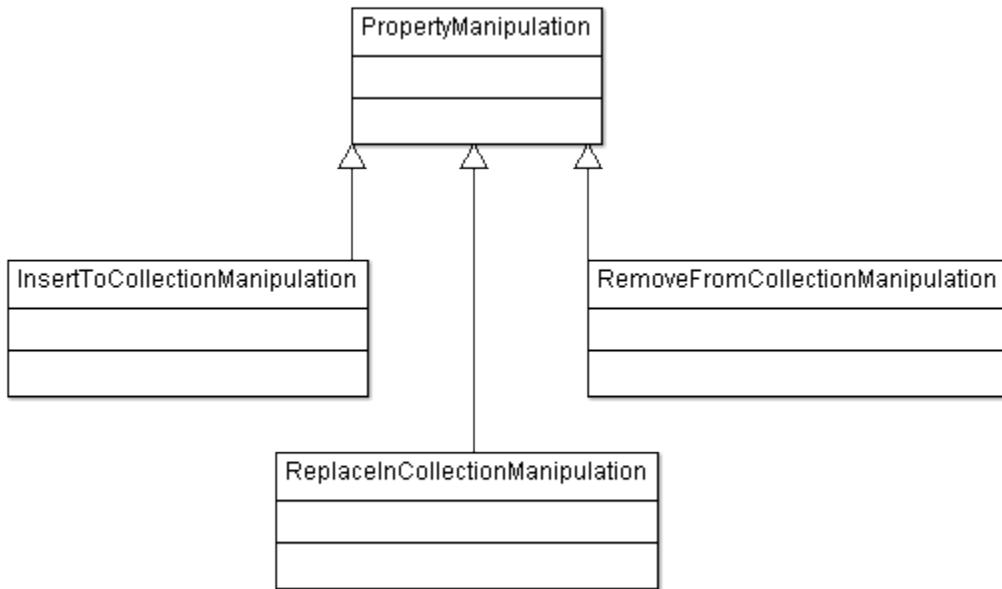
BulkReplaceInCollectionManipulation

Name	Type	Description
itemsToReplace	Map<Object, Object>	A map with the data to replace. If the target's a one-dimensional map, the keys are to be used.

Property target type	Parameter composition
List	Map<Index, Value>
Map	Set<Key, Value>
Set	Set<Old value, New value>

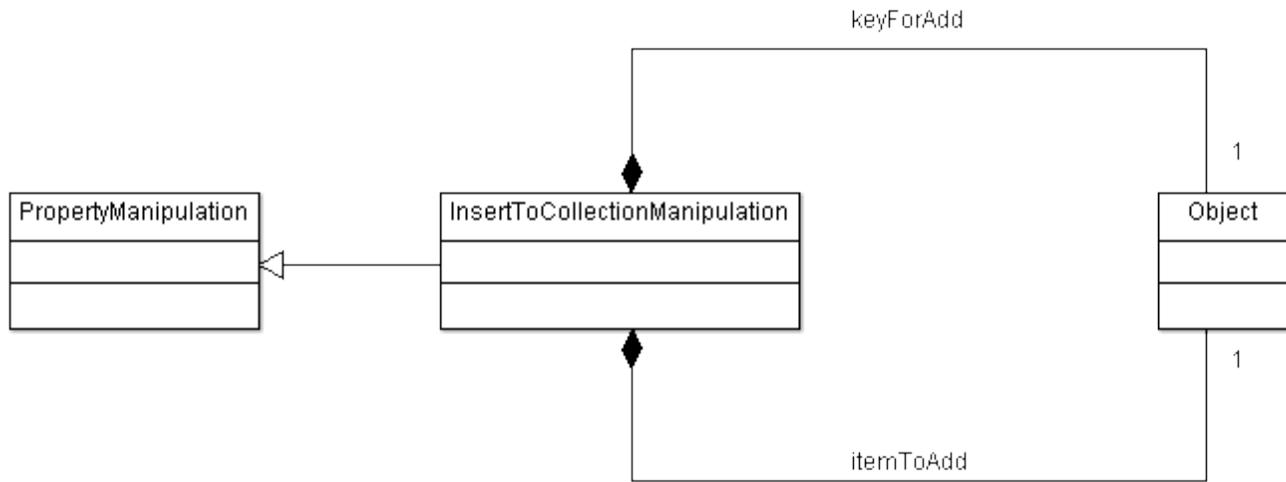
Single collection manipulations

Single collection manipulations manipulate collection with a single parameter.



InsertToCollectionManipulation

This manipulation describes the insertion of a single parameter into a collection.

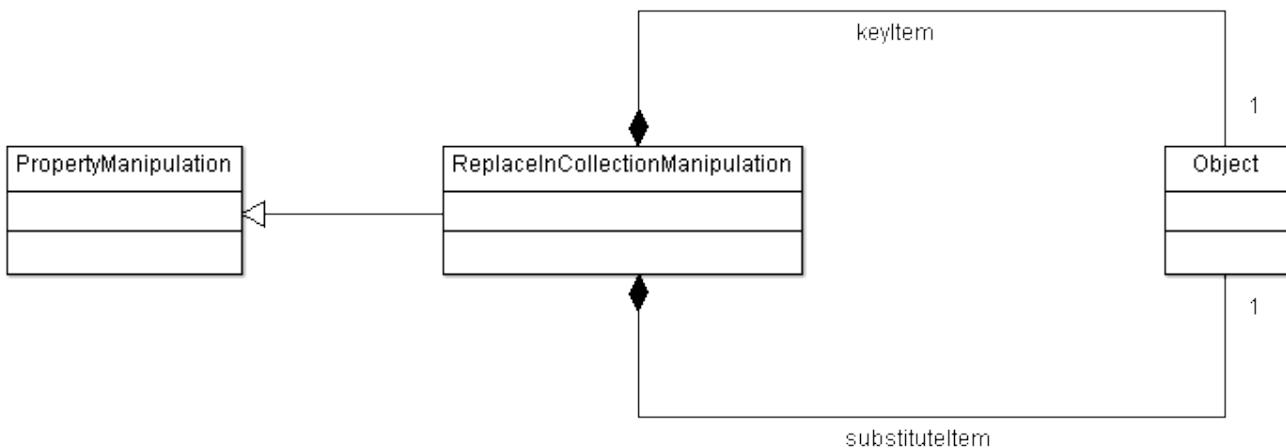


Name	Type	Description
keyForAdd	Object	An object identifying the location of the value inside the collection.
itemToAdd	Object	The object to add to the collection.

Property target type	Parameter composition of key
List	Index
Map	Key
Set	Ignore

ReplaceInCollectionManipulation

Replaces a single value in a collection.

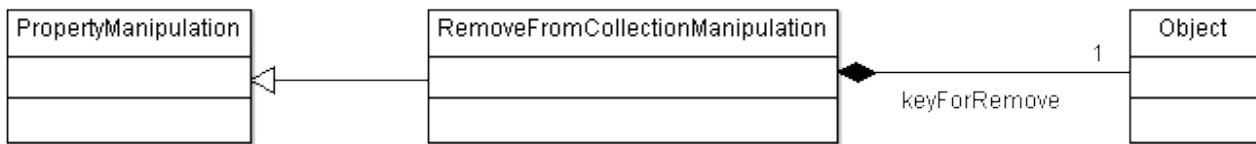


Name	Type	Description
keyItem	Object	An object identifying the location of the value inside the collection.

substitute item	Object	The object to replace the old value with
Property target type	Parameter composition of keyItem	
List	Index	
Map	Key	
Set	Value to replace	

RemovesFromCollectionManipulation

Removes a single value from a collection.

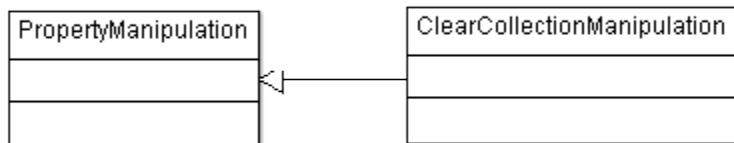


Name	Type	Description
keyForRemove	Object	An object identifying the location of the value inside the collection.

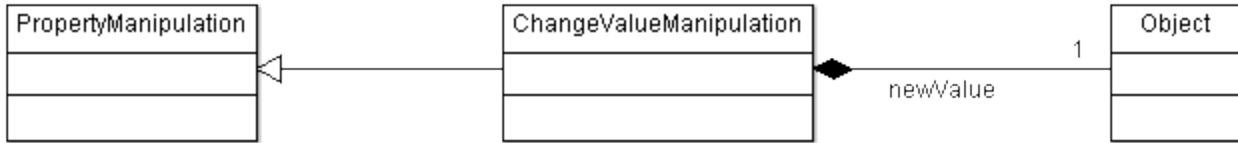
Property target type	Parameter composition of keyItem	
List	Index	
Map	Key	
Set	Value to remove	

ClearCollectionManipulation

Clears a collection.

**ChangeValueManipulation**

Denotes the changing of a property's value.



Name	Type	Description
new Value	Object	The new value of the property.

Creation / deletion manipulations

InstantiationManipulations

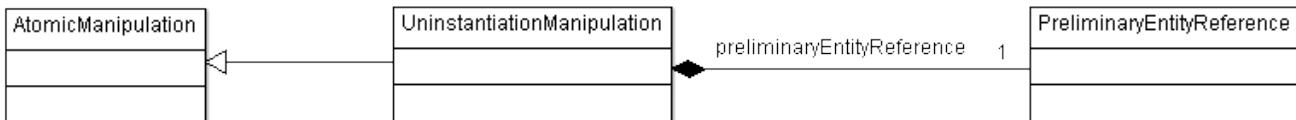
Describes a new instantiation of an entity.



Name	Type	Description
preliminaryEntityReference	PreliminaryEntityReference	The preliminary entity reference of the newly created entity.

UninstantiationManipulation

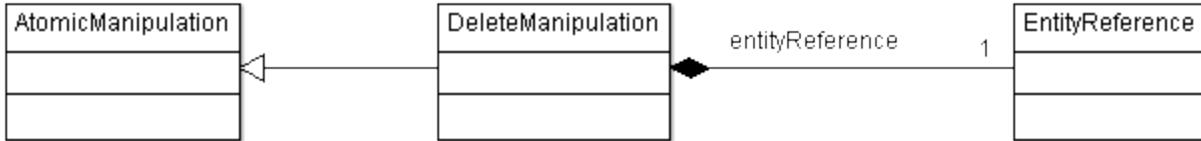
Describes the uninstantiation of an entity, i.e. the deletion of an entity that only had a preliminary reference attached to it (was never persisted).



Name	Type	Description
preliminaryEntityReference	PreliminaryEntityReference	The preliminary entity reference of the newly created entity that is to be deleted..

DeleteManipulation

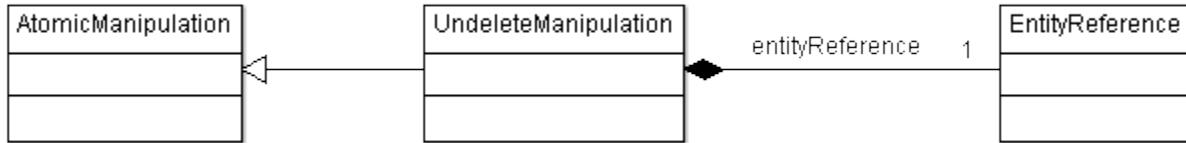
The delete manipulation denotes a deletion event of an existing element, i.e. one has been persisted once.



Name	Type	Description
entityReference	EntityReference	The reference of the entity that is to be deleted.

UndeleteManipulation

The undelete manipulation is the inverse operation to the delete manipulation.



Name	Type	Description
entityReference	EntityReference	The reference of the entity that is to be deleted.

Process Model

Derives from

▼ show/hide

 BTT-1975 - Build ProcessModel ( Resolved)

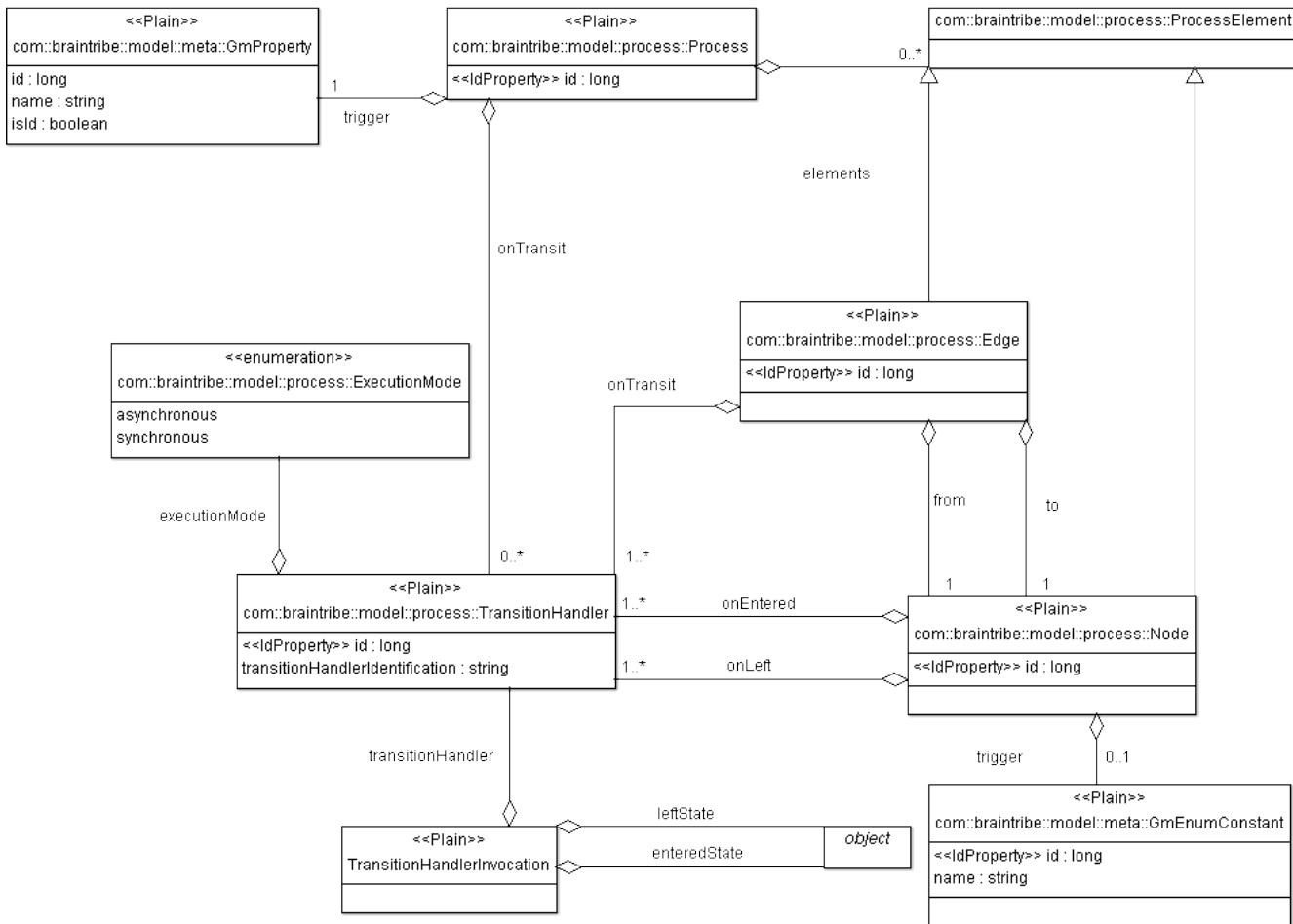
Table of Contents

▼ show/hide

- 1 Introduction
- 2 The Types of the Model
 - 2.1 Process
 - 2.2 ProcessElements
 - 2.3 Node
 - 2.4 Edge
 - 2.5 TransitionHandler
 - 2.6 TransitionHandlerInvocation

Introduction

The process model describes the states of a process. It is an integral part of the process engine, i.e. the LightWeightStateProcessor 2.0 interpretes entities of this model and dispatches calls to the associated handlers accordingly.

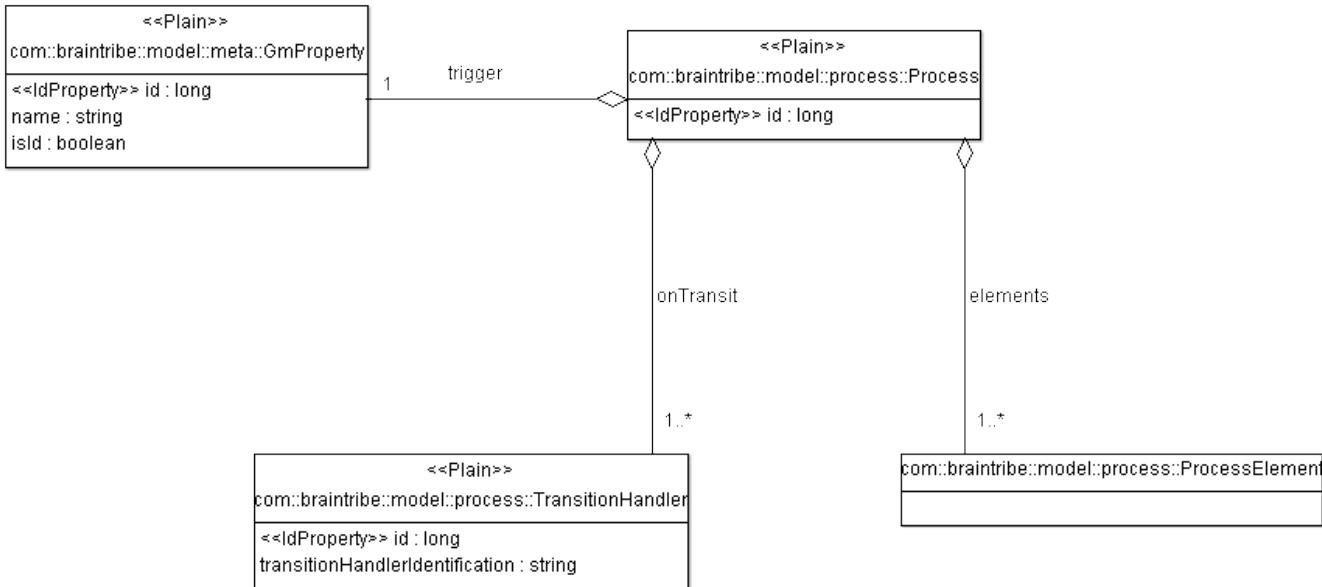


Basically, the model describes processes that are linked to a GmProperty. It is thought that the GmProperty is on type GmEnumType, and the associated steps are attached to distinct values of the enum.

The Types of the Model

Process

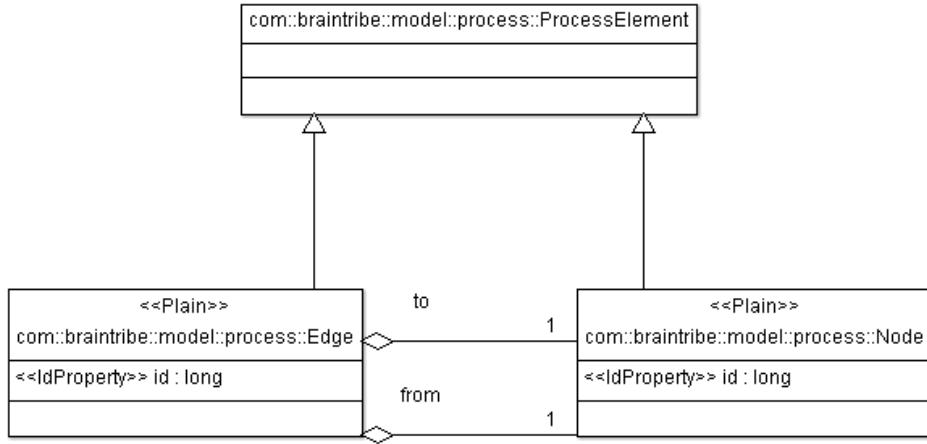
A process is linked to a property. The state handler is triggered if it detects a ChangeValueManipulation with the associated GmProperty. It then looks at the model and uses the information contained in the ChangeValueManipulation to query the model.



Name	Type	Description
<code>id</code>	<code>long</code>	Id property of the process
<code>trigger</code>	<code>GmProperty</code>	Property the process is associated with
<code>elements</code>	<code>Set<ProcessElement></code>	Set of associated nodes and edges
<code>onTransit</code>	<code>Set<TransitionHandler></code>	Transition handlers that should be called whenever something related to the process happens

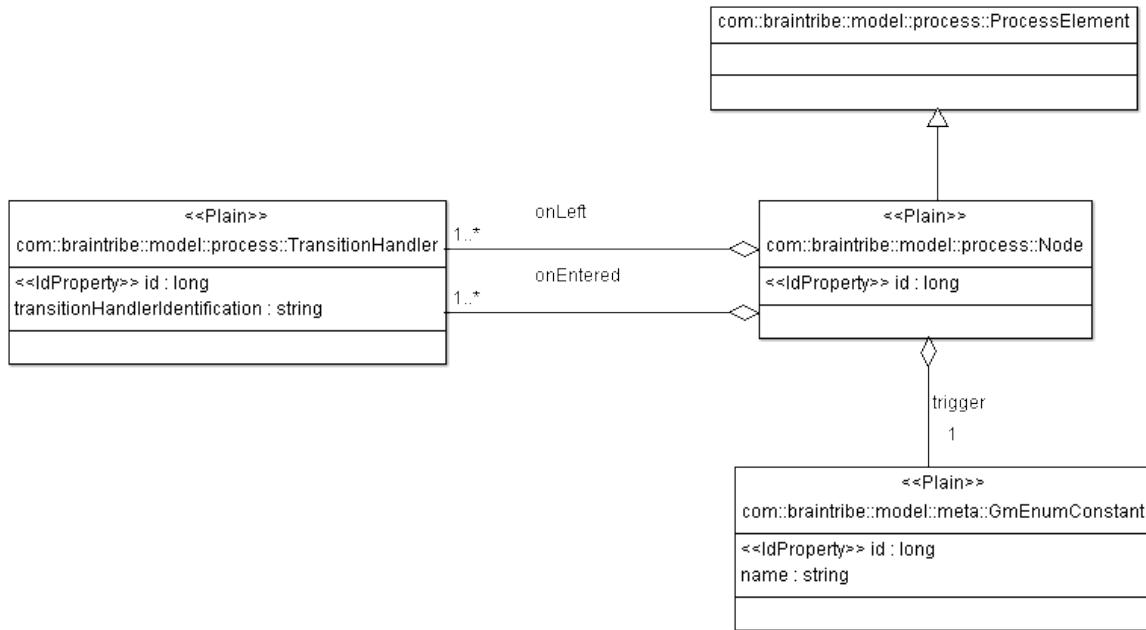
ProcessElements

The process element is a common denominator for Node and Edge. They are both associated with a process as they model the possible entry points for the handlers.



Node

A node reflects a specific state within a chain of states. It derives from the generic `ProcessElement`. Semantically, it has two different aspects - in one context it reflects the state that has been entered, and in the other context it reflects the state that has been left. In each context, the node can have an associated transition handler. Associated with the node is an identifying state value.

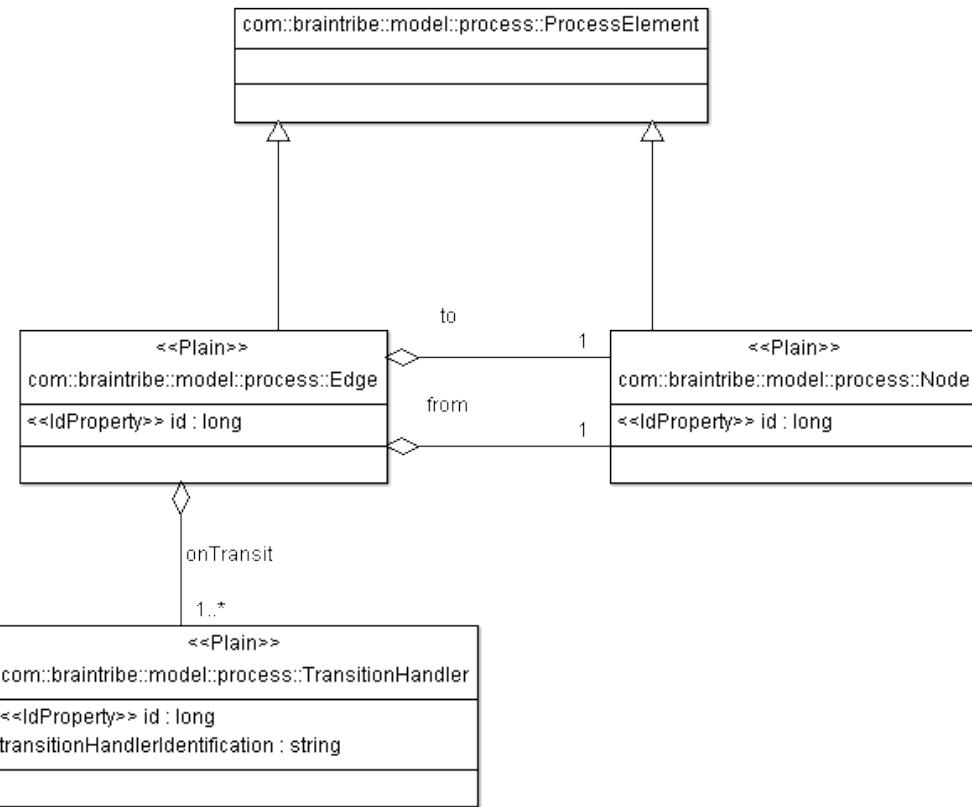


Name	Type	Description
<code>id</code>	<code>long</code>	Id Property
<code>trigger</code>	<code>GmEnumConstant</code>	The distinct enum value that denotes the state the node is associated with
<code>onLeft</code>	<code>List<TransitionHandler></code>	List of transition handler that should be run if the node is in the context of having been left

onEntered	List<TransitionHandler>	List transition handler to be run if the node is reached in the context of having been entered
-----------	-------------------------	--

Edge

An edge describes the transition between two states, it connects to nodes. If a node describes a single state (that has been left or has been entered), an edge describes a connection between two nodes, i.e. a transition between two distinct states.

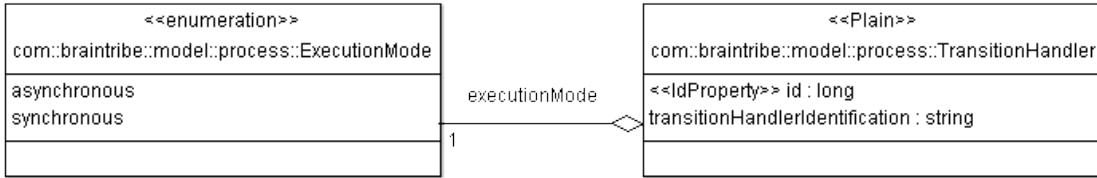


Name	Type	Description
id	long	Id property
to	Node	The node that is the start point of the transition
from	Node	The node that is the end point of the transition
onTransit	List<TransitionHandler>	The associated transition handlers

TransitionHandler

The transition handler declares the class name of the expert that is to be called when required. Basically, it is

nothing more than the fully qualified class name of the expert, and a hint of how the execution should be scheduled.



Name	Type	Description
id	long	Id Property
transitionHandlerIdentification	string	The fully qualified name of the associated expert
executionModel	Execution Mode	An GmEnumType with two possible values denoting whether the handler should be scheduled synchronously or rather asynchronously

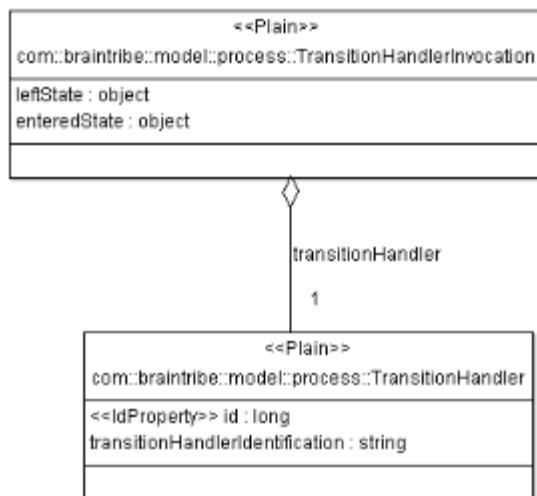
In all cases where transition handlers are associated, there are multiple instances allowed, i.e. they are referenced as an ordered collection in the process, edge and node types. Consequence is of course twofold:

- the transition handlers are called in the order they exist in the list (keep in mind that in case of a distributed process engine, that still doesn't necessarily mean that they are executed in that order as only the sequence how the calls to the publisher is synchronized and the actual execution of the handler code solely depends on the consumers)
- a state change in one transition handler can lead to the execution of further transition handlers, so you cannot assume that the state of affairs your handlers encounter are the exact state that lead to their calling - you can minimize the chance of this happening by moving the single state changing transition handler to the end of the list.. only if all handlers are marked as to be synchronously called and you only have a single state changing handler, then you can safely rely on a static state of affairs during the processing of all transition handlers.

Even if these restrictions apply, it is still a powerful feature to have more than a single transition handler, as there are always handlers imaginable that do not lead to a state change - such as tracing handlers, message brokers, persistence experts and so on.

TransitionHandlerInvocation

The transition handler invocation is what the lightweight state processor uses to call the transition handler. It contains the actual transition handler definition and the two values associated, i.e. the value of the property before and the value of the property after the change.



The MetaModel aka a model that models models

Table of Contents

▼ Expand / collapse

- The Meta Model
 - Types
 - MetaData
 - Constraints
 - DisplayInfo
 - BasicDisplayInfo
 - FulltextMetaData
 - ManipulationPostprocessing
 - Prompting
 - Resources
 - Security
 - Selectors

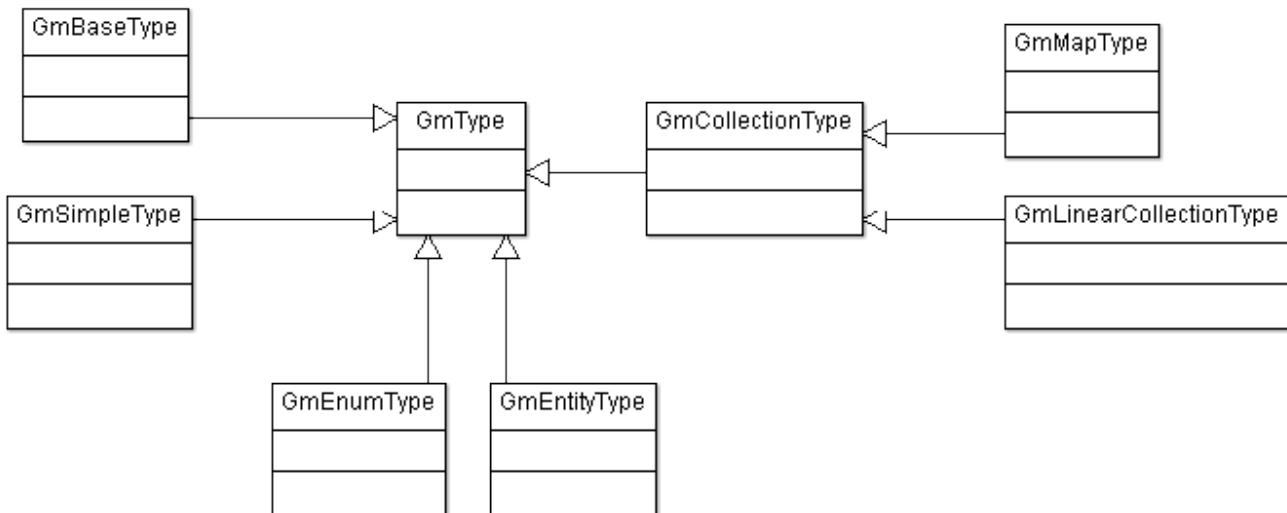
Introduction

The Meta Model is the model that models models no, that subtitle's not a joke. The Meta Model is exactly that it allows to talk about models in a generic manner. Due to that, it is the backbone of GM.

The Meta Model

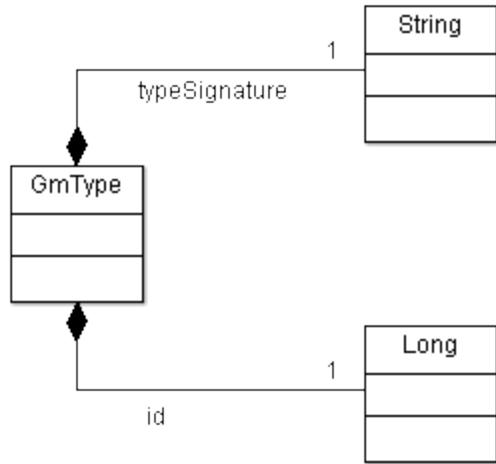
Types

Types of course are used to reflect the types we encounter in the real world.



GmType

The GmType is the mother of all types.



Name	Type	Description
id	Long	The id property of the type
typeSignature	String	the fully qualified name of the class it represents.

GmBaseType

The **GmBaseType** is a wrapper of any class not further specified. Any object matches the base type.



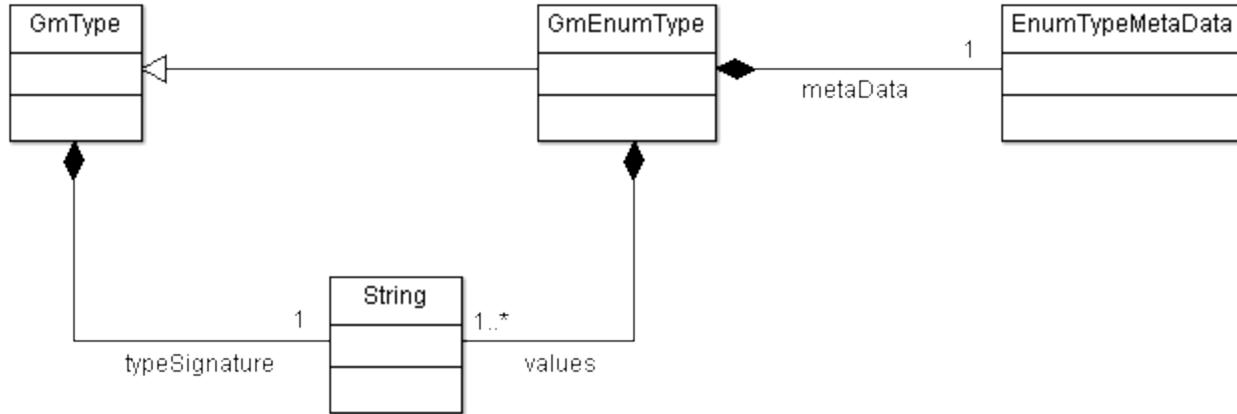
GmSimpleType

The **GmSimpleType** represents simple types like the `java.lang.*Types`.



GmEnumType

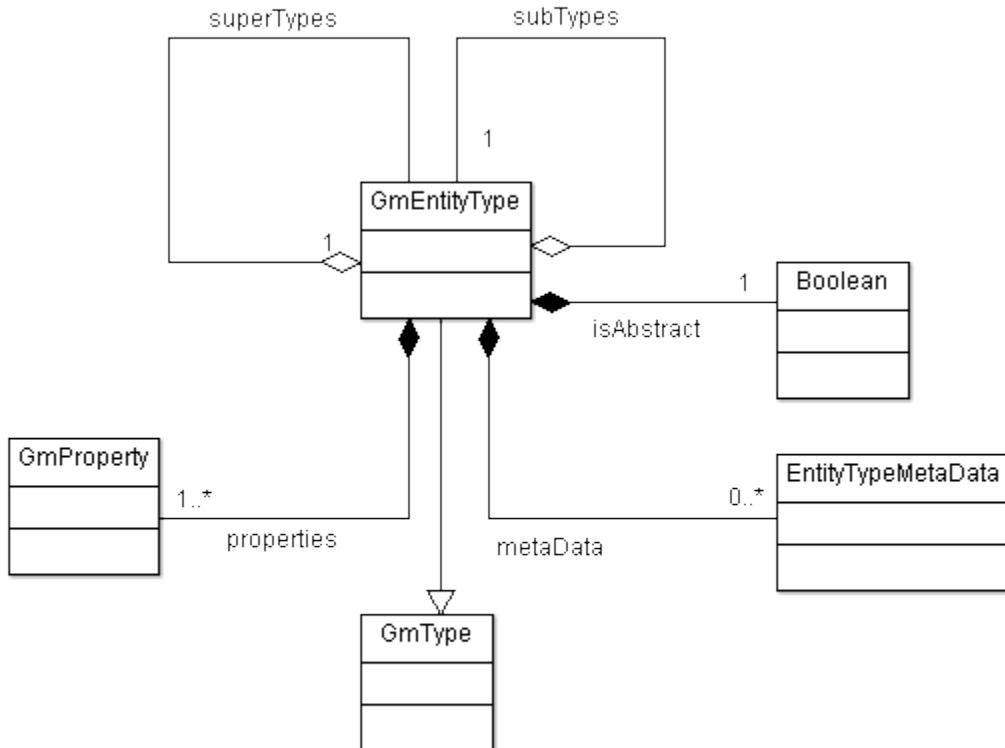
The **GmEnumType** describes enums.



Name	Type	Description
metaData	EnumTypeMetaData	Metadata for the enum.
values	String	The string representation of the enum's values.

GmEntityType

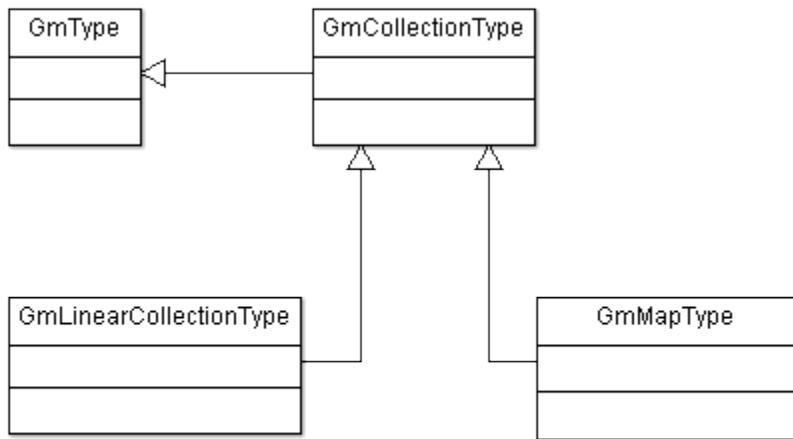
The **GmEntityType** describes a type that derives from **GenericEntityType**.



Name	Type	Description
superTypes	Set<GmEntityType>	A set with the super types of this GmEntityType.
subTypes	Set<GmEntityType>	A set with the sub types of this GmEntityType.
isAbstract	Boolean	A flag whether this type's abstract.
metaData	EntityTypeMetaData	MetaData for the GmEntityType
properties	Set<GmProperty>	A set of GmProperties for the GmEntityType.

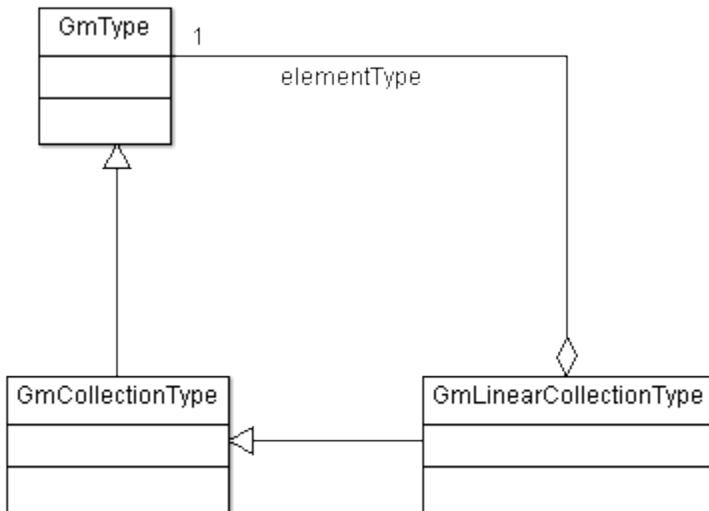
GmCollectionType

The GmCollection type itself is the base class for the two subtypes GmLinearCollectionType and GmMapType.



GmLinearCollectionType

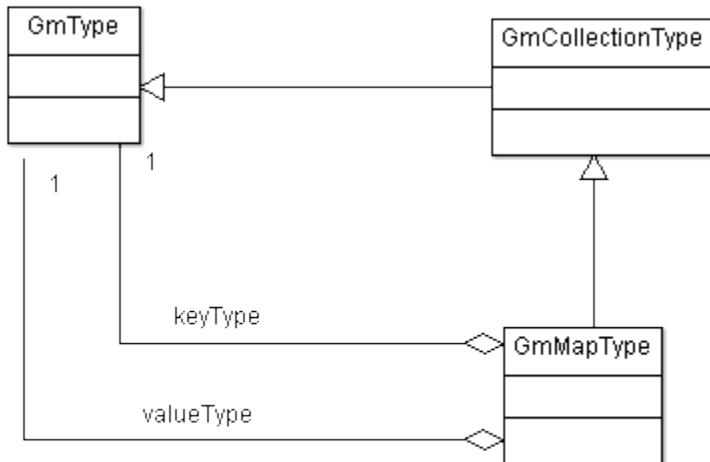
The GmLinearCollectionType stands for any onedimensional collection such as lists, sets and so on.



Name	Type	Description
elementType	GmType	The GmType that is the element type of the collection.

GmMapType

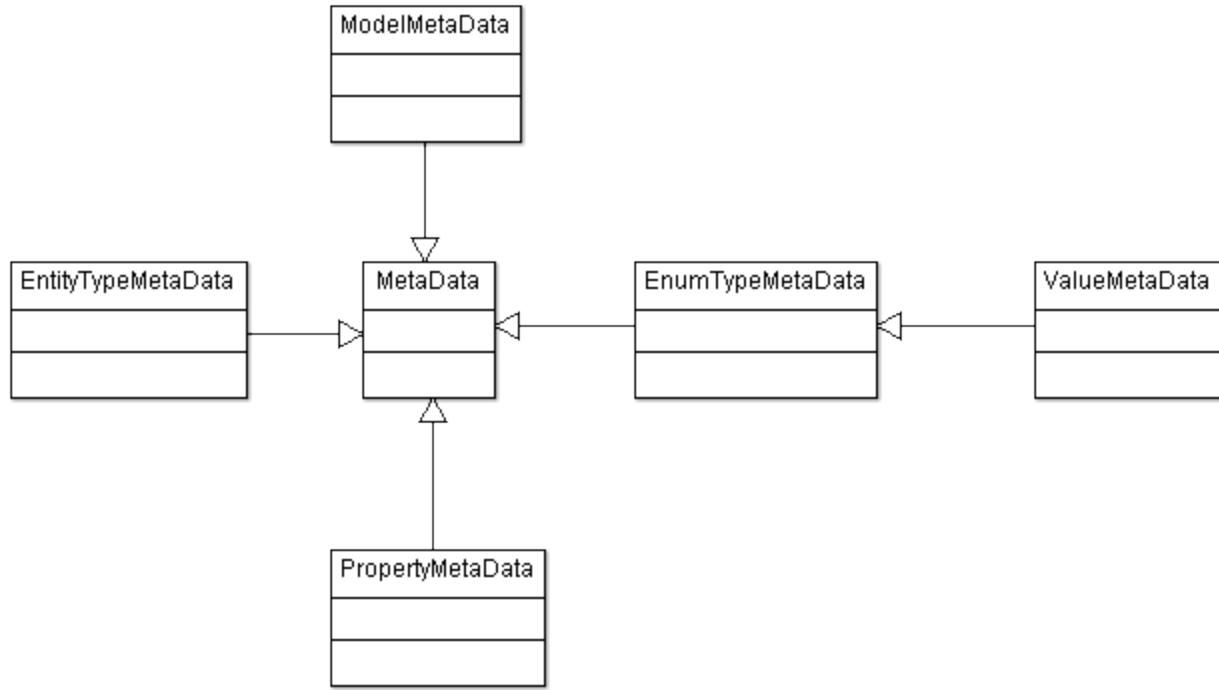
The GmMapType describes a two dimensional collection, such as maps.



Name	Type	Description
keyType	GmType	The GmType that stands for the key of the map.
valueType	GmType	The GmType of the values in the map.

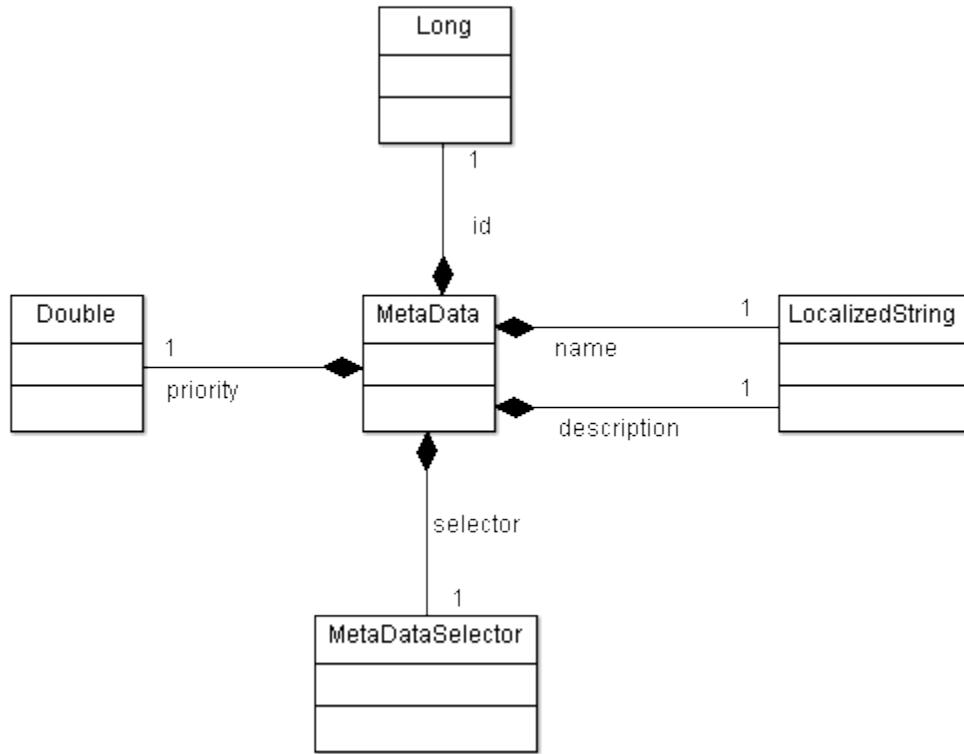
MetaData

MetaData is data that is attached to the types. They contain any form of data that gives additional information about the types.



MetaData

The base class for all Meta Data.



Name	Type	Description
------	------	-------------

id	Long	The id property.
priority	Double	The priority, between 0.0 and 1.0.
name	LocalizedString	The name of the MetaData as multilanguage string.
description	LocalizedString	The description of the MetaData as multilanguage string.
selector	MetaDataSelector	A selector for the conditional activation / relevance of the MetaData's content.

ModelMetaData

MetaData for a Model

**EntityTypeMetaData**

MetaData for an EntityType

**PropertyMetaData**

MetaData for a property.

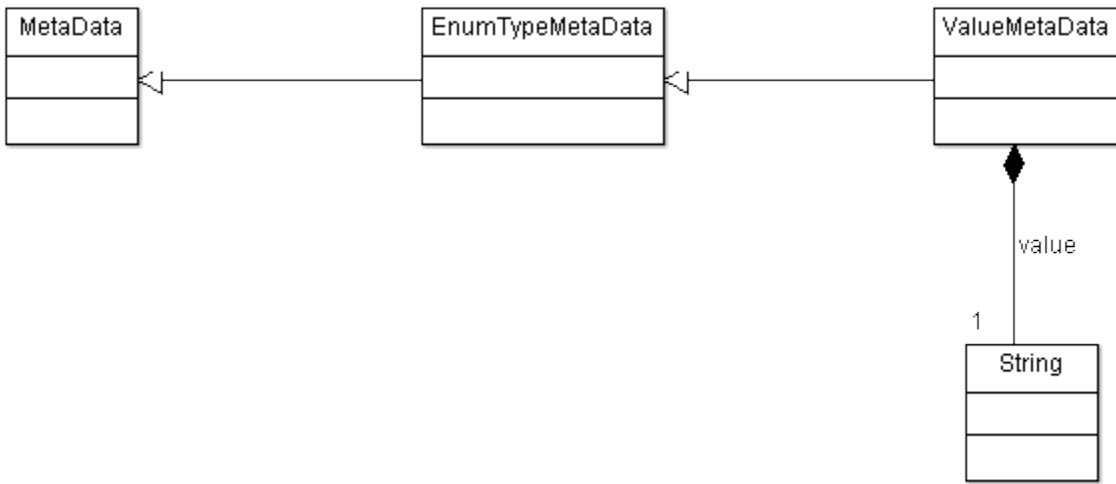
**EnumTypeMetaData**

MetaData for an EnumType.



ValueMetaData

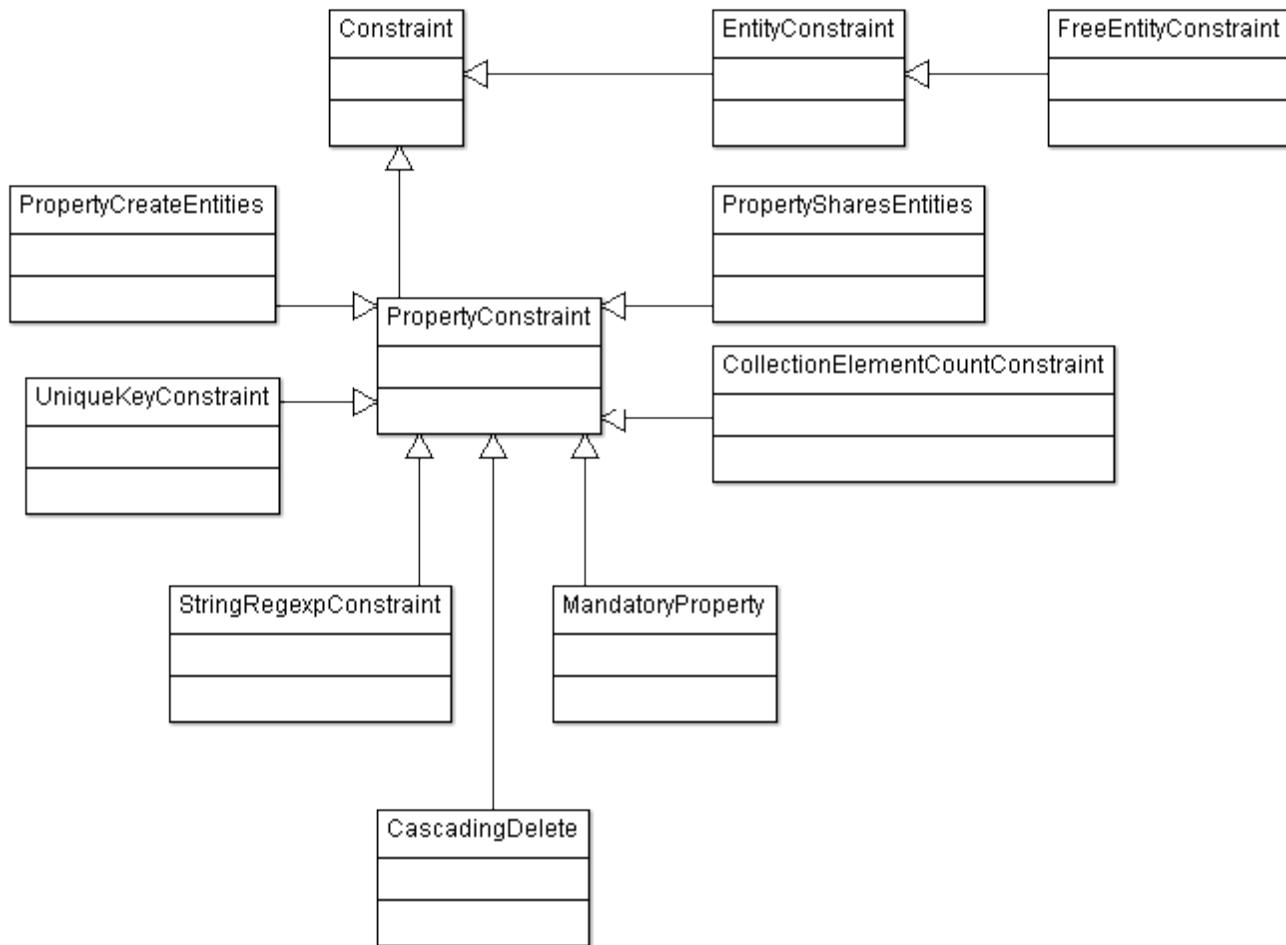
MetaData for a value of an EnumType.



Name	Type	Description
value	String	The name of the value the MetaData applies to.

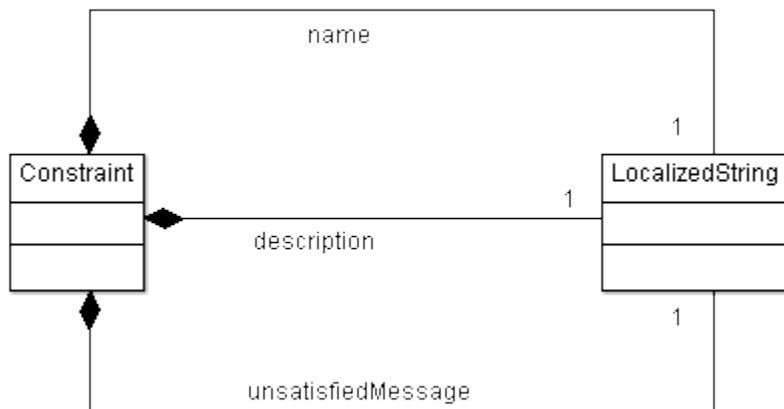
Constraints

Constraints are a form of MetaData that is some aspect reflect restrictions



Constraint

The Constraint is the base entity for all constraints.

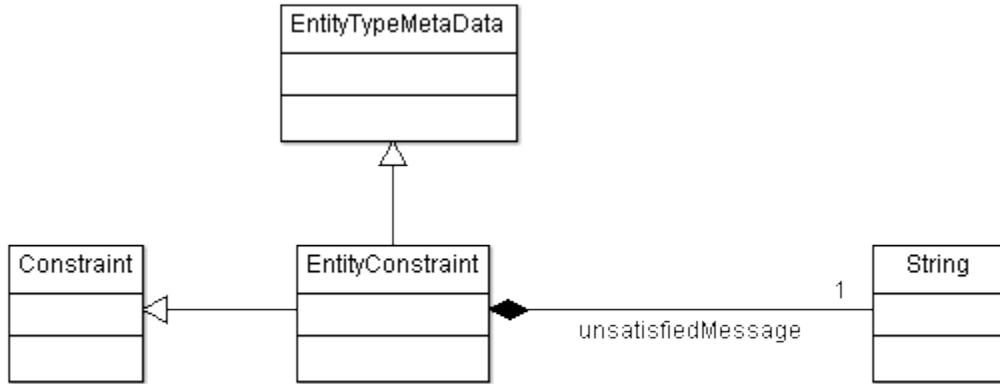


Name	Type	Description
name (interface only)	LocalizedString	retrieves the name of the constraint

description (interface only)	LocalizedString	retrieves the description of the constraint
unsatisfied Message (interface only)	LocalizedString	retrieves the message that is shown when the constraint's not satisfied..

EntityConstraint

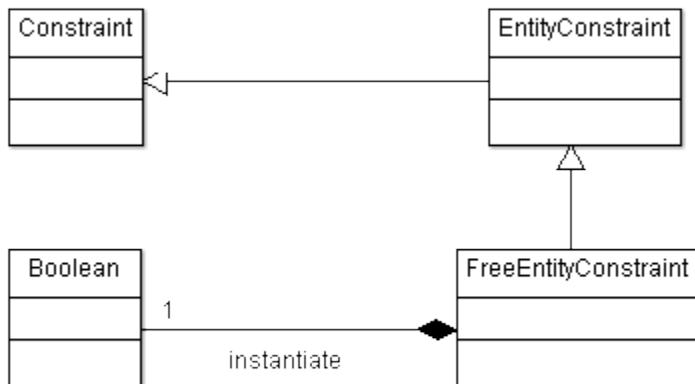
The base for entity related constraints.



Name	Type	Description
unsatisfiedMessage	LocalizedString	retrieves the message that is shown when the constraint's not satisfied..

FreeEntityConstraint

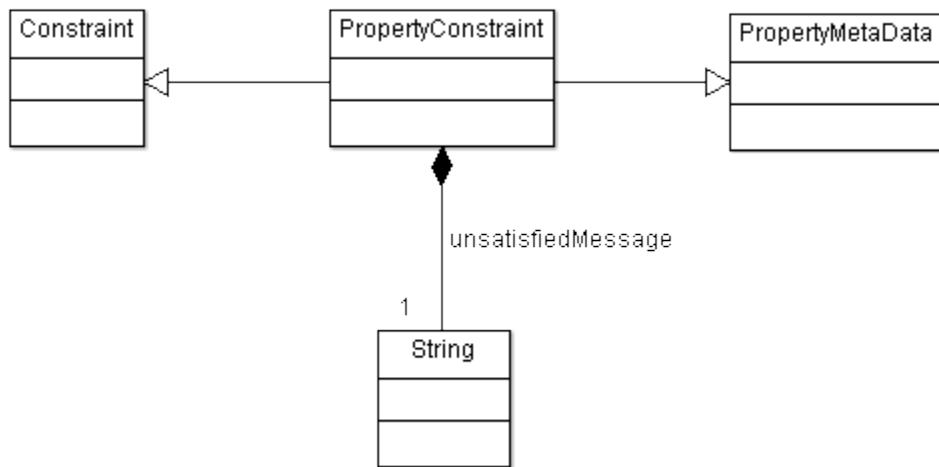
Description missing.



Name	Type	Description
instantiate	Boolean	Flag to describe whether this constraints applies to an instantiation - please verify

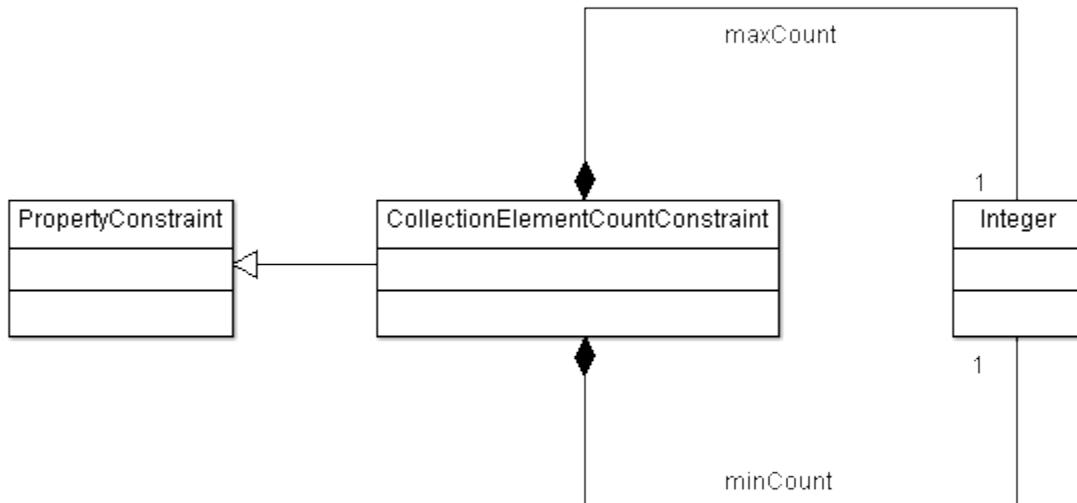
PropertyConstraint

The base for all property related constraints.



CollectionElementCountConstraint

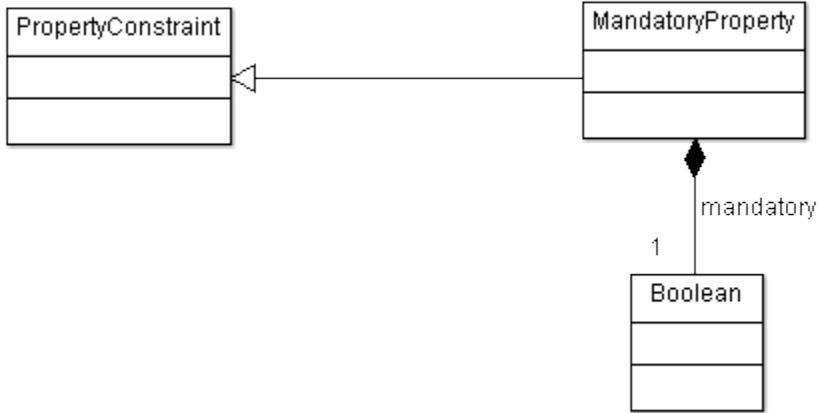
A constraint where the size of a collection is relevant.



Name	Type	Description
minCount	Integer	Minimal size or null if not relevant
maxCount	Integer	Maximal size or null if not relevant

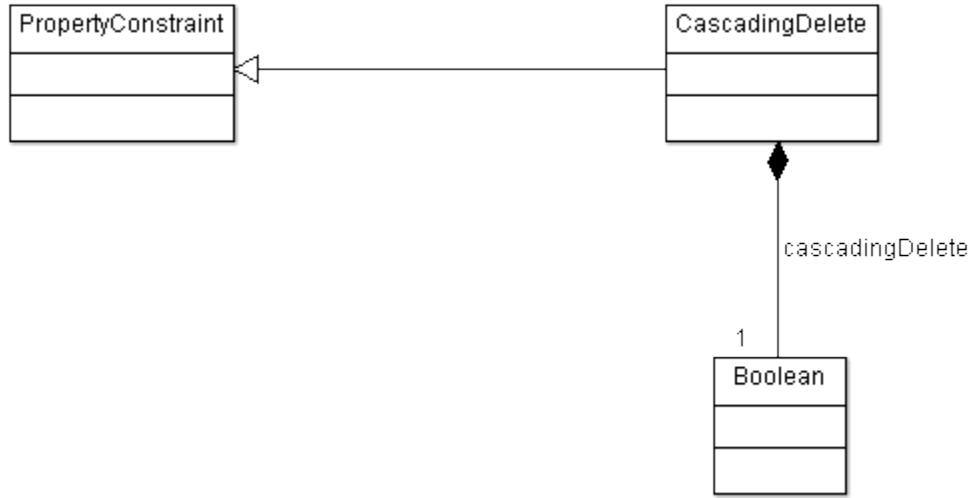
MandatoryProperty

This constraint marks that the property it is attached to is mandatory (if the flag's set of course)



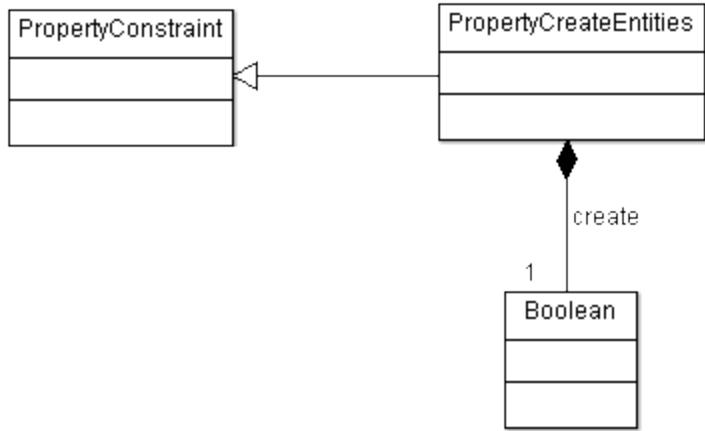
Name	Type	Description
mandatory	Boolean	If set, the attached property is marked as being mandatory.

CascadingDelete



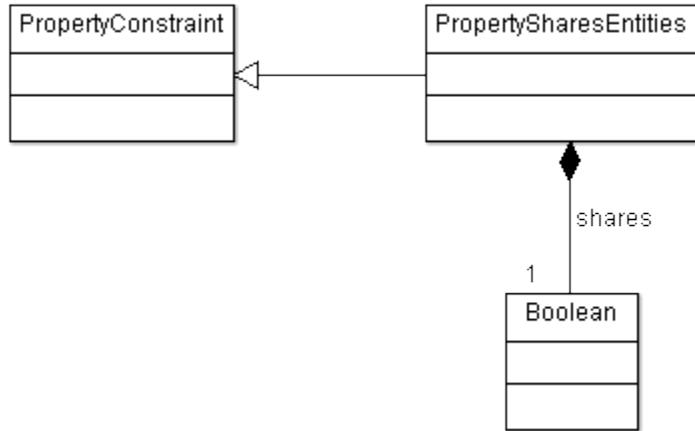
Name	Type	Description
cascadingDelete	Boolean	If set, the constraint is activated.

PropertyCreateEntities



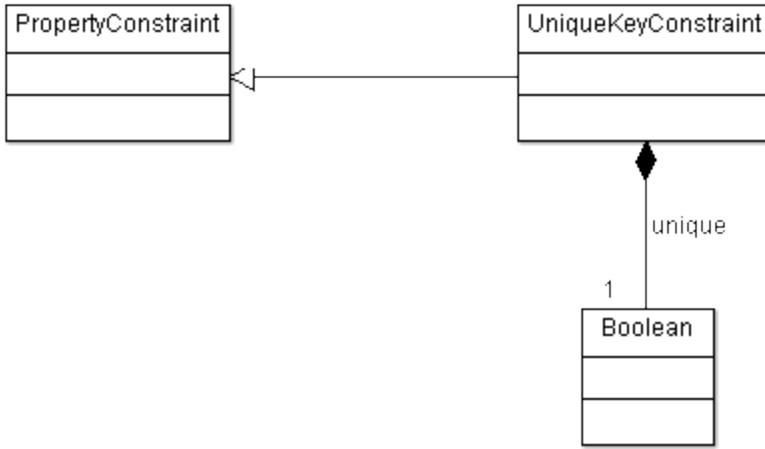
Name	Type	Description
create	Boolean	If set, the constraint is activated.

PropertyShareEntities



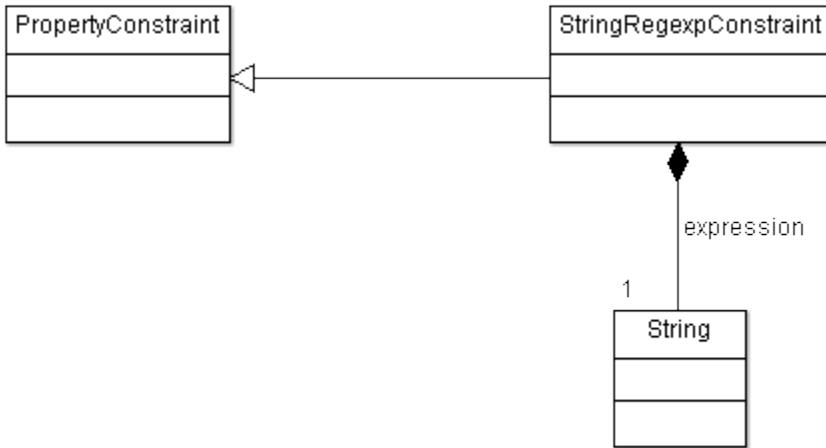
Name	Type	Description
shares	Boolean	If set, the constraint is activated.

UniqueKeyConstraint



Name	Type	Description
shres	Boolean	If set, the constraint is activated.

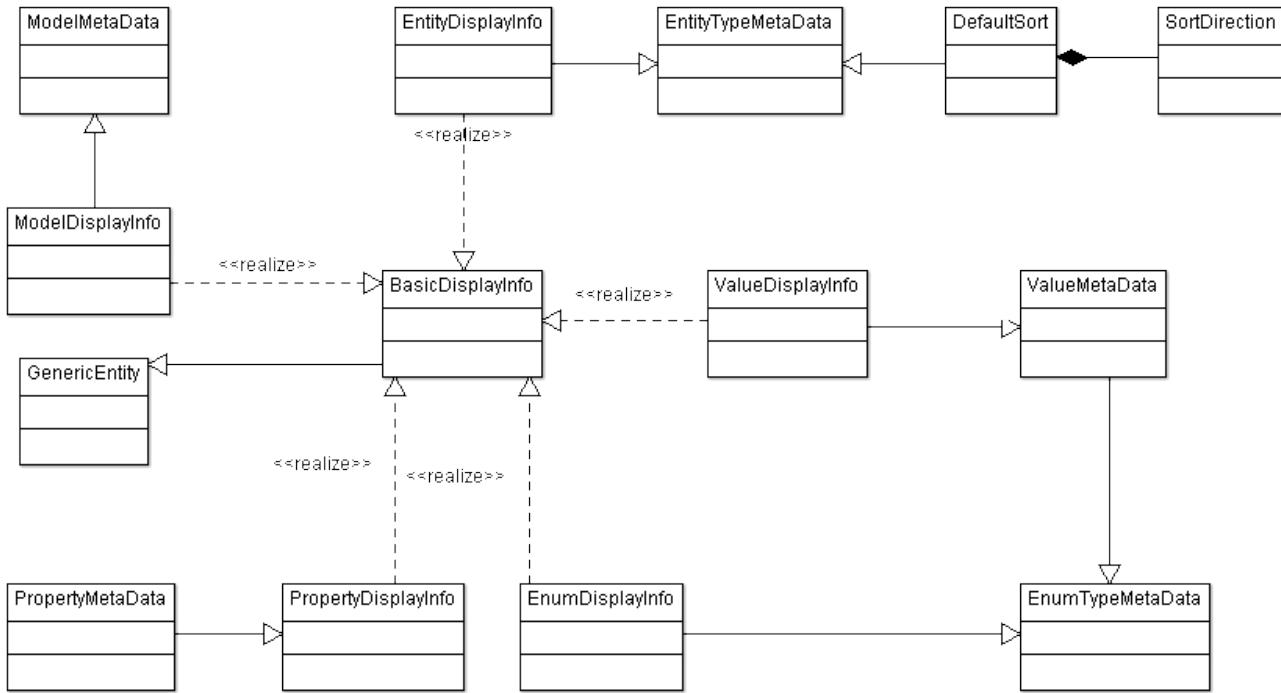
StringRegexpConstraint



Name	Type	Description
expression	String	The regular expression that the property's value must match.

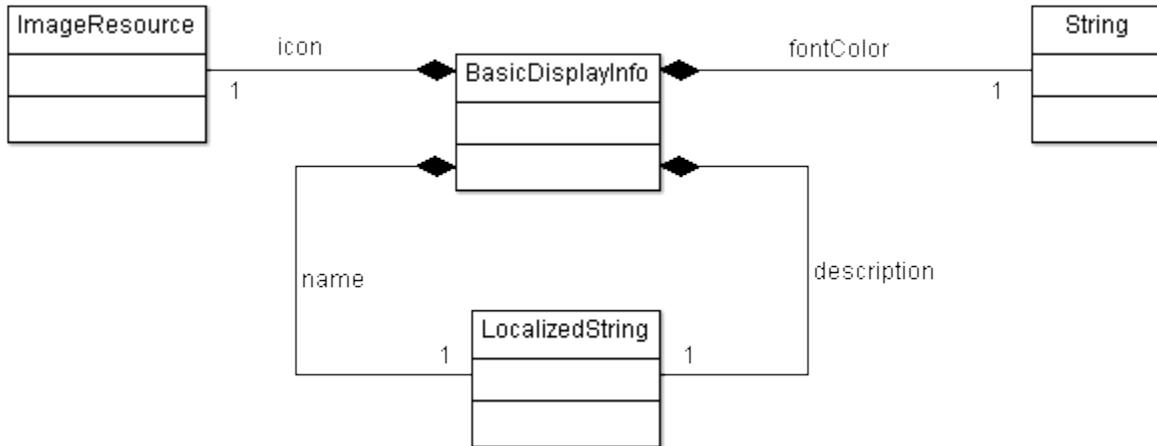
DisplayInfo

DisplayInfo is intended to provide GME with information about how to fashion the UI representation of the element that the display info is attached to.



BasicDisplayInfo

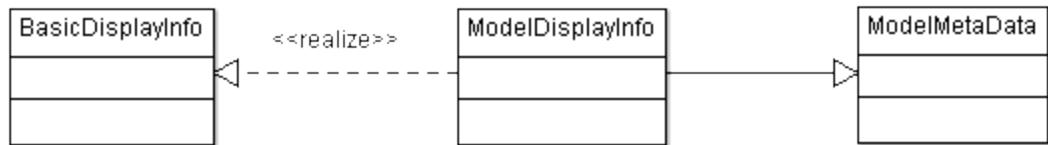
Again, this is the basic interface all DisplayInfo entities realize.



Name	Type	Description
name (interface)	LocalizedString	the name of the display info
description (interface)	LocalizedString	the description of it
icon (interface)	ImageResource	an associated icon
fontColor (interface)	String	the font color

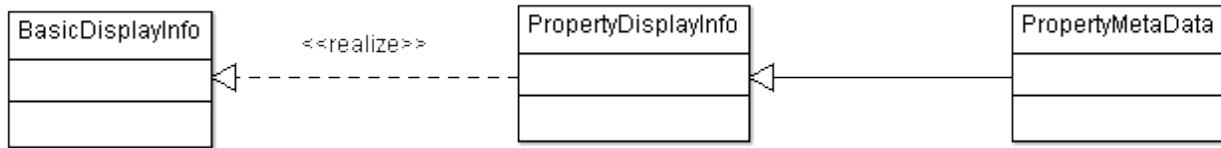
ModelDisplayInfo

ModelDisplay info is intended for Models. Obviously.



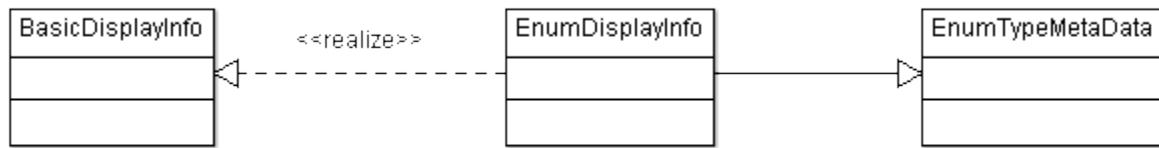
PropertyDisplayInfo

Again, this is for properties.



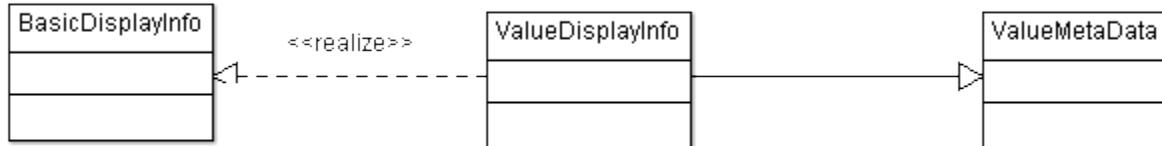
EnumDisplayInfo

And this is for enums.



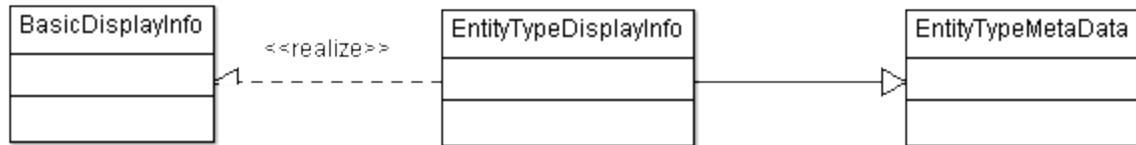
ValueDisplayInfo

For enum values.



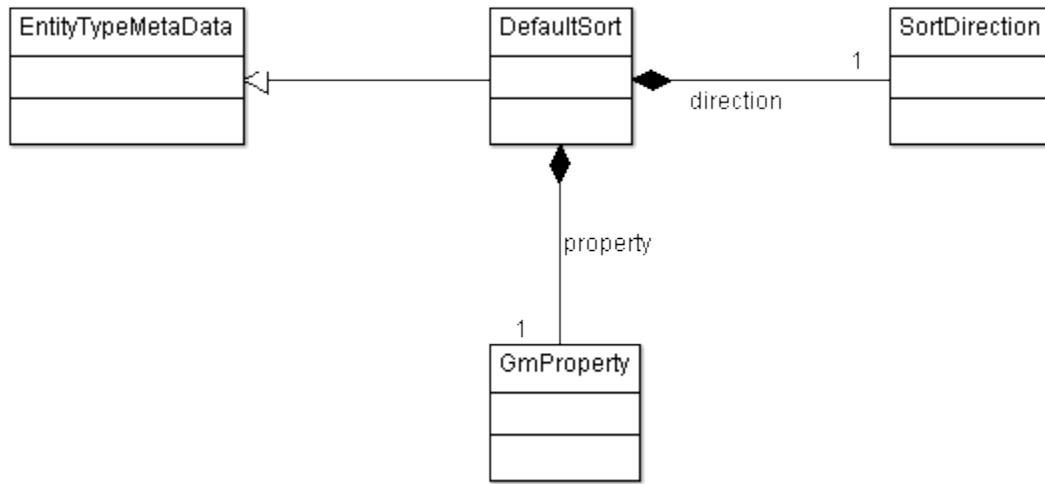
EntityTypeDisplayInfo

For entities.



DefaultSort

A sort info attached to Entities, yet property specific.



Name	Type	Description
property	GmProperty	The property the sort is attached to.
direction	SortDirection	The direction of the sort (see SortDirection Values for possible values)

SortDirection

The possible values of the SortDirection.

Value	Description
ascending	Ascending sorting
descending	Descending sorting

SelectiveInformation

Selective information is used to describe what GME should show instead of the standard which is the entity's type name.

Name	Type	Description
template	LocalizedString	The template that should be parsed if the entity's main tag should be shown.

The template is a string sequence, where properties of the respective entity can be referenced in the following manner:

```
 ${<property name>}
```

You can mix text, such as in these sequences:

```
 ${key}=${value}  
 ${build}, version ${release}
```

The property's value can be null, in this case, null is shown. It must exist however or an exception's thrown at display time.

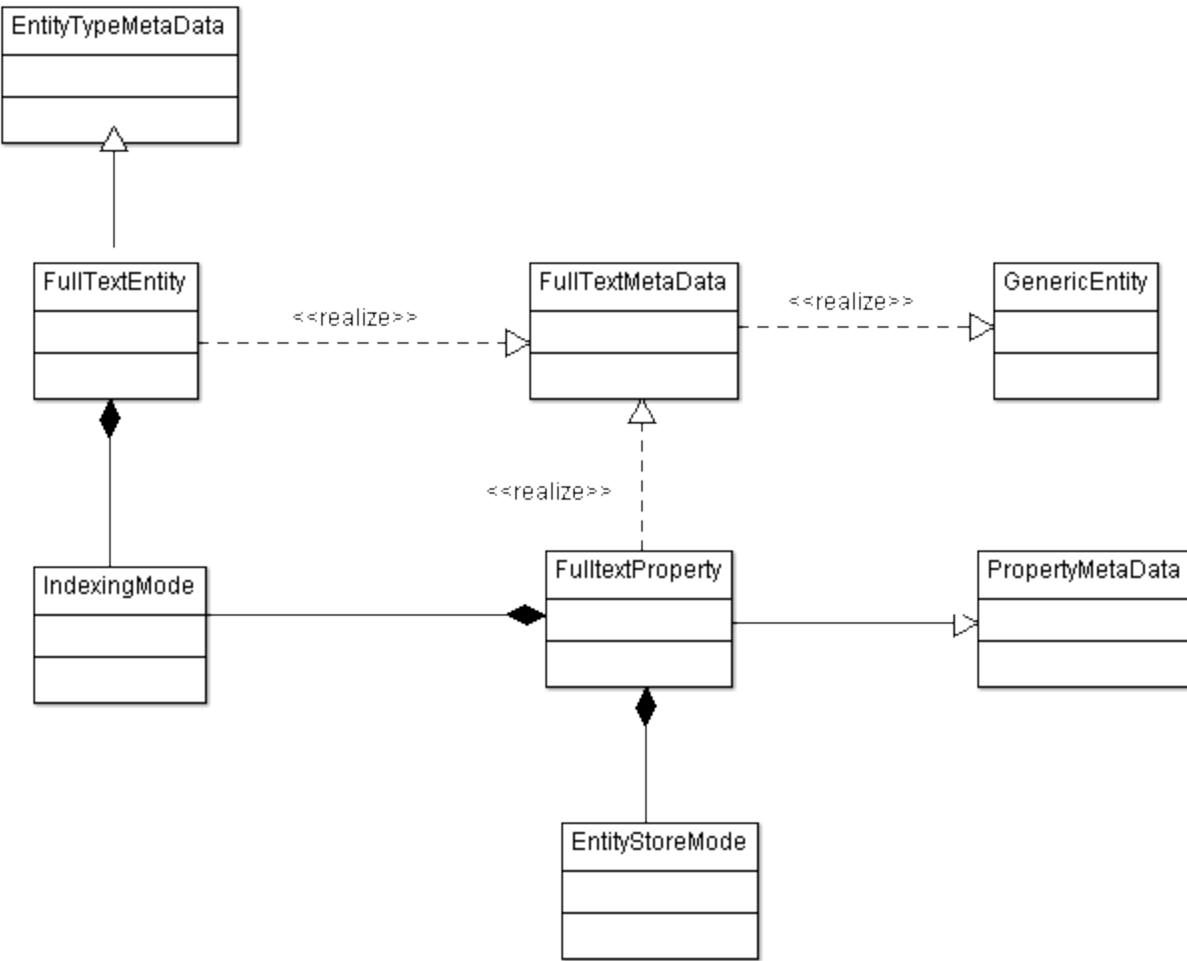
You can access values of member entities, but then must make sure that the referenced entity isn't absent at display time.

```
 ${release.build}
```

It can also be set via an annotation directly in the source code. However, if ITW(Instant Type Weaving) is active, this information is lost, therefore you are encouraged to set this only via an enriching process.

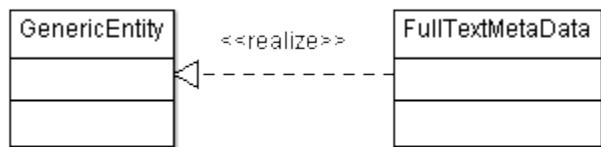
FulltextMetaData

?

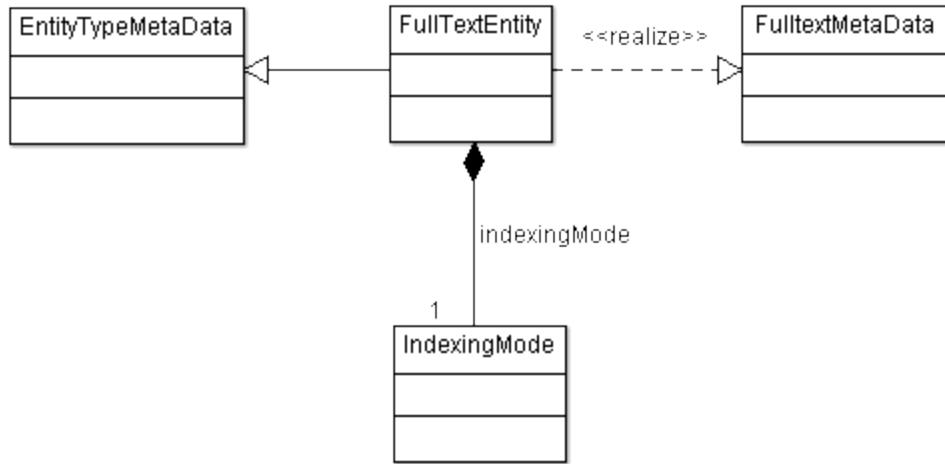


FulltextMetaData

?

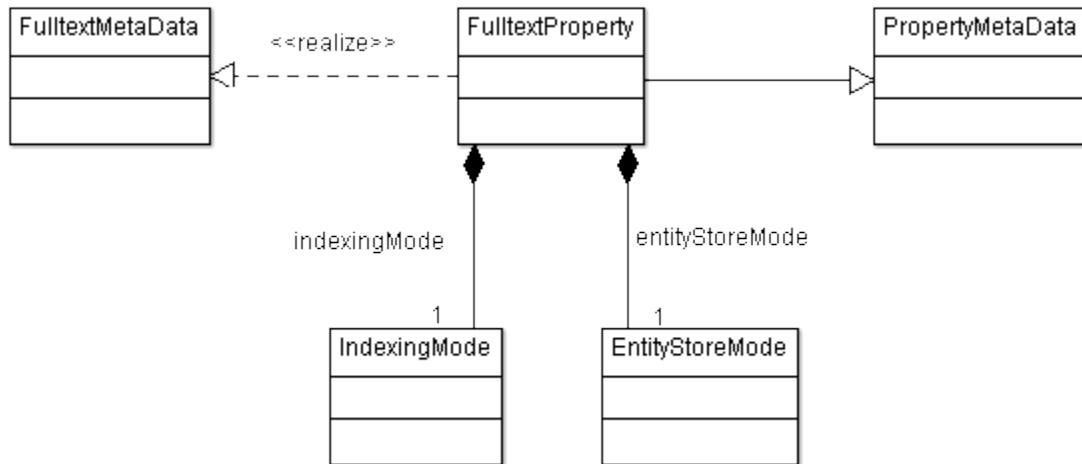


FulltextEntity



Name	Type	Description
indexingMode	IndexingMode	? see declaration of the Enum IndexingMode for possible values.

FulltextProperty



Name	Type	Description
indexingMode	IndexingMode	See the declaration of the Enum IndexingMode for possible values.
entityStoreMode	EntityStoreMode	See the declaration of the Enum EntityStoreMode for possible values.

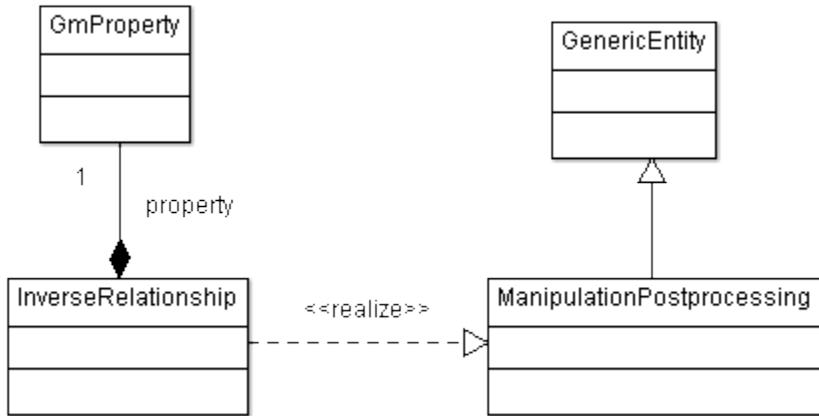
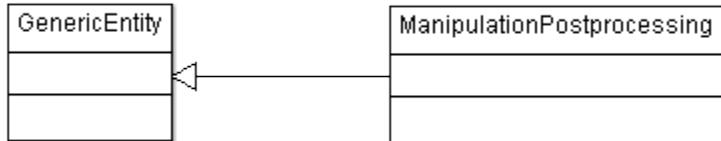
IndexingMode

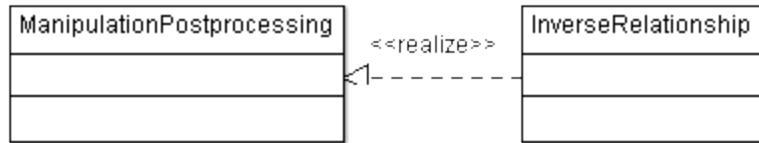
Value	Description

none	
indexedAndStored	
indexedOnly	
storedOnly	

EntityStoreMode

Value	Description
reference	
embedded	
encoded	

ManipulationPostprocessing**ManipulationPostprocessing****InverseRelationship**



Name	Type	Description
property	GmProperty	?

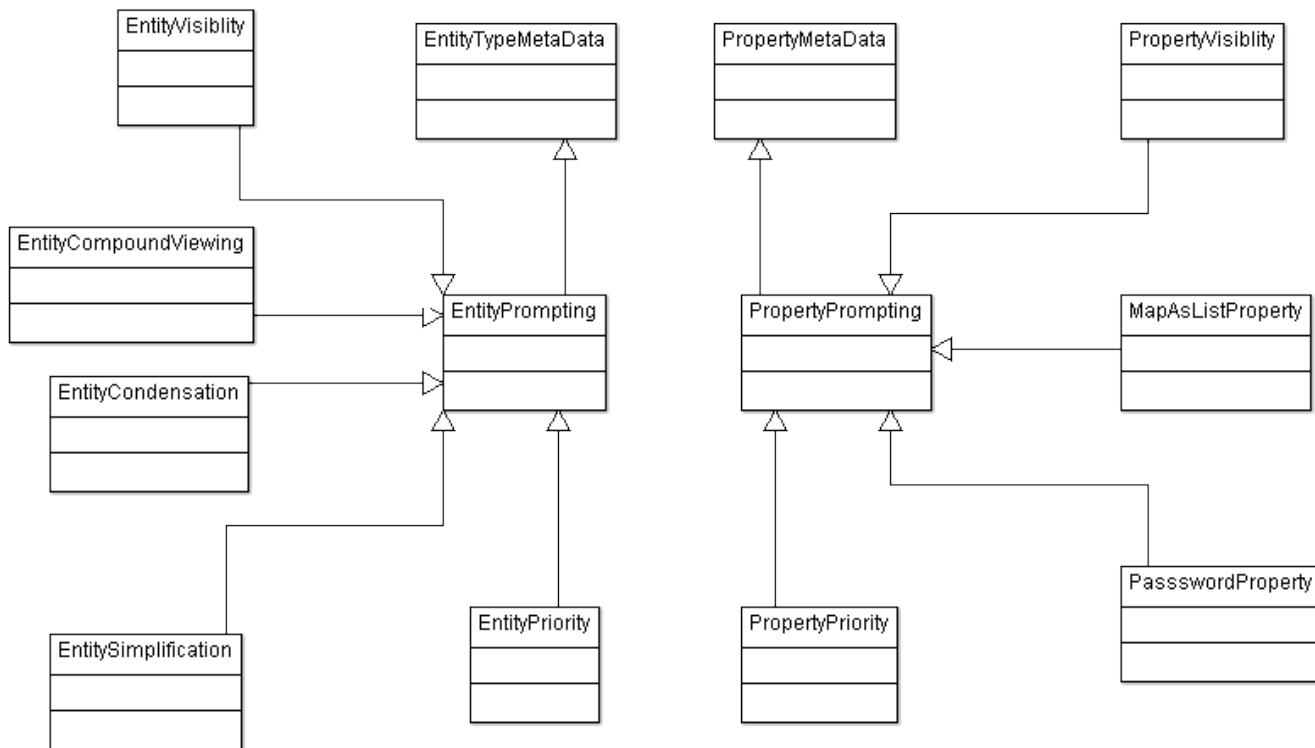
Prompting

Prompting controls how the entities or properties should be displayed in the UI.



Note

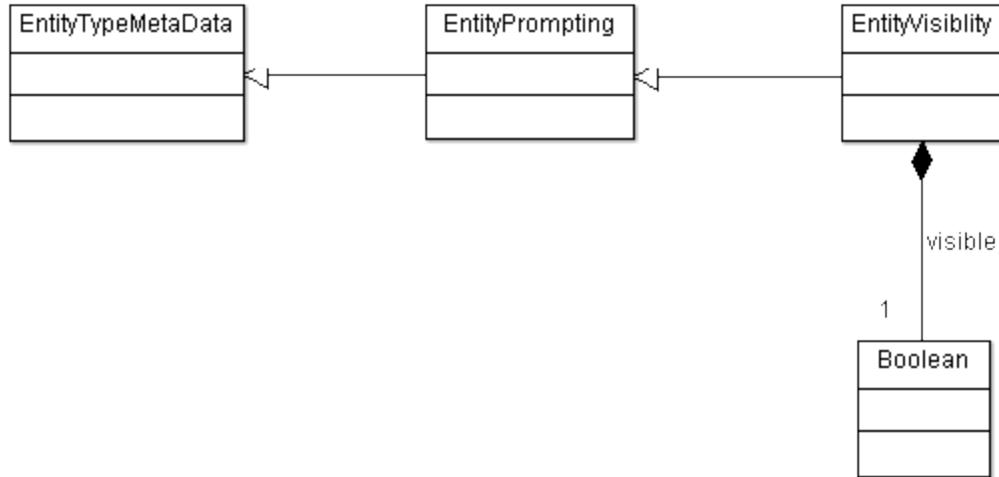
prompting in this context means showing, displaying in the UI and not as prompting can be understood in its actual, narrow meaning (as in Webster's definition: to cause (someone) to do something)



EntityPrompting

EntityVisibility

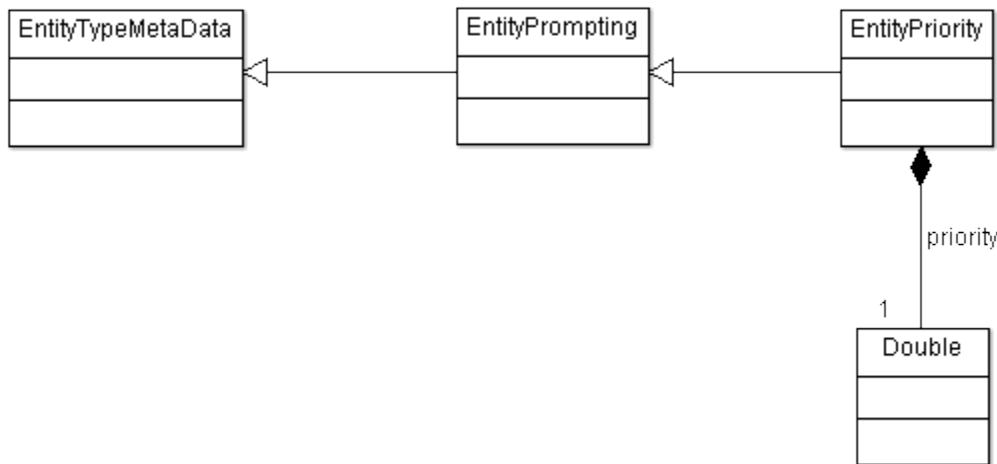
Controls the visibility of an entity. Currently, only the initial display is influenced by this setting, i.e. the left most list of entry points.



Name	Type	Description
visible	boolean	Flag whether the entity's visible.

EntityPriority

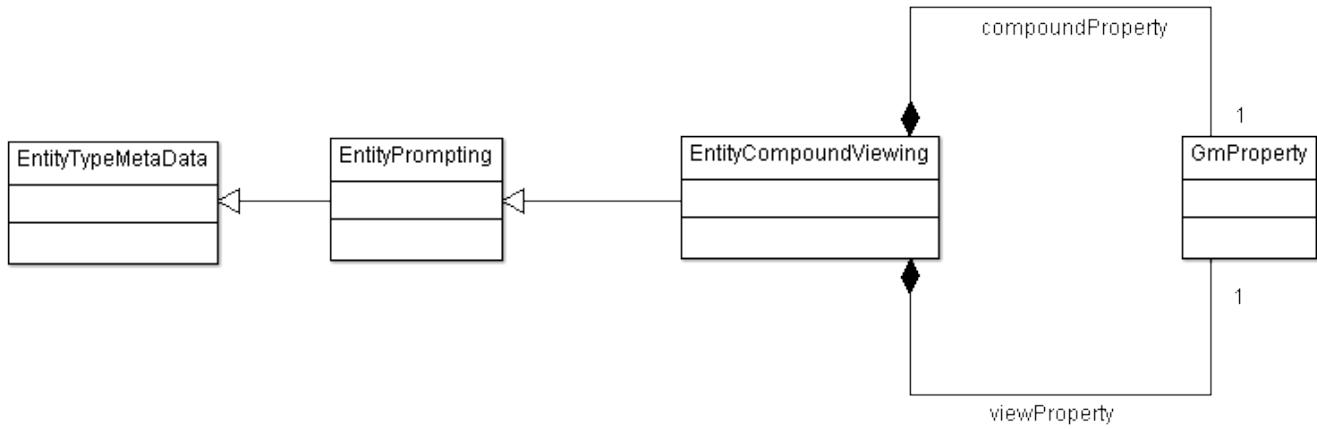
?



Name	Type	Description
priority	double	Priority in a double value between 0.0 and 1.0.

EntityCompoundViewing

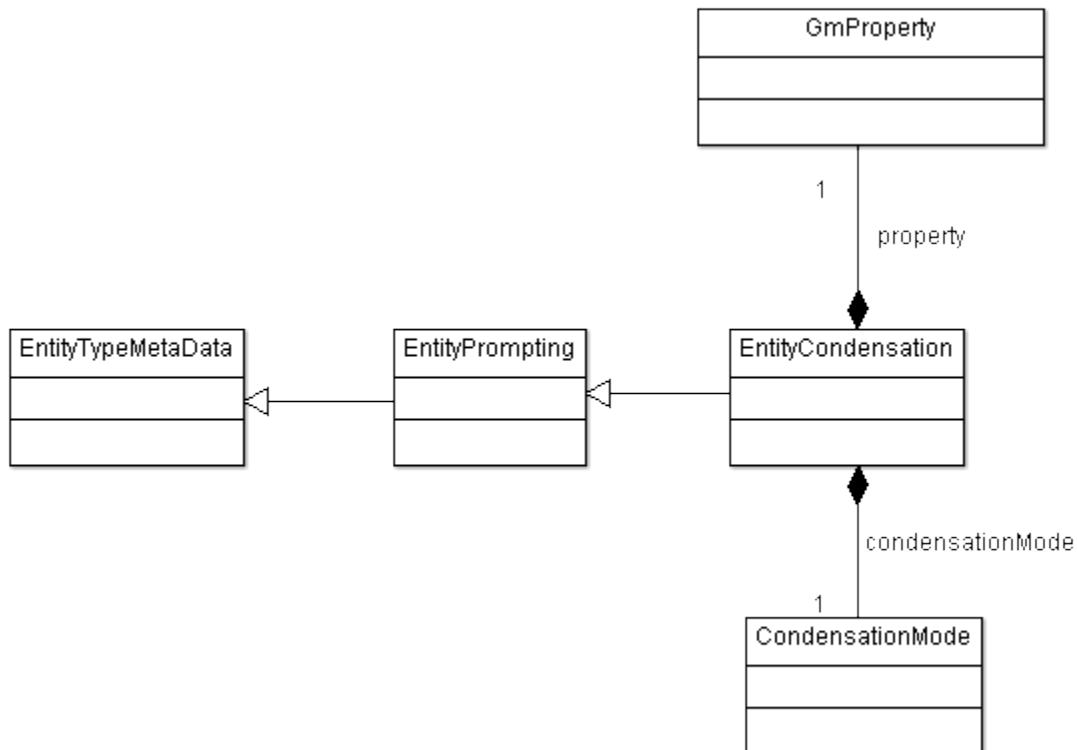
?



Name	Type	Description
compundProperty	GmProperty	?
viewProperty	GmProperty	?

EntityCondensation

MetaData that controls if an Entity has a condensed view of one of its properties.



Name	Type	Description
condensationMode	CondensationMode	For possible values see the declaration of the Enum CondensationMode below

property	GmProperty	The property the condensed view applies to.
----------	------------	---

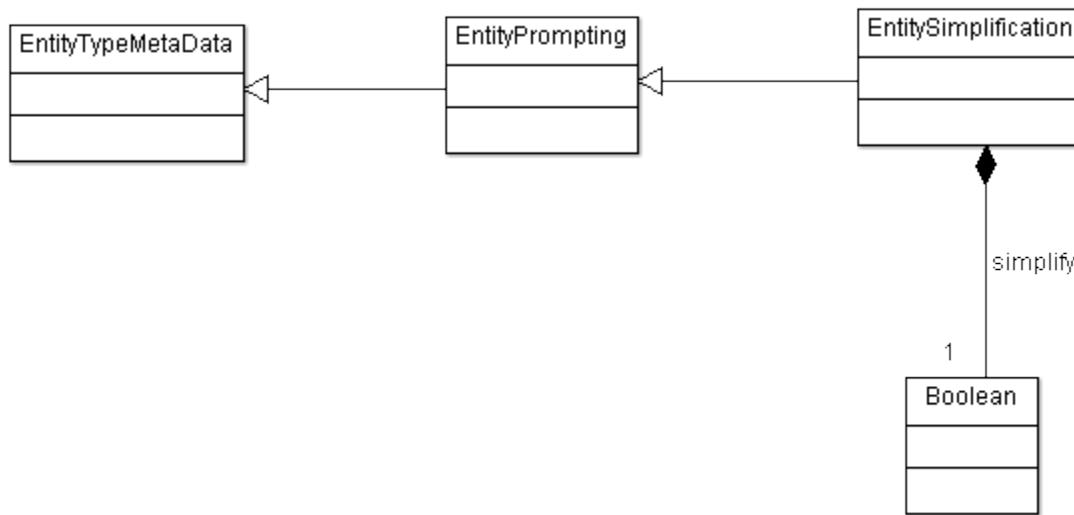
CondensationMode

Enum that declares the possible condensation modes

Value	Description
optional	Condensation view is optional.
auto	Condensation mode is controlled outside this element.
forced	Condensation mode cannot be changed, and is active.

EntitySimplification

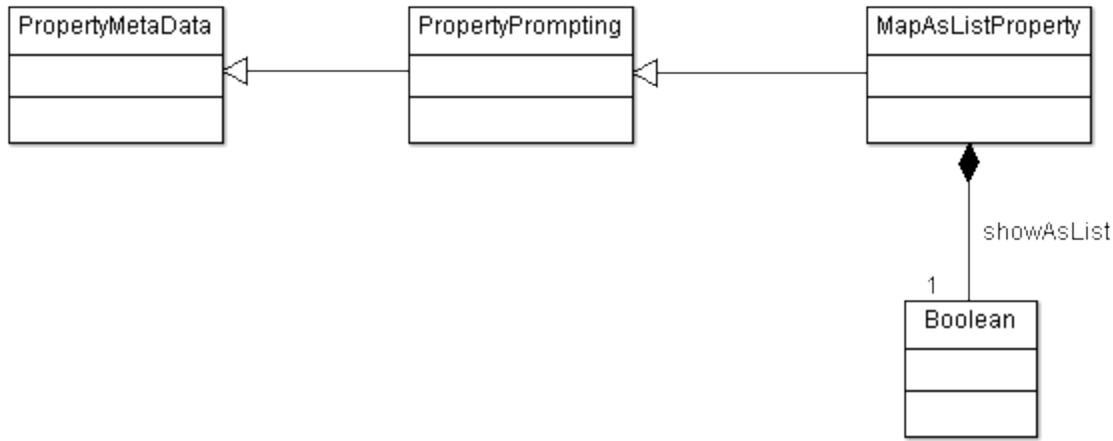
MetaData that controls whether the Entity's display is simplified ?



Name	Type	Description
simplify	boolean	If active, the entities view should be simplified, whatever that is.

PropertyPrompting**MapAsListProperty**

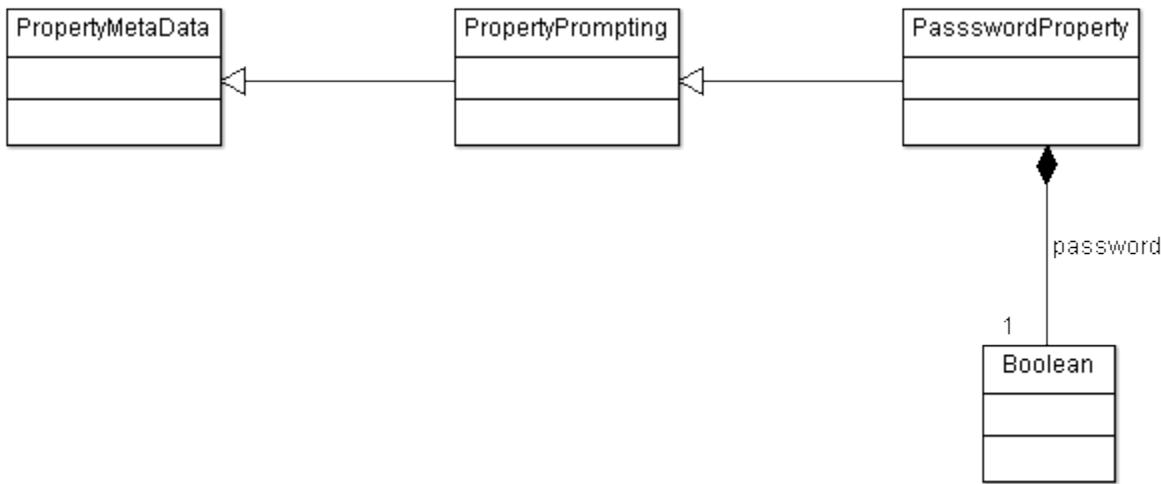
Tells whether the property being a map should be displayed as a list.



Name	Type	Description
showAsList	boolean	If active, the property should be displayed as a list.

PasswordProperty

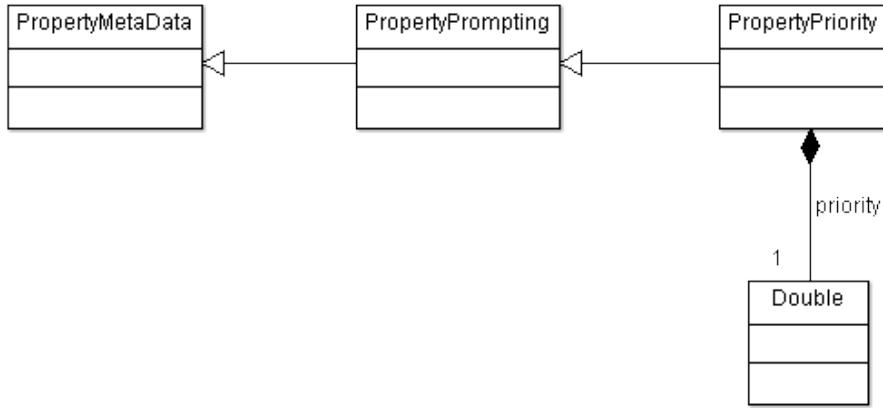
Flags a property to be displayed as a password.



Name	Type	Description
password	boolean	If active, the property should be displayed as a password.

PropertyPriority

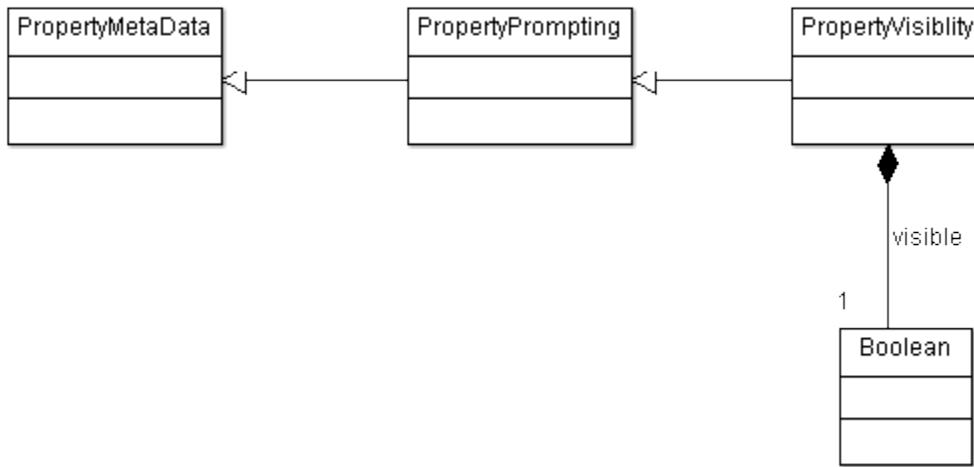
Determines the priority of the property.



Name	Type	Description
priority	double	Priorit in a double value between 0.0 and 1.0

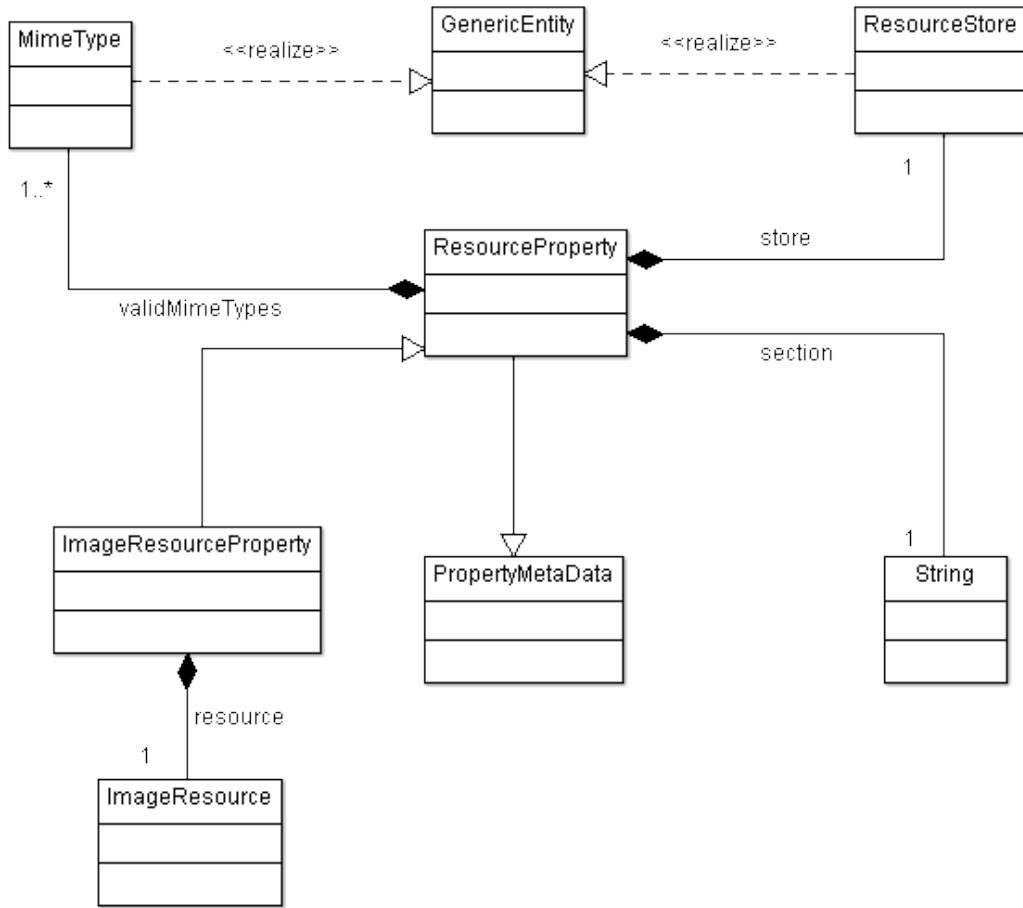
PropertyVisibility

Controls the visibility of the property.



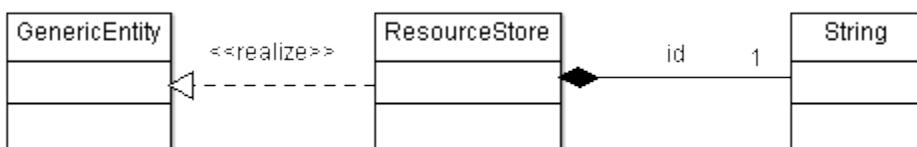
Name	Type	Description
visible	boolean	Flag whether the property's visible.

Resources



ResourceStore

??



Name	Type	Description

id	String	If of the storage record of the resource.
----	--------	---

MimeType

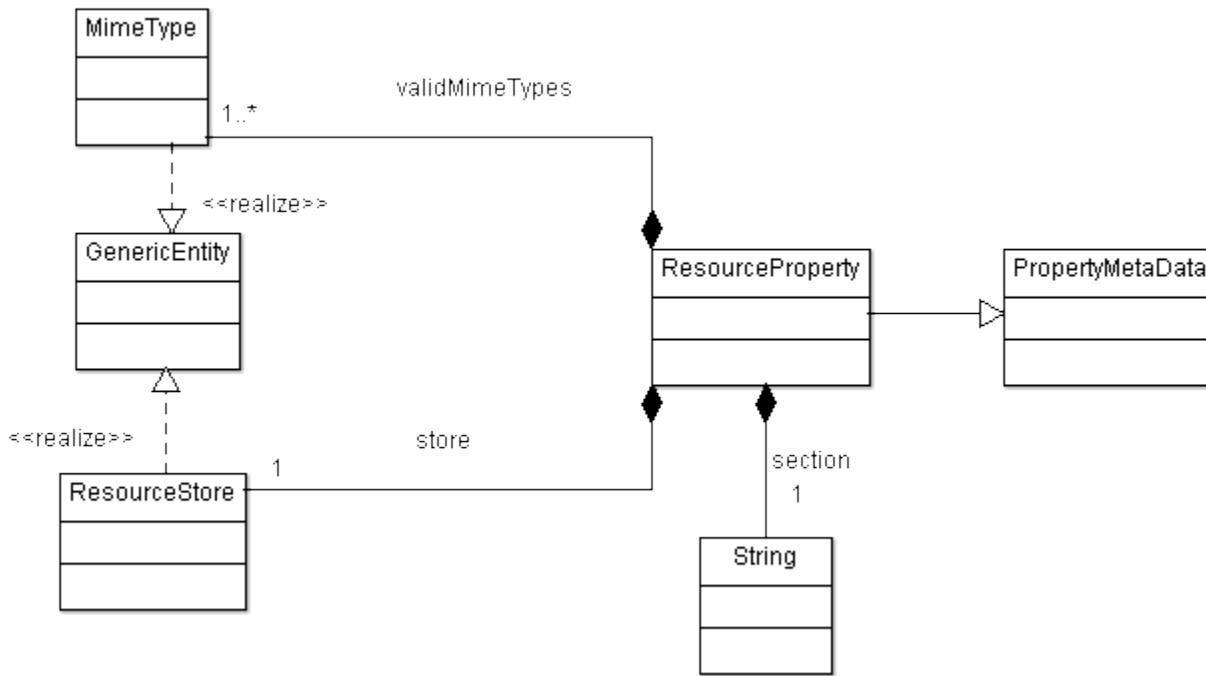
A MimeType represents a mime type, as it says by its name.



Name	Type	Description
name	String	The name of the mime type.

ResourceProperty

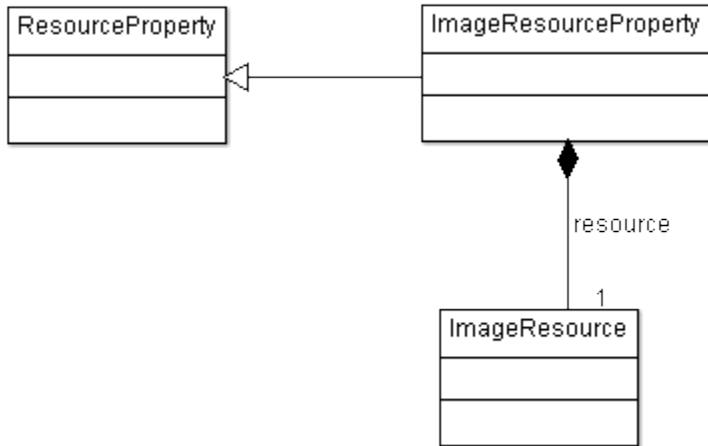
A ResourceProperty represents a resource, specified by its mime type and id in the persistence level.



Name	Type	Description
validMimeTypes	Set<MimeType>	A set of valid mime types for the property.
store	ResourceStore	persistence container.
section	String	?

ImageResourceProperty

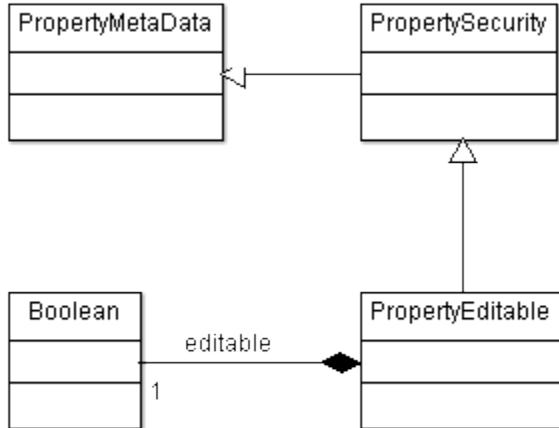
An image-specific resource property.



Name	Type	Description
resource	ImageResource	Associated image resource.

Security

Securities control the access to a property. Currently, only the editing of a property is controlled via this mechanism.

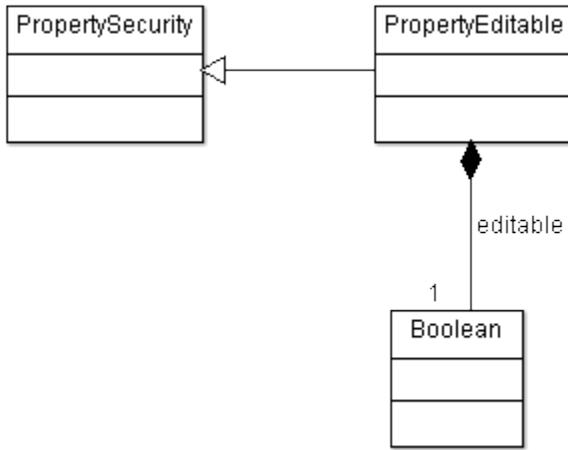


PropertySecurity

The base class for all property related security issues.



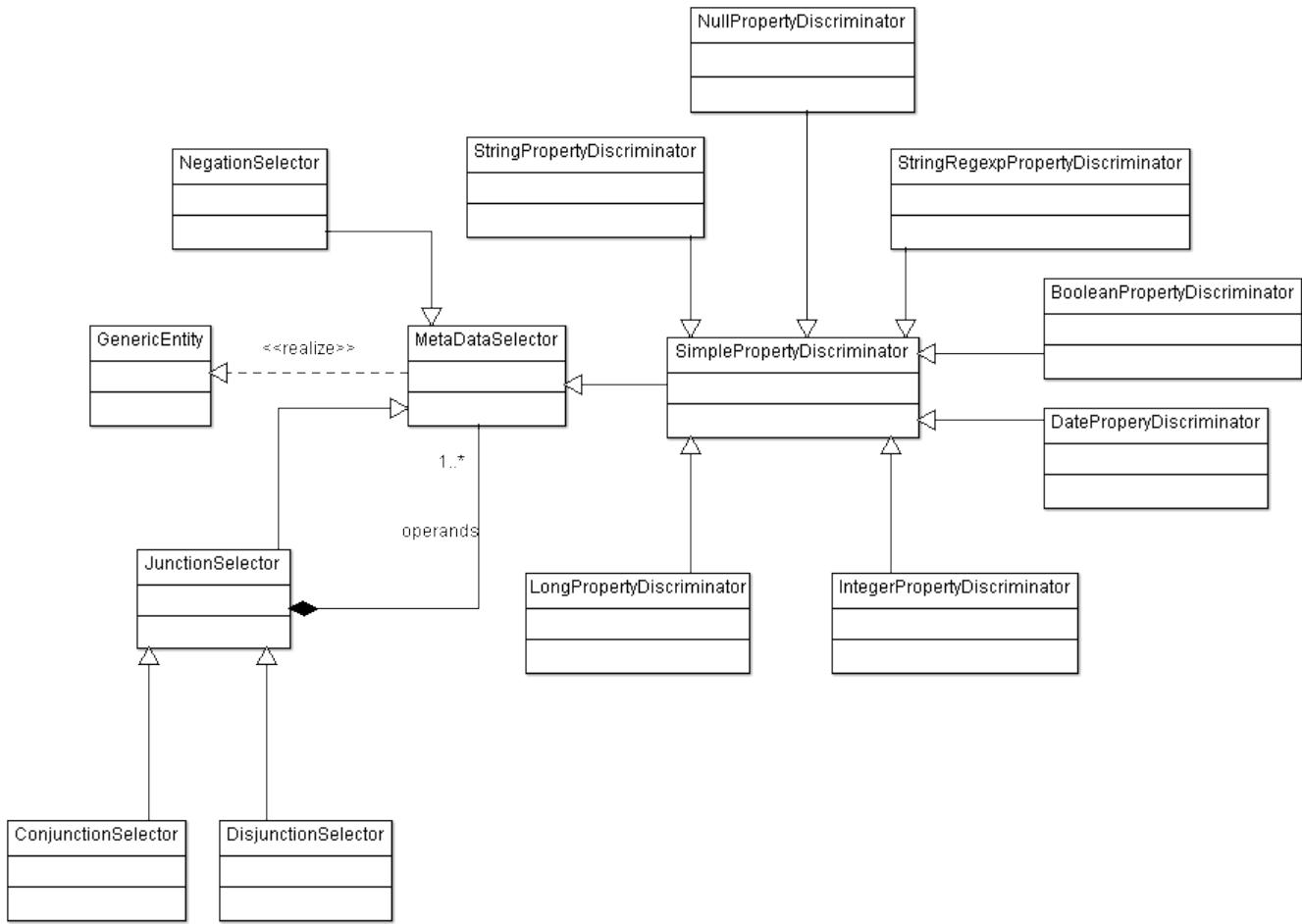
PropertyEditable



Name	Type	Description
editable	boolean	Flag whether the property can be edited.

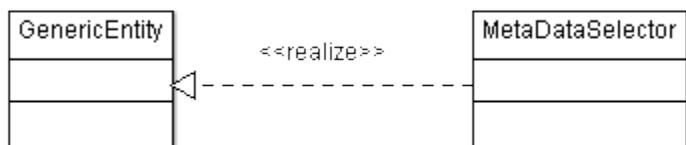
Selectors

Selectors control whether the associated meta data is applicable or not, i.e. they are a kind of activation criteria. A selector in the end evaluates to either true or false, i.e. it matches or it doesn't.

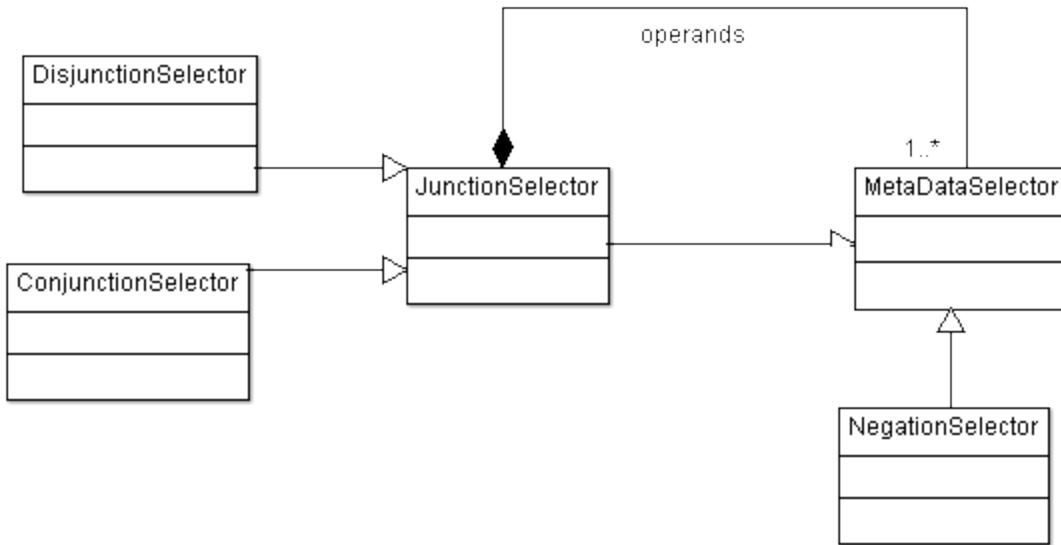


MetaDataSelector

The MetaDataSelector is the base class of all selectors.



Logical selectors



NegationSelector

The negation selector just negates the intrinsic matching value.

JunctionSelector

A junction selector logically joins all operands - each a MetaDataSelector again - it owns.

Name	Type	Description
operands	List<MetaDataSelector>	A list MetaDataSelector whose evaluated values are to be combined.

ConjunctionSelector

A conjunction selector is an junction selector that joins its operand's value as a conjunction i.e. in order for it to match, all operands must match.

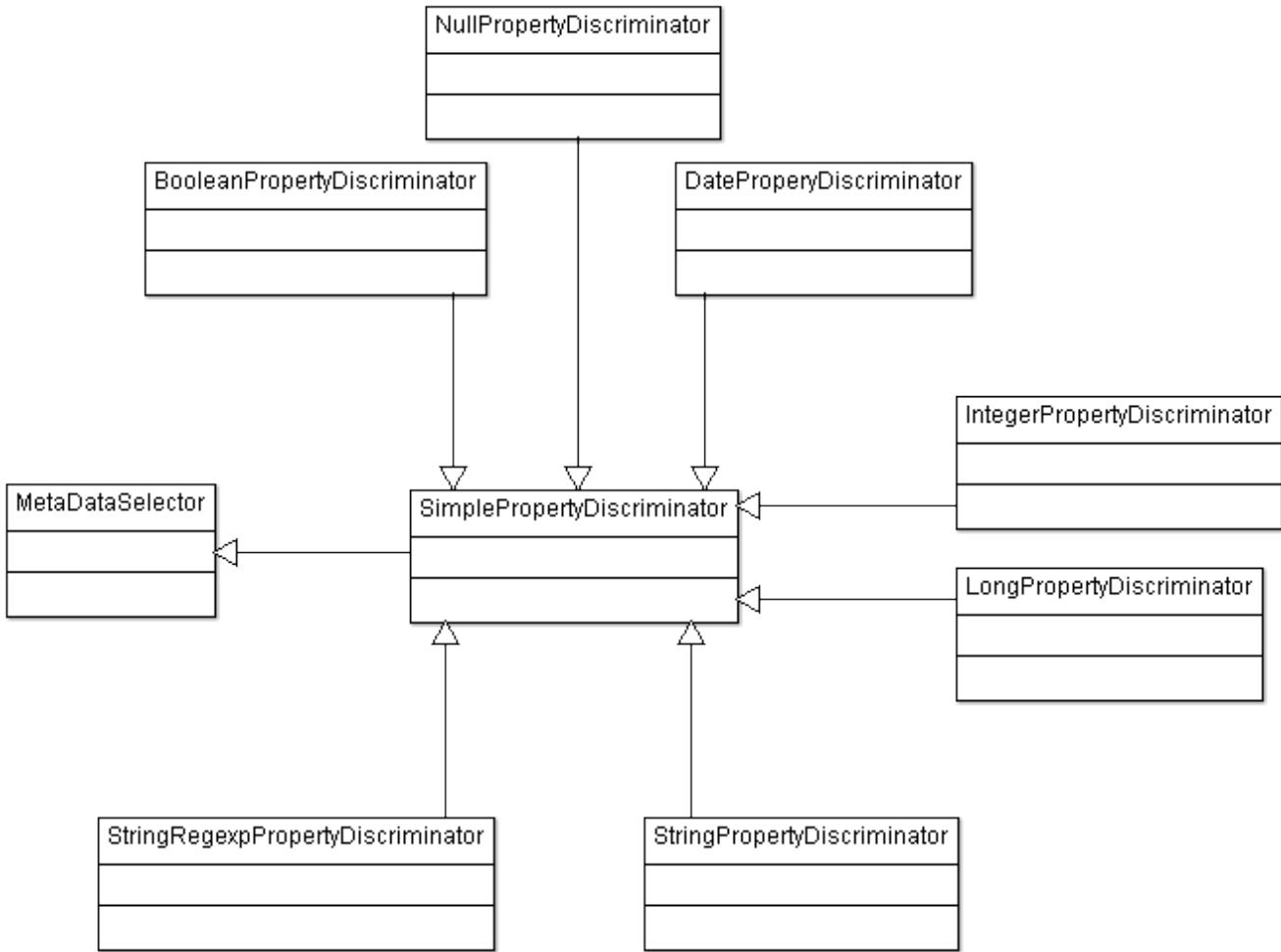
DisjunctionsSelector

A conjunction selector is an junction selector that joins its operand's value as a disjunction i.e. in order for it to match, a single operand of the list must match.

Property discriminators

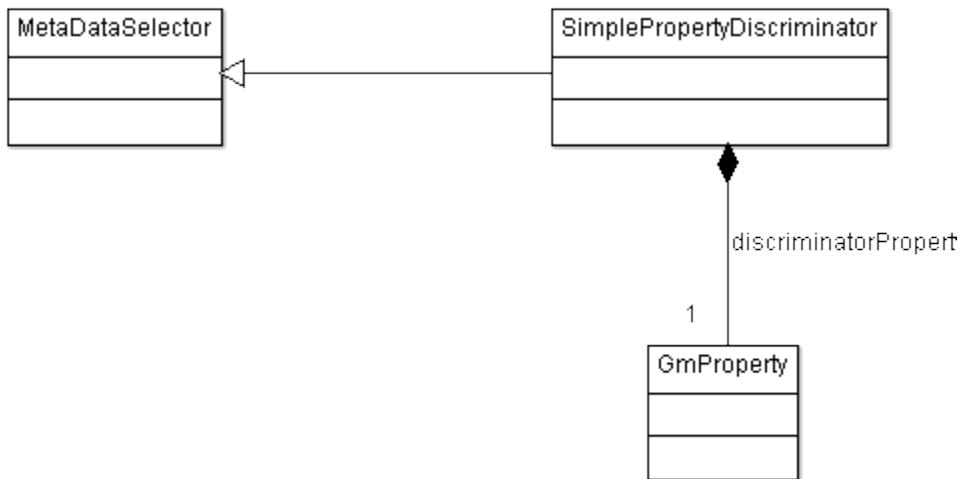
Discriminators are the same as selectors (they share the derivation axis of the logical selectors), but by whatever reasons they're named differently.

The discriminator all possess a “discriminator value” which is nothing more than a stable comparison value, i.e. the value of the associated property must match the value in order to let the discriminator evaluate to true.



SimplePropertyDiscriminator

The base class of all property discriminators is this discriminator. It introduces the property that the discriminator is attached to.

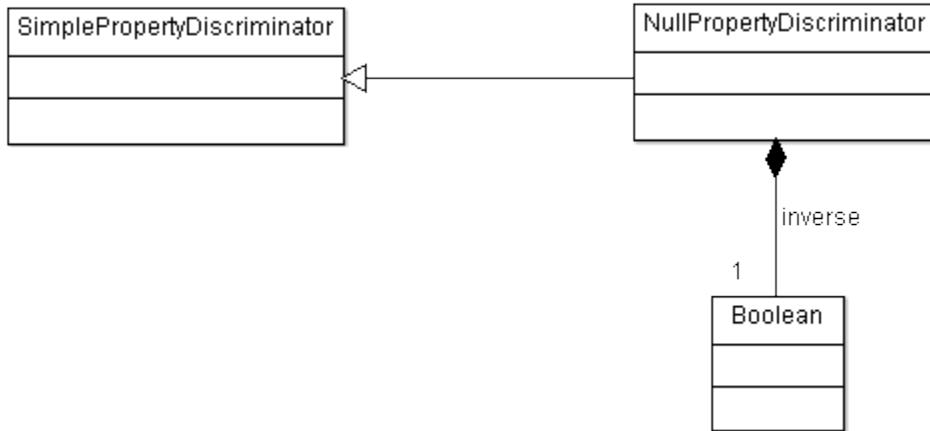


Name	Type	Description
------	------	-------------

discriminatoryProperty	GmProperty	The property the discriminator discriminates against - just joking: the property that the discriminator is attached to.
------------------------	------------	---

NullPropertyDiscriminator

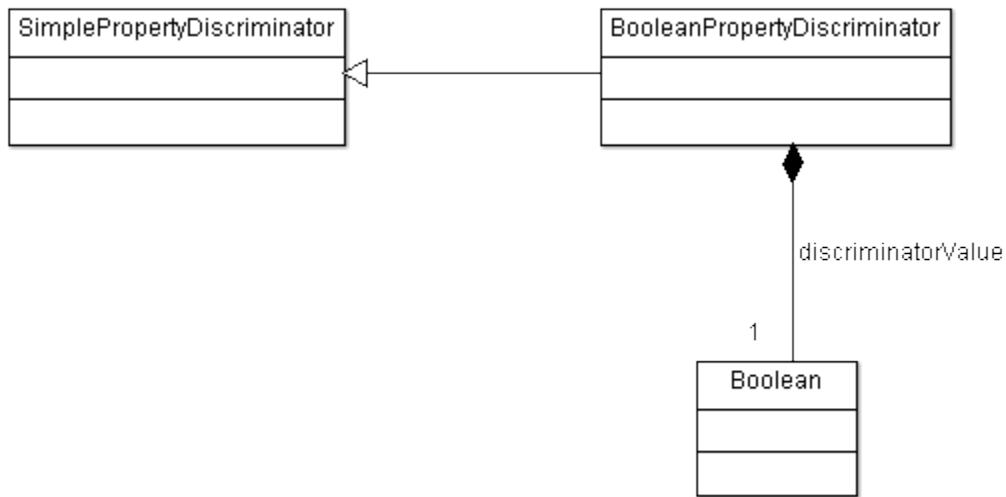
A null property discriminator allows to match on a null or non-null value of a property.



Name	Type	Description
inverse	boolean	Flag whether the discriminator matches a null or non-null value. Alternative to using a NegationSelector.

BooleanPropertyDiscriminator

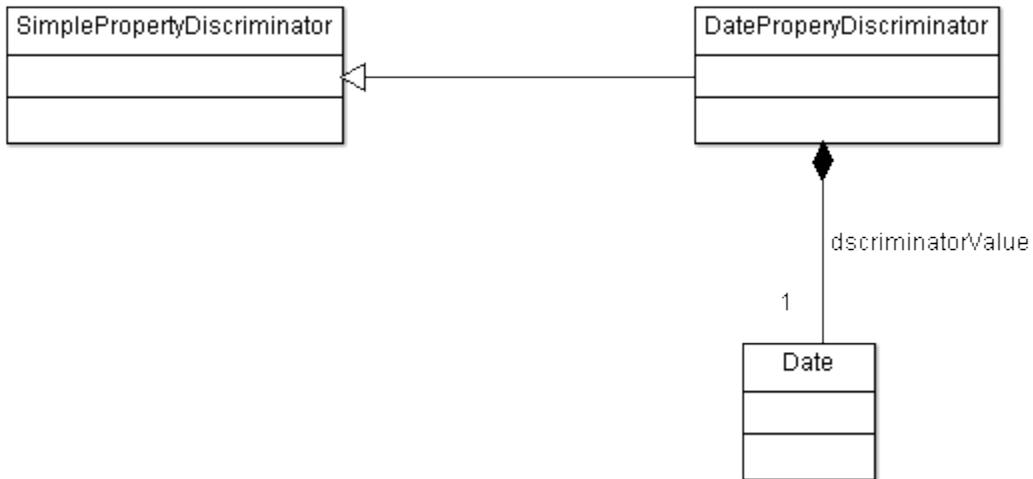
A boolean property discriminator matches on a boolean value of a property.



Name	Type	Description
discriminatorValue	boolean	The comparison value is a boolean value.

DatePropertyDiscriminator

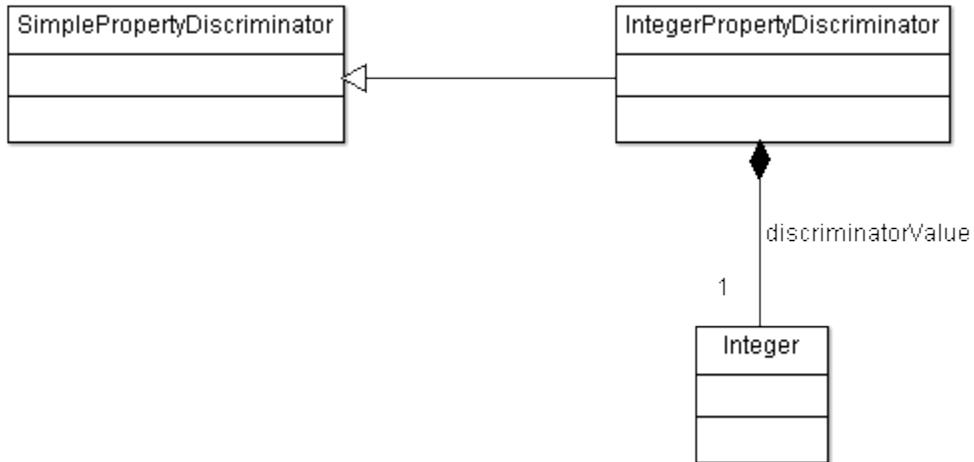
A data property discriminator matches on a date value of property.



Name	Type	Description
discriminatorValue	Date	The comparison value is a date value.

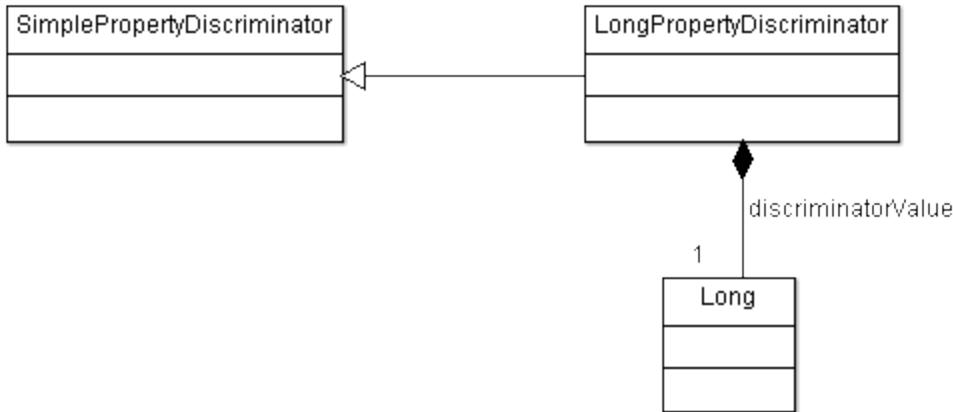
IntegerPropertyDiscriminator

An integer property discriminator matches on



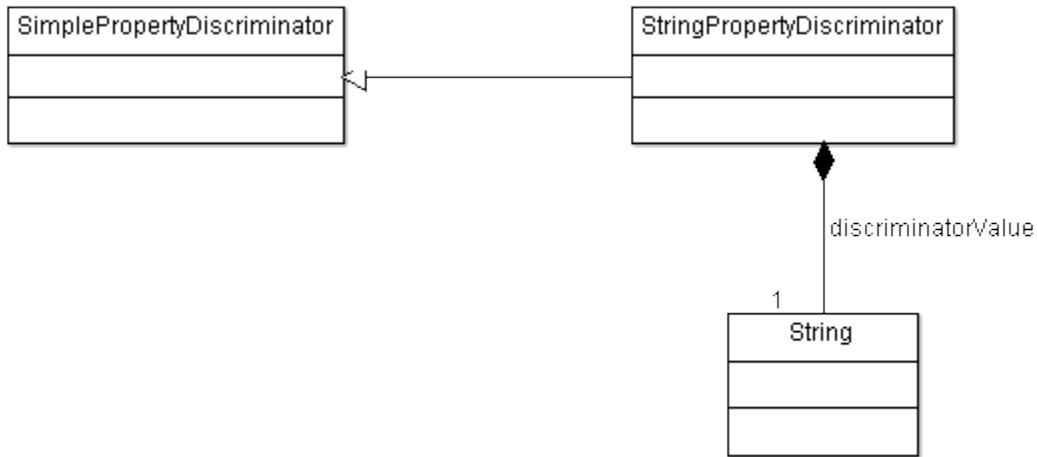
Name	Type	Description
discriminatorValue	int	The comparison value is an integer value

LongPropertyDiscriminator



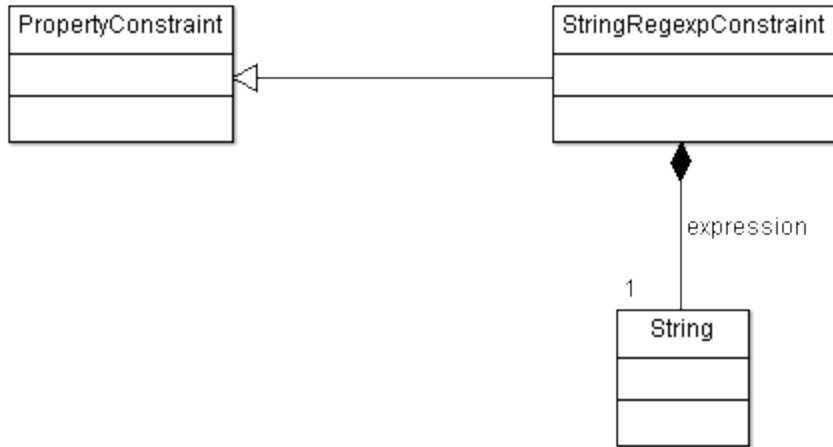
Name	Type	Description
discriminatorValue	long	The comparison value is a long.

StringPropertyDiscriminator



Name	Type	Description
discriminatorValue	string	The comparison value is a string.

StringRegexpPropertyDiscriminator



Name	Type	Description
discriminatorRegexp	String	The regular expression pattern to be used for comparison.

The Query Model

Table of Contents

Expand / collapse

- [Introduction](#)
- [Overview](#)
 - [Conditions an 'Query' base class](#)
 - [Queries](#)
 - [Results](#)
 - [Queries](#)
 - [Populations](#)
 - [Restrictions](#)
 - [Ordering](#)

Introduction

One of the most important models of all is the QueryModel. It allows to describe a query in a generic way that can be interpreted by the actual query processors of the different persistence layers. As it is central, crucial, but most importantly valid for any persistence system, a lot of thought went into it.

Even if the QueryModel is very well suited to describe a query, it is not suited to be processed by a query processor. Therefore, the QueryPlanModel was conceived, see its description at <https://docs.google.com/a/braintribe.com/document/d/1BRNrGkDv0dW6yhkY47cwyHiwmEqQ21uYajUrll0pj8M>

So if you ever thought that querying is a piece of cake welcome to the club and at the same time, abandon all hope.

Overview

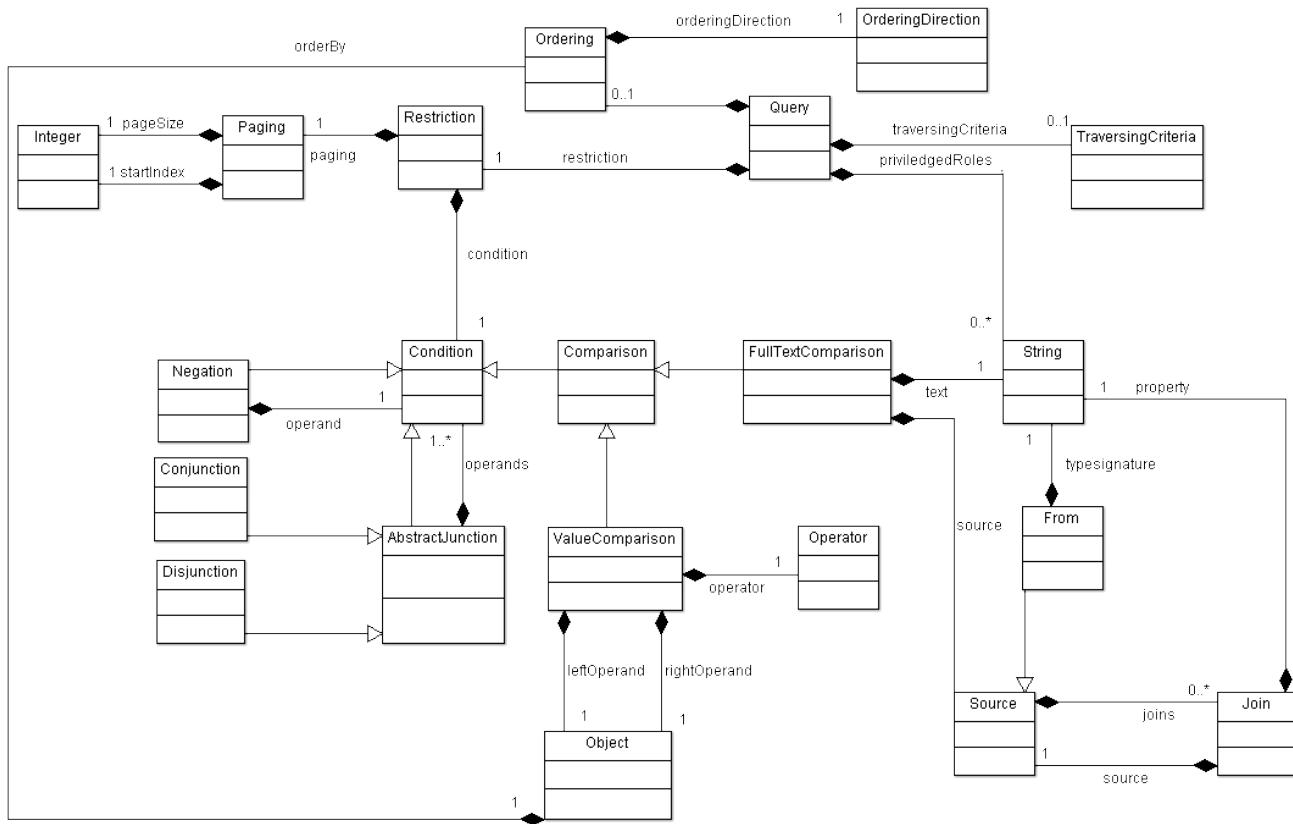
Conditions an 'Query' base class

The overview gives a partial impression of the complexity of the model even though quite a few classes and relations are deliberately not shown. We'll discuss everything later on, so use this overview just to get your bearings.

Missing are the different kinds of queries as only the base class 'Query' is shown.

Unfortunately, we had to break the expressiveness of the model in order to incorporate the flexibility we need. In some places, see the classes 'Ordering', 'ValueComparison' for instance, we had to use the lowest possible common denominator 'Object' to allow to dock complex structures to it.

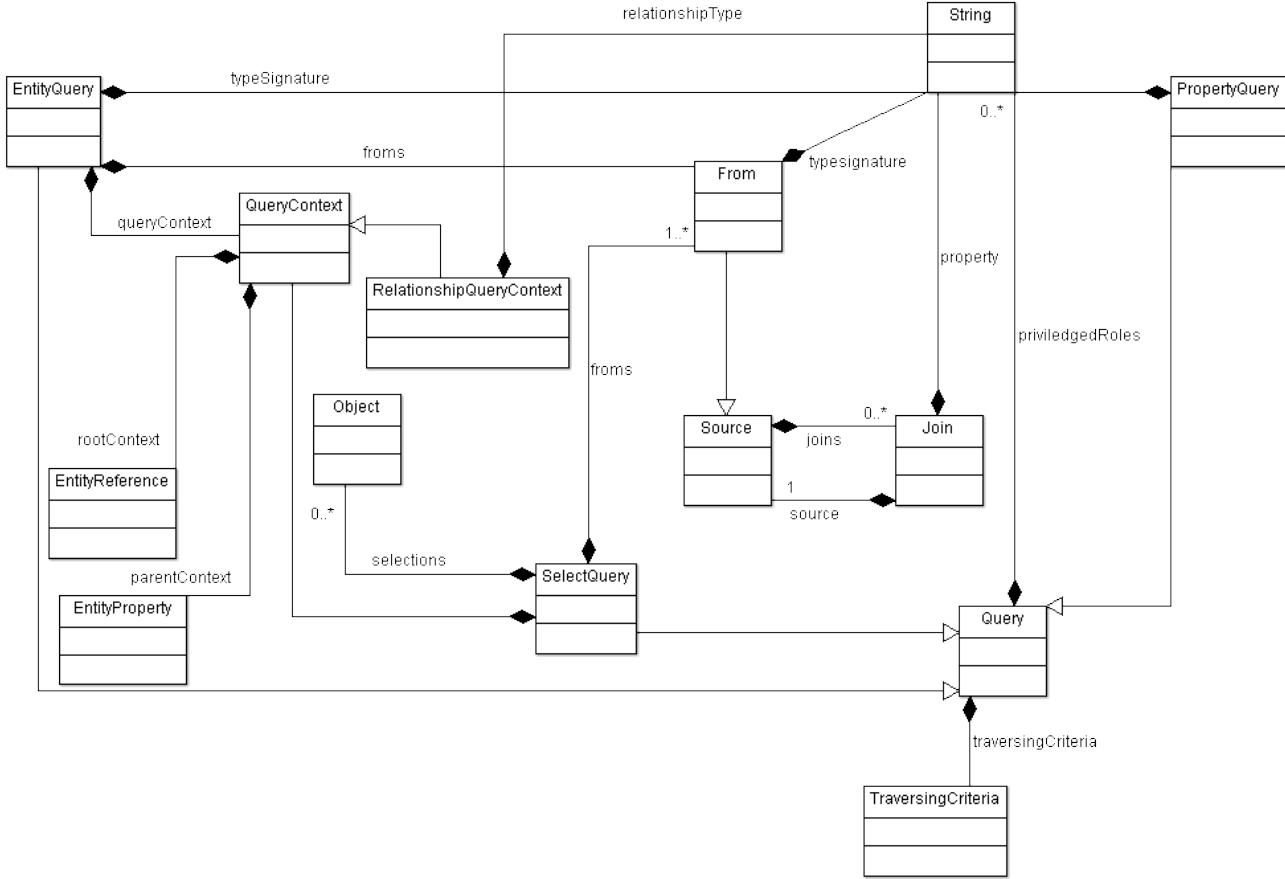
In case of the 'Ordering' class, it's because of the requirement to allow for cascading or nested sorting in case of the 'ValueComparison', the need to be able to freely inject query functions into the mix, was triggering the decision. And, no: we're not happy with it.



Queries

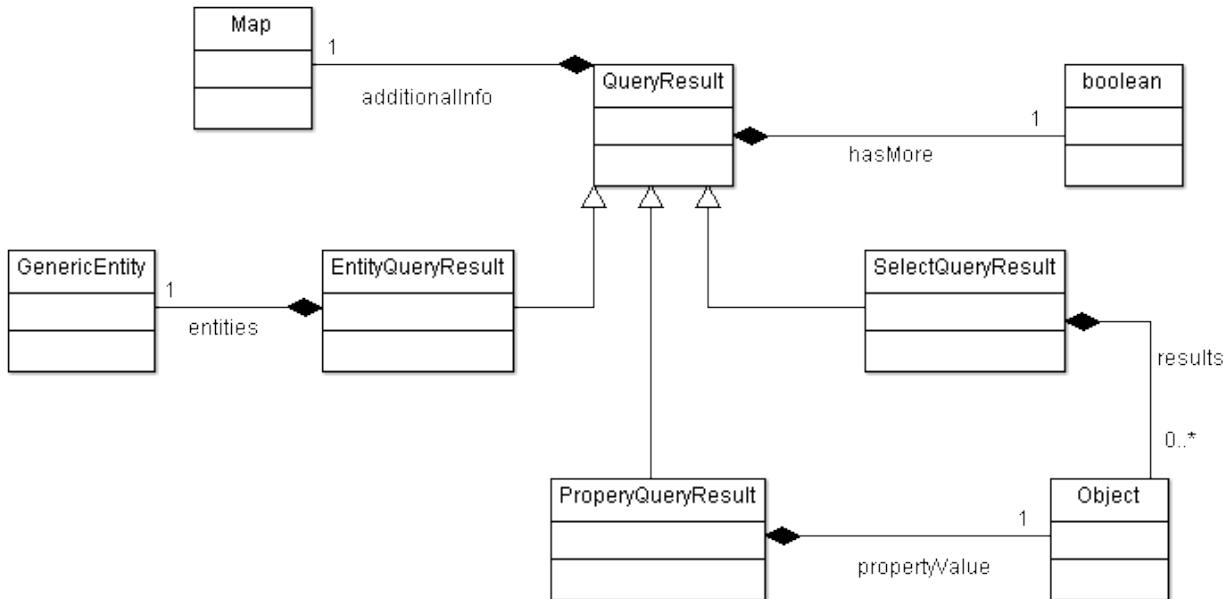
This view here shows the queries that were missing in the overview diagram above. The base class 'Query' is the starting point of the diagram. Some elements, such as the EntityReference, EntityProperty and the TraversingCriteria are imported from other models.

Imported types are specially marked in the type discussions.



Results

Results of the queries are similarly complex each query has its own specific result set.

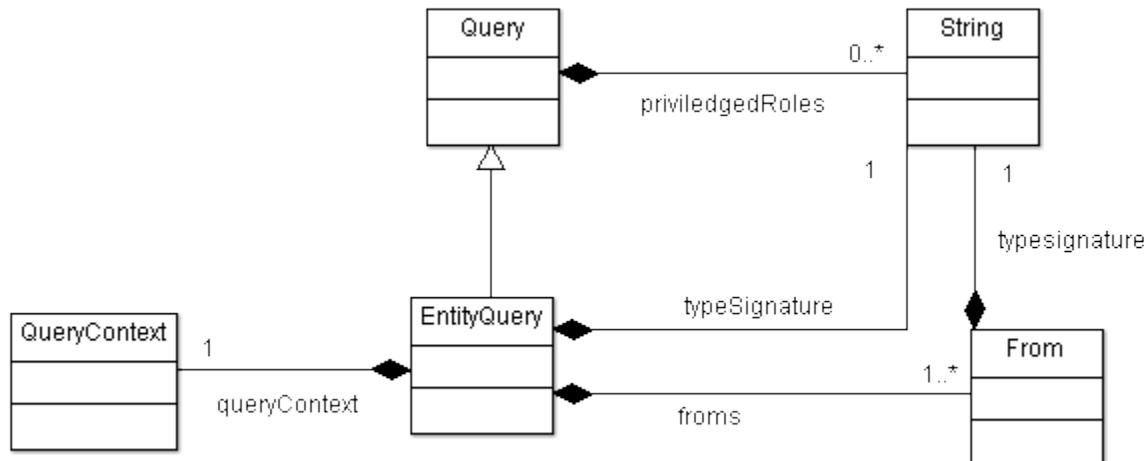


Queries

Please note that not all persistence layers support all query types.

EntityQuery

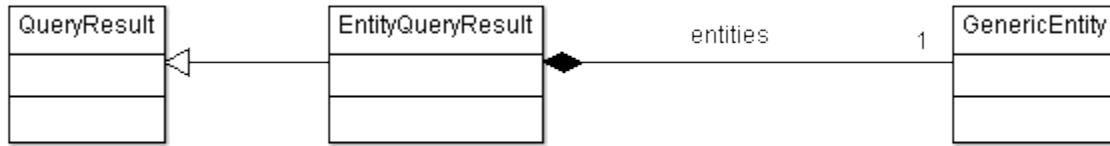
An EntityQuery aims to retrieve elements that match the criteria passed.



Name	Type	Description
queryContext	QueryContext	The associated query context.
typeSignature	String	The type signature of the population to query
froms	From	A list of associated Froms

EntityQueryResult

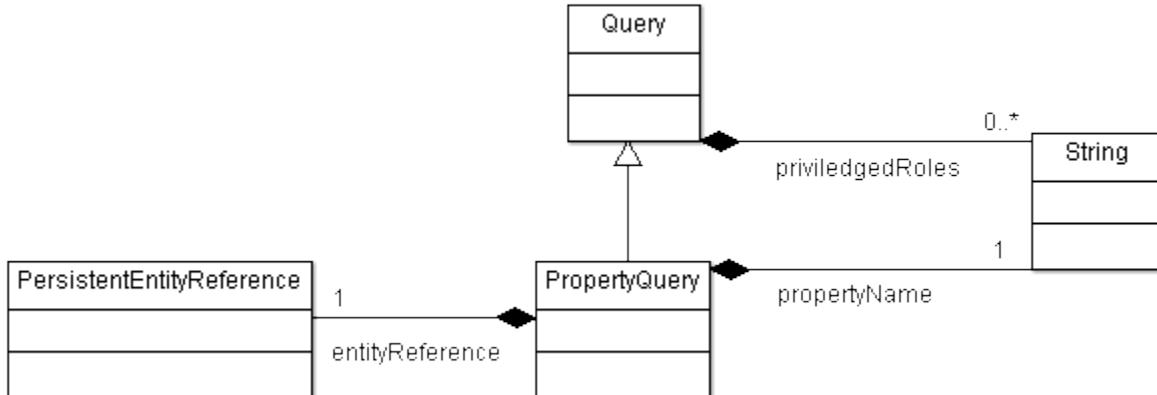
The query just returns a simple list of entities



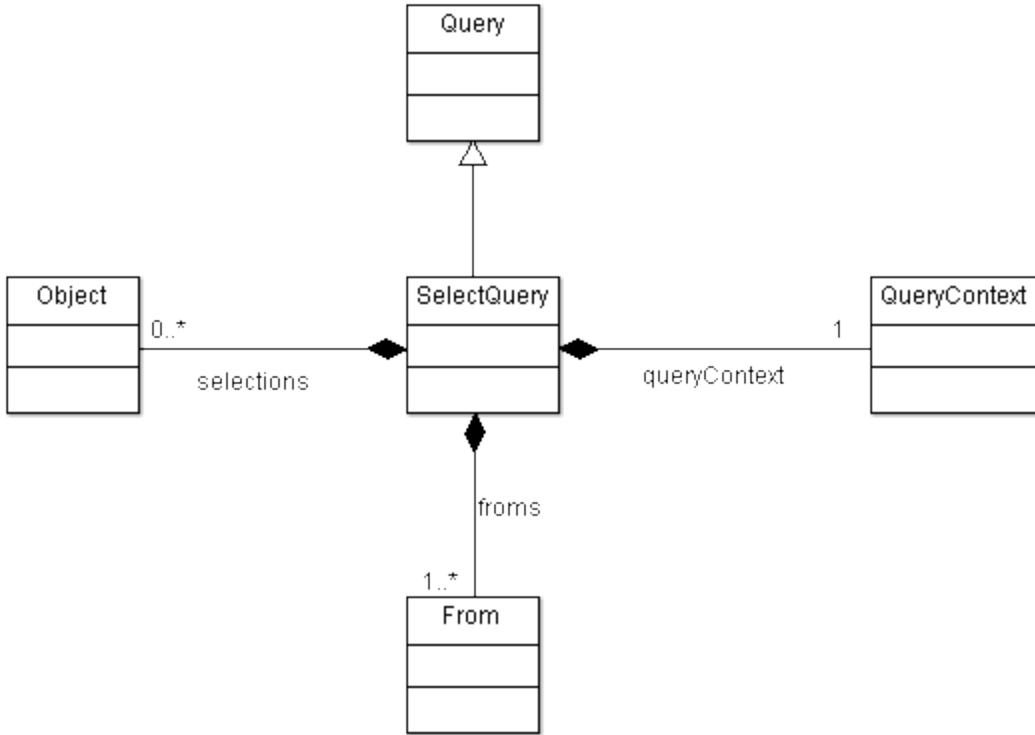
Name	Type	Description
entities	List<GenericEntity>	A list of GenericEntities that fulfilled the queries, sorted and portioned as requested.

PropertyQuery

A Property Query

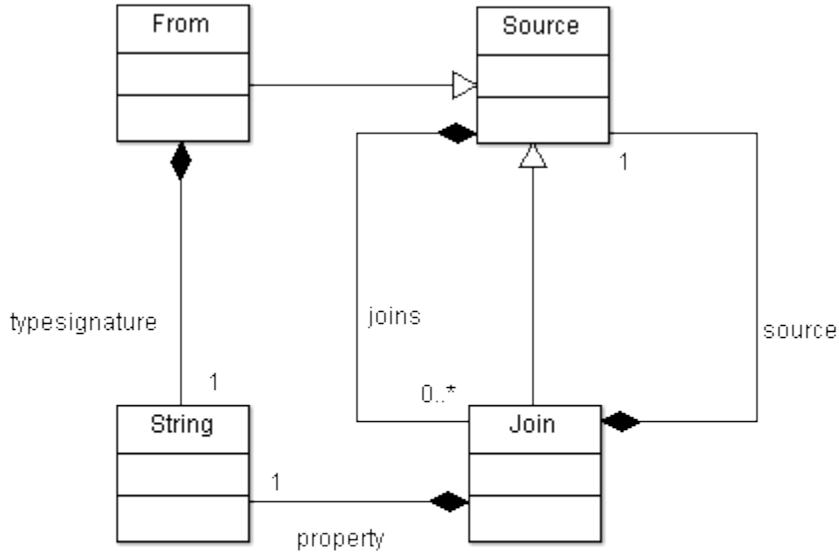


SelectQuery



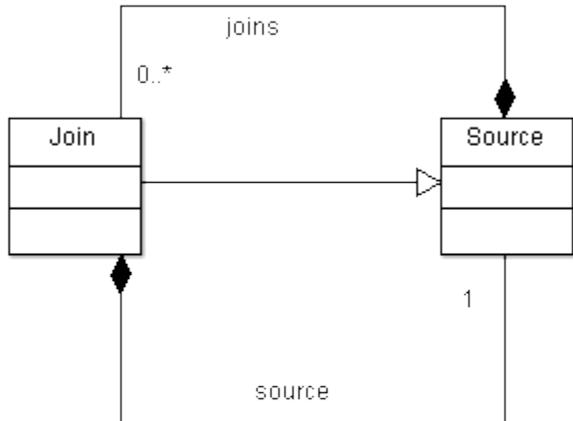
Populations

As with every query, we must specify from the data are to be extracted, i.e. declare the sources and therefore the From and the Join.



Source

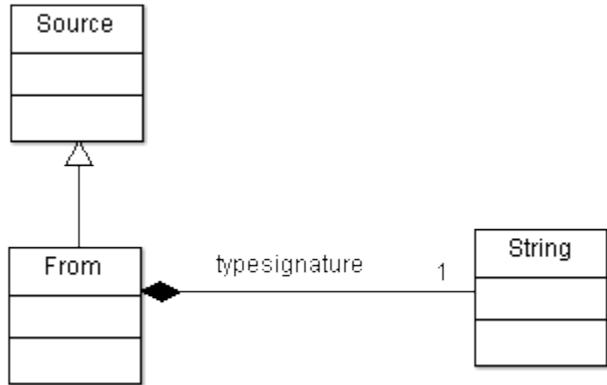
The Source is the base of both From and Join.



Name	Type	Description
joins	Set<Join>	A set of associated Join objects

From

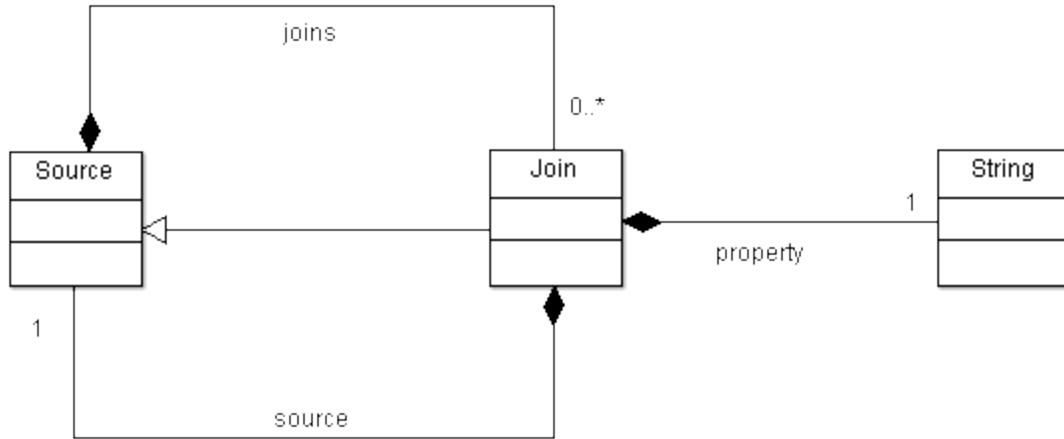
A From represents a single source of elements, aka a population.



Name	Type	Description
typeSignature	String	The fully qualified type signature of the population's class

Join

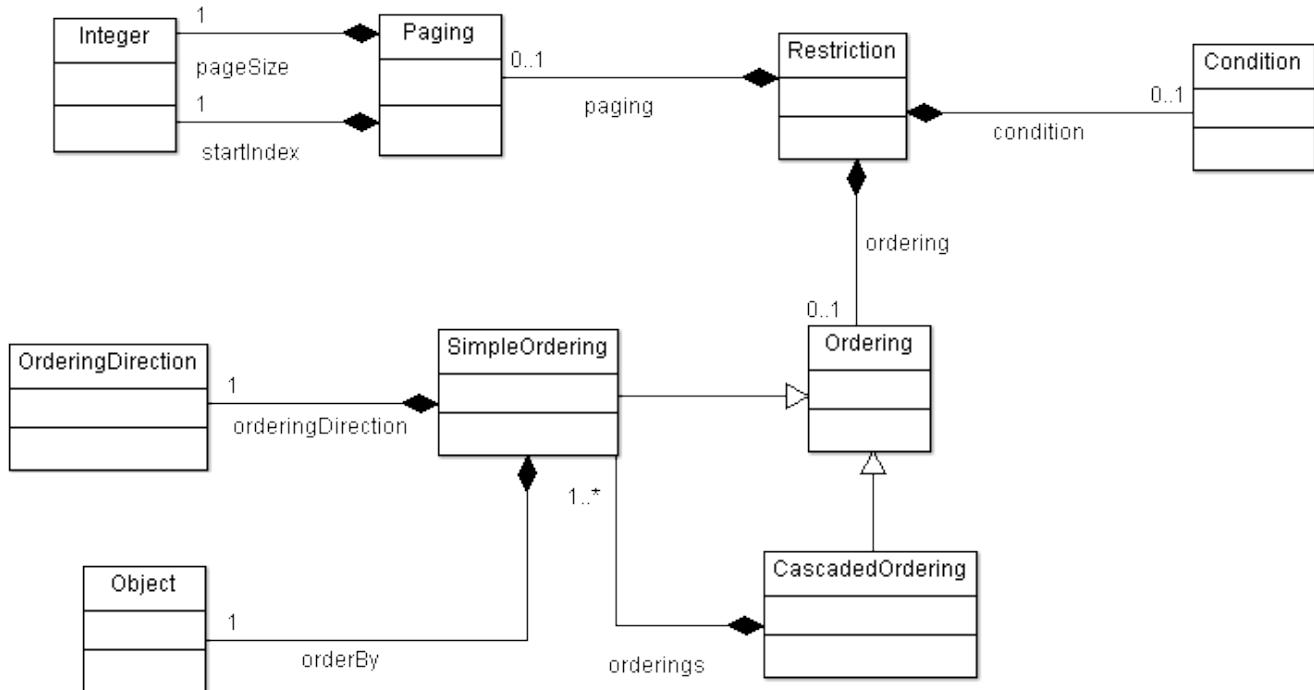
A Join joins two populations (or Froms, depending in which terminology you think), it derives from Source



Name	Type	Description
source	Source	The source we're joining with
property	String	The property that we're joining by.

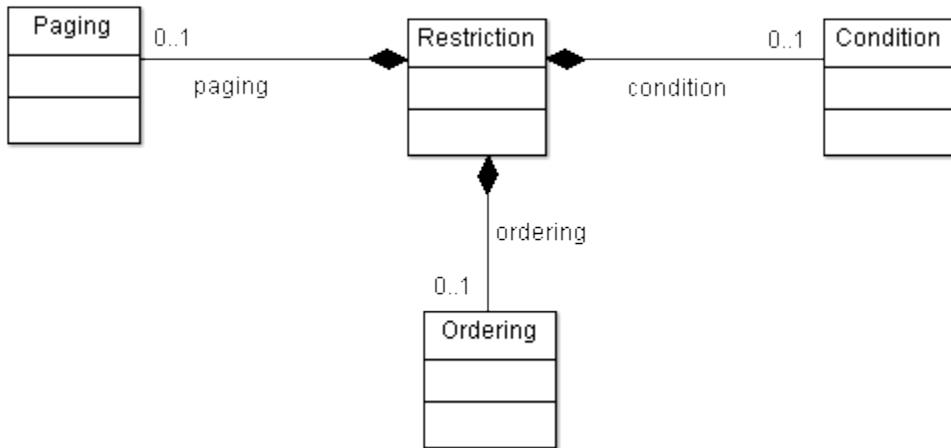
Restrictions

Obviously, a query is a means of restricting the view of a population, so there is always a kind of Restriction involved. Intuitively, one would think of the conditions of the query, but there's also the aspect of cutting up the result in smaller chunks and sorting the data within the chunks.



Restriction

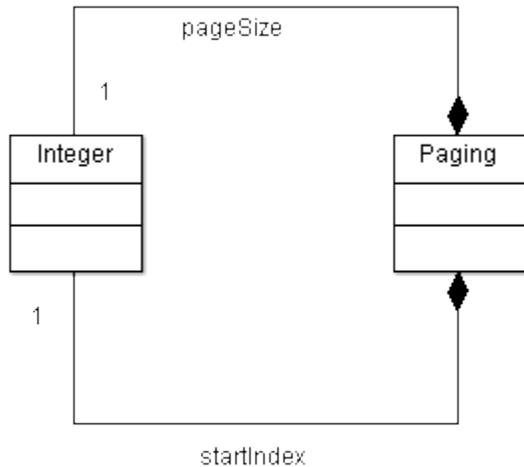
The actual restriction will be passed a parameter to the query.



Name	Type	Description
paging	Paging	The paging object that controls the size of the data to return. May be null no paging occurs,
ordering	Ordering	The ordering object that controls how the result should be sorted. May be null no ordering occurs,
condition	Condition	The condition object that declares how the data should be filtered. May be null no filtering occurs.

Paging

The Paging object declares how the resulting data should be fed back to the caller. It is especially important in a not very performant environment and made even more important when dealing with huge data sets.



Name	Type	Description
pageSize	Integer	the size of a single page

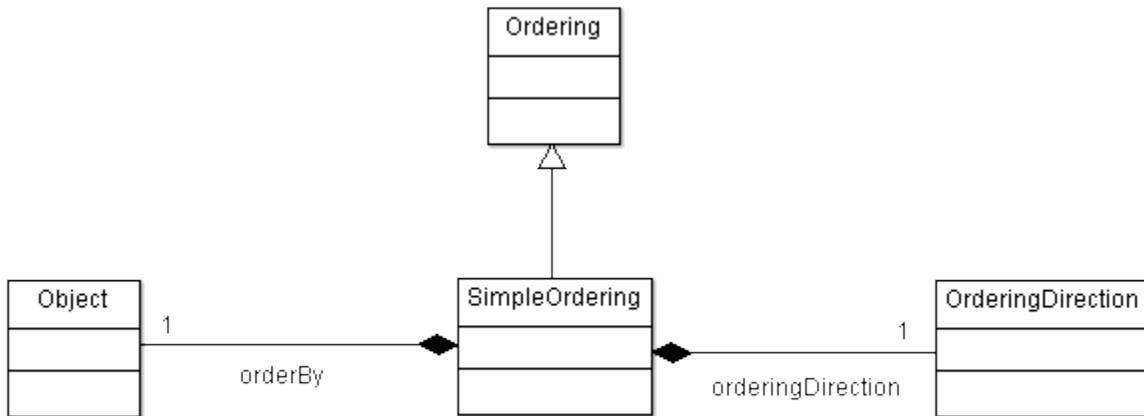
startIndex	Integer	the index of the first element to return.
------------	---------	---

Ordering

Ordering controls the sorting. The class itself is abstract as there two kinds of Ordering that implement the specific behavior.

SimpleOrdering

The SimpleOrdering specifies exactly one sorting parameter. It has a ([missing?](#))



Name	Type	Description
orderBy	Object	The Object we need to interpret on how we should sort. It may be a String (in case of the simple LQP), a PropertyOperand or a QueryFunction.
orderingDirection	OrderingDirection	The direction the sorting should occur.

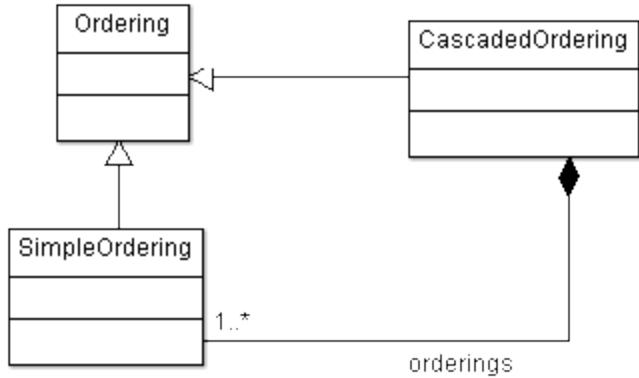
OrderingDirection

The OrderingDirection is an Enum with the following possible values.

Value	Description
OrderingDirection.ascending	Ascending sort requested.
OrderingDirection.descending	Descending sort requested.

CascadedOrdering

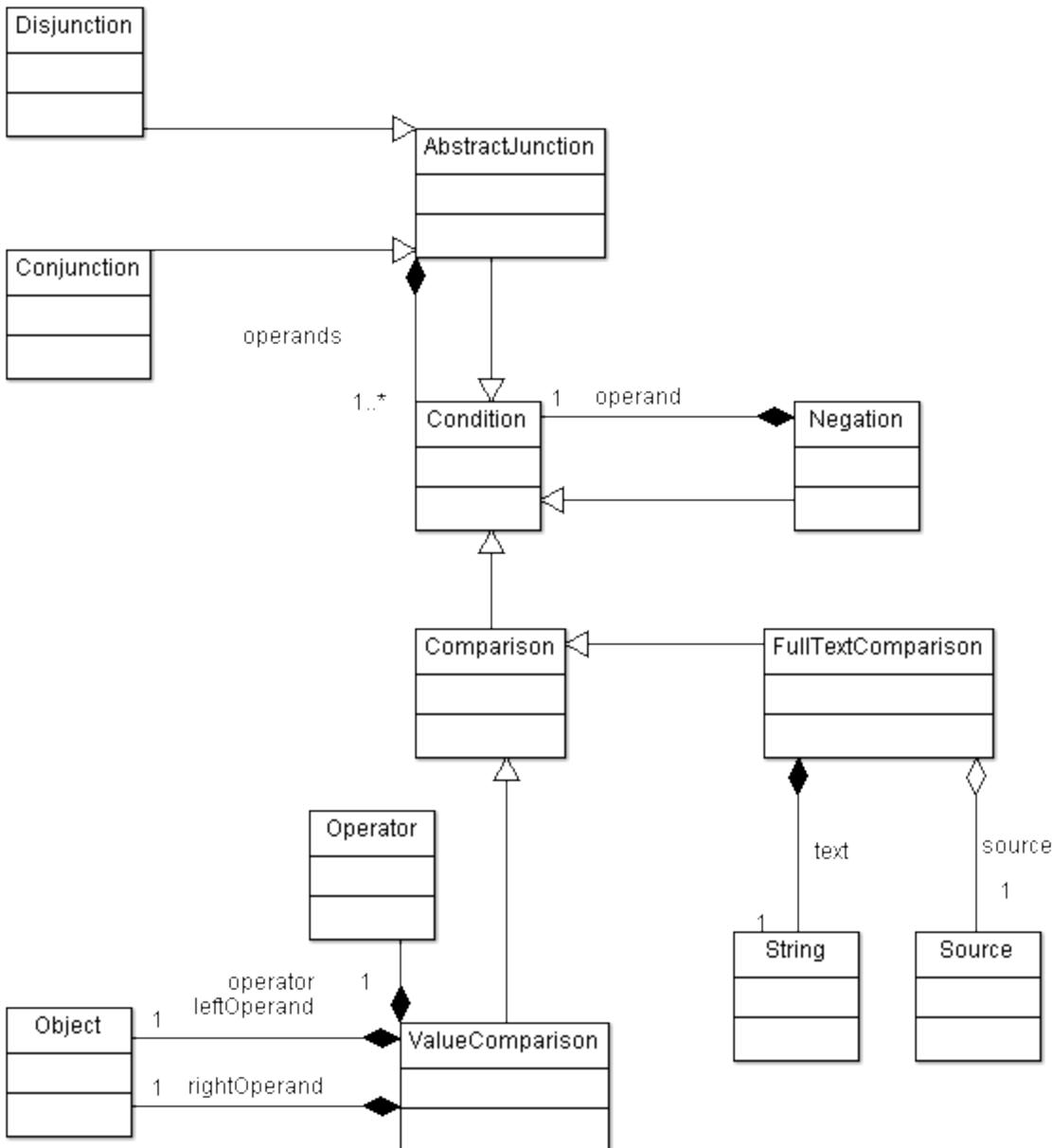
The CascadedOrdering is used if a cascading sorting logic must applied. If the first ordering is inconclusive (i.e. is deemed equal), then the next ordering is applied, and so on.



Name	Type	Description
orderings	List<Ordering>	A list orderings that should be applied in sequence.

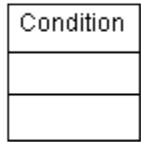
Conditions

Conditions are the mainstay of a query. They describe the complex selection process of results by logically structuring the search codes. As they restrict the output of a query, they're part of the Restriction above.



Condition

The **Condition** is the base class for all following conditions. It is a simple Generic Entity



In principle, there are two kind of conditions: One one hand the actual comparisons, and on the other hand the logical operators who act on the those comparisons.

Logical operators

The logical operators allow to build a logical structure according Bool.

Negation

A negation is a simple negation of the operand that is assigned to it. So if the operand evaluates to true, the negation returns false and vice versa.



Name	Type	Description
operand	Condition	The single operand whose's value is to be negated.

AbstractJunction

The AbstractJunction is a concatenation of several operands. How their values are joined is specified by the derived classes.



Name	Type	Description
operands	List<Condition>	A list of operands whose boolean values are to be collapsed into a single value.

Conjunction

A conjunction combines the values of its operands according the AND operator, i.e. all respective values must be true in order to make it return true. Otherwise, it will return false.



Disjunction

A disjunction acts like an OR operator, i.e. a single value of its operands must evaluate to true in order to make it return true. It will return false only if all values evaluate to false.



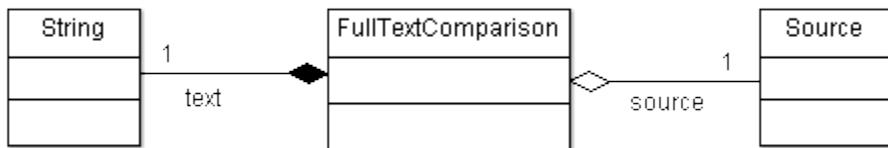
Comparisons

Comparison is the base class for the currently two comparison types. Of course, it's a Condition, so it can be used in the concatenations of the logical operators.



FullTextComparison

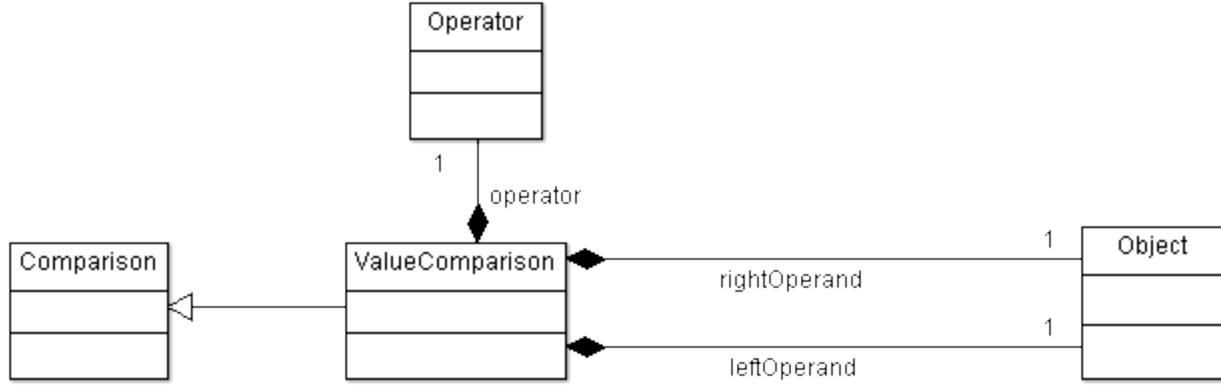
A FullTextComparison scans for a text in a full population (aka a source).



Name	Type	Description
text	String	The search string
source	Source	the population to scan for the text.

ValueComparison

A ValueComparison is a bit more complicated. Basically, it compares to values the question is how those two values are constituted.



Name	Type	Description
operator	Operator	An enum describing the kind of comparison is involved.
leftOperand	Object	The left operand. Even it's declared as Object, it can be of the types PropertyOperand, QueryFunction, or a simple Object or Collection
rightOperand		The right operand. The same applies here.

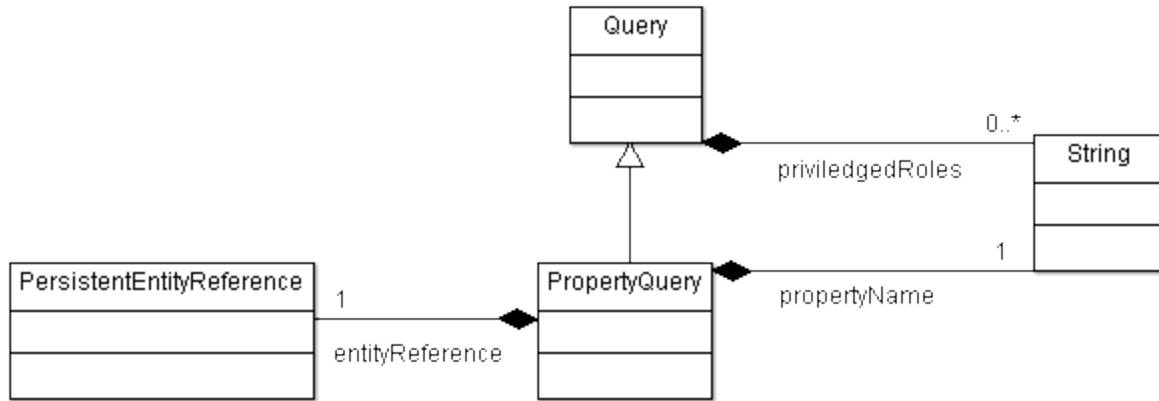
Operator

The Operator is a standalone declared enum with the following possible values:

Value	String	Description
contains	contains	left contains right, so left must be a collection that could contain an element like right.
in	in	left is in right, so right must be a collection that could contain an element like left.
equal	=	left is equal to right. Both values must be of a comparable type.
notEqual	!=	left is not equal to right. Both values must be of a comparable type.
greater	>	left is greater than right.
greaterOrEqual	>=	left is either greater or at least equal to right.
less	<	left is less than right
lessOrEqual	<=	left is either less or at least equal to right
like	like	left can contain the wildcards '*' and '?' and is matched to right.
ilike	ilike	left can contain the wildcards '*' and '?' and is matched to right while case insensitive.

PropertyOperand

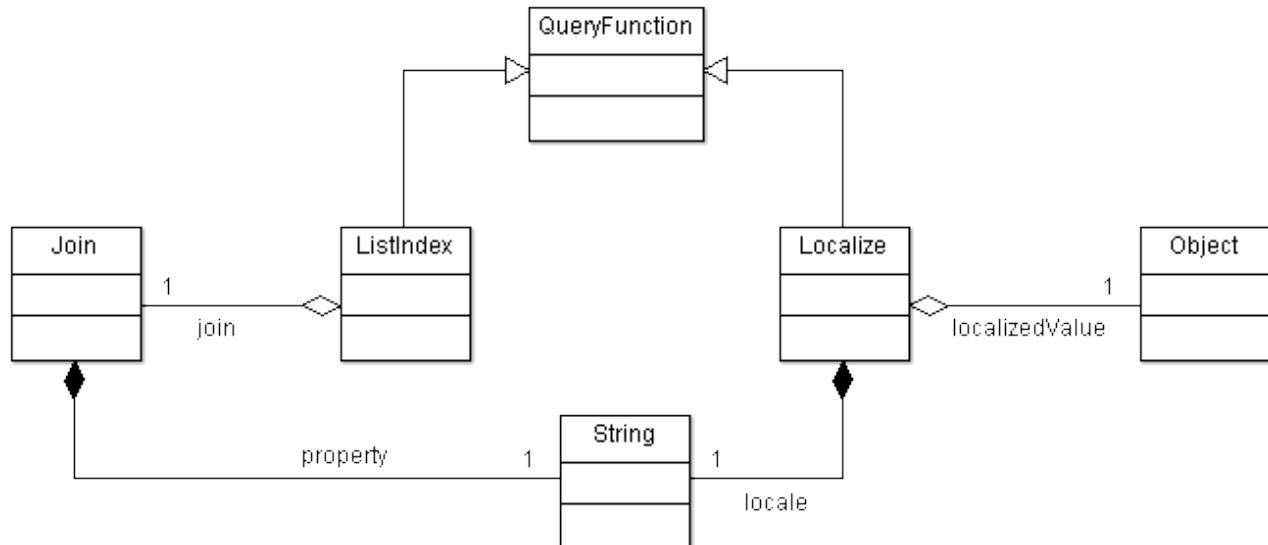
The PropertyOperand contains all data required to extract an actual value for the comparison.



Name	Type	Description
source	Source	The source of the data.
propertyName	String	The name of the property whose value is to be extracted.

QueryFunction

QueryFunctions are basically functions that can appear (mostly) anywhere in a query. They can standin as operands to a ValueComparison or in a Selection. Currently, three functions are known to the model: ListIndex, MapKey and Localize.



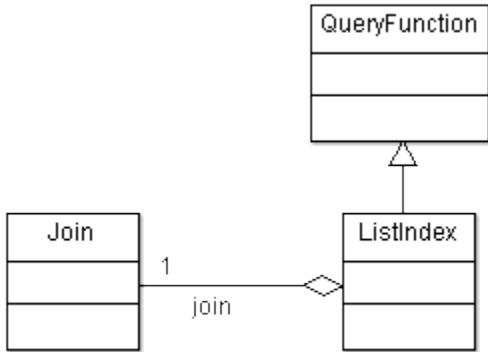
PropertyFunction

PropertyFunction is an abstract implementation for functions that operate on properties and require a Join as data source.

Name	Type	Description
join	Join	The join that is the source for the function

ListIndex

The ListIndex function will return the index of the current object in the property that the Join its associated with specifies.



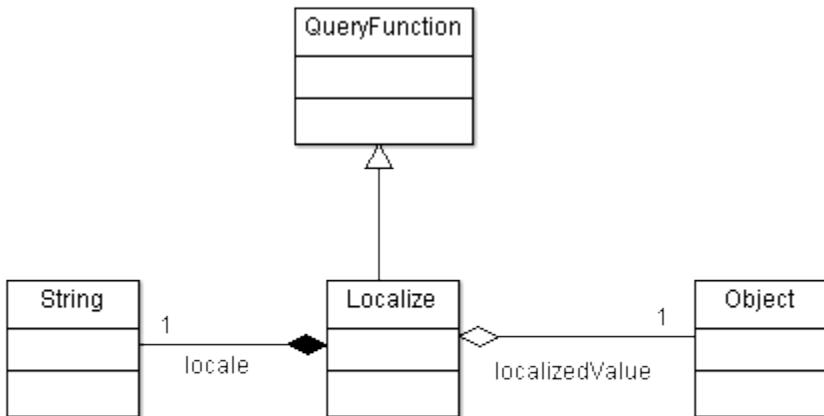
MapKey

The MapKey function provides access to the keys of properties of key/valuepairdatatype (like a java.util.Map in Java).

Localize

The Localize function is thought to return an appropriate value as defined by the locale from an object that contains several localized values.

At least, that what I think it does. Jami, could you please check on this? Why is 'localizedValue' an Object? It could be LocalizedString and what else?



Name	Type	Description
locale	String	The locale we want it's value from the localized value.
localizedValue	Object	An object that somehow contains localized values.

REST, Restlet and tribefire

Introduction

In this page we will examine what is REST, what is Restlet, how it is used to create a RESTful application and how it is related to the new implementation of tribefire (based on the servlet tribefire services, as of March 31, 2014)

What is REST?

REST (Representation State Transfer) is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed [hypermedia](#) system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

The term *representational state transfer* was introduced and defined in 2000 by [Roy Fielding](#) in his doctoral dissertation at [UC Irvine](#).

REST has been applied to describe desired web architecture, to identify existing problems, to compare alternative solutions, and to ensure that protocol extensions would not violate the core constraints that make the Web successful. Fielding used REST to design [HTTP 1.1](#) and [Uniform Resource Identifiers \(URI\)](#).

The REST architectural style is also applied to the development of [Web services](#) as an alternative to other distributed-computing specifications such as [SOAP](#).

What is Restlet?

Restlet is a lightweight, comprehensive, [open source](#) REST-based **framework** for the [Java](#) platform. It provides **reusable and extendable uses of classes** and practically multiple ways to connect with a server, thus helping developers to create so-called **RESTful applications**(adhere to the REST constraints).

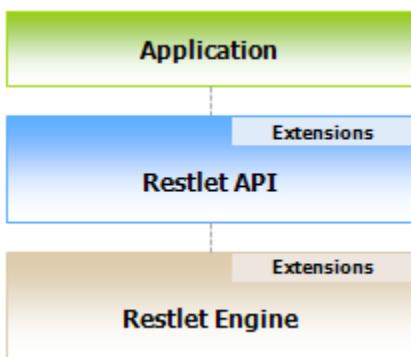
Restlet is suitable for both server and client Web applications. It supports major Internet transport, data format, and service description standards like [HTTP](#) and [HTTPS](#), [SMTP](#), [XML](#), [JSON](#), Atom, and [WADL](#). A [GWT](#) port of the client-side library is also available. It supports all REST concepts (Resource, Representation, Connector, Component etc.) and builds on them.

The first version was released in April 2007 by Jerome Louvel of Noelios Consulting. The latest version (as of March 2014) is Restlet 2.2

The Restlet framework is composed of two main parts.

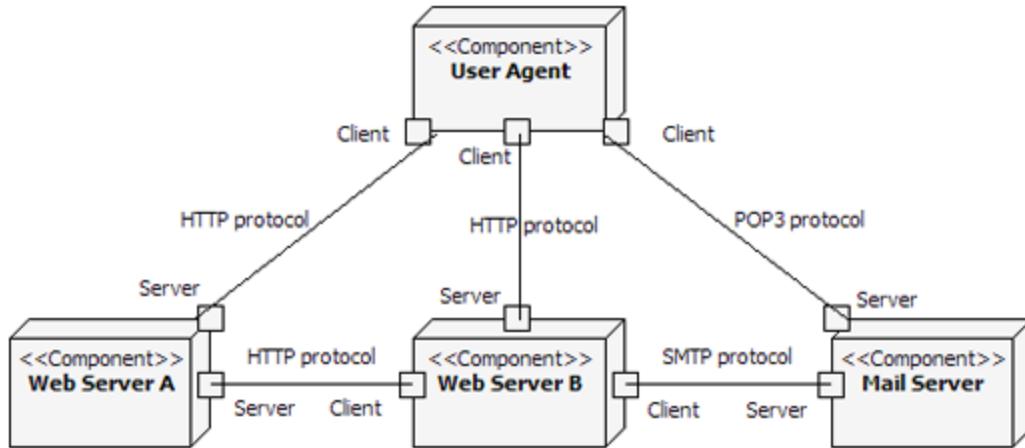
1) Restlet API: a neutral API supporting the concepts of REST and facilitating the handling of calls for both client-side and server-side applications.

2) Restlet Engine: Supports the API before it can effectively be used. It is based on a Restlet edition (or implementation). Multiple editions could be provided (open source projects or commercial products).



A network component can be at the same time client and server, depending on the context of the transaction. There are multiple ways for a client and a server to be connected, defined by methods called '**Connectors**'.

Let's step back a little and consider typical web architectures from a REST point of view. In the diagram below, ports represent the connector that enables the communication between components which are represented by the larger boxes. The links represents the particular protocol (HTTP, SMTP, etc.) used for the actual communication.



Note that the same component can have any number of client and server connectors attached to it. Web Server B, for example, has both a server connector to respond to requests from the User Agent component, and client connectors to send requests to Web Server A and the Mail Server.

Restlet Editions

The Restlet Framework is available in 6 editions:

1. Java SE edition, to run Restlet applications in regular JVMs
2. Java EE edition, to run Restlet applications in Servlet containers
3. GAE Edition, to run Restlet applications in Google App Engine cloud platform
4. GWT Edition, to run Web browser clients, without plugins
5. Android Edition, to deploy Restlet application on mobile Android devices.
6. OSGi Edition, to deploy Restlet on dynamic and embedded OSGi environments.

Restlet Licences

The Restlet framework is available under four different licenses:

- Apache
- CDDL(Common Development and Distribution License, a free software license produced by Sun Microsystems)
- LGPL(Lesser General Public License by the Free Software Foundation) and
- EPL(Eclipse Public License)

Resource for further learning

The online [restlet community](#) provides extensive tutorials javadocs and additional resources. Anyone can get support and meet the team behind it, become a contributor by reporting issues, fixing bugs, adding new features and documentation.

Connection with the new tribefire implementation

The new tribefire implementation is not anymore a server, but a servlet, hosted (contained) by an Apache Tomcat server. This servlet is a RESTful web-application which means the REST calls directed to this servlet **use the capabilities of the Restlet framework in their URL parameters**. The eventual .war file can be deployed in any Java Server. In the case of tribefire, the Server for the deployment is the Java EE (Enterprise Edition).

The CRUD(Create, Read, Update, Delete) operations and HTTP methods GET(), POST(), PUT(), DELETE() are supported by a set of classes, that are represented by entities in the underlying model. They can cover functionality of more than one method, for example the operation 'Fetch URL' supports the methods GET and POST, while the operation 'Creation URL' supports 3 methods: GET, POST and PUT. Its functionality (and of course our needs) determine which HTTP methods are used when invoking an Entity with a RESTcall.

Projections use literals or classes. They can be directly from the existing ones of the Restlet framework or user-defined. In the case of tribefire, depending on the Entity's function, there are the following user-defined literals:payload, envelope, none. Additionally for Workbench Actions there are goto, message, code.

Information Resolving Entities fetch data without altering them.

Metadata Entities have the ability to describe themselves. They also fetch data without altering them.

Depths show us the extent of the information fetched. That is decided by the Traversing Criteria. The default value (in case no Traversing Criteria exists) is 'shallow', which refers to the first level of possible separation only.

Tribefire Restlet Calls

Restlet URLs

- Restlet URLs
 - Common Paramters
 - CRUD Operations
 - Collections
 - Session Management
 - Meta Data Resolving
 - Workbench Actions
 - Supported Workbench Actions
- RestApiModel
- Artifacts

Common Paramters

These parameters can be used with every rest call. AccessId and sessionID are required for every call!

- **accessId** (required): accessId of the addressed access.
- **sessionId** (required): sessionId for the authentication.
- **codec** (optional): output format (gm/xml, gm/json, ...)
- **pseudoContentType** (optional): used for forcing browser for handling known mimeTypes (text/plain, ...)
- **depth** (optional): see comment at operations
- **projection** (optional): see comment at operations
- **axis** (optional): data (default), workbench, meta



Note

The operation "authenticate" doesn't require access nor sessionId. Also "logout" doesn't require accessId.

CRUD Operations

Generic Query URL

Url Path: ./rest/query

Supported Methods: GET

Url Parameters:

- **statement** (required): written statement which will be parsed via ANTLR and executed (needs to be escaped/encoded because of the colliding delimiters (=, blank, ...))

Projections:

- **payload** (default): SelectQueryResult.results, PropertyQueryResult.propertyValue, EntityQueryResult.entities
- **first-of-payload**: SelectQueryResult.results[0], PropertyQueryResult.propertyValue[0], EntityQueryResult.entities[0]
- **envelope**: SelectQueryResult, PropertyQueryResult, EntityQueryResult

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Example

```
http://localhost:8080/tribefire-services/rest/query?accessId=cortex&sessionId=b0de409b-e770-4acl-8e29-f66dd9dcd9e1&statement=from%20com.braintripe.model.meta.GmMetaModel&projection=first-of-payload&codec=text/xml
```

Entity Fetch URL

Url Path: ./rest/fetch

Supported Methods: GET, POST

Url Parameters:

- **type** (required): typeSignature (qualified name) of the entity.
- **\$xyz** (optional): name of the property of the entity used for a simple comparison (equals).
- **orderBy** (optional): name of the property which is ordered by.
- **orderDir** (optional): ordering direction (asc, desc).
- **pageStart** (optional): 0 based start index of the paging.
- **pageSize** (optional): size of the limitation of the result count.

Projections:

- **payload**: EntityQueryResult.entities
- **first-of-payload**: EntityQueryResult.entities[0]
- **envelope**: EntityQueryResult

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Entity Identity Resolution URL

Url Path: ./rest/entity

Supported Methods: GET, POST

Url Parameters:

- **type** (required): typeSignature (qualified name) of the entity.
- **id** (required): id of the entity which is addressed

Projections:

- **payload**: EntityQueryResult.entities[0]
- **envelope**: EntityQueryResult

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Property Fetch URL

Url Path: /rest/property-fetch

Supported Methods: GET, POST

Url Parameters:

- **type** (required): typeSignature (qualified name) of the entity.
- **id** (required): id of the entity which is addressed.
- **property** (required): name of the property being fetched.
- **\$xyz** (optional): name of the property of the entity used for a simple comparison (equals).
- **orderBy** (optional): name of the property which is ordered by.
- **orderDir** (optional): ordering direction (asc, desc).
- **pageStart** (optional): 0 based start index of the paging.
- **pageSize** (optional): size of the limitation of the result count.

Projections:

- **payload** (default): PropertyQueryResult.propertyValue
- **first-of-payload**: PropertyQueryResult.propertyValue[0]
- **envelope**: PropertyQueryResult

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Entity Creation URLs

Url Path: /rest/create

Supported Methods: GET, POST, PUT

Url Parameters:

- **type** (required): typeSignature (qualified name) of the entity.
- **\$xyz** (optional): name of the property of the entity used for creation.

Projections:

- **entity** (default): new created entity
- **id**: id of new created entity
- **envelope**: ManipulationResponse of session.commit()
- **payload**: ManipulationResponse.inducedManipulations

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Entity Updating URLs

Url Path: /rest/update

Supported Methods: GET, POST, PUT

Url Parameters:

- **type** (required): typeSignature (qualified name) of the entity.
- **id** (required): id of the entity which is updated.
- **\$xyz** (optional): name of the property of the entity used for updating.

Projections:

- **entity**: updated entity
- **envelope**: ManipulationResponse of session.commit()
- **payload**: ManipulationResponse.inducedManipulations
- **none** (default): none

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Entity Deletion URL

Url Path: /rest/delete

Supported Methods: GET, POST, PUT, DELETE

Url Parameters:

- **type** (required): type of the entity to delete.
- **id** (required): id of the entity to delete.

Projections:

- **envelope**: ManipulationResponse of session.commit()
- **payload**: ManipulationResponse.inducedManipulations
- **none** (default): none

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Assembly Based Manipulation URLs (Complex CRUD)

Url Path: /rest/assembly-manipulate

Supported Methods: GET, POST, PUT

Url Parameters:

- **body** (required): serialized representation of the entity to be created.
- **projection** (optional): entity (none), id, entities, ids, envelope, payload, none

Projections:

- **entity**: the rootvalue if it is an entity
- **id**: id of the rootvalue if it is an entity
- **entities**: entities being in a list if the rootvalue is a list of entities
- **ids**: ids of the entities in a list if the rootvalue is a list of entities
- **envelope**: ManipulationResponse of session.commit()
- **payload**: ManipulationResponse.inducedManipulations
- **none** (default): none

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Url Path: /rest/assembly-stream-manipulate

Supported Methods: POST, PUT

Url Parameters:

- **accessId** (required): accessId of the business access.
- **sessionId** (required): sessionId for auth.
- **streamBody** (required with encoding from contentType):
- **projection** (optional): entity (none), id, entities, ids, envelope, payload, none
- **depth** (optional):
- **codec** (optional):
- **pseudoContentType** (optional):

Projections:

- **entity**: the rootvalue if it is an entity
- **id**: id of the rootvalue if it is an entity
- **entities**: entities being in a list if the rootvalue is a list of entities
- **ids**: ids of the entities in a list if the rootvalue is a list of entities
- **envelope**: ManipulationResponse of session.commit()
- **payload**: ManipulationResponse.inducedManipulations
- **none** (default): none

Depths:

- **reachable**: every reachable information (simple, entity, enum and collection of those)
- **shallow** (default): just the first level. just simple. no entities and collections

Collections

List Insert URL

Url Path: /rest/list-insert

Supported Methods: GET, POST (form url encoded), PUT

Url Parameters:

- **type** (required): type of the addressed entity
- **id** (required): id of the addressed entity
- **property** (required): name of the collection property of the entity.
- **index** (optional): start index, integer value, default 0.
- **value** (required): value(s) to be inserted.

Projections:

- **payload**: ManipulationResponse.inducedManipulations
- **envelope**: ManipulationResponse
- **none** (default): none

List Remove URL

Url Path: /rest/list-remove

Supported Methods: GET, POST (form url encoded), PUT

Url Parameters:

- **type** (required): type of the addressed entity.
- **id** (required): id of the addressed entity.
- **property** (required): name of the collection property of the entity.
- **index** (required):

Projections:

- **payload**: ManipulationResponse.inducedManipulations
- **envelope**: ManipulationResponse
- **none** (default): none

Set Insert URL

Url Path: /rest/set-insert

Supported Methods: GET, POST (form url encoded), PUT

Url Parameters:

- **type** (required): type of the addressed entity
- **id** (required): id of the addressed entity
- **property** (required): name of the collection property of the entity.
- **value** (required): value(s) to be inserted.

Projections:

- **payload**: ManipulationResponse.inducedManipulations
- **envelope**: ManipulationResponse
- **none** (default): none

Set Remove URL

Url Path: /rest/set-remove

Supported Methods: GET, POST (form url encoded), PUT

Url Parameters:

- **type** (required): type of the addressed entity.
- **id** (required): id of the addressed entity.
- **property** (required): name of the collection property of the entity.
- **index** (required):

Projections:

- **payload**: ManipulationResponse.inducedManipulations
- **envelope**: ManipulationResponse
- **none** (default): none

Map Put URL

Url Path: /rest/map-put

Supported Methods: GET, POST (form url encoded), PUT

Url Parameters:

- **type** (required): type of the addressed entity.
- **id** (required): id of the addressed entity.
- **property** (required): name of the collection property of the entity.
- **key** (required):
- **value** (required):

Projections:

- **payload**: ManipulationResponse.inducedManipulations
- **envelope**: ManipulationResponse
- **none** (default): none

Map Remove URL

Url Path: /rest/map-remove

Supported Methods: GET, POST (form url encoded), PUT

Url Parameters:

- **type** (required): type of the addressed entity.
- **id** (required): id of the addressed entity.
- **property** (required): name of the collection property of the entity.
- **key** (required):

Projections:

- **payload**: ManipulationResponse.inducedManipulations
- **envelope**: ManipulationResponse
- **none** (default): none

Session Management

Authenticate URL

Url Path: /rest/authenticate

Url Parameters:

- **user** (required): username
- **password** (required): password

Example

```
http://localhost:8080/tribefire-services/rest/authenticate?user=cortex&password=cortex
&projection=payload&depth=reachable
```

Logout URL

Url Path: /rest/logout

Reflection

Access Reflection URL

Url Path: /rest/reflect-accesses

Model Reflection URL

Url Path: /rest/reflect-model

Types Reflection URL

Url Path: /rest/reflect-types

Url Parameters:

- **abstraction** (optional): object, simple, enum, entity(default), #typeSignature#
- **signature** (optional + multi): com.braintripe.model.* -> wildcard support aka like

Property Reflection URL

Url Path: /rest/reflect-properties

Url Parameters:

- **type** (required): entity type signature
- **name** (optional+multi):

ID Property Reflection URL

Url Path: /rest/reflect-id-property

Meta Data Resolving

Model Meta Data URL

Url Path: /rest/model-md

Url Parameters:

- **metaData** (required): type of the meta data you want to resolve.
- **useCase** (optional):
- **exclusive** (optional): default=true;

EntityType Meta Data URL

Url Path: /rest/entity-md

Url Parameters:

- **type** (required): type name of the entity
- **metaData** (required): type of the meta data you want to resolve
- **useCase** (optional):
- **exclusive** (optional): default=true;

Property Meta Data URL

Url Path: /rest/property-md

Url Parameters:

- **type** (required): type name of the entity
- **property** (required): name of the property
- **metaData** (required): type of the meta data you want to resolve
- **useCase** (optional):
- **exclusive** (optional): default=true;

EnumType Meta Data URL

Url Path: /rest/enum-md

Url Parameters:

- **type** (required): type name of the entity
- **metaData** (required): type of the meta data you want to resolve
- **useCase** (optional):
- **exclusive** (optional): default=true;

Enum Constant Meta Data URL

Url Path: /rest/enum-constant-md

Url Parameters:

- **type** (required): type name of the entity
- **metaData** (required): type of the meta data you want to resolve
- **useCase** (optional):
- **exclusive** (optional): default=true;

Workbench Actions

Workbench Action URL

Url Path: /rest/action

Supported Methods: GET, POST

Url Parameters:

- **name** (required): name of the Folder from the workbench to get the FolderContent (WorkbenchAction) from
- **\$xyz** (optional): name of a variable from a template if the action is a template based action

Projections:

- **payload**:
- **envelope**:
- **none** (default): none

Supported Workbench Actions

Rpc Actions

- RpcAction
- TemplateRpcAction

Rpc Actions will be executed via GmWebRpc and the result will be returned as a serialization. Supported projections are:

- payload(*RpcResponse.returnValue*)
- envelope (*RpcResponse*)

Query Actions

- SimpleQueryAction
- PrototypeQueryAction
- TemplateQueryAction

Query Actions are evaluated via the AccessService and the result will be returned as a serialization. Supported projections are:

- see generic query request

Instantiation Actions

- SimpleInstantiationAction

- TemplateInstantiationAction

Instantiation Actions will be executed and the generated value will be return as serialization. Supported projections are:

- see manipulation based requests

Custom Actions

- CustomAction
- TemplateCustomAction

Custom Actions will be executed as GME does. Supported projections are:

- envelope (*ActionResponse*)
- payload (?)
- goto (*ActionResponse.goTo*)
- message (*ActionResponse.message*)
- code (*ActionResponse.code*)

RestApiModel

- interface RestRequest
 - long requestId (persistence id)
 - string accessId
 - string sessionId
 - string pseudoContentType
- interface HasProjection
 - string projection
 - string depth
- interface HasBody extends HasCodec
 - string body
- interface HasStreamBody
 - string streamBody
- interface HasCodec
 - string codec
- interface HasEntityType
 - string type
- interface HasEntityIdentification extends HasEntityType
 - string id
- interface HasDynamicParameters
 - map<string,string> dynamicParameters
- interface HasPropertyIdentification extends HasEntityType
 - string property
- interface HasCollectionValues extends HasPropertyIdentification
 - list<object> value
- interface HasMapKey extends HasPropertyIdentification
 - object key
- interface HasMapValue extends HasPropertyIdentification
 - object value
- interface HasMapEntry extends HasMapKey, HasMapValue
- interface QueryRequest extends RestRequest, HasProjection
 - string statement
- interface FetchRequest extends RestRequest, HasProjection, HasEntityIdentification, HasDynamicParameters
 - string orderBy
 - OrderDirection orderDir

- int pageStart
 - int pageSize
-
- interface interface EntityIdentificationResolutionRequest extends RestRequest, HasCodec, HasProjection, HasEntityIdentification
 - interface PropertyFetchRequest extends FetchRequest, HasPropertyIdentfcation,HasEntityIdentification
-
- interface CreateRequest extends RestRequest, HasCodec, HasEntityType, HasProjection, HasDynamicParameters
-
- interface UpdateRequest extends RestRequest, HasCodec, HasEntityIdentification, HasProjection, HasDynamicParameters
-
- interface DeleteRequest extends RestRequest, HasCodec, HasEntityIdentification
 - interface AssemblyManipulateRequest extends RestRequest, HasBody, HasProjection
-
- interface AssemblyStreamManipulateRequest extendsRestRequest, HasCodec, HasProjection, HasStreamBody
-
- interface ActionRequest extends RestRequest, HasCodec, HasProjection, HasDynamicParameters
 - string name
 - interface CollectionRequest extends RestRequest, HasCodec, HasProjection, HasEntityIdentification, HasCollectionValues
 - interface ListInsertRequest extends CollectionRequest
 - int pageStart
 - interface ListRemoveRequest extends CollectionRequest
 - interface SetInsertRequest extends CollectionRequest
 - interface SetRemoveRequest extends CollectionRequest
 - interface MapPutRequest extends RestRequest, HasCodec, HasProjection, HasEntityIdentification, HasMapEntry
 - interface MapRemoveRequest extends RestRequest, HasCodec, HasProjection, HasEntityIdentification, HasMapKey
 - interface MetaDataRequest extends RestRequest, HasCodec, HasProjection, HasEntityType, HasPropertyIdentfcation
 - string metaData
 - string useCase

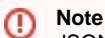
Artifacts

1. com.braintribe.model:GmRestApiModel-1.0
2. com.braintribe.model.processing.web:GmRestApi-1.0
3. com.braintribe.model.processing.web:GmRestRestlet-2.0

GmWebRpcServer - Example

GmWebRpcServer is a servlet that enables remote method invocation of services built using generic models through plain http.

This article shows a simple use case of a RPC call using JSON representation of generic models.



Note

JSON is not the only option. XML and serialized binary representation could also be used.

Configuring the example

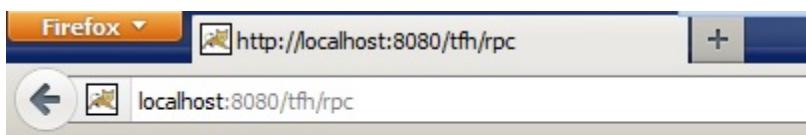
Deploy the DemoTribeFirehost#2.0 (tfh.war)

[tfh.war](#)

Deploy the given war-files into your servlet container (in case of tomcat, just place it into the "webapps" directory or your TOMCAT_HOME)

Check if the GmWebRpcExample is running by accessing <http://localhost:8080/tfh/rpc>

If you see the message "GmWebRpc functional interface", the GmWebRpcServer is up and running.



Deploy the test html client (tfh-json-test.war)

[tfh-json-test.war](#)

The given tfh_json-test.war application defines no server-side coding whatsoever, just pure html/javascript that will delegate calls to the previously deployed tfh.war.

It has to be deployed just like the tfh.war.

After deployment, check if the test html is running by accessing <http://localhost:8080/tfh-json-test/GmWebRpcExample.html>

A html button labeled "Test Authentication" should be shown. By hitting this button, a JSON representation is printed on the web page. This means everything works.

Example - client perspective

Accessing the source code of the GmWebRpcExample.html page, you will see a quite simple piece of javascript that simply:

- creates a JSON representation of a generic entity called RpcRequest.
- posts it through ajax to the tribefire hosts GmWebRpcServer running in the same container.
- gets the return from the ajax call (JSON representation of a generic entity called RpcResponse) and append it to the html page body.

The request: RpcRequest

In this entity's JSON representation, you can notice 5 properties:

JSON

```
{
    _type : <value>,
    serviceId : <value>,
    methodName : <value>,
    parameterTypeSignatures: <value>,
    parameters : <value>
};
```

`_type`: this property must be present in every JSON representation of a generic entity since it determines the underlying Generic Entity type signature.

`serviceId`: this is the id of the service to be called. In this case, we knew upfront that there was a service mapped to the id "SERVICE". More information on this in the server-section.

`methodName`: the name of the method to be invoked, that must be defined in the service given by `serviceId`.

`parameterTypeSignatures`: the signatures of the entity types expected as parameters for the method defined by `parameterTypeSignatures`.

`parameters`: the JSON representation of the parameters, according to the signatures given by `parameterTypeSignatures`.

In this example's source code these parameters were valued as:

```
_type: "com.braintribe.model.rpc.RpcRequest",
serviceId: "SECURITY",
methodName: "authenticate",
parameterTypeSignatures:
[ "com.braintribe.model.securityservice.AuthenticationRequest" ],
parameters: [
{
    _type: "com.braintribe.model.securityservice.AuthenticationRequest",
    credentials: {
        _type:
"com.braintribe.model.securityservice.credentials.UserPasswordCredentials",
        userIdentification: {
            _type:
"com.braintribe.model.securityservice.credentials.identification.UserNameIdentificatio
n",
            userName: userName
        },
        password: password
    }
}
]
```

`_type`: this property is set to "com.braintribe.model.rpc.RpcRequest" as this must be the root entity of every RPC call against GmWebRpcServer.

`serviceId`: in this case, we knew upfront that there was a service mapped to the id "SECURITY".

`methodName`: the method to be invoked in this case called "authenticate", since we knew upfront that the service mapped to the id "SECURITY" exposes a method with this name.

tribefire Training Model

SampleBase.java

```
package com.braintribe.model.processing.bttraining;
import com.braintribe.model.processing.session.impl.remote.SimpleConfiguration;
import com.braintribe.model.processing.session.impl.remote.RemoteGmSessionFactory;
public abstract class SampleBase {
    private RemoteGmSessionFactory sessionFactory = null;
    private String accessId = "bttrainingModel.access";
    private String host = "localhost";
    private String server = "tfh";
    private int smiPort = 9010;
    private int streamPort = 9011;
    private String user = "cortex";
    private String password = "cortex";

    public RemoteGmSessionFactory getSessionFactory() throws Exception {
        if (sessionFactory == null) {
            sessionFactory = new RemoteGmSessionFactory();

            SimpleConfiguration configuration = new SimpleConfiguration();
            configuration.setAccessId(accessId);
            configuration.setHost(host);
            configuration.setServer(server);
            configuration.setSmiPort(smiPort);
            configuration.setStreamPort(streamPort);
            configuration.setUser(user);
            configuration.setPassword(password);

            sessionFactory.init(configuration);
        }
        return sessionFactory;
    }
}
```

This class creates an attendee object as well as a company. Both are linked and then committed.

CreateAttendee.java

```
package com.braintribe.model.processing.bttraining;

import java.util.HashSet;

import com.braintribe.model.bttraining.Attendee;
import com.braintribe.model.bttraining.Company;
import com.braintribe.model.processing.session.impl.remote.RemoteGmSession;

/**
 * This sample class instantiates a new Attendee object, fills them with random data
and commits it to the GmSession.
 */
public class CreateAttendee extends SampleBase {

    public void run() throws Exception {

        RemoteGmSession session = getSessionFactory().provide();
        try {

            // Create an attendee with name John Doe
            Attendee attendee = session.createEntity(Attendee.class); // Each new
GenericEntity needs to be instantiated via the session. Don't use the ordinary new
Operator!
            attendee.setFirstName("John");
            attendee.setLastName("Doe");
            attendee.setEmailAddress("john@doe.com");
            attendee.setJobTitle("Trainee");

            // Create a company called Braintribe
            Company company = session.createEntity(Company.class);
            company.setCompanyName("Braintribe");

            // Link the company and the attendee
            attendee.setCompany(company);
            company.setEmployee(new HashSet<Attendee>());
            company.getEmployee().add(attendee);

            // Persist the changes.
            session.commit();

            System.out.println("Successfully created Attendee.");
        } finally {
            // Finally we close the session
            session.close();
        }
    }

    public static void main(String[] args) throws Exception {
        new CreateAttendee().run();
    }
}
```

QueryAttendee.java

```

package com.braintribe.model.processing.bttraining;
import java.util.List;
import com.braintribe.model.bttraining.Attendee;
import com.braintribe.model.processing.query.fluent.EntityQueryBuilder;
import com.braintribe.model.processing.session.impl.remote.RemoteGmSession;
import com.braintribe.model.query.EntityQuery;
/**
 * This sample class queries the session for an Attendee named "John Doe" and prints
the resulting objects to system.out.
 */
public class QueryAttendee extends SampleBase {
    public void run() throws Exception {

        RemoteGmSession session = getSessionFactory().provide();
        try {

            // Create a query that filters for firstName=John AND lastName=Doe
            EntityQuery query =
                EntityQueryBuilder
                    .from(Attendee.class)
                    .where()
                    .conjunction()
                    .property("firstName").eq("John")
                    .property("lastName").eq("Doe")
                    .close()
                    .done();

            // Execute the query and get list of resulting attendees
            List<Attendee> attendees = session.query().entities(query).list();

            // Loop over the result and print each attendee to System.out using the
internal toString() implementation of GenericEntities.
            int i = 0;
            for (Attendee attendee : attendees) {
                System.out.println("Attendee "+(++i)+": "+attendee);
            }

        } finally {
            session.close();
        }
    }

    public static void main(String[] args) throws Exception {
        new QueryAttendee().run();
    }
}

```

REST API Documentation

Table of Contents

- ▼ show/hide
 - authenticate
 - authenticateSso
 - logout
 - sessionValidation
 - entity?accessKind=query
 - entity?accessKind=update
 - entity?accessKind=create
 - entity?accessKind=delete
 - entity/resource

authenticate

host:port/EcmRestlet/authenticate?user=string&password=string[&local=boolean]

Example

http://localhost:8080/EcmRestlet/authenticate?user=MyUser,&password=MyPass

Parameters

- user
- password
- locale

Return value: JSON Object

```
{
  "_id": "",
  "_type": "",
  "sessionId": "",
  "roles": { "_type": "", "value": [ "" ] },
  "user": "",
  "profile": {
    "_type": "",
    "value": [
      { "key": "", "value": "" },
      . . .
      { "key": "", "value": "" }
    ]
  }
}
```

authenticateSso

host:port/EcmRestlet/authenticateSso[&local=boolean]

The Restlet creates a technical session. Again, the return value will be a session object.

Parameter

- locale

Return value: JSON Object

```
{
    "_id": "",
    "_type": "",
    "sessionId": "",
    "roles": { "_type": "", "value": [ "" ] },
    "user": "",
    "profile": {
        "_type": "",
        "value": [
            { "key": "", "value": "" },
            ...
            { "key": "", "value": "" }
        ]
    }
}
```

logout

host:port/EcmRestlet/logout?sessionId=string

Invoking this method will terminate the session with the passed sessionId.

Parameter

- sessionId

Return value: none**sessionValidation**

host:port/EcmRestlet/sessionValidation?sessionId=string

This method checks whether the passed sessionId is still valid and can still be used.

Parameter

- sessionId

Return value: boolean

true/false

entity?accessKind=query

host:port/EcmRestlet/entity?_type=string&accessKind=query&queryType=string&sessionId=string
[&orderBy=string&orderDir=string&page=number&pageSize=number&tc=json Object]

host:port/EcmRestlet/entity?_type=string&accessKind=query&condition=string&sessionId=string
[&orderBy=string&orderDir=string&page=number&pageSize=number&tc=json Object]

A query against the interface can be done two different ways:

One the one hand by a queryType and on the other hand by a self-defined condition. Further the type of the entity must be declared and a

sessionId must be added. The other parameters are optional.

Currently there is one pre-defined queryType (byId=number) available.

Example for a queryType call:

```
http://localhost:8080/EcmRestlet/entity?entityType=com.braintripe.model.example.Person&accessKind=query&queryType=byId&id=5&sessionId=120123122204426268fc1c6da36e649e
```

Example call of a property with parameter propertyName:

```
http://localhost:8080/EcmRestlet/entity?entityType=com.braintripe.model.example.Person&propertyName=image&id=1&sessionId=120206105220110c6e0f7ff3c40ff757
```

Parameters

- entityType
- accessKind
- queryType (byId)
- propertyName
- condition
- sessionId
- orderBy
- orderDir
- page
- pageSize
- tc

Return value: JSON Object, Array of JSON Objects

```
{...}  
[ { ... }, { ... } ]
```

entity?accessKind=update

```
host:port/EcmRestlet/entity?accessKind=update&data={_type:string,id:string,"key": "val"}&sessionId=string
```

```
host:port/EcmRestlet/entity?accessKind=update&data={_type:string,id:string,{jsonObject Array}}&sessionId=string
```

With accessKind update single values or - by usage of an array - also several values at the same time can be changed. For this the property data needs the id and the type of the respective entity.

Example call for changing a property:

```
http://localhost:8080/EcmRestlet/entity?accessKind=update&data={_type: "com.braintripe.model.example.Person", id:5, name: "Paul"}&sessionId=120123122204426268fc1c6da36e649e
```

Parameters

- data
- sessionId

Return value: none

entity?accessKind=create

```
host:port/EcmRestlet/entity?accessKind=create?data={"_type":"string","key1":val1,...,"keyN":val  
N:  
&sessionId=string}
```

By passing a JSON object to the data property an entry can be created. The object must contain the entity type.

Example call for creating a resource:

```
http://localhost:8080/EcmRestlet/entity?accessKind=create&data={"_type":"com.braintrib  
e.model.generic.resources.ImageResource","animated":false,"height":160,"name":"foto_Di  
eter_Taufer.jpg","width":120,"left":0,"uuid":"6994a2e8-39ac-4ee9-88b6-323ee4c3734a","m  
imeType":"image/jpeg","top":0}&sessionId=120125100412100375a59136730d7d3d
```

Parameters

- data
- sessionId

Return value: Array

```
[ Number ]
```

entity?accessKind=delete

```
host:port/EcmRestlet/entity?accessKind=delete
```

 Not yet implemented.

Parameter

- sessionId

Return value: none

entity/resource

```
host:port/EcmRestlet/entity/resource?queryType=customQuery&sessionId=string
```

It is possible to define custom queries in the Restlet for arbitrary resources that can be accessed like described above. The parameters of the call are default by the respective method declaration. The configuration is done at **resourcedef.spring.xml** of the Restlet.

Example query:

```
http://localhost:8080/EcmRestlet/entity/person?queryType=byNameOrLastName&name=Peter&  
sessionId=120125100412100375a59136730d7d3d
```

Parameter

- queryType

Return value: any

Standard Metadata

Derives from
 BTT-3104 - MetaData in GME (In Progress)


Info

For more information about the MetaModel please visit [The MetaModel aka a model that models models](#)

- Available standard properties
 - MetaData

Available standard properties

These properties are provided by the metamodel. Specific properties should never be put into the metamodel but into other models.

MetaData

MetaData is data that is attached to the types. They contain any form of data that gives additional information about the types.

Name	Type	Description
id	Long	The id property.
priority	Double	The priority, between 0.0 and 1.0.
name	LocalizedString	The name of the MetaData as multilanguage string.
description	LocalizedString	The description of the MetaData as multilanguage string.
selector	MetaDataSelector	A selector for the conditional activation / relevance of the MetaData's content.
value	String	The name of the value the MetaData applies to.

Modelling with UML

Table of Contents

- ▼ show/hide
 - 1 Introduction
 - 2 Deliverables
 - 3 Instructions
 - 3.1 How to model in UML
 - 3.2 Relationships
 - 3.2.1 Generalization
 - 3.2.2 Aggregation
 - 3.3 Dos and don'ts
 - 4 How to model Generic Models with UML
 - 4.1 Simple types
 - 4.2 Classes vs Interfaces
 - 4.3 Attributes vs Associations
 - 4.4 Special Markers
 - 4.4.1 Stereotypes
 - 4.4.2 Tag-Definitions
 - 4.5 Collections
 - 4.5.1 Set
 - 4.5.2 List
 - 4.5.3 Maps
 - 5 Base Aggregator Models for Future Modeling Tasks
 - 5.1 BaseAggregatormodel-1.0
 - 5.2 DmsBaseAggregatorModel-1.0
 - 6 Standard ant targets

Introduction

The GM/UML - tagged as “**UrMeL**” now - initiative aims to give modellers to choice of how to create and/or modify a model. It introduces modelling via an XMI compatible graphic modeller as an alternative.

The modelling is transparent, i.e. there is no basic restriction in either method of modelling other than it is restricted to the common features of UML and GM - so metadata cannot be modelled.

Currently, the UML modelling is implemented by a group of bidirectional codecs:

- **Artifact to MetaModel codec** - encodes artifacts to meta models and meta models to artifacts
- **Metamodel to XMI codec** - encodes meta models to XMI and XMI to meta models
- **XMI To ARGO codec** - encodes XMI files to ARGO files and ARGO files to XMI files

It also depends on a Malaclypse supplied dependency walk.

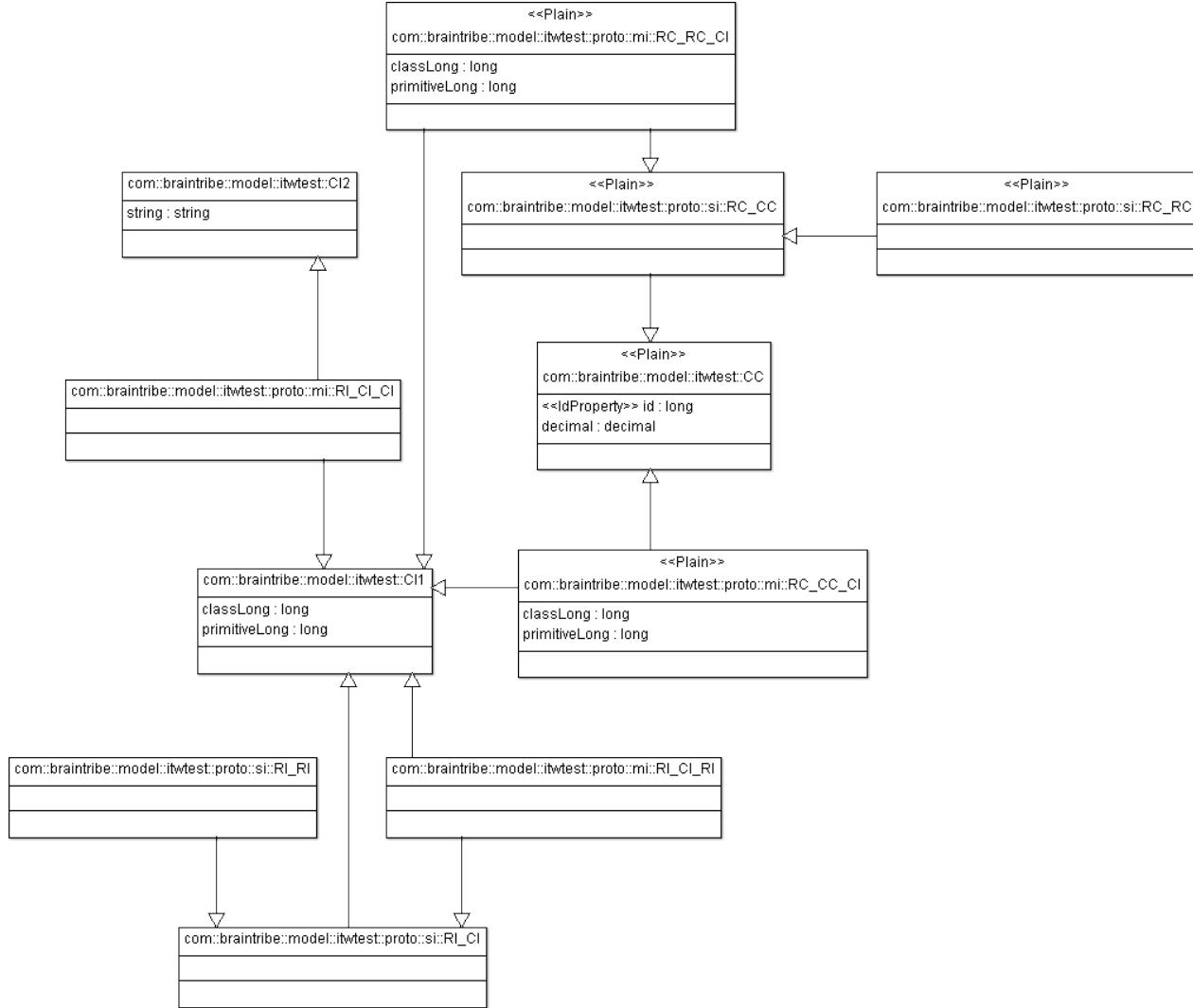
Deliverables

ARGO UML is a open-source project. You can get it from here:

 argouml.tigris.org

Basically, the GM/UML interface consists of two compound ANT tasks, which parts you'll find in the **BtAntTask Library 1.8** forwards. Drop the contents of the BtAntTasks-1.8.zip into your ANT lib directory. Any ANT higher than 1.8 should do (it was built with 1.8.2)

Instructions



An example of a class diagram for a tribefire Model in UML. The model shown is the `com.braintribe.model.InheritanceTestModel#1.0`. Apart from providing tests for UrMeL, it also shows one of the features of GM: You can use multiple inheritance on GenericEntities.

How to model in UML

This is not an UML tutorial! If you are not familiar with UML you may want to refer to:
[tutorialspoint.com/uml/uml_tutorial \(PDF\)](http://tutorialspoint.com/uml/uml_tutorial (PDF))

Find below the few things you'll need to know for modelling Generic Models.

Relationships

For you, there are only two kinds of relationships that are of interest, “**generalization**” and “**aggregation**” which is a flavoured association. All other connection types found in UML are not needed for our purpose.

Generalization

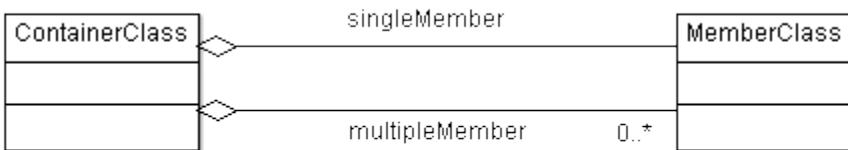
The first one is a “generalization”, which denotes a derivation or inheritance relationship. UML models that in the reverse - a generalization points from the sub class (the deriving class) to the super class (the derived from class).



Aggregation

An “aggregation” is what we use to denote a member variable of a Generic Entity which is by itself a complex type, i.e. another Generic Entity.. It is a kind of association, but differs in that it has a direction (it tells us what the containing class and what the type of the association are) and a multiplicity (or cardinality).

If you do not specify the multiplicity, it will default to a multiplicity value of 1, as in the example “singleMember” below. The multiplicity “0..*” (or “1..*”) tells the generator that a collection of values is meant. See below for details on aggregations.



Dos and don'ts



Never specify a class in plural

If your model contains classes like “doctors”, “patients” or “documents”, you have a serious flaw in your model.

Always use a singular and specify multiplicity using an association.



Never use any other associations but generalization and aggregation

In UML an association is exactly what the word means - a thing that somewhat connects to classes - it has no further meaning.

If you use that, you'll leave the guy that has to work on the model always guessing what you meant by it.

Anyhow, the generator can't figure it out anyway. So the tools will simply fail.

How to model Generic Models with UML



Of course, Generic Models aren't just plain models, but models *with a twist*. Therefore, UML can't quite model GM models right out of the box. As long as we don't have our proper graphical stage for the GME, we must live with the problem that UML isn't properly adapted to GM.

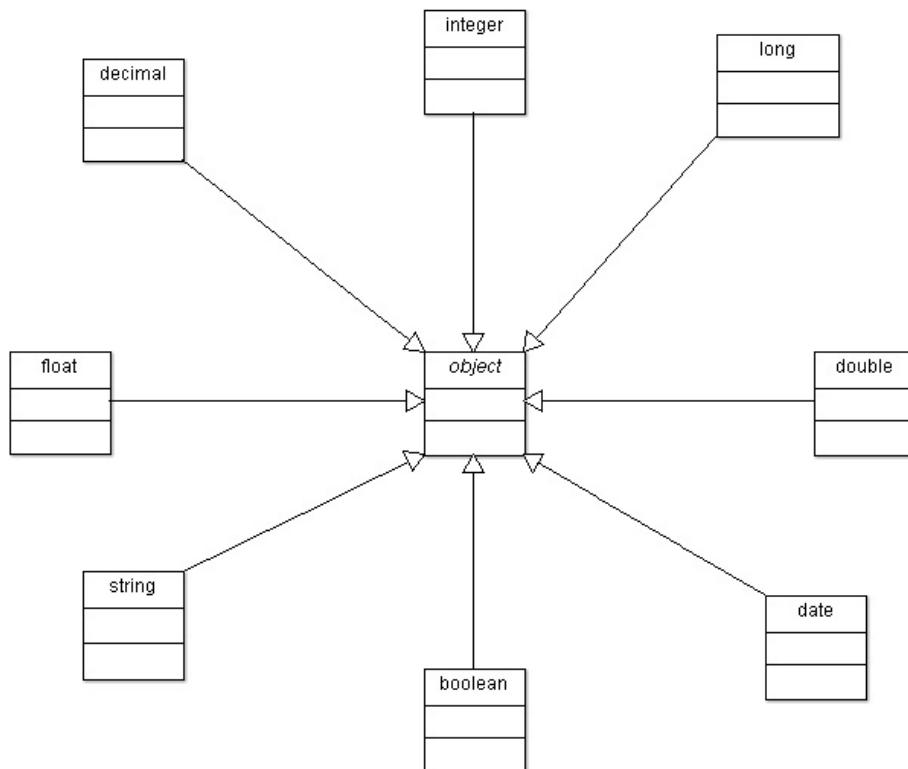
In most of the cases, we can restrict UML to where a GM model is a subset of what UML can model. In some cases however, UML is restricted and cannot model what a GM model is. And I'm not talking about meta data - which is something absolutely beyond the scope of UML (see OCL for that).

In most cases we came up with a smart work-around, but in some cases, it means that this aspect simply cannot be modelled without our proper stage. So, all in all, this will remain a stop-gap measure. But I still think it's cool.

Simple types

Your favourite simple types are always part of an UML model. They are automatically injected by the codec's encoder, each modelled correctly deriving from the simple type object:

- object
- integer
- long
- float
- double
- decimal
- boolean
- string



You can always select them from the drop down box of available types, if you need to specify an attribute's value. Please note that they are NOT read by the codec's decoder (of course, the references are), so treat them as read only.

Classes vs Interfaces



In UML, GenericEntities are represented as classes, even if they're interfaces in the JAVA code.

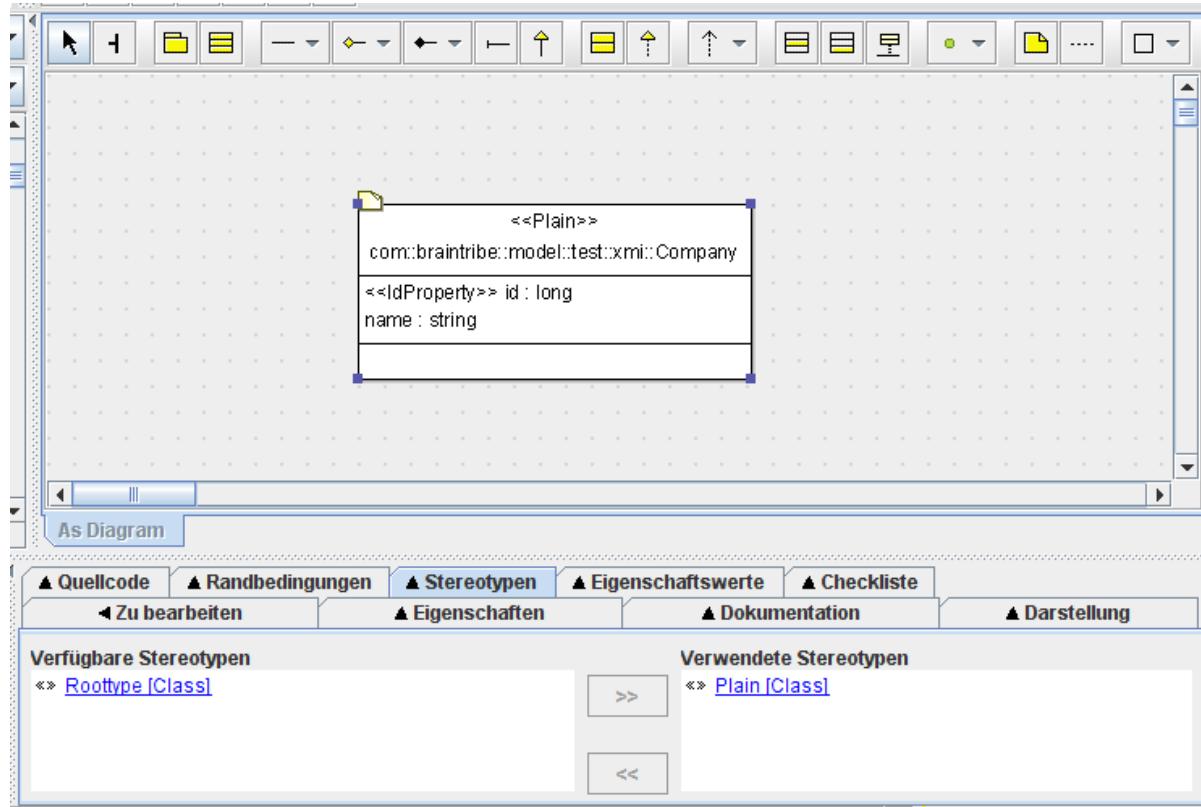
You do not specify getter/setter functions, but just declare attributes for simple types and associations for complex types (such as other GenericEntities or collections).

Anyhow, as there are NO methods in a generic entity, you might as well switch your UML tool to hide methods.

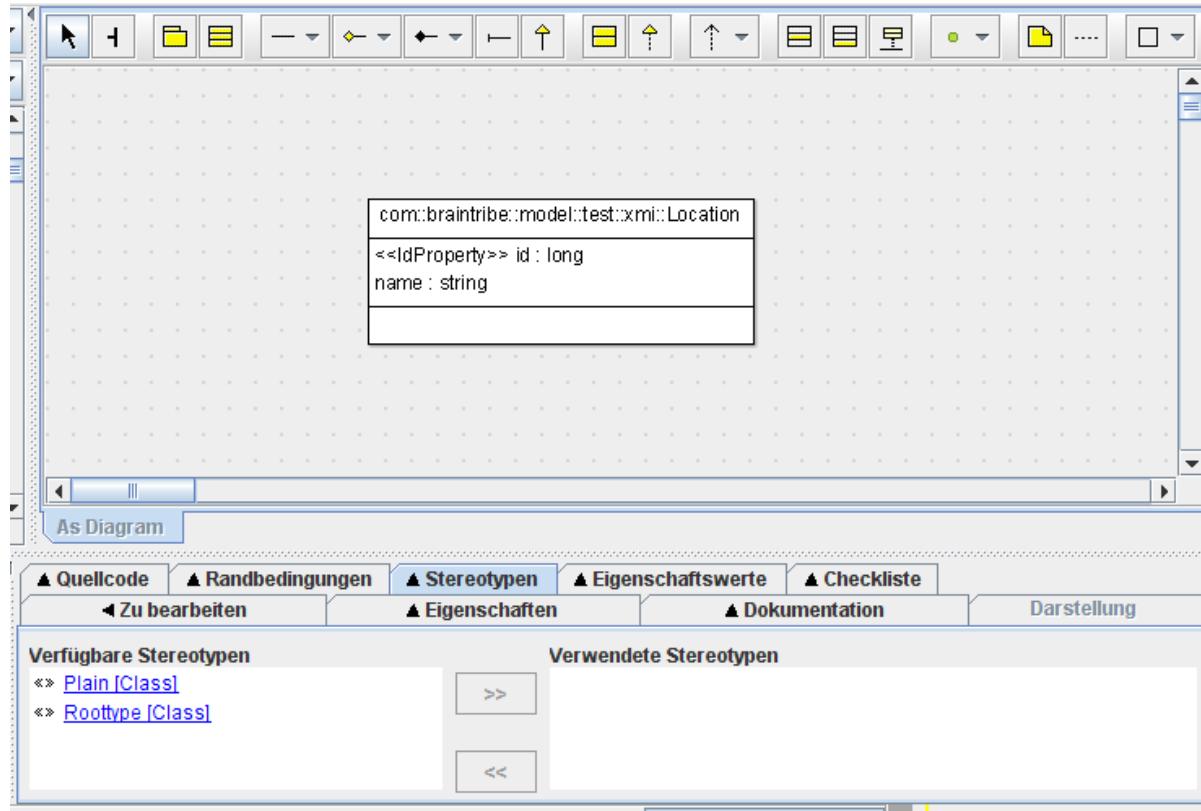
- Basically, any UML class is represented in the JAVA code as an interface. If you want it to be generated as a class, you must add the "Plain"-stereotype as a marker.

See below for more info on markers.

So, this is a class:



And this is an interface:



Attributes vs Associations

As said above, properties are declared as attributes or associations.

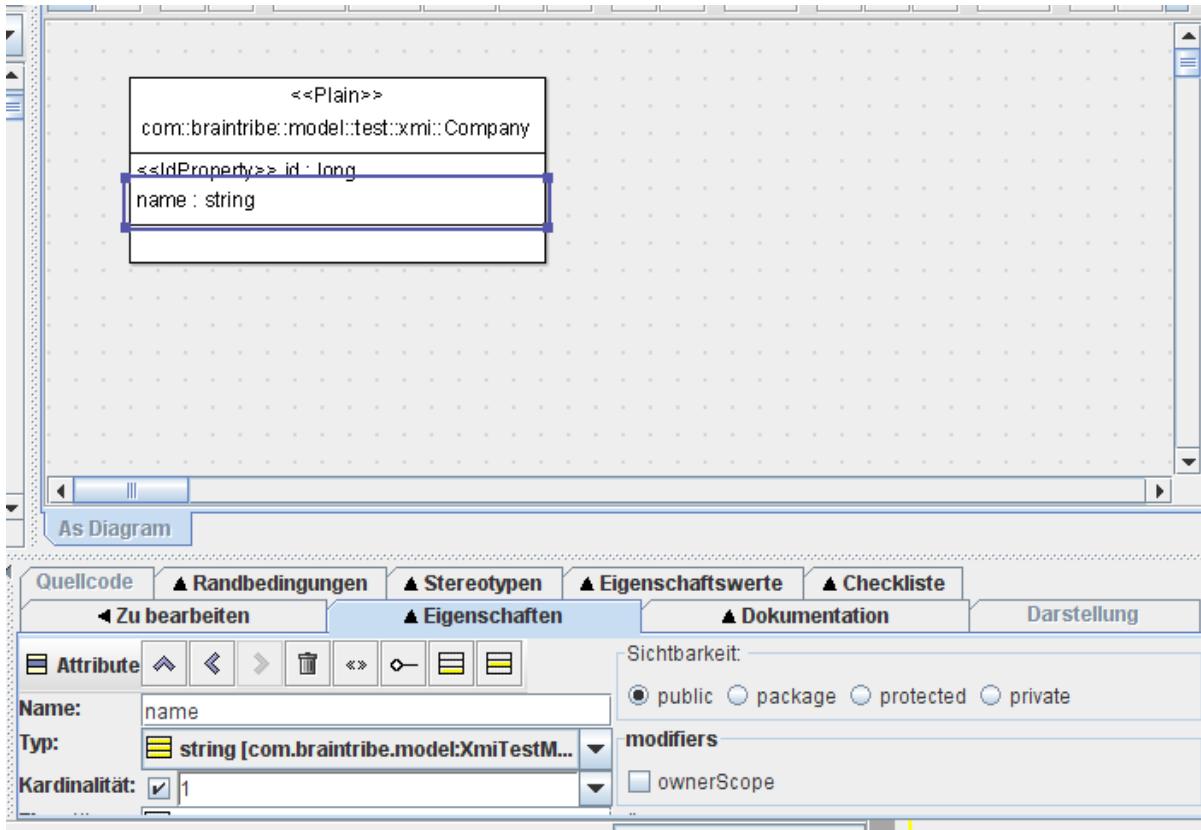
Golden Rule

The golden rule is that you declare:

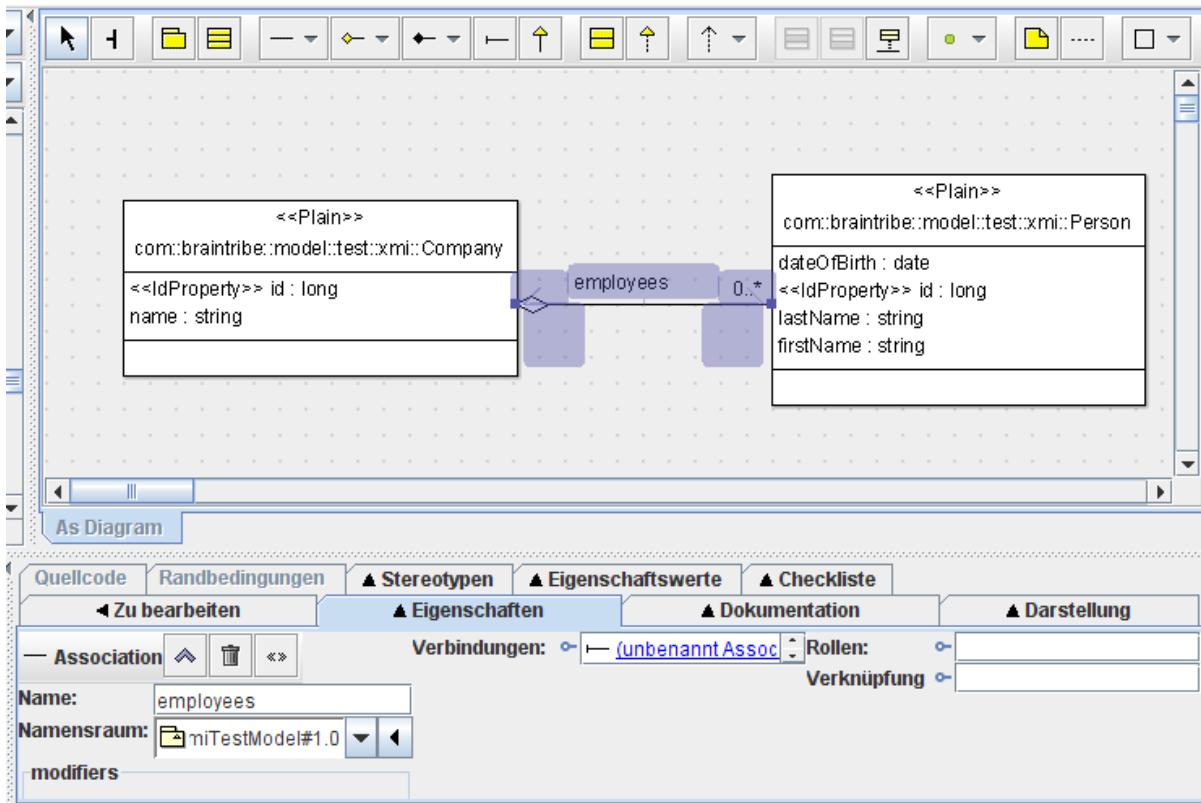
- an **attribute** if your property uses a **simple type** and
- an **association** for **complex types**, such as all collections and **entity types**

The difference being that attributes are declared directly within the UML representation of the entity, but associations are declared by adding it in the graphics display.

Declaring an attribute:



Declaring an association:



Special Markers

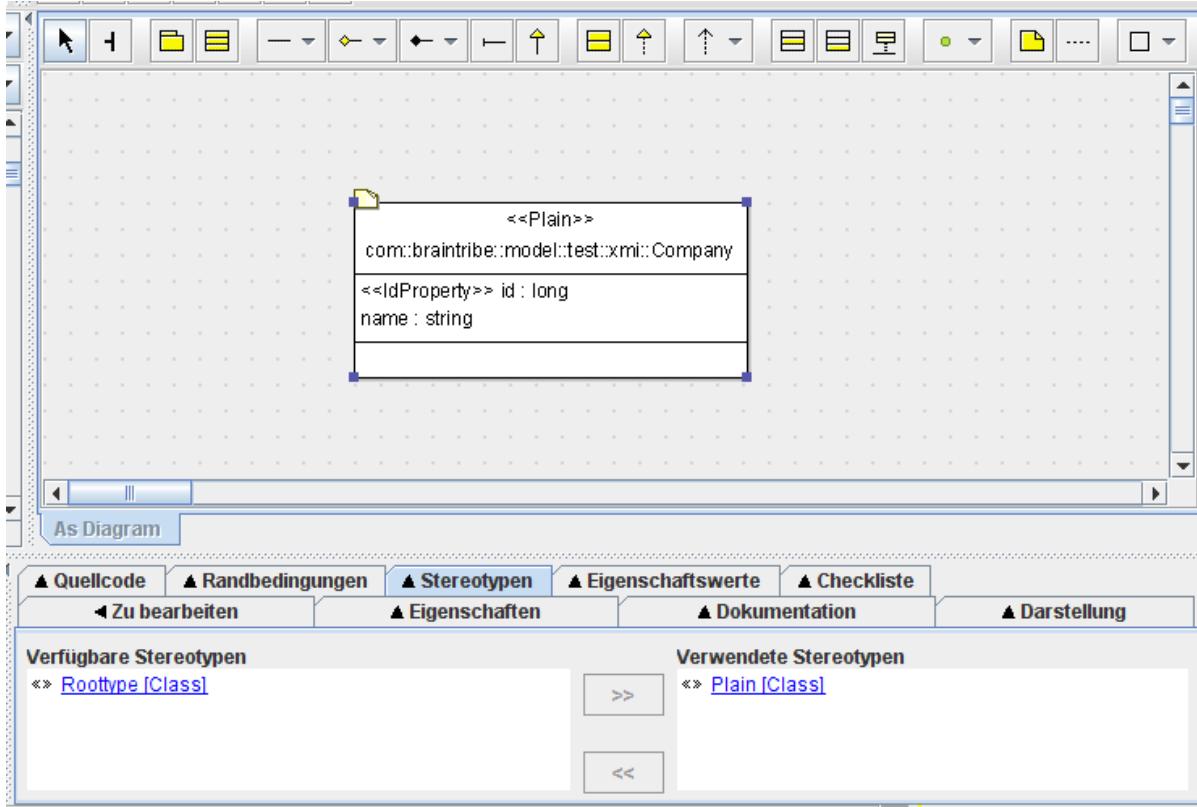
There are a few special markers to add information that are not part of UML. The UML feature we use for that purpose are called **stereotypes** and **tag definitions**. Stereotypes are simple markers, and tag definitions are containers for simple string values.

Stereotypes

Stereotypes are simple markers and can be added to either classes or attributes (actually any UML element, but we only use it for attributes and classes). You add a stereotype by selecting the tab for stereotypes. The dialog now shows all markers that are applicable for the selected item on the left list. Selecting one will move it to the right side, to the list of applied markers.

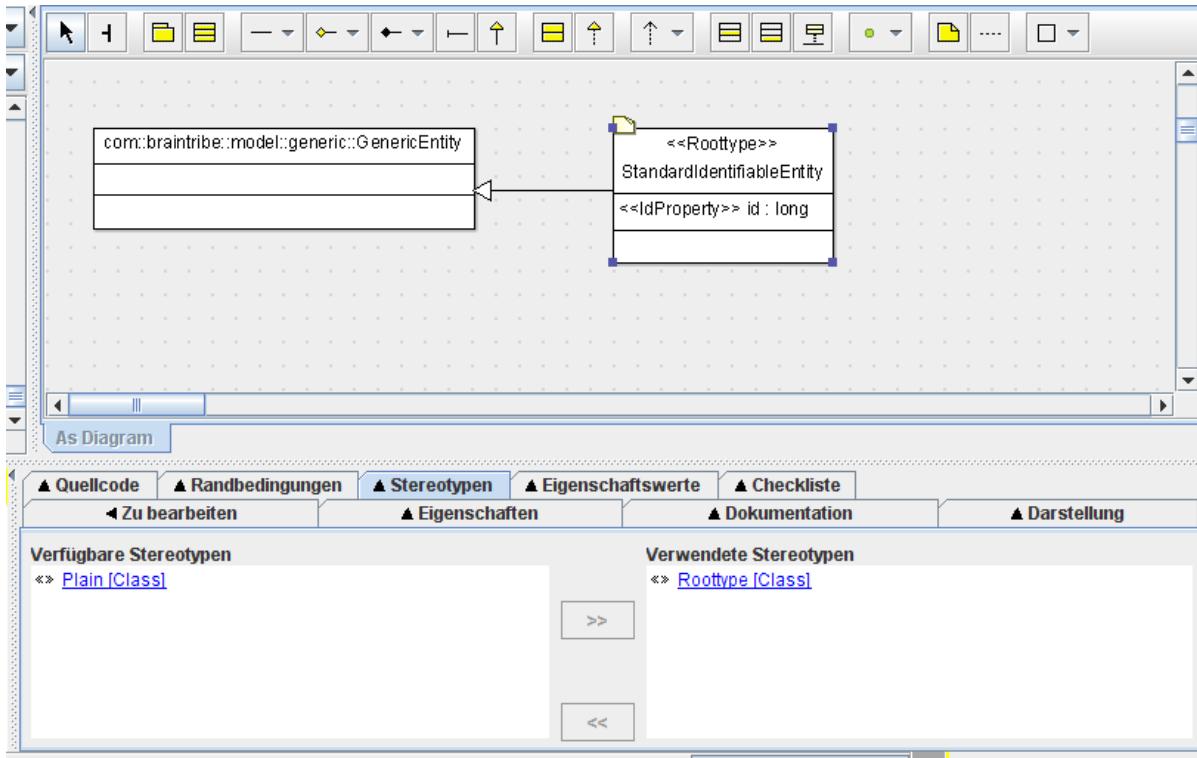
Plain

The **Plain stereotype** is only applicable to classes and marks the class that it should be generated as class in JAVA code. If you want an abstract class, add this stereotype and then set the abstract tick in the classes data. If the plain stereotype's missing, the entity is generated as an interface and the abstract tick ignored.



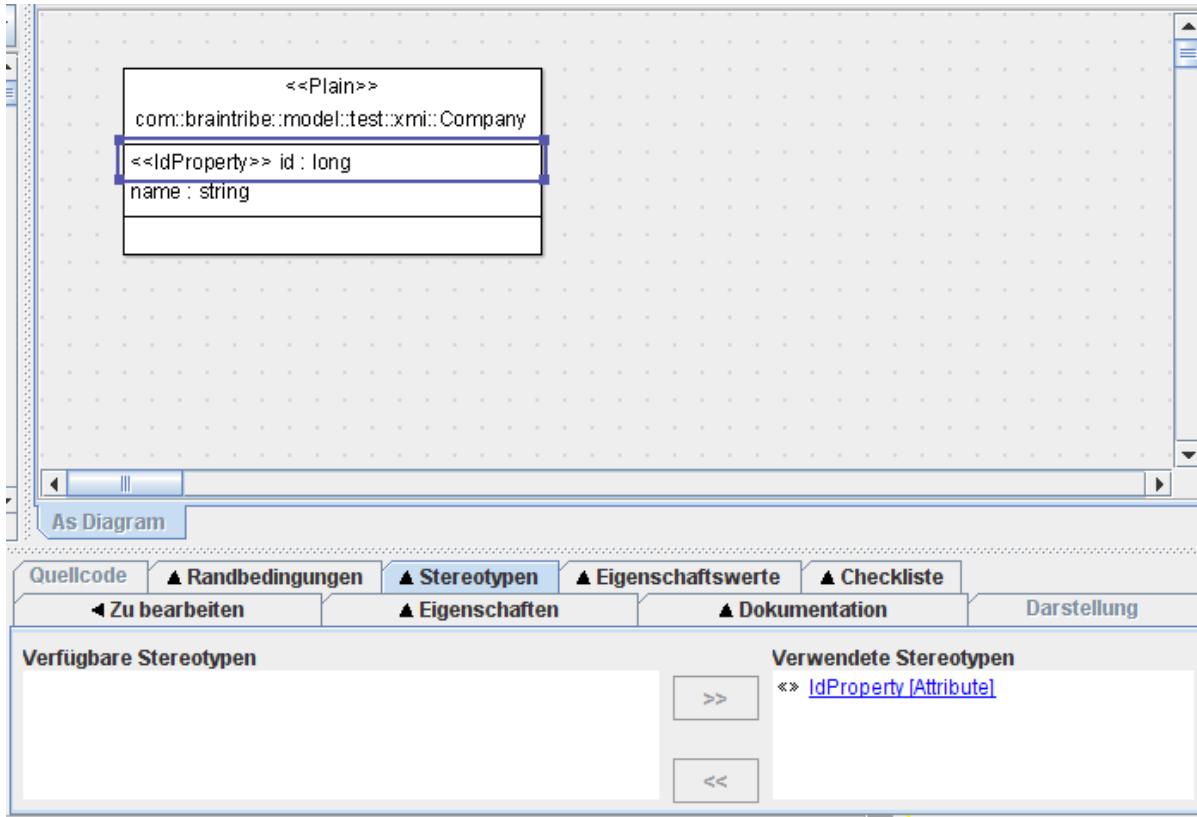
Roottype

The **Roottype stereotype** is a marker introduced for your convenience: You do not need to derive all classes from the GenericEntity - the codec's decoder will see that and automatically generate a reference to GenericEntity. If you have an alternative root type, you can add the RootType stereotype to it and the codec will automatically add references to this root type if not already present in the list of super-classes (or deriving from GenericEntity).



IdProperty

The **IdProperty stereotype** is used to declare an attribute as the id property - it will get the "@IdProperty"-annotation in the JAVA code.



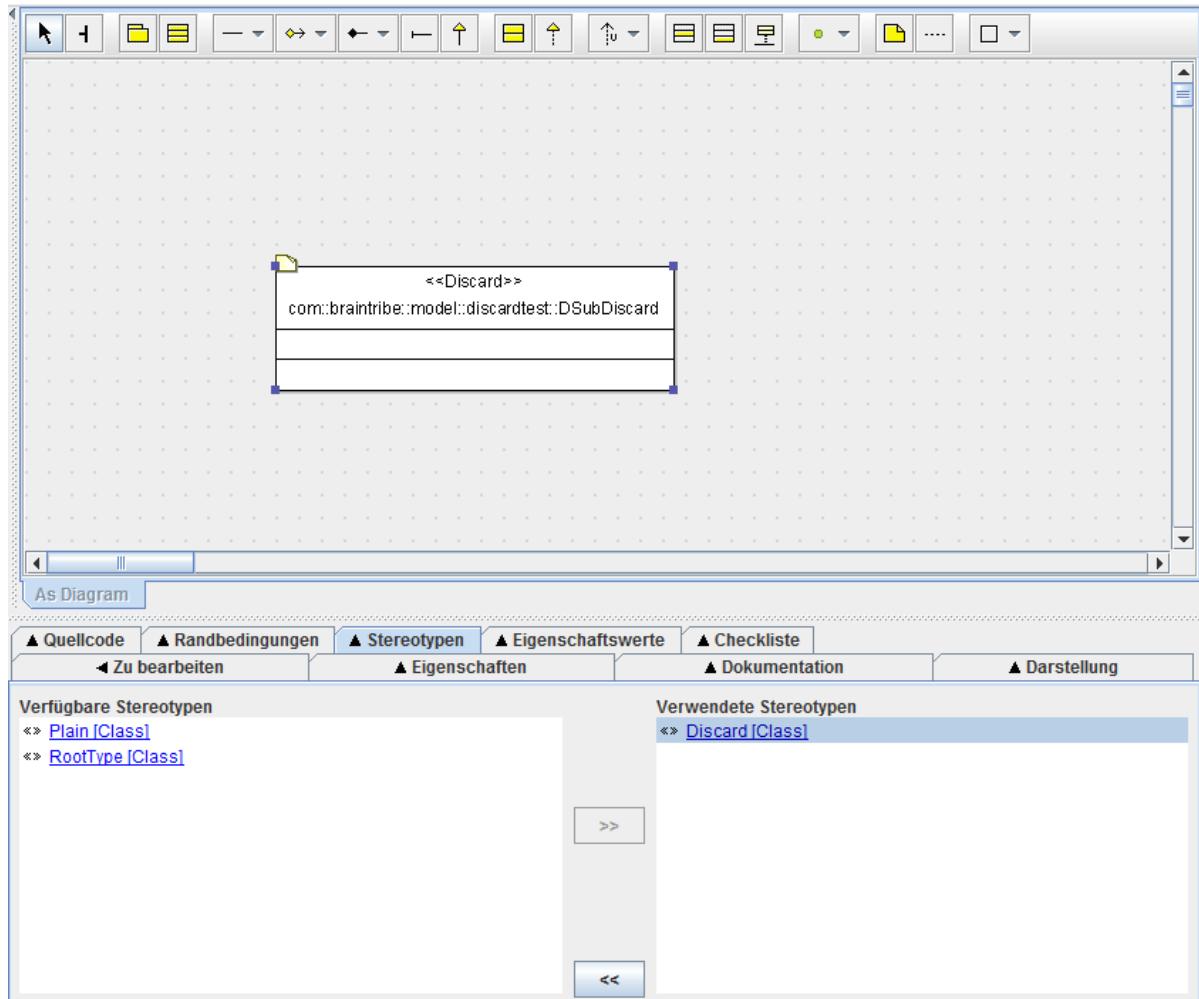
Discard

The **discard stereotype** is available for types, i.e. GmEntityType and GmEnumType instances. They are used to tag a GmType to be an internal

type that should be discarded in a deployment model.

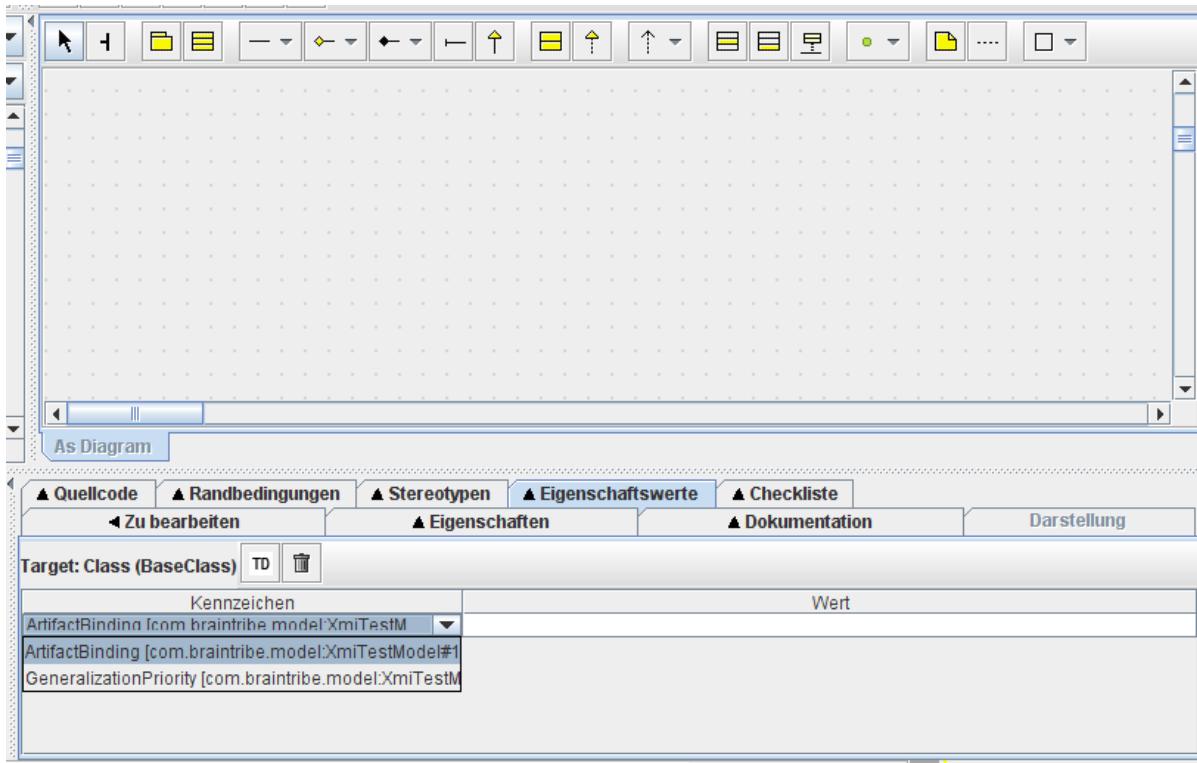
 See the following documentation for the rationale behind the discarding feature here:
[Refactoring the packing-info \(Google Doc\)](#)

The stereotype reflects the associated property in the GmType.



Tag-Definitions

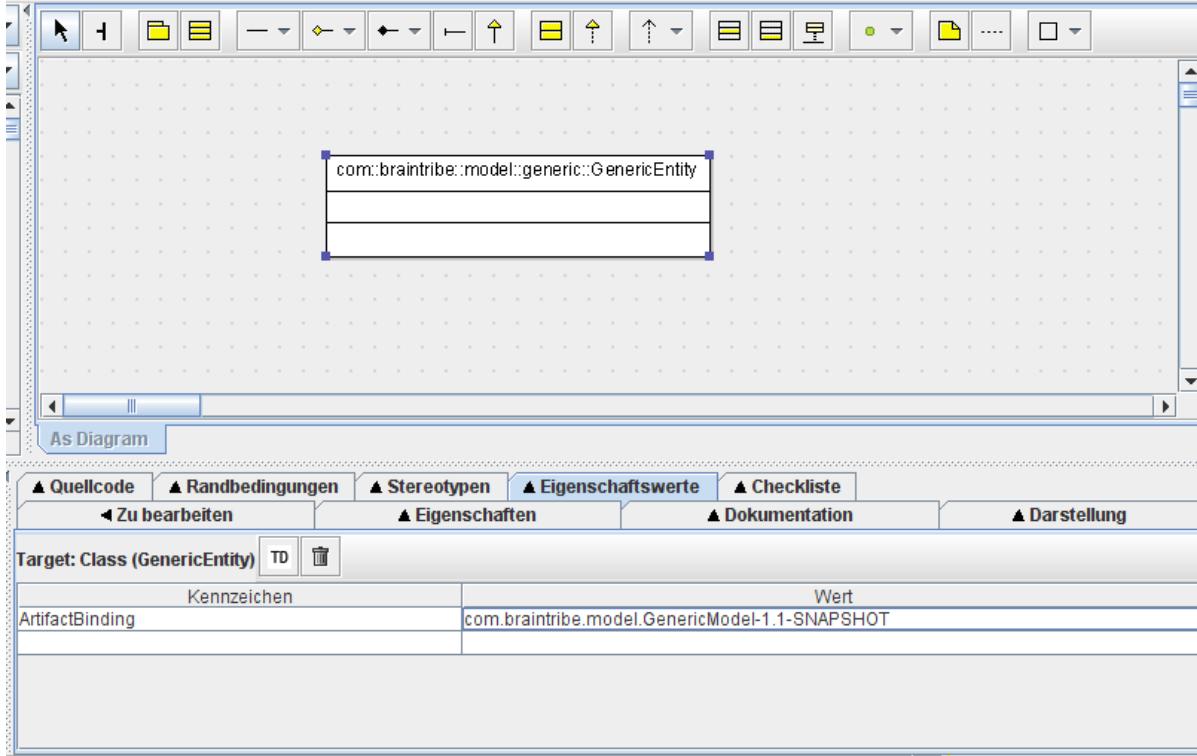
Tag definitions are markers that contain a value. You select the tab called "Properties". Then you select the tag definition on the left drop down list. Once you selected it, you can add a value.



ArtifactBinding

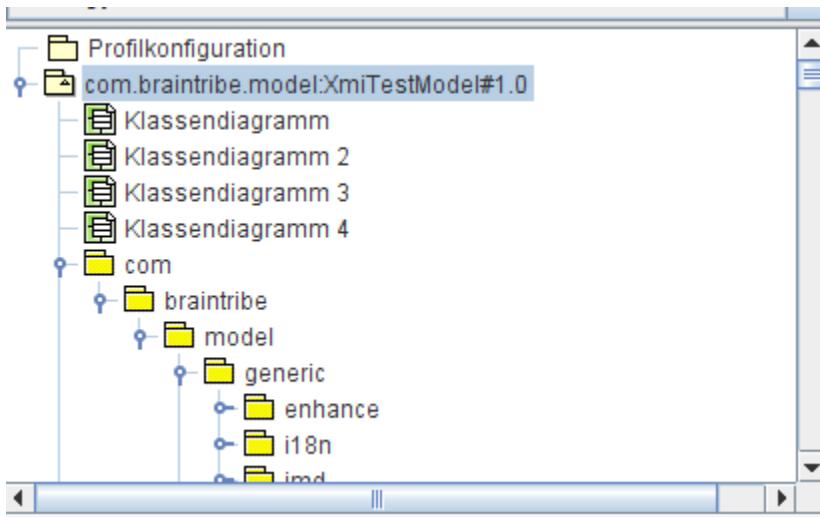
The artifact binding tag definition is used to tag an entity type or enum type to which artifact it belongs to. In most cases, you do not have to worry about this tag definition as it is written and read by the codec.

There are no user interactions required.

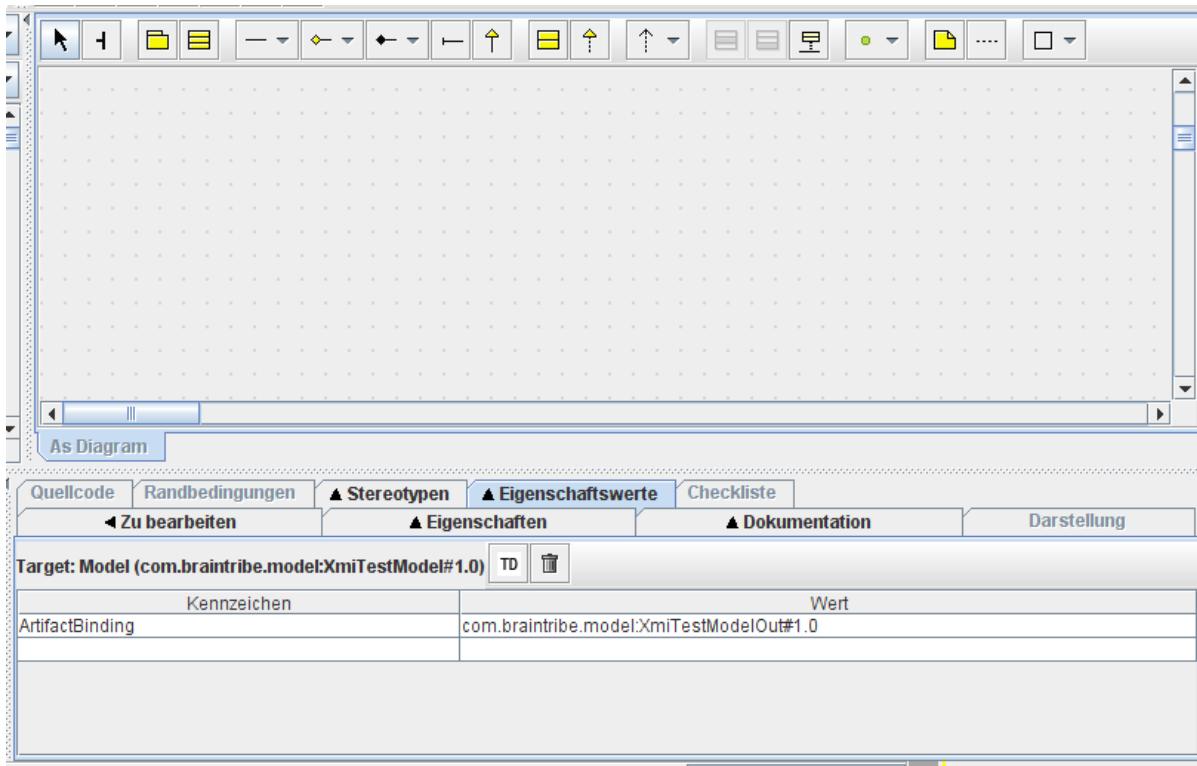


Note that the model entry itself has such a tag definition. It defines the actual target name of the model or better: the default binding. Any entity or enum type that has NO artifact binding specified is automatically regarded as being part of the models binding.

If you want to see - or change it, see below - you select the model in the package explorer's view.



In the dialog below the graphics, the settings for the model is shown and you can edit the value.



i This comes in handy if you want to create an new model using an existing model as base.

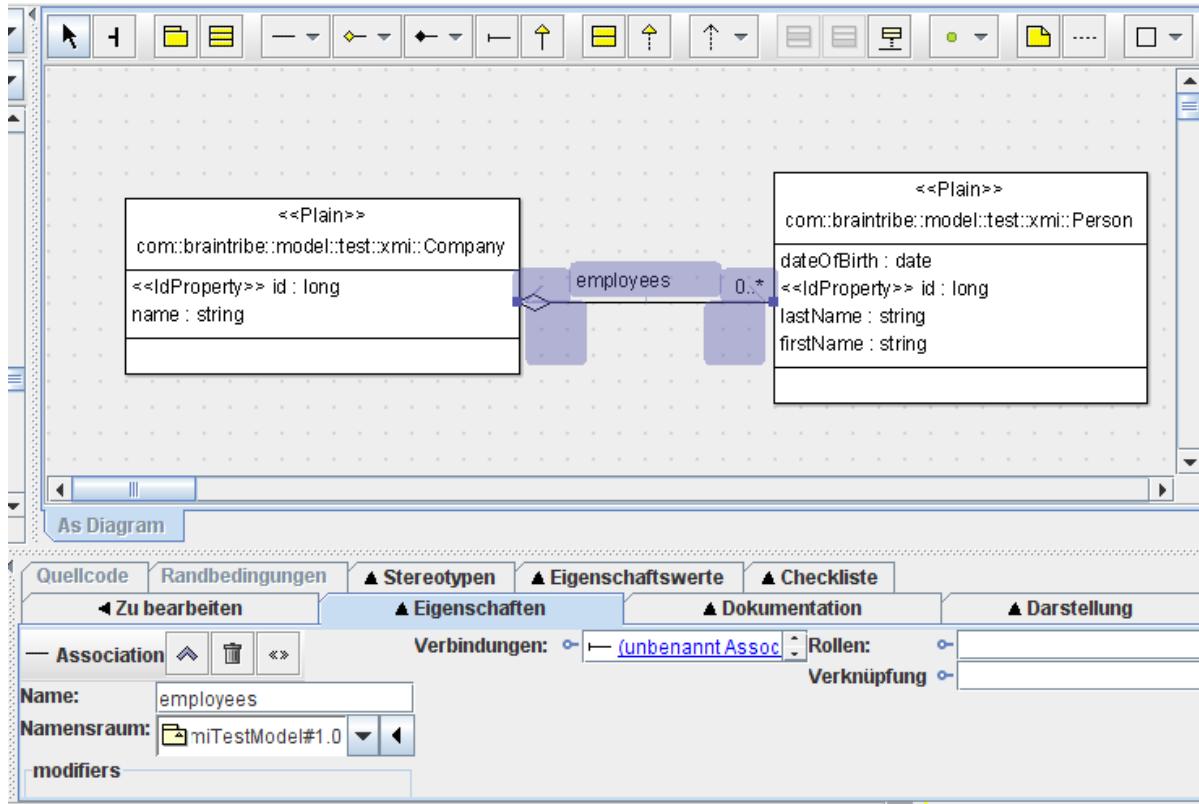
Please keep in mind that the codec will not specify a binding if a type belongs to the same model that you are extracting (i.e. share group id, artifact id and version you passed as argument). So if you change the target binding, all the types without a matching binding will be generated into the new model. So you want to use an aggregation model as a starting point, which is a model that has NO proper types, i.e. only consists of dependent types.

Then you can do the following:

Change the model's tag definition's value to the target artifact you want. Then start adding the new types you want, save the ARGO UML and generate the source code. The codec's decoder will automatically generate source for all types without a binding into the new artifact's source, and you end up with a brand new model artifact containing all your new types.

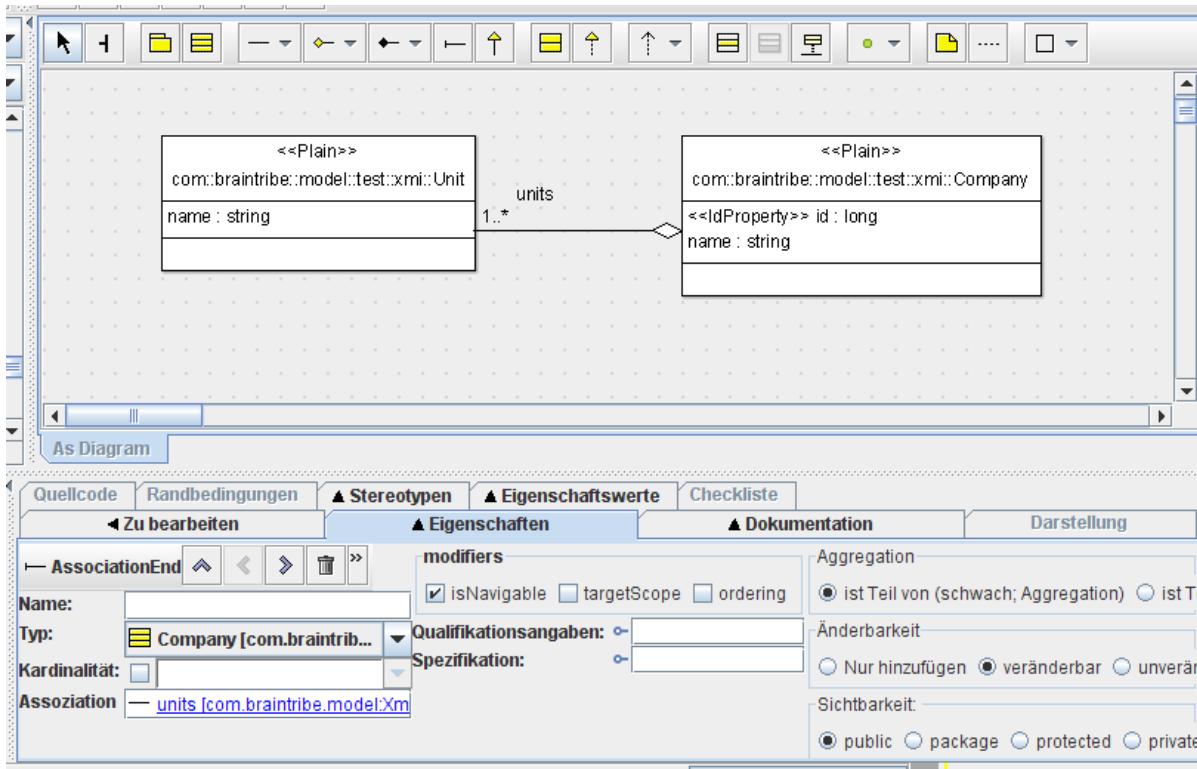
Collections

As mentioned above, collections are specified as associations, i.e. you link to classes with each other. The cardinality defines whether its a single association or if its a collective association. Only cardinalities 1 to n are supported. The name of the association declares the name of the member, and the ordered tick in its properties tells whether it's a set (unordered) or a list (ordered).



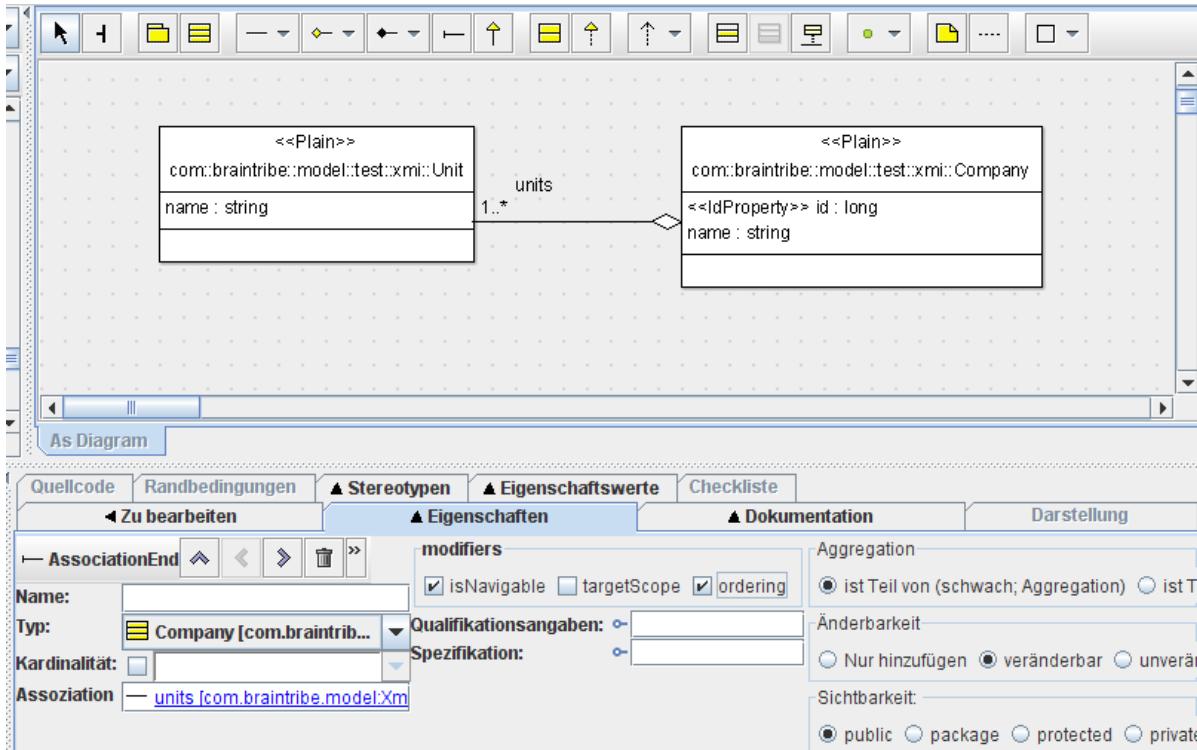
Set

A set is an association with a 1 to n cardinality and its ordered tick NOT set.



List

A list is an association with a 1 to n cardinality and its ordered tick set.



Maps

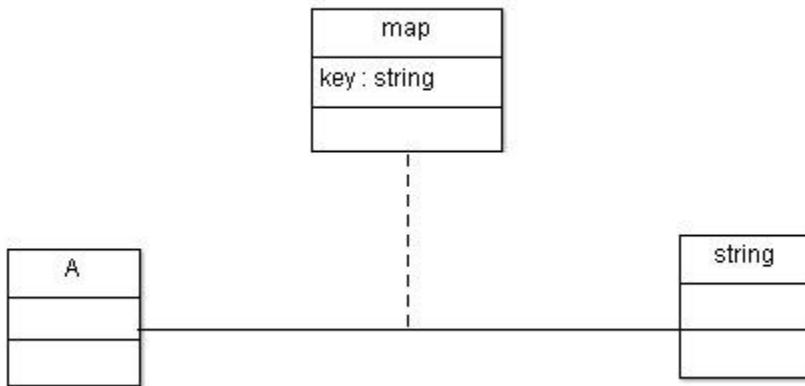
A map looks differently. Currently, there's not a standard UML way to do it. We decided to model it via a an AssociationClass or as a typed association.

Unfortunately, the way how you define an association class in ARGO UML and how it is displayed after it has been automatically generated is different. As a consequence, association classes you've defined in ARGO UML will always look different like the ones defined via the XMI codec.

If you want to declare a map in ARGO UML, you first make a simple association. Then you select the AssociationClass tool and you attach it to the graphical representation of the association. Then you can add an attribute to the AssociationClass.

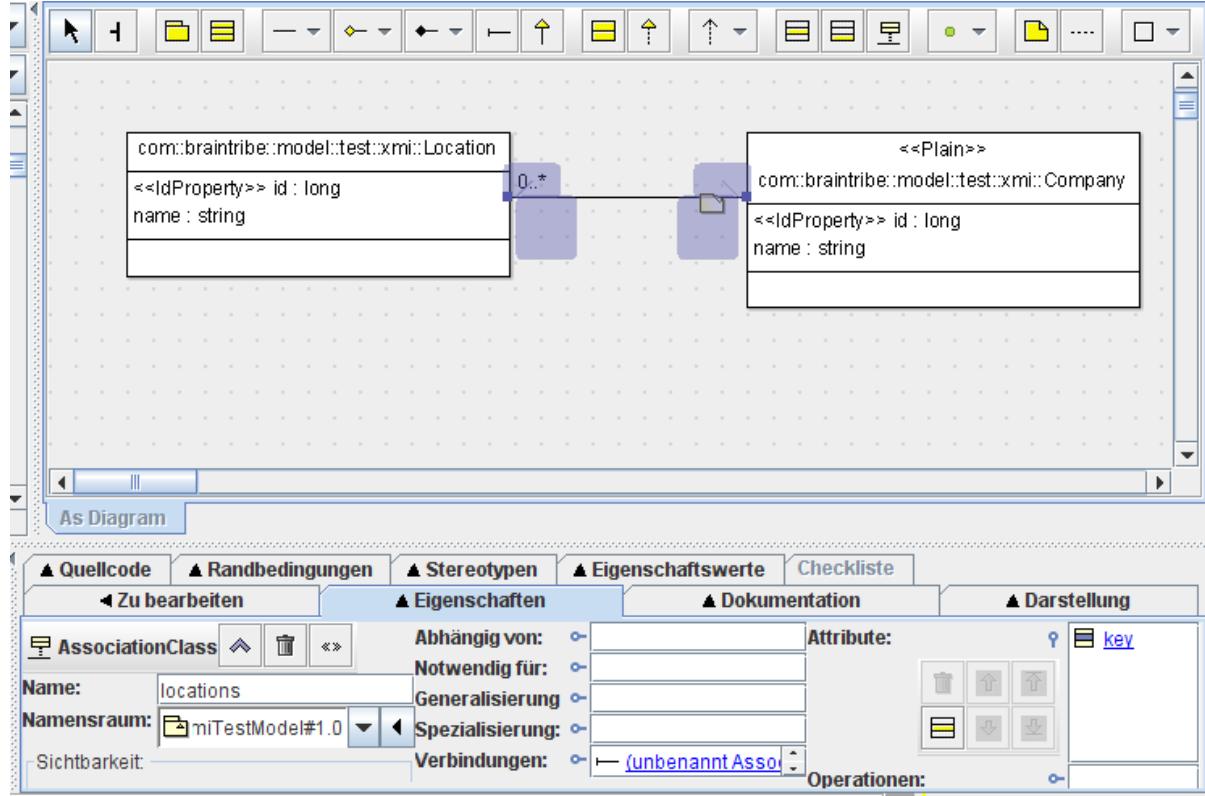
As a convention, the key of the AssociationClass MUST be named key in order to be interpreted as a map.

So, it will look about like this:

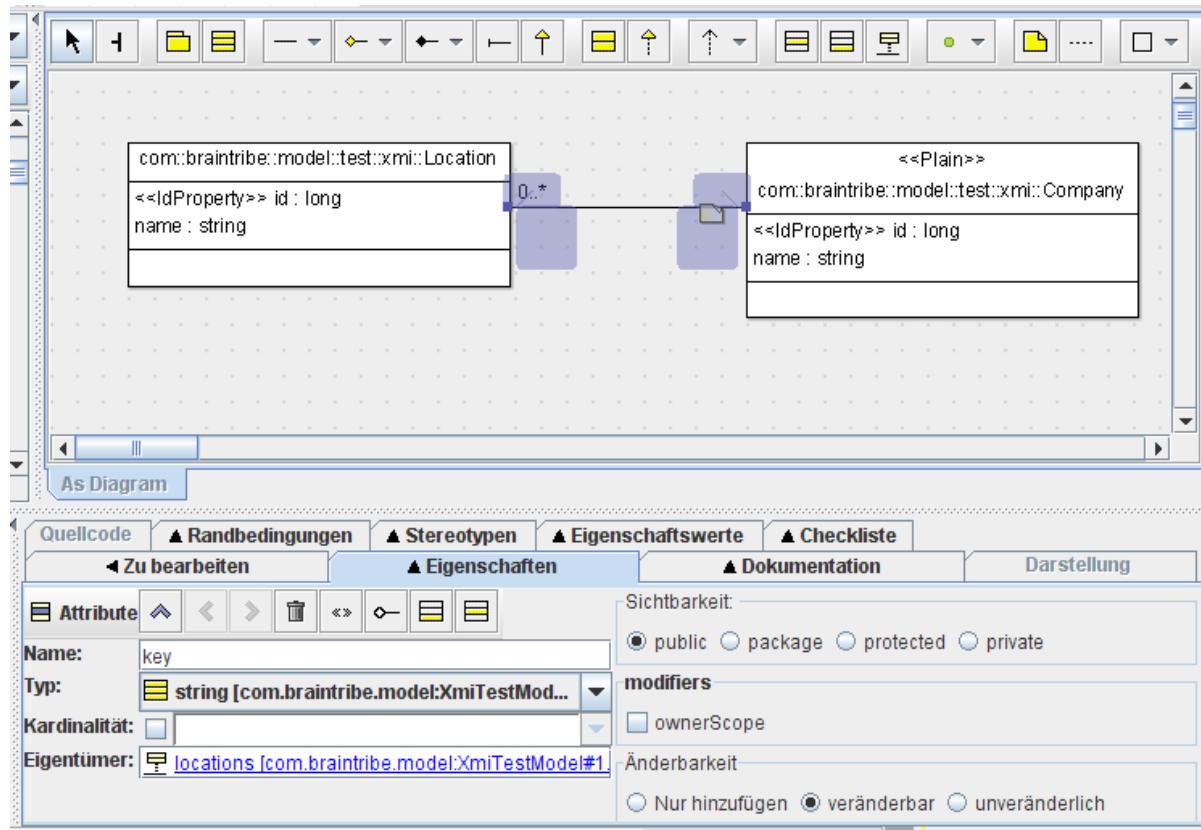


The picture shows the generic entity “A” which has a property called “map” of type Map<string, string>. The codec’s decoder will automatically generate a GmMapType for such an entry.

If the AssociationClass has been defined in the XMI, and loaded into ARGO UML, and then dragged into the graphics, it will look like this:



As you can see, the graphics differ. Unfortunately, neither the name nor the key value is shown in the graphics. But if you select the association in the display, the correct settings are shown in the dialog box below. Selecting the attribute “key” on the right side will show this dialog:



As expected, the key value itself is a standard attribute.

Base Aggregator Models for Future Modeling Tasks

A base aggregator model is a model that contains no GenericEntities by itself, and hence is perfectly suited to be used as a base for new models.

All you have to do is to modify the artifact binding of the model, add your classes to the model (without any artifact binding as they will inherit the binding of the model) and then parametrize the ant task to write your model.

Currently, two .ZARGO exist that can be used as a base for future models.

BaseAggregatormodel-1.0

This model is a basic model. Currently, it only includes GenericModel-1.1, but that might be expanded in the future. Use this model if you're planning to do a new model that has no other dependencies to our models.

DmsBaseAggregatorModel-1.0

Derives from the BaseAggregatorModel, and includes the DocumentModel-1.2. Use this model as a base for models that use the DMS basic functionality.

Standard ant targets

There are now new ant targets defined and imported in the build.xml files of the models.

The targets are:

- **sources-to-model**
Reads the project and extracts the model. The model file is always named as the model, with an added extension "xml".
- **model-to-zargo**
Reads the model file and creates the zargo. The zargo file is always named as the model, with an added extension "zargo".
- **zargo-to-model**
Reads the zargo file and extracts the model file. The model file is always named as the model, with an added extension "xml".
- **model-to-source**
Reads the model file and creates the sources in your working copy. Any existing sources are backed-up.
- **sources-to-zargo**
Reads your project and automatically creates a zargo file from it. It combines sources-to-model and model-to-zargo.
- **zargo-to-sources**
Reads your zargo and automatically creates the sources from it. It combines zargo-to-model and model-to-sources.

You can call them as any other build.xml target.

If they are missing in the build file of your model, you can add them as here:

```

<target name="sources-to-model" depends="compile">

    <mkdir dir="${dist}" />

    <bt:dependencies description="Launches Malaclypse dependency walk"
        pomFile="pom.xml"
        packagingFile ="${dist}/packaging.xml"
        filesetId="classpathFileSet"
        sourcesFilesetId="sourceFileSet"
        pathId="classpath"
        solutionListFile = "${dist}/solutions.xml"
    />

    <bt:generateMetaModelFromArtifact description="extracts a metamodel from an artifact
and its dependencies"
        buildDirectory ="${build}"
        groupId ="${maven.project.groupId}"
        artifactId ="${maven.project.artifactId}"
        version ="${maven.project.version}"
        filesetId="classpathFileSet"
        metaModelFile ="${dist}/${maven.project.artifactId}-${maven.project.version}.xml"
        solutionListFile ="${dist}/solutions.xml"
    />
</target>

<target name="model-to-zargo" depends="compile">
    <bt:generateZargoFromMetaModel description="generates a XMI with wrapping ZARGO from
an metamodel"
        metaModelFile ="${dist}/${maven.project.artifactId}-${maven.project.version}.xml"
        zargoFile ="${dist}/${maven.project.artifactId}-${maven.project.version}.zargo"
    />
</target>

<target name="sources-to-zargo" depends="sources-to-model,model-to-zargo" />

<target name="zargo-to-model" depends="init">
    <bt:generateMetaModelFromZargo
        zargoFile ="${dist}/${maven.project.artifactId}-${maven.project.version}.zargo"
        metaModelFile ="${maven.project.artifactId}-${maven.project.version}.xml" />
</target>

<target name="model-to-sources" depends="init">
    <bt:generateArtifactFromMetaModel
        metaModelFile ="${dist}/${maven.project.artifactId}-${maven.project.version}.xml"
        workingCopy ="${workingCopy}"
        groupId ="${maven.project.groupId}"
        artifactId ="${maven.project.artifactId}"
        version ="${maven.project.version}"
    />
</target>

<target name="zargo-to-sources" depends="zargo-to-model,model-to-sources" />

```


Dependency Management - Controlling the Uncontrollable

Table of Contents

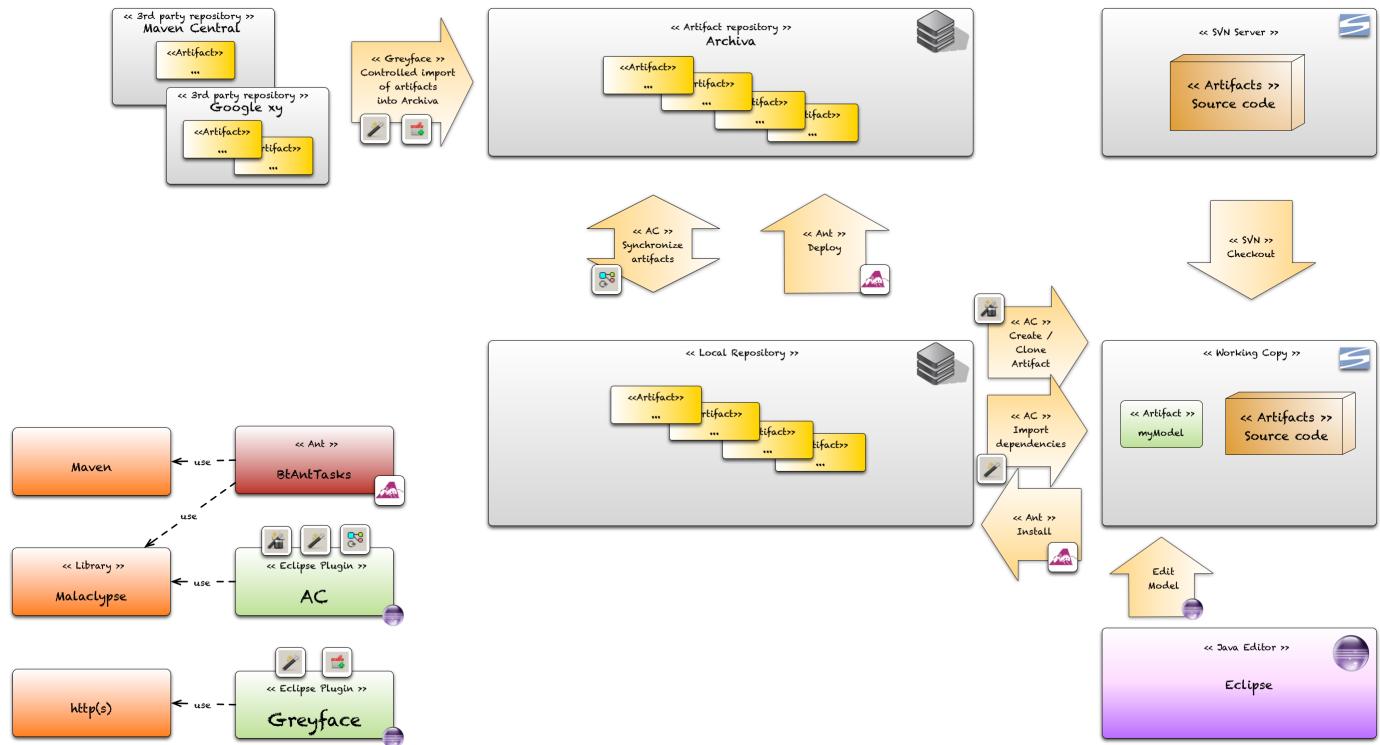
- ▼ Expand / collapse
 - Dependency Management
 - Overview
 - Components
 - Archiva
 - AC
 - BtAntTasks
 - Greyface
 - Malaclypse
 - Ravenhurst

Children Pages

- ▼ Expand / collapse
 - TFG-ACDC
 - TFG-BTAntTasks
 - TFG-Greyface

Dependency Management

Overview



Components

Component	Description
-----------	-------------

Archiva	<ul style="list-style-type: none"> • Company repository for artifacts • More information: <ul style="list-style-type: none"> • Archiva Website
AC	<ul style="list-style-type: none"> • Eclipse Plugin • AC: Artifact Container • Implements a dynamic classpath library (container) and implements several features for dependency management, build tools and artifact creation features. • Uses the Malaclypse library • Accessible via Toolbox in Eclipse: <ul style="list-style-type: none"> • Artifact Wizard:  Create or clone new artifacts: • Synchronize (Selected project or whole workspace):  Synchronizing the local artifacts with those contained at the global repository (Archiva) • Dependency Analysis:  Check for dependencies of an artifact. • Imports dependencies as projects to your working copy. • Synchronizes artifacts in local repositories with those in Archiva • Starts the Greyface plugin if artifacts in Archiva are needed • Detailed description: <ul style="list-style-type: none"> • ACDC (Confluence)
BtAntTasks	<ul style="list-style-type: none"> • Provide following targets: <ul style="list-style-type: none"> • install: Compiles the artifact from the working copy (your eclipse project) and puts it into the local repository as an artifact. • dist: Builds a distribution of the artifact. • deploy: same as install, but also uploads the artifacts into the remote repository (archiva). • assemble:same: same as install, but also creates a "deployable installation" e.g., BTAntTask: a zip file; Servlets: a war file. Note that not every model has the assemble task. • install-deps: Installs all dependencies as artifacts into the local repository. • update-wc: synchronises all dependencies of the artifact with svn. • BtAntTasks description: <ul style="list-style-type: none"> • BTAntTasks (Confluence)
Greyface	<ul style="list-style-type: none"> • Eclipse Plugin • Check for dependencies and import artifacts from 3rd-party repositories them into Archiva in a controlled manner • Handling of dependencies-clashes by giving you the chance to choose the proper version manually • Started via Toolbox in Eclipse: <ul style="list-style-type: none"> • Dependency Analysis  or • its own wizard  • Detailed description <ul style="list-style-type: none"> • Greyface (Confluence)

Malaclypse	<ul style="list-style-type: none">• Library• Implements features centered on artifacts and their dependencies (can be used in Eclipse, ANT, Gradle etc.).• Used by AC and BtAntTasks.• Detailed description:<ul style="list-style-type: none">• Malaclypse (PDF)
Ravenhurst	<ul style="list-style-type: none">• Servlet• Accesses the database of the Archiva installation and retrieves information about the installed artifacts, such as what artifact changed after a certain time stamp or what the actual time stamp of the parts of an artifact is• Used by Malaclypse's global-repository expert.• Detailed description:<ul style="list-style-type: none">• Malaclypse (PDF) (Yes, its in there)

TODO

▼ Expand / collapse

- Description of BtAntTasks in conjunction with Dependency Management
- Extend this page with the QA and release aspects of Panther (Current situation vs. where do we want to go) Publishing Artifacts aNd Testing eNhanced Release Manager

TFG-ACDC

AC / DC

Artifact Container / Dependency Controller

current version : 2.4.0

[Artifact Container / Dependency Controller](#)
[How to get AC](#)
[Artifact Model](#)
[Artifact Container](#)
[Plugin Preferences:](#)
[Adding an artifact container to the classpath](#)
[Container properties](#)
[Project kind](#)
[Builders](#)
[Clash resolvers](#)
[Magic Scopes](#)
[Compile](#)
[Runtime](#)
[AC as a dynamic library](#)
[Standard behaviour](#)
[Dynamic behaviour](#)
[Commands](#)
[Dependency wizard](#)
[Parameter](#)
[Local repository only \(no synchronising\)](#)
[Synchronize with remote repository](#)
[Update settings for artifacts](#)
[Update settings for parts](#)
[Further settings](#)
[Dependency View](#)
[Unresolved dependencies](#)
[Dependency clashes](#)
[Undetermined dependencies](#)
[Undefined dependencies](#)
[Solution view](#)
[Ambiguous solutions*](#)
[Synchronizing view](#)
[Auto synchronizing](#)
[Transitive builds](#)
[Artifact related svn tasks](#)
[GWT features](#)
[Saving a gwt.xml file](#)

AC/DC actually consists of five projects:

Project	Description
Artifact Model	A Generic Model that models all classes involved in artifacts and their dependencies
Malaclypse	a library that implements all features (can be used in Eclipse, ANT, Gradle etc)
ArtifactContainer AC	the eclipse plugin to control the classpath and implement other useful features of the artifact code-base.
Greyface	the eclipse plugin to import artifacts into our company repository.
AcDevloader	A class loader for Tomcat 7 and AC based projects as a replacement for the standard. Improves usability in conjunction with the Sysdeo plugin.

Ravenhurst	A servlet that accessed the database of our Archiva installation and retrieves information about the installed artifacts, such as what artifact changed after a certain time stamp or that the actual timestamp of the parts of an artifact is. It is used by Malaclypse's global-repository expert.
<i>Dependency Controller (DC)</i>	<i>a servlet that gives access to company wide settings and features, such as release/publish procedures, check builds etc.</i>
<i>Khayyam</i>	<i>a graphical viewer for dependency trees.</i>

Italics and grey means that I haven't found the time to seriously work on those projects.

How to get AC

You can get the newest version from Google docs;

AC: Check the file in the RoadMap/Public section or from the main section. There should only be one AC hanging around - if not, always select the newest one.

Right, the name: as all plugins we do, it is fully qualified, i.e.
com.braintribe.ArtifactContainer.<version>.<qualifier>.jar.

or the update sites:

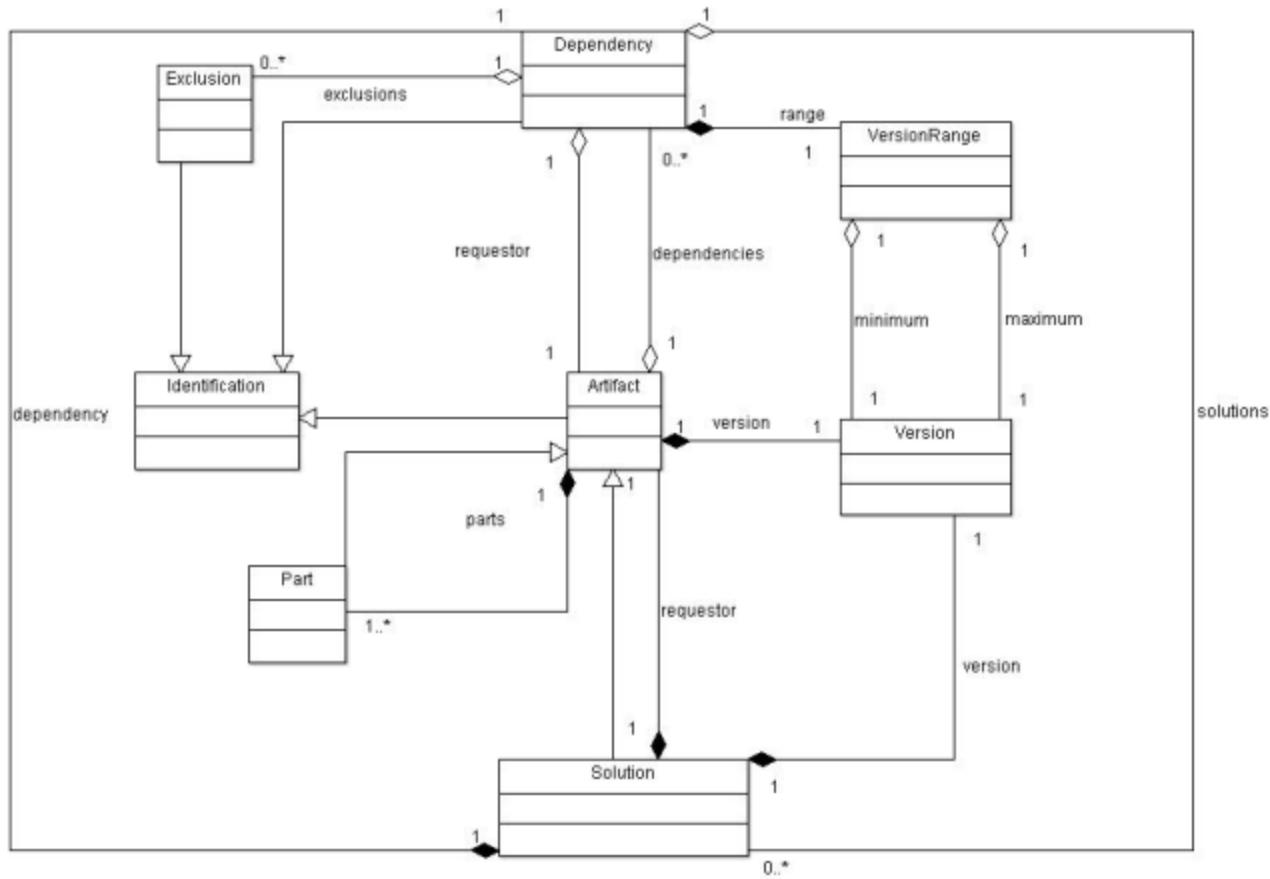
{+}<http://kwaqwagga.ch/update-site>{+}

{+}<http://inf-web.braintribe:9999/update-site>{+} (VPN)

You add AC to your build path like you did with AC 1.x, just select "add library" in "edit buildpath".

Artifact Model

Underlying all projects is the new Artifact Model, a Generic Model.



The elements in detail:

Name	Description
Version	exactly what its name implies
VersionRange	a range of versions (or a single version), precise or fuzzy
Identification	the combination of group- and artifact-id
Exclusion	actually a Identification itself (appears in poms to exclude a identification from parsing)
Dependency	the thing that appears in a pom as a dependency (which has a VersionRange)
Artifact	derives from Identification and introduces a version
Part	a file associated with an Artifact (jar or java-doc or sources whatever)
Solution	an existing artifact to a dependency (which has a Version)

A few things that you need to know about dependencies and solutions

As mentioned above, there is relationship between a dependency and a solution. The dependency is the declaration of what an artifact needs (i.e. exactly what is written in the pom as dependency entry). The solution however is what has actually been found. There is a one-to-many relationship here, as a dependency's version range may be fuzzy, i.e. consist of an interval of a minimal and maximum version or even be an undetermined version range (i.e. not specifying a version information at all). This simply means that one single dependency may have several solutions to it and - obviously as the class path can only have one solution per dependency - the resulting set of solutions of a dependency walk must be sorted out once we reach the end.

Clash resolving

Clash resolving is an important issue in dealing with dependencies. It happens on two levels.

- Dependency level: This is the obvious case - if there are two conflicting dependencies in the tree, they must be resolved. Resolving can either lead to either use of them or - if they're of the interval kind - to a new dependency with a merged range.

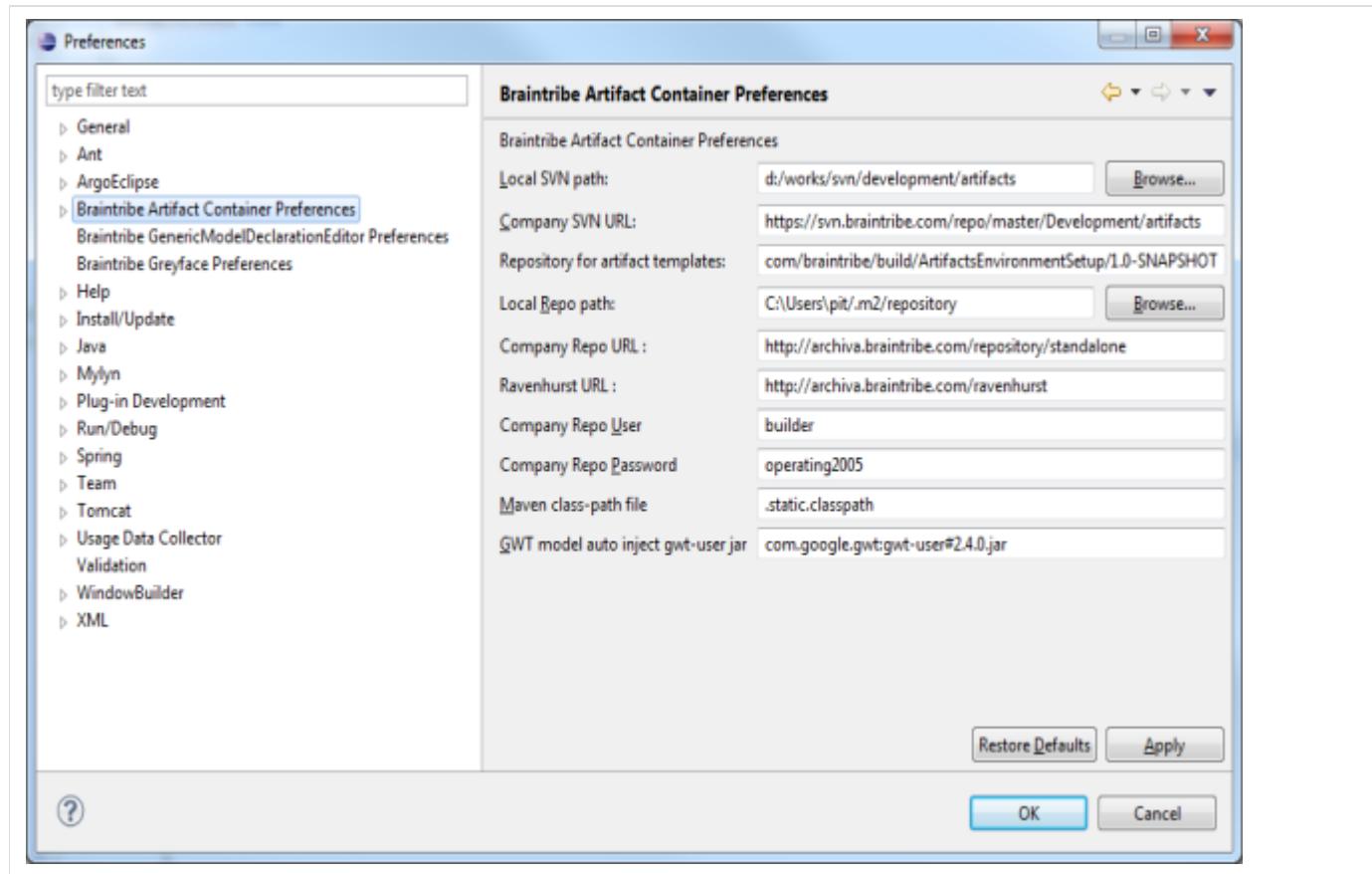
- Solution level: This occurs if a dependency is either undetermined (the requesting dependency has NO version preference) or if it is fuzzy, i.e. the range has several possible matching solutions.

There are three automatic clash resolvers.

- Optimistic clash resolver: It will always prefer the version that is regarded as being the higher version.
- Maven-style resolver: It will choose the dependency which has been encountered first. (Obviously, this fails in case of the fuzzy solutions as they're solutions to the same dependency).
- Top-level clash resolver: It will select the dependency that has been introduced in the highest recursive level, i.e. that one specified in the pom of the current terminal artifact will win. (Obviously again, this fails in case of the fuzzy solutions as they're solutions to the same dependency).

Artifact Container

Plugin Preferences:

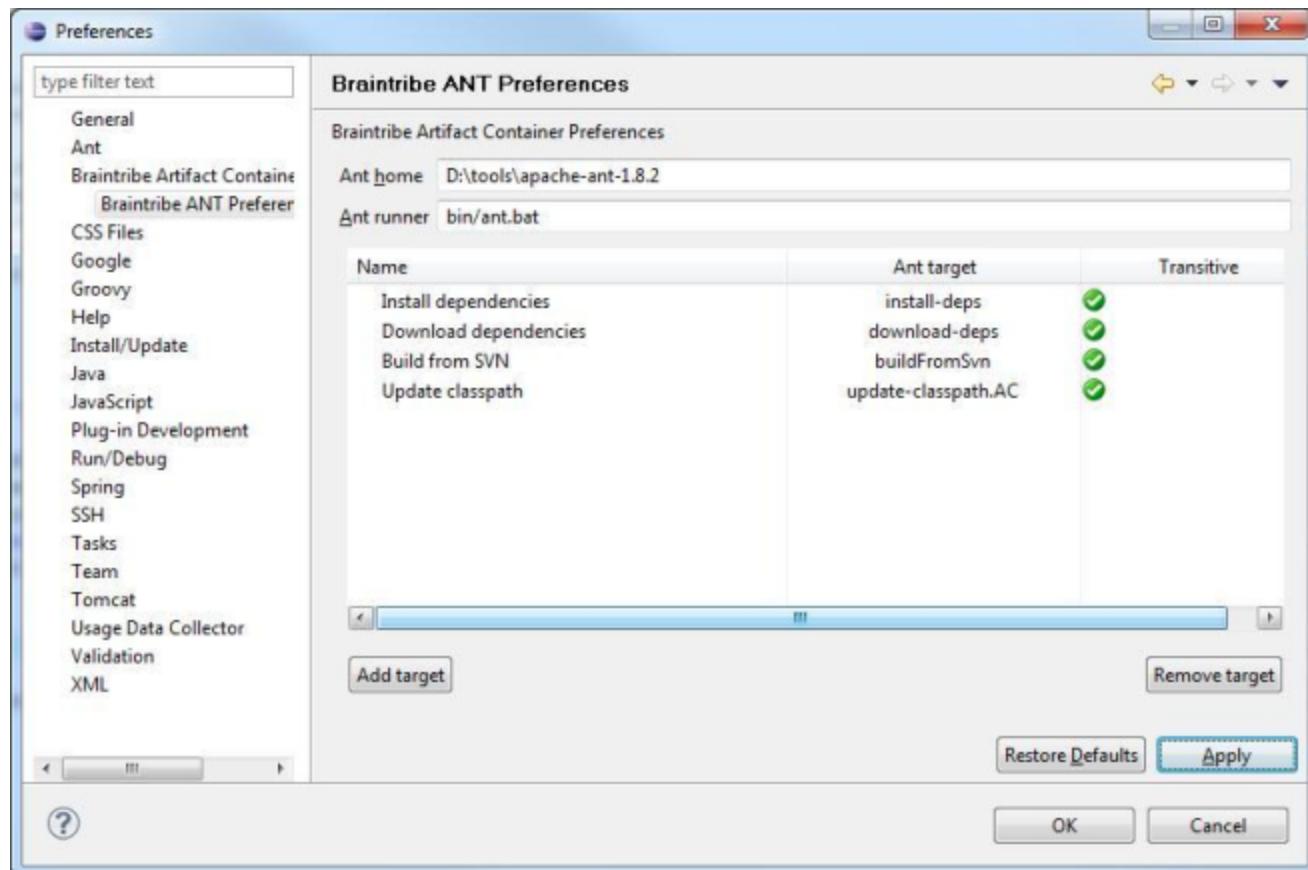


The preferences are stored in a XML file located in your user's home directory. So you just have to adapt them once to your system. If you open a new workspace, the stored settings are ready and incorporated.

The settings in detail:

Name	Description

Local SVN Path	Path to your local svn working copy
Company SVN URL	URL of the company svn repository
Local Repo path	Path to your local artifact repository
Repository for artifact templates	Ignore for now, just keep default setting
Company REPO URL	URL of the company artifact repository that is <i>not</i> linked to other repositories.
Ravenhurst URL	URL of the Ravenhurst servlet (required to retrieve update information from the company repository)
Company REPO user	The user for the repository access (the default user "builder" is a restricted user that has exactly sufficient rights to up- and download artifacts).
Company REPO pwd	The appropriate password
Maven classpath file	If AC's set the use Maven as dependency controller, set the static classpath file name here. Standard is ".static.classpath" (from BtAntTasks-1.6 up)
GWT model auto-inject gwt-user jar	The gwt-user.jar that will be automatically injected for Models and GWT projects.

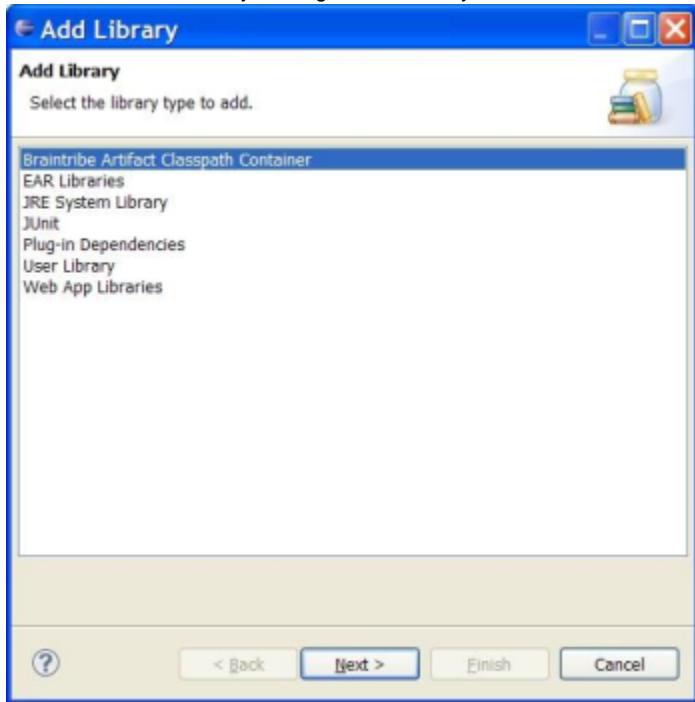


The settings for ant in detail:

- Ant Home : ANT home directory
- Ant Runner: the actual ANT runner, relative to the Ant home directory
- Transitive ant targets: You can add transitive ant targets that you want to be able to run from the Ant-Wizard (see below). Currently, only transitive targets are enabled, so you can safely ignore the third column, i.e. leave it on transitive as standard.

Adding an artifact container to the classpath

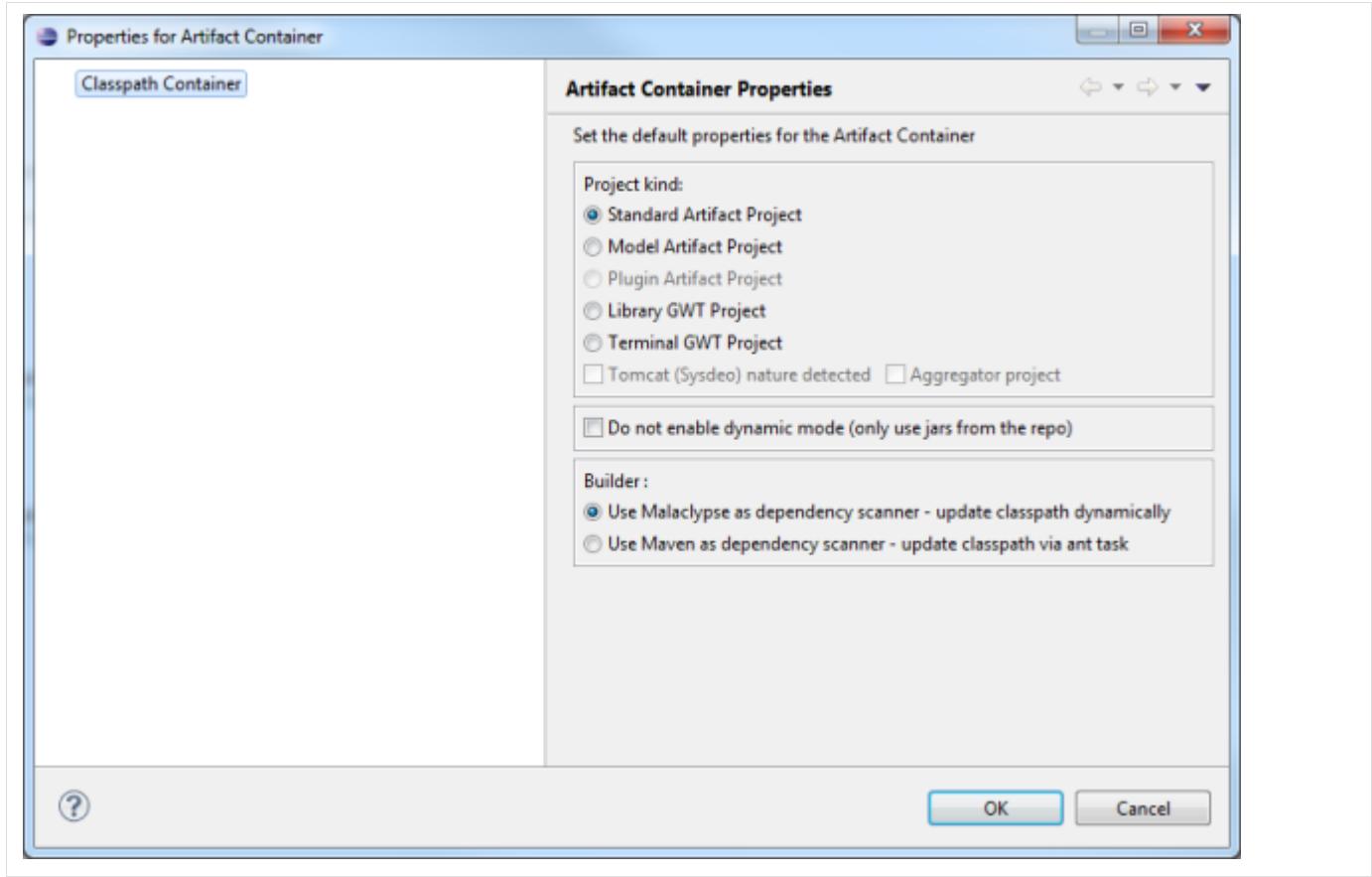
Once you set that, select the project you want to insert the container in. Go to Project/Properties/Build path or Build path/Configure Build Path in the right button menu. First, remove any M2_REPO references, you won't need them anymore. Then add the container by clicking on Add Library.



Select the entry for the container and press Next.

Container properties

Now the container preferences will show. You can set them here, but you can change the properties anytime by accessing the properties sub-menu from the context menu of the package explorer.



Each container has its own settings.

Project kind

- **Standard project** : The project is a standard artifact.
- **Model project** : The project is Generic Model. If checked with the GWT feature, the gwt-user library declared in the preferences will automatically added be to the classpath passed to the GWT feature.
- **Plugin Artifact**: experimental setting for an Eclipse plugin. If interested, contact me.
- **Library GWT project**: Same features as above are activated. The distinction was made to support future enhancements.
- **Terminal GWT project**. The project is a terminal GWT artifact. If you set that switch, AC will create duplicate path entries for any artifact-dependency, one for the jar and one for the sources (instead of using the standard complex Eclipse entry that incorporates all in one). And any dependency that is currently loaded in the workspace will also get the source classpath-entry additionally to the project classpath-entry.

Read-only toggles

- **Tomcat (Sysdeo) nature** : the project is a servlet. Special support is activated, see the documentation : https://docs.google.com/a/braintribe.com/document/d/1bmoHLhmsKSW_uScpvvSLBTzMD75fvLgRXoqEkySsdrw+
- **Aggregator project** : If you set the packaging in XML to pom, AC will treat the project as an aggregator project, i.e. a "pom only project". It will include the file in the walks and scan its dependencies, but it will not create a solution to add to the classpath. Yes, that's standard behaviour anyway if AC doesn't find an associated jar, but using this as a fallback will generate ugly error messages in the log.

Dynamic mode toggle

- **Do not enable dynamic mode**: This toggle activated, AC will no longer generate project dependencies and only use jar dependencies. It was introduced to circumvent a problem in Eclipse handling of the internal classpath when projects were involved (it didn't do a proper clash resolving, even if AC did). It is most probably no longer required from AC 2.4 on.

Builders

There are currently two classpath builders available.

- **Maven:** the standard maven dependency resolver is linked in via an ant task. In order to activate that, you'll need to install the BtAntTask-1.6 into ant. AC will still track whether a dependencies is loaded or unloaded, but will not react on changes of pom.xml. On the contrary you must manually call "update dependencies" which will run the ant task and then incorporate the results. It is included as a choice and as a fallback - should we get wrong results with Malaclypse, we can switch back to Maven and still use AC. *One note here: In this mode, AC will scan the resulting classpath file from Maven and extract the contents of the container from it. Due to the fact that versions and classifiers cannot be properly distinguished once built into a name (a.b.c-1.0-x-y-z.jar), AC may not be able to correctly determine what artifact is meant and most probably won't be able to add source and/or javadoc support.*
- **Malaclypse:** Our own classpath resolver is more flexible and more transparent than the one from Maven. It is our plan to use it exclusively, but it is still in a testing phase. And frankly, it's always good to have a fallback. If this builder is selected, the pom.xml is tracked and any changes are directly reflected in the classpath.

If you open the properties window of a container and leave by pressing the "OK"-Button, AC will automatically update the dependencies.

Magic Scopes

AC from 2.3 on (or rather Malaclypse) uses magic scopes to determine the classpath or the packaging path. Therefore, you can no longer set the scopes for a specific project. The magic scope themselves consist of a combination of standard scopes.

Note: all other scopes are automatically disregarded. Optionals as well.

Compile

One magic scope - the one used for the classpath which is of interest here - is the **compile** scope.

Included scope	Description
compile	This scope is the default scope. So any dependency that has no scope declared is automatically deemed to have this scope.
provided	Provided is a scope that tells that the dependency is needed to compile, but is not to be packaged as the target environment already has the dependency resolved.

Runtime

The second magic scope - runtime - is used for packaging runs.

Included scope	Description
compile	Again, the default scope.
runtime	Runtime is the scope for dependencies that are not required to compile the project, but are indirectly required to RUN the project.

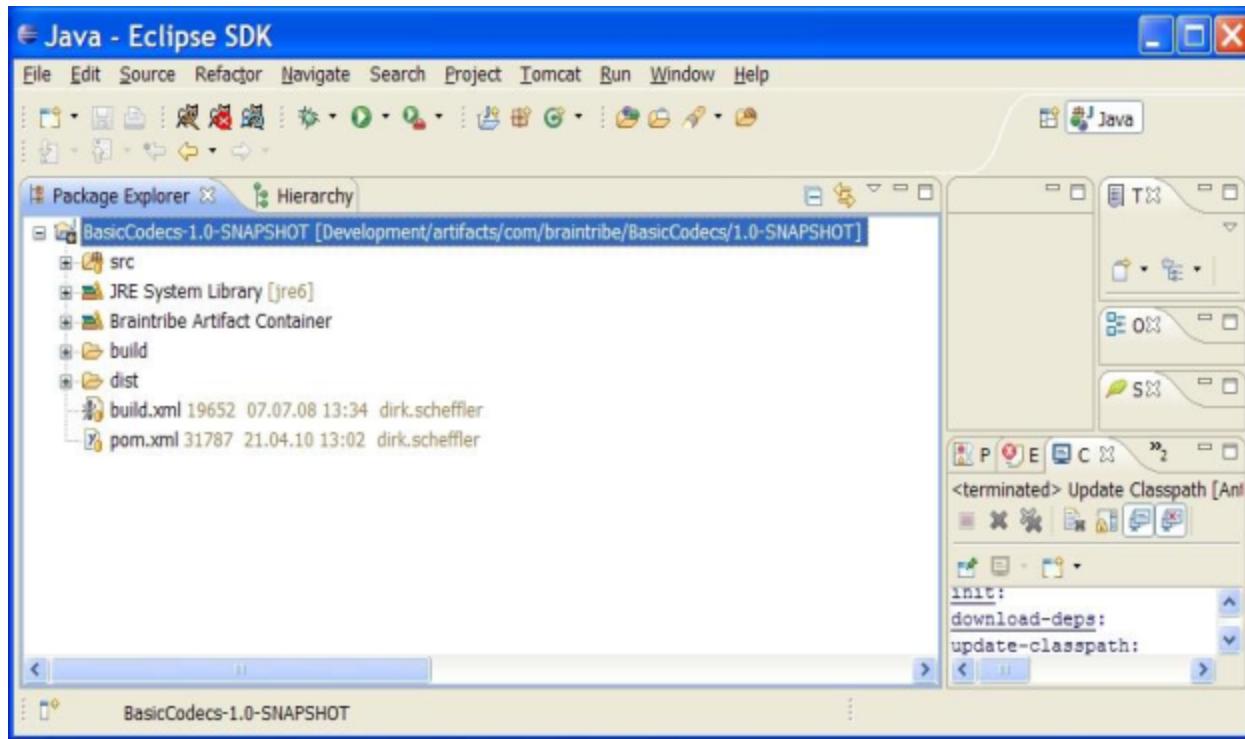
AC as a dynamic library

Classpath containers in Eclipse are nothing new. What makes AC special, is that it tries to implement a dynamic container - a container which content changes according to several criteria.

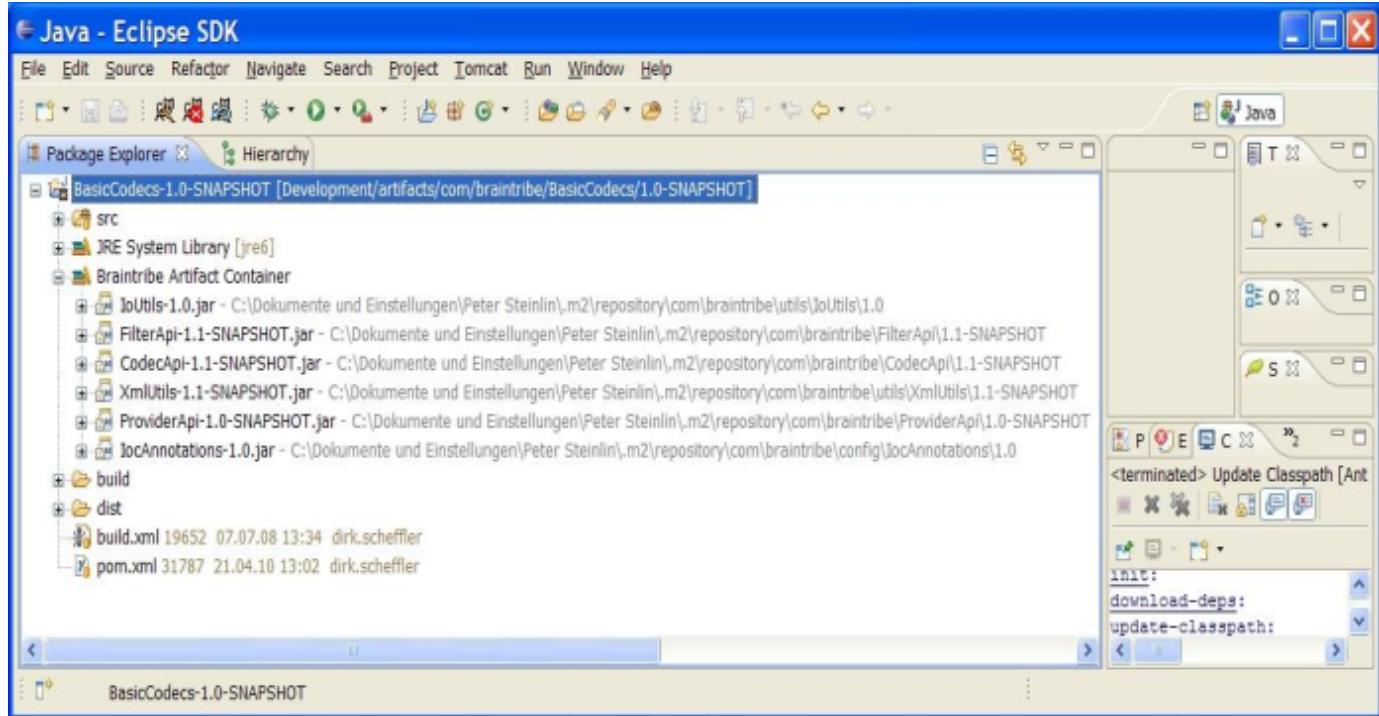
Eclipse doesn't really support such an idea. So it wasn't easy to find a convenient way to integrate such behaviour.

Standard behaviour

First of all, let's look at its standard behaviour: It's just like a standard classpath library:



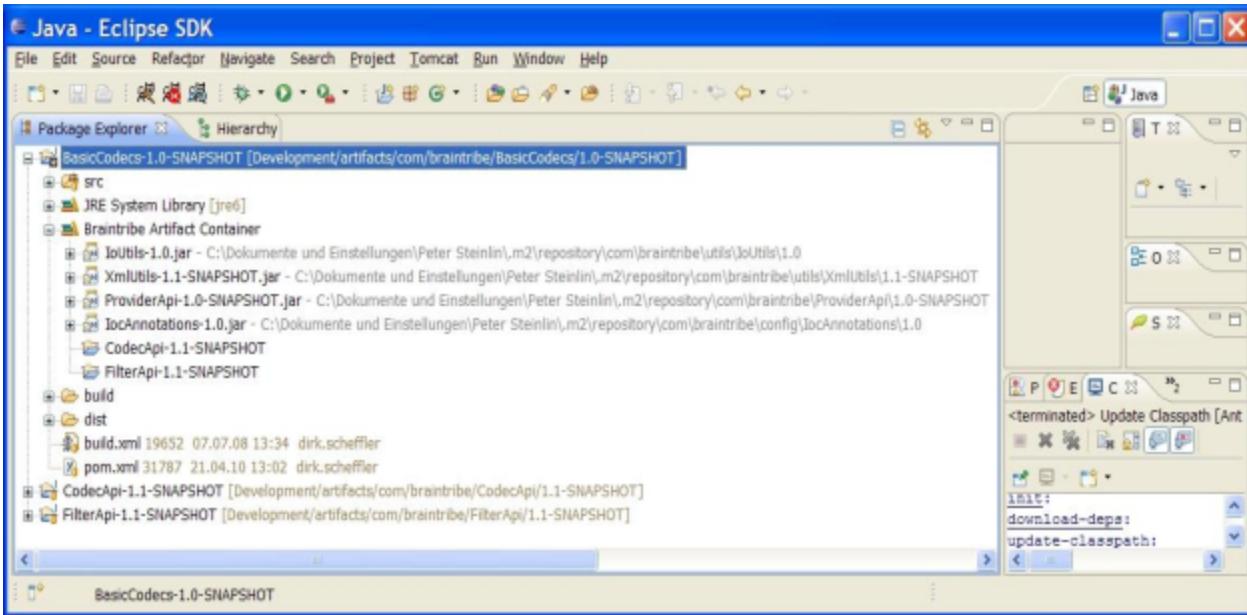
As you can see, there is now a second container in the classpath, just below the JRE System Library Container. Open the container and you'll see that it references all the dependencies of your artifact.



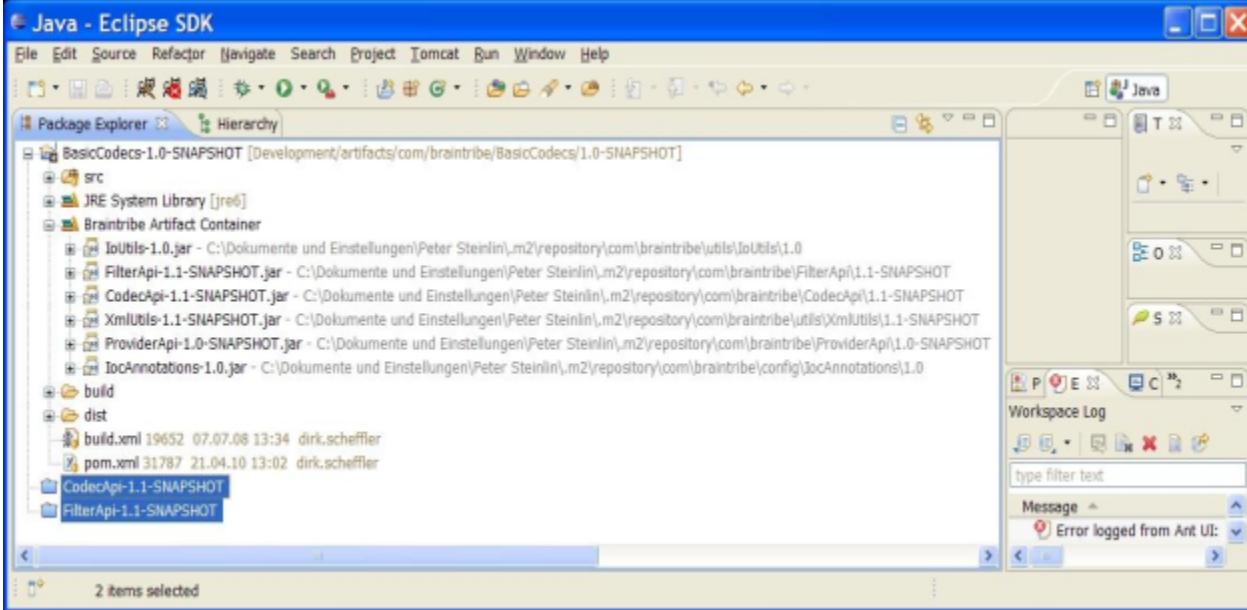
Any sources that are associated with the jar file of a referenced artifact will be automatically added to the classpath's entry of the referenced artifact. The same applies to any associated java-doc files.

Dynamic behaviour

As AC is a dynamic Classpath Container, any changes in the workspace will reflect in the classpath. If AC finds that a referenced Artifact is present in the workspace, it will replace the jar-reference with a workspace reference.



And vice versa. It reacts on changes in the workspace, i.e. does it automatically.



If you choose to modify the pom.xml of a loaded project, the plugin will automatically parse it and retrieve any added dependencies and reference them appropriately. If the dependencies are found in your local repository, they are automatically downloaded, if they exists as projects, they will be referenced as such.

Commands

AC implements some commands and triggers.

- Commands are :
 - Synchronizes your local repo with the remote repo and updates the container. has two modes: synchronize and update current project synchronize and update all projects in workspace
 - Hammer: kicks Eclipse's shins to make it read the changes in the classpath (as Eclipse doesn't like to refresh regularly) - shouldn't be required that often any more in 2.4.0
 - Run the GWT dependency checker
 - Trigger transitive ant targets (opens transitive target wizard)

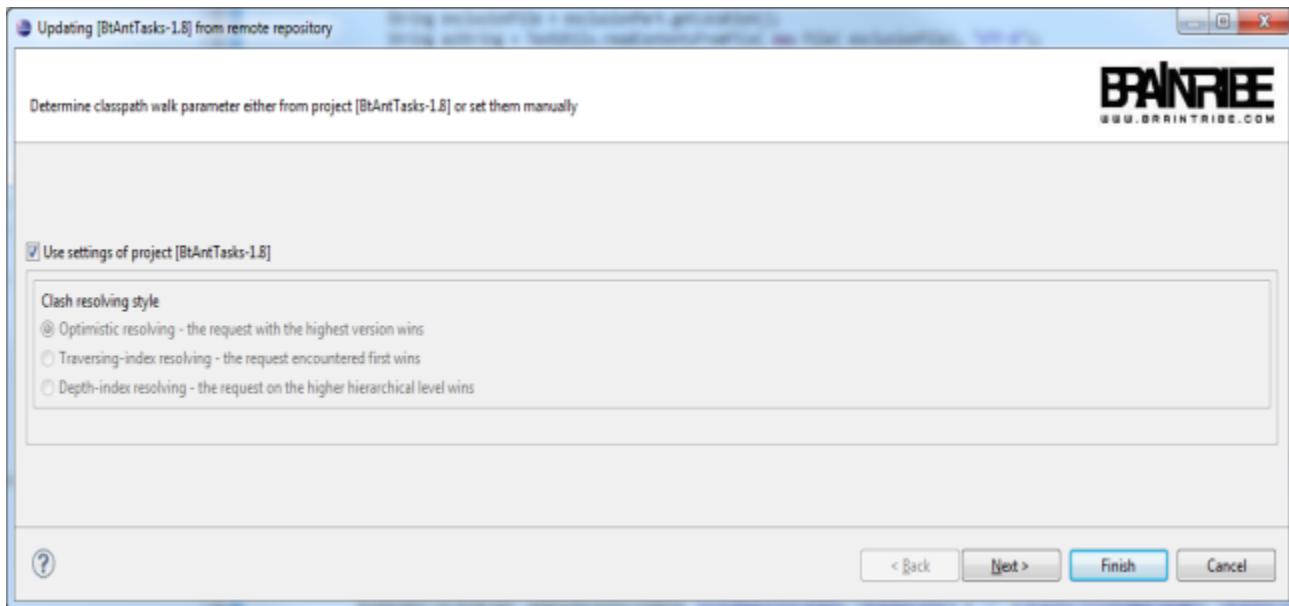
-  View dependency analysis

The commands & triggers that are shown here with an icon are also available in the toolbar and top menu.
Note that only the svn command wizard and the transitive ant wizard can run with or without a project selected in the package explorer. All others require a preselected artifact.

Dependency wizard

New in AC 2.0.5 is that the formerly independent dependency and synchronizer wizards are now merged into one wizard.
You control the different behavior by selecting the repositories you want to work with; either just have a look at the dependencies or synchronize them, even import missing artifacts.

Parameter



Project settings

You can specify to use the settings that are stored in the container - as shown in the disabled part of the dialog - or you can specify your own parameters. In the newer AC versions, all you can select is the type of clash resolving desired.

Once you've done that, press next.

Dependency View

Updating [MaelstromModelProcessing-1.0] from remote repository

Shows dependency information of the project [MaelstromModelProcessing-1.0]

BRAINTRIBE
WWW.BRAINTRIBE.COM

Dependencies Unresolved dependencies Clashing dependencies Undetermined dependencies Solutions Ambiguous solutions Synchronized solutions

Artifact	Assigned Version	Requested Version	Scope
com.braintribe.model.maelstrom.processing:MaelstromModelProcessing	1.0		
com.braintribe.model:QueryModel	1.0		
com.braintribe:ProviderApi	1.0-SNAPSHOT		compile
com.braintribe:FilterApi	1.1-SNAPSHOT		optional
com.braintribe.model:GenericModel	1.1-SNAPSHOT		compile
com.braintribe.model:GenericModel	1.1-SNAPSHOT		
com.braintribe.model.maelstrom:MaelstromModel	1.0		
com.braintribe.model.processing:LightweightStateProcessor	1.0		
com.braintribe.codecs:PainCodec	1.0		
com.braintribe.xml:Validator	1.0		
org.apache.velocity:velocity	1.5		
javax.mail:mail	1.4		provided
org.springframework:spring-context	3.0.5.RELEASE		provided
com.braintribe.model.maelstrom.processing:MaelstromCsProcessor	1.0		
org.springframework:spring-context	3.0.5.RELEASE		
org.springframework.inject-tck	1.0.0-PFD-3-jboss-1		test
org.springframework:spring-aop	3.0.5.RELEASE		compile
org.springframework:spring-beans	3.0.5.RELEASE		compile
org.springframework:spring-core	3.0.5.RELEASE		compile
org.springframework:spring-expression	3.0.5.RELEASE		compile
org.springframework:spring-asrn	3.0.5.RELEASE		compile
javax.xml:jaxrpc-api	1.1		test
commons-pool:commons-pool	1.5.3		test
commons-dbc:commons-dbc	1.2.2		test
org.easymock:easymock	3.0		test
org.slf4j:slf4j-log4j12	UNDEFINED		test
junit:junit	4.10	UNDEFINED	test
log4j:log4j	1.2.12	UNDEFINED	provided
com.braintribe.model:PainModel	1.0		
com.braintribe.utils:Logging	1.0-SNAPSHOT		provided
com.braintribe:ProviderApi	1.0-SNAPSHOT		compile
com.braintribe:CodecApi	1.1-SNAPSHOT		compile
com.braintribe.codecs:PainCodec	1.0		
com.braintribe.model:PainModel	1.0		
com.braintribe.model.processing:GenericModelSession	1.0		
com.braintribe.xml.parser:DomParser	1.0-SNAPSHOT		

Import selected projects to workspace

?

< Back Next > Finish Cancel

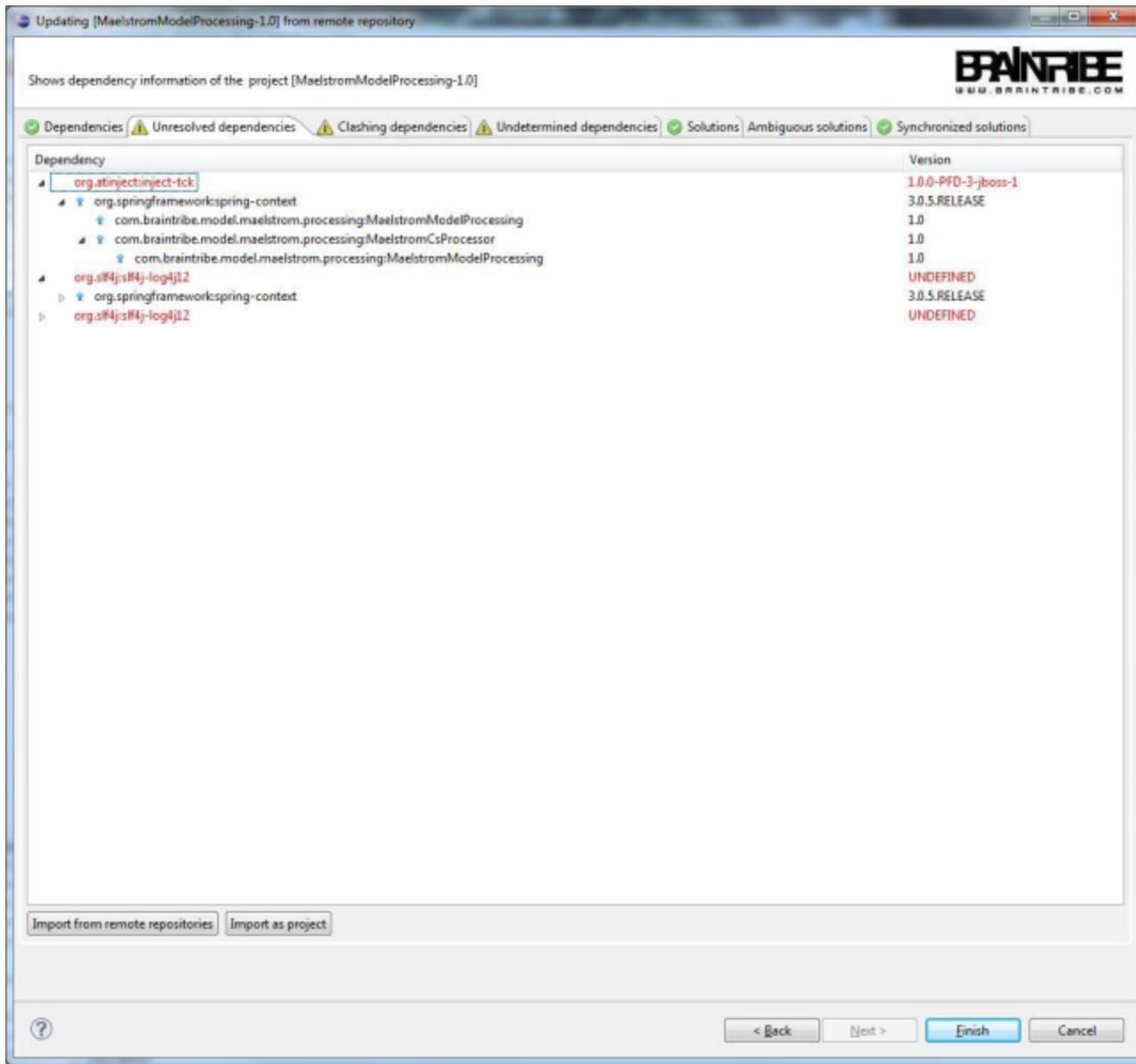
Items shown with a round plus sign exist in the local svn repository, and can be imported into the current workspace, items shown in *italics* are either not present in the svn or are already loaded in the workspace. You don't have to leave the dialog, the import process happens in another thread. So even if you can select multiple projects to import, you can as well import one after the other.

Items shown in red are dependencies that couldn't have been resolved. See the specialized view for these files.

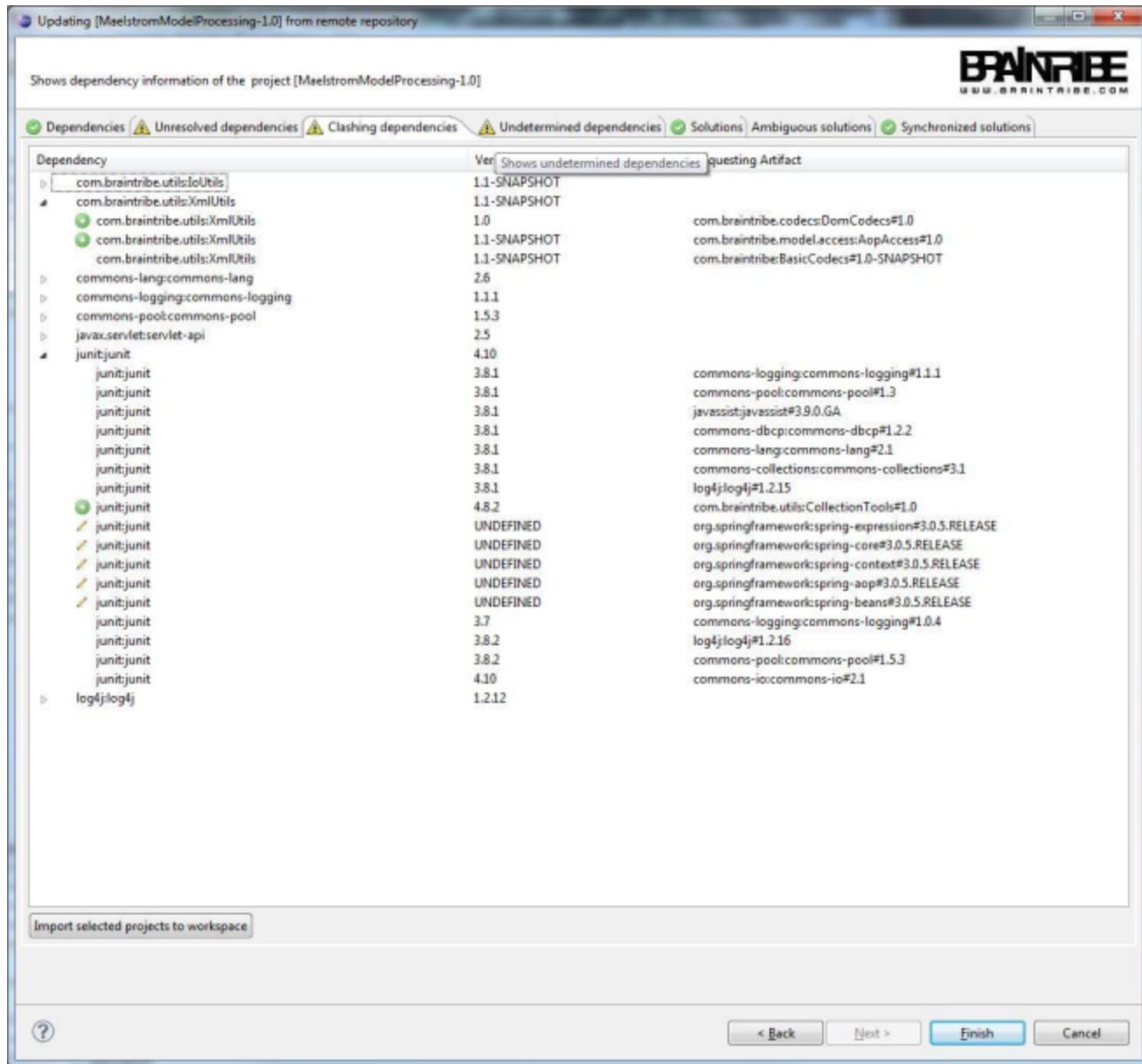
Unresolved dependencies

In some cases, dependencies cannot be found, i.e. no access exists to their pom-file.

You can try to get the dependency via Greyface by selecting the button "Import from remote repositories" or - if it's a project - try to import it as a project, either from the working-copy or the remote svn repository.



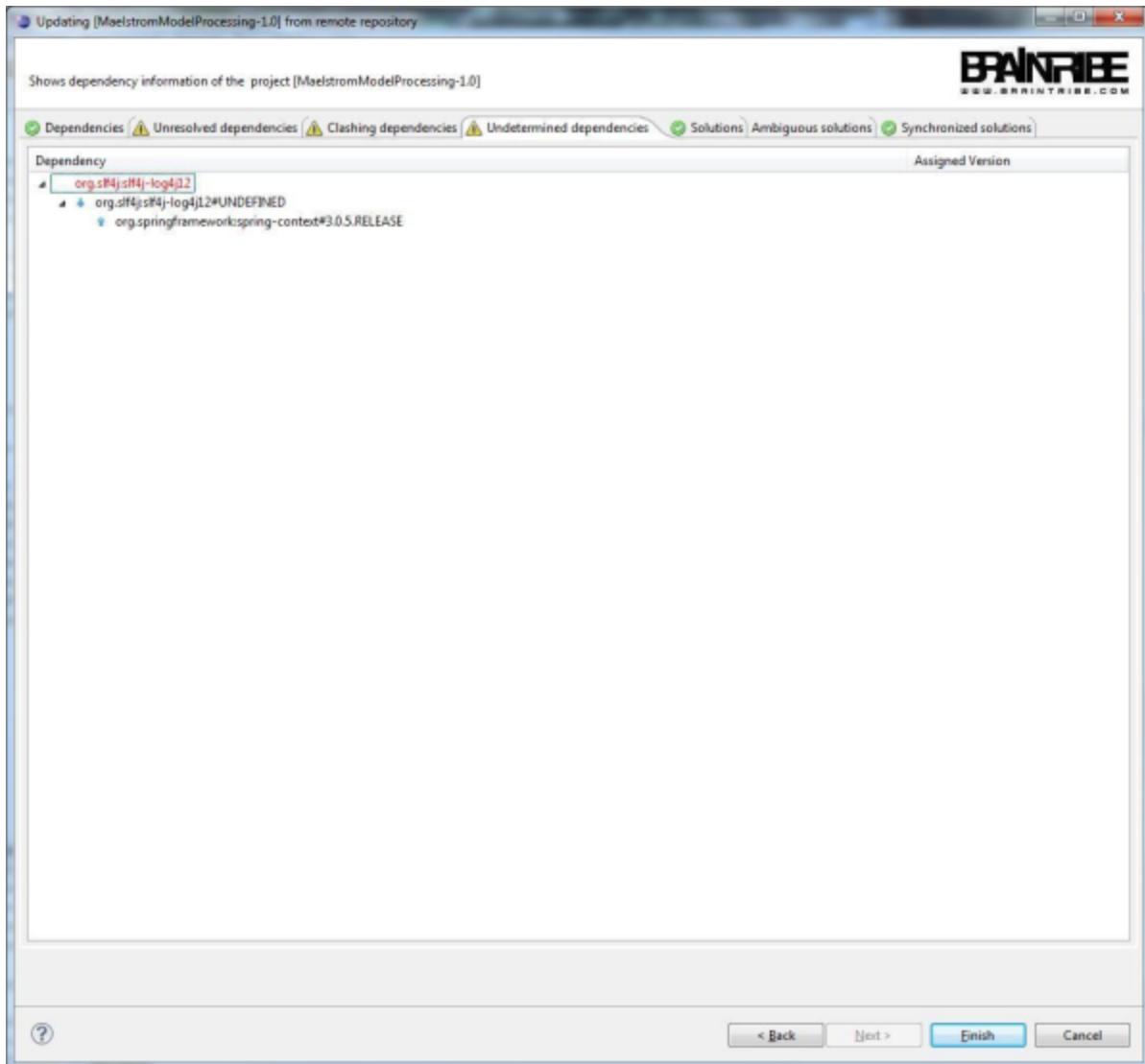
Dependency clashes



In this view, the clashing dependencies are shown (not to be confused with clashing solutions, see next section). Top entry is the dependency that was chosen by the current clash resolver, followed by the version and the artifact that was requesting this dependency (in other words, the owner of the pom file that contained the dependency). Attached to this "winning" dependency are the found clashing dependencies.

Undetermined dependencies

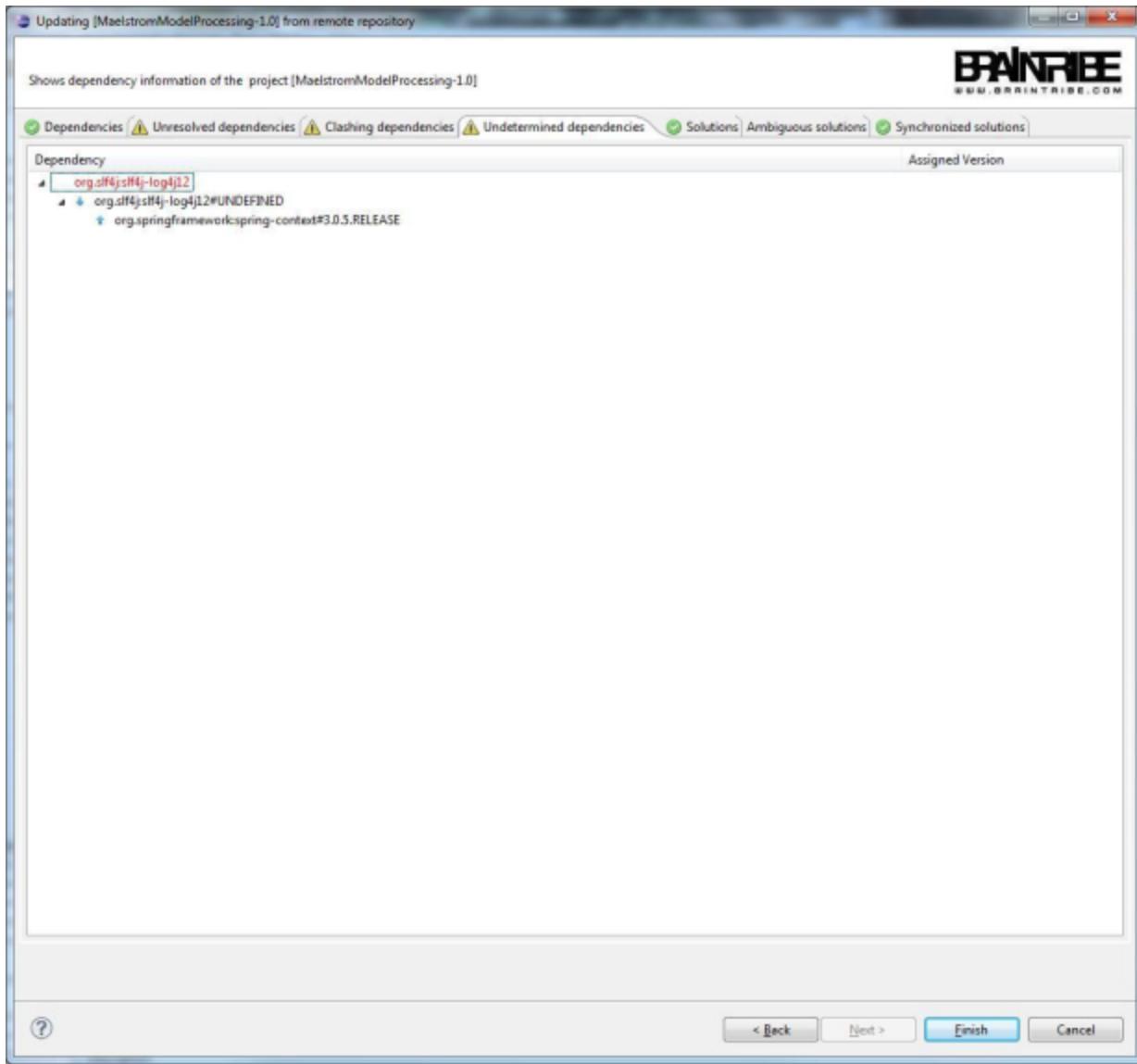
In some artifacts - third party artifacts - sometimes dependency declarations lack a version all together, i.e. the version tag's missing. Malaclypse calls such dependencies "undetermined dependencies" and resolves them after the run. It will see what solution was chosen via another, fully specified dependency and complete the dependency accordingly. This view shows which solution had the undetermined dependency declared and which dependency was actually used to build the class-path.



Unresolved dependencies are marked in red. Below are the requesting artifacts marked, up to the terminal artifact.
 If there are un resolved dependencies, you can try to look them up in two different ways: You can use Greyface to import the missing third party artifact or you can import an artifact as a project from either the local working copy or remote repository.
 If you select any of the top nodes of the list and press "look up dependency", Greyface's import wizard is started with the dependencies' data preset.
 If you select the button "look up project", you'll get present with a tabbed dialog showing the projects in your working copy, but also in the svn. Selecting the svn will either update or check-out the project, depending on whether it already exists in your local working copy. In any case, the selected project's loaded into you workspace. You might need to update AC's container and Eclipse's class path after the import.

Unresolved dependencies

Even if malaclypse does its best to resolve all dependencies, it sometimes runs into problems.. Especially in those cases where no version is specified . Such undefined dependencies are shown in this view.



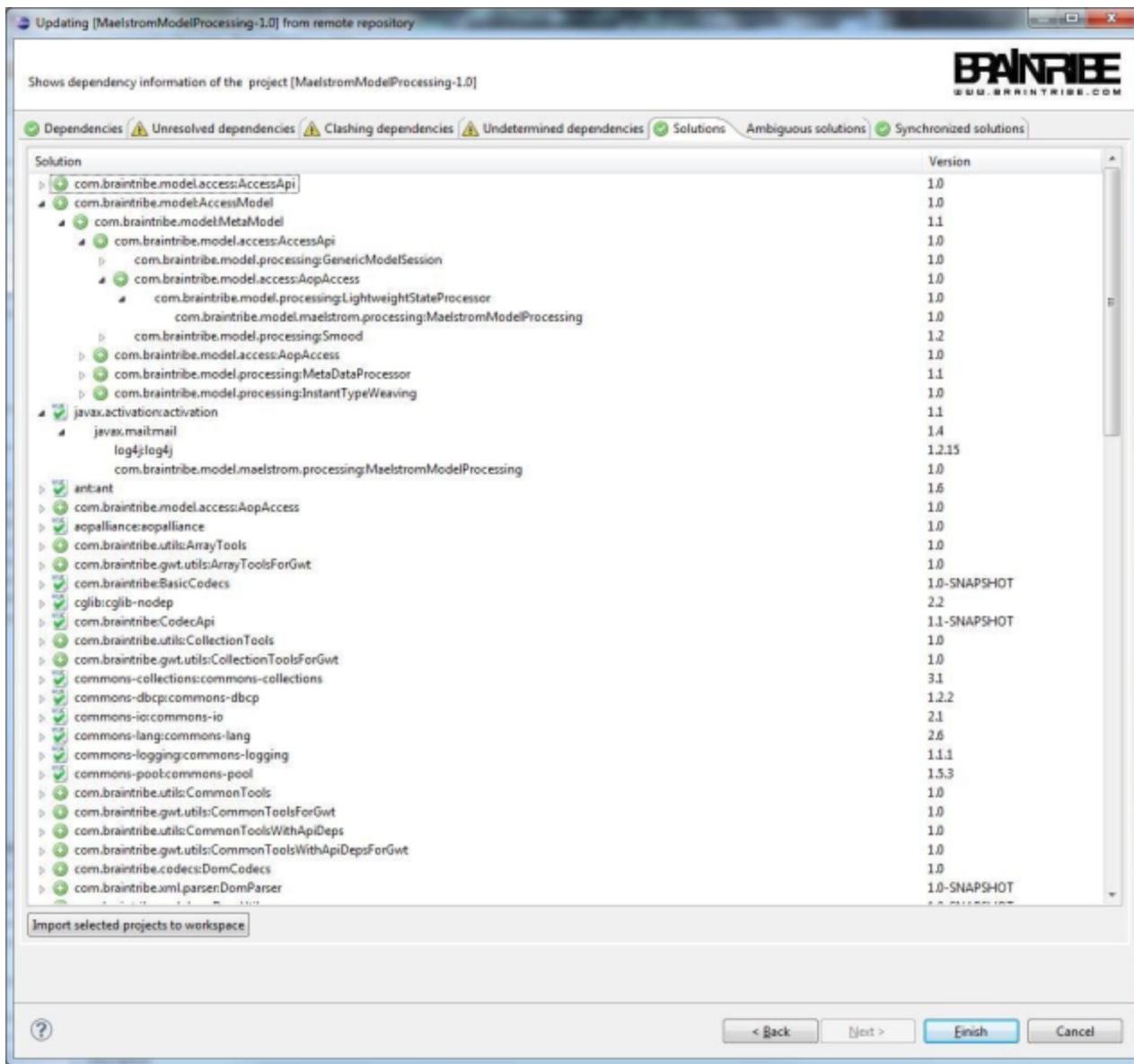
If malaclypse was able to determine a version - that means at least one other artifact explicitly requests a certain version, then the version's shown. If not, then the version's field remains empty.

New is that you can now import a dependency in two ways:

- if it's one of our artifacts, you can import it as a project, either from your local working copy or by retrieving it from our svn repository.
- if it's an external dependency, then it's not present in our repository. You can automatically call Greyface to import it (but do me a favour and think about whether it's worth it to import that; we might have a version in our repo that is sufficient for your needs).

Solution view

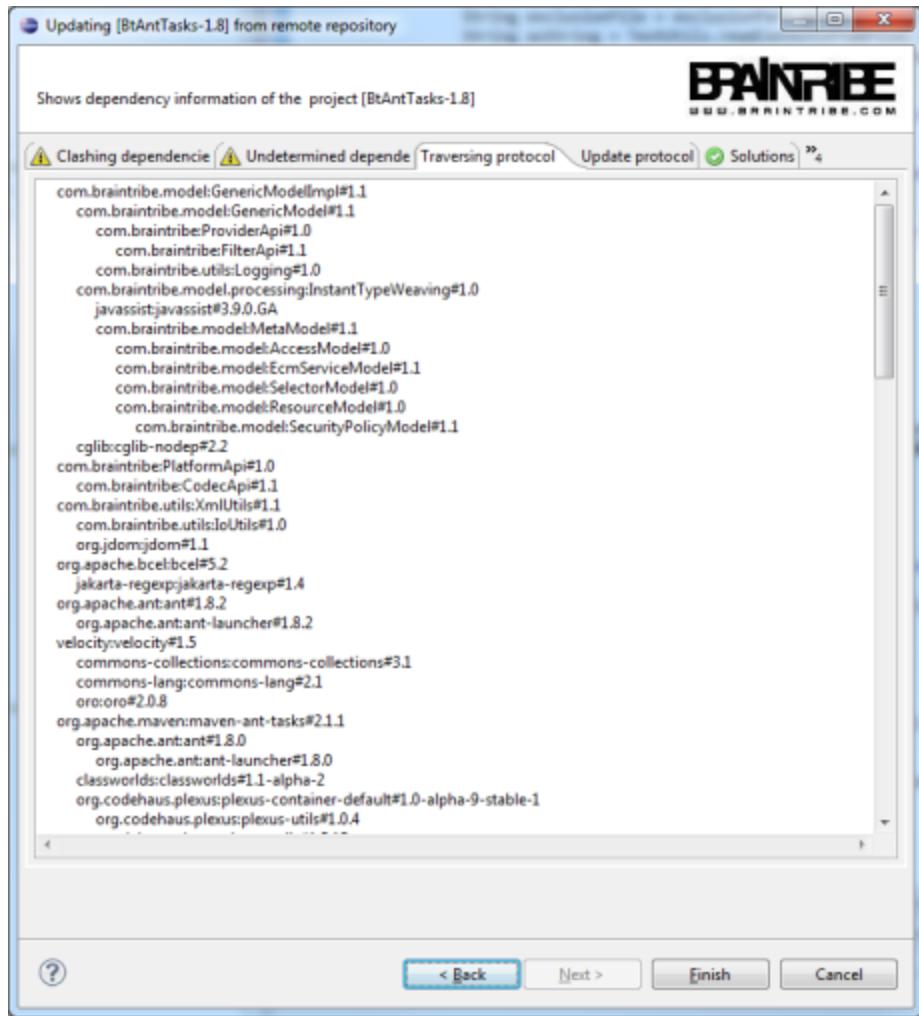
For each solution referenced in the pom, the reverse dependency list is shown, d.h. below the solution, the requesting artifact is shown, up to the terminal artifact.



Again, any solution marked in bold is backed by a project that can be loaded by selecting the button below the tree.

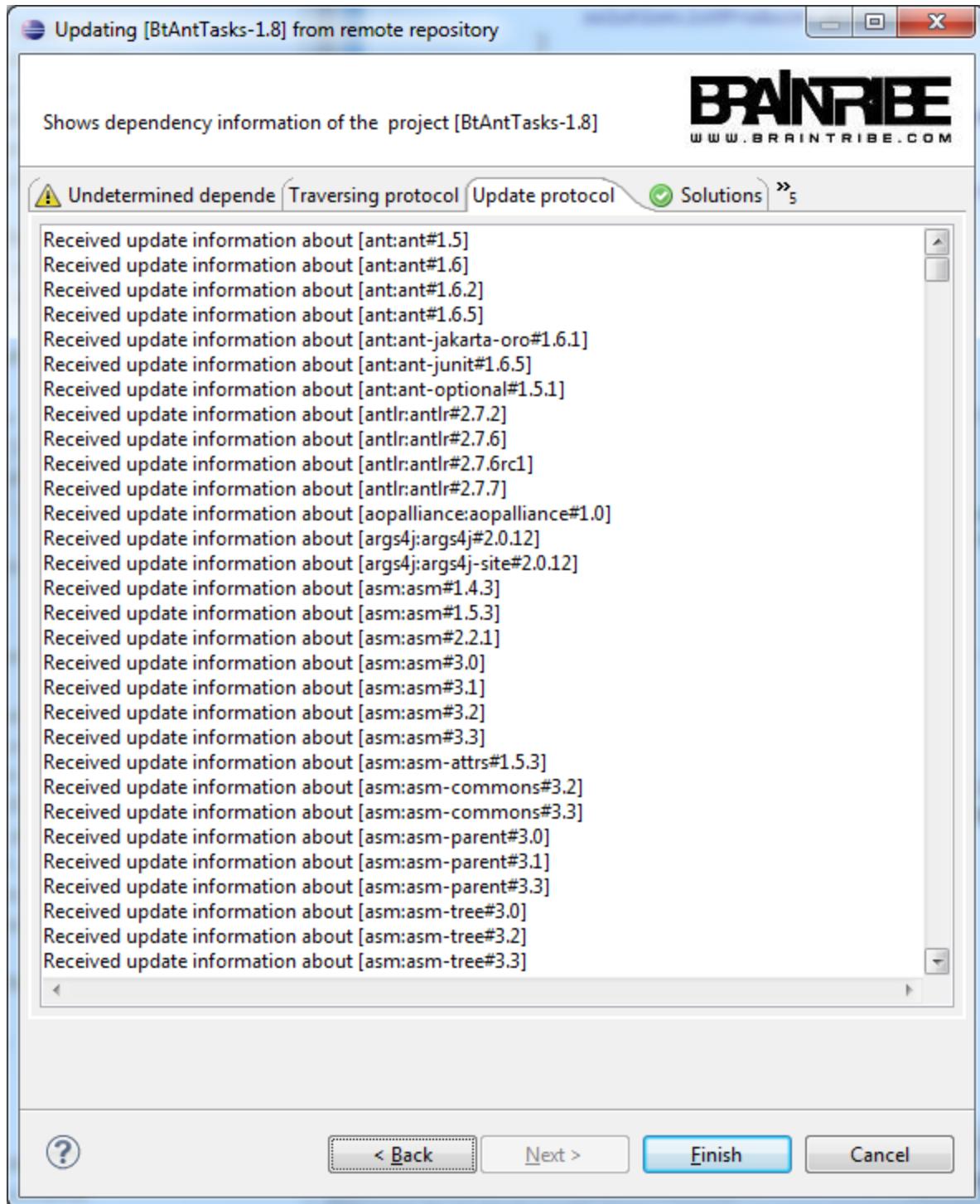
Traversing protocol

The traversing protocol viewer shows a trace of how the poms were traversed while building the solution list.



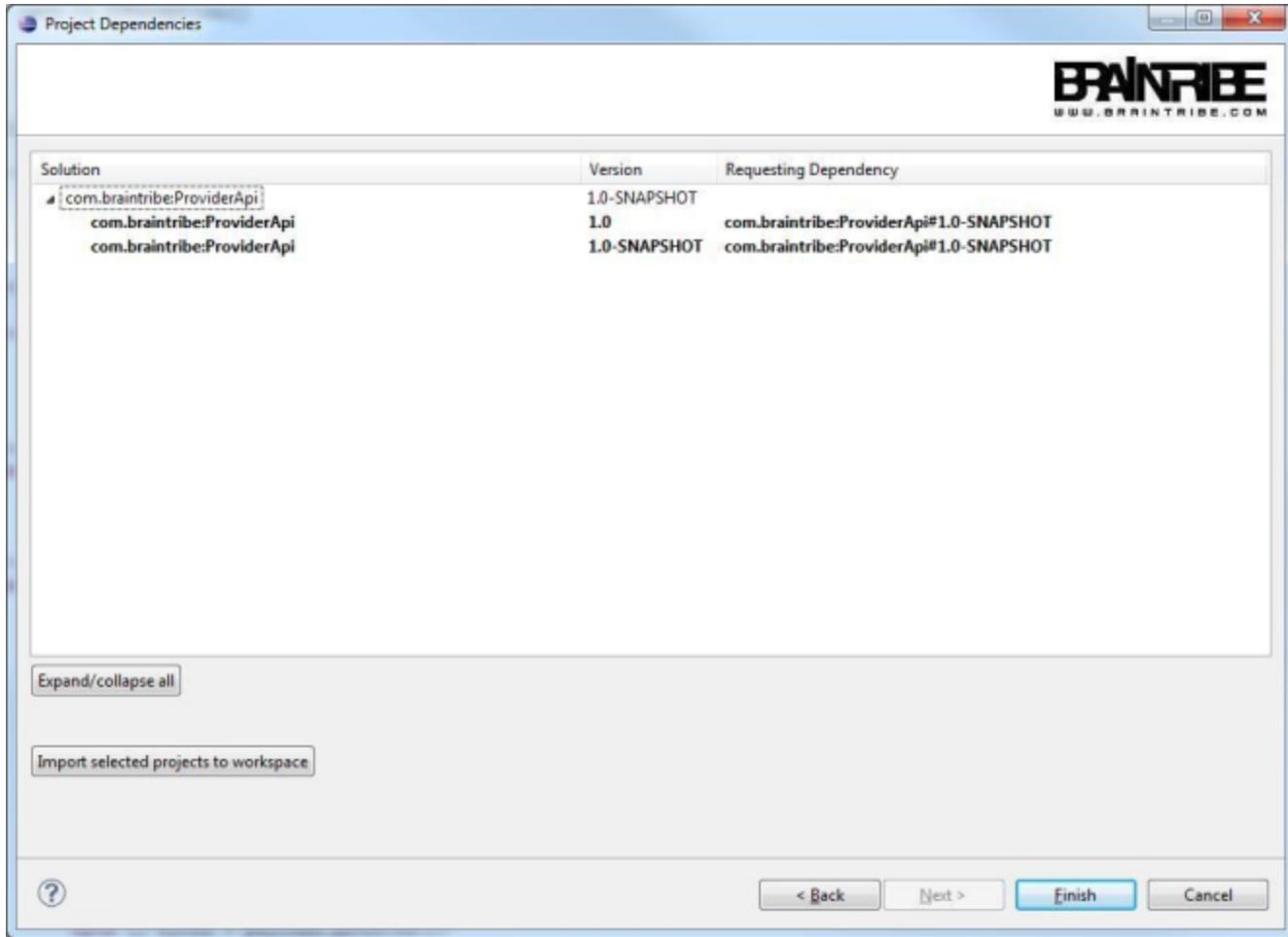
Update protocol

As from 2.4.0 on, AC uses MC's connection to Ravenhurst to find any changes in the remote repository. The update protocol shows the information retrieved from Ravenhurst.



If - as in the pictured run - no prior info was stored, all artifacts are returned as changed. Your local data is never overwritten, so if you need to have this done, you must delete your local copy beforehand (inclusively all .updated files)

Ambiguous solutions



This section is only populated, if version ranges com into play. For example, a range could say
(1.0, 1.2) (*Bloody google docs doesn't do square brackets*)

or simply

1.0^

In such a case several solutions will match the version range. But as we need to boil everything down to exactly one solution, we need a clash resolving mechanism, just analog the dependency clash resolving mechanism.

Note: 1.0 is always deemed a more recent version than 1.0-SNAPSHOT!

Note again: In all cases of clash resolvers, it is always the optimistic clash resolver that does that resolving on solutions. On other words: you can only influence the clash resolving on dependencies, not on solutions.

Synchronizing view

As of 2.4.0, the semantics of this view has been changed, as synchronization is done automatically whenever an update request on the dependencies of an artifact is issued.

Updating [BtAntTasks-1.8] from remote repository

Shows dependency information of the project [BtAntTasks-1.8]

BRAINTRIBE
www.BRAINTRIBE.COM

Clashing dependencies Undetermined dependencies Traversing protocol Update protocol Solutions Ambiguous solutions Synchronized solution »2

Artifact	Version	Type	Comment
gnu.getopt:java-getopt	1.0.13		
velocity:velocity	1.5		
org.apache.maven:maven-settings	2.2.1		
com.braintribes.web.velocity:VelocityTemplateRenderer	1.0		
com.braintribes.model.io:GenericModelSourceWriter	1.1		
com.braintribes.model.processing:InstantTypeWeaving	2.0		
org.springframework:spring-expression	3.0.5.RELEASE		
com.braintribes.spring:SpringSupport	1.0		
com.braintribes.spring:SpringSupport		:pom	
com.braintribes.spring:SpringSupport		sources:jar	
com.braintribes.spring:SpringSupport		:jar	
plexus:plexus-utils	1.0.2		
plexus:plexus-utils		:pom	
org.apache.maven:maven-ant-tasks	2.1.1		
org.apache.maven:maven-ant-tasks		:javadoc	
org.apache.maven:maven-ant-tasks		sources:jar	
org.apache.maven:maven-ant-tasks		:pom	
org.apache.maven:maven-ant-tasks		:jar	
com.braintribes.config:locAnnotations	1.0		
org.springframework:spring-context	3.0.5.RELEASE		
org.apache.ant:ant	1.8.2		
com.braintribes.ProviderApi	1.0		
oro:oro	2.0.8		
org.apache.maven:maven-error-diagnostics	2.2.1		
org.apache.maven:maven-artifact-manager	2.2.1		
commons-httpclient:commons-httpclient	3.1		

Import selected projects to workspace

?

< Back Next > Finish Cancel

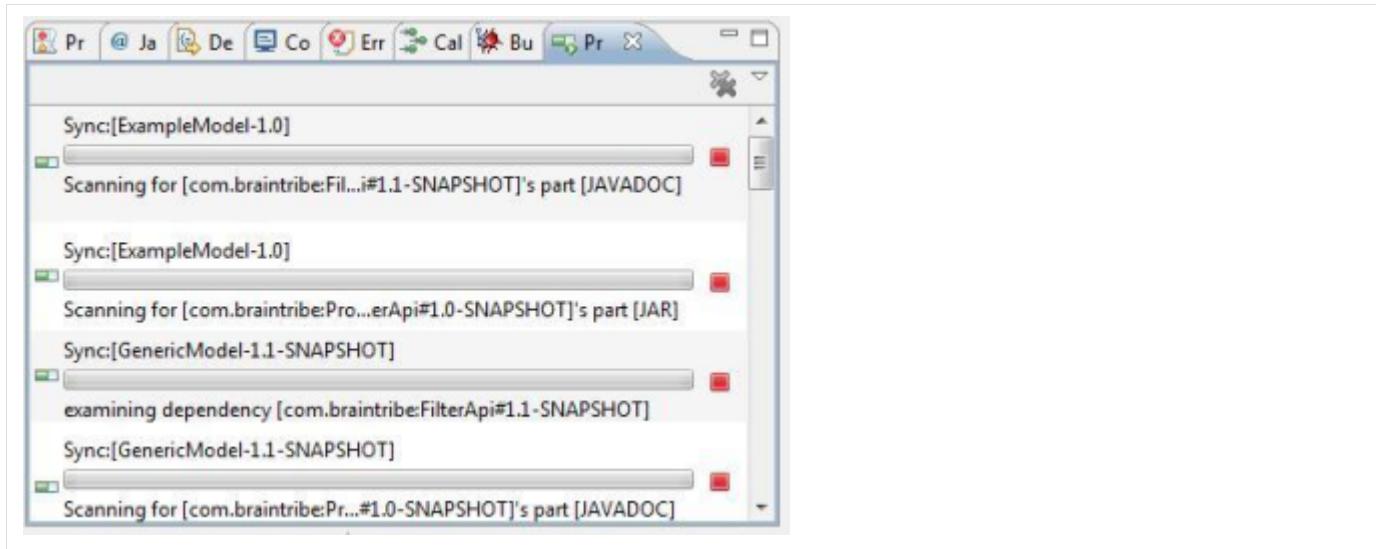
What the view shows is all parts of the solutions that make up the classpath.

An entry for a solution can have the following states:

- Green circle with a plus sign: This solution is backed a project in your local working copy and can be loaded into your workspace.
- Yellow triangle with a question mark: This solution has a problem - mostly it just shows that the jar is missing.
- Green circle with a checkmark: Default for an external solution - the pom is valid.
- Red circle with a checkmark: The pom of this solution could not be read as it's invalid.

Auto synchronizing

The workspace-version of the command will **fork** walks on any artifact projects loaded in the workspace. You won't see it happen, but it is enough to click once. If you want to see the progress (and that also applies to any background process run by AC) open the progress view.



Results - if any - you'll find in the error log, but not as errors obviously.

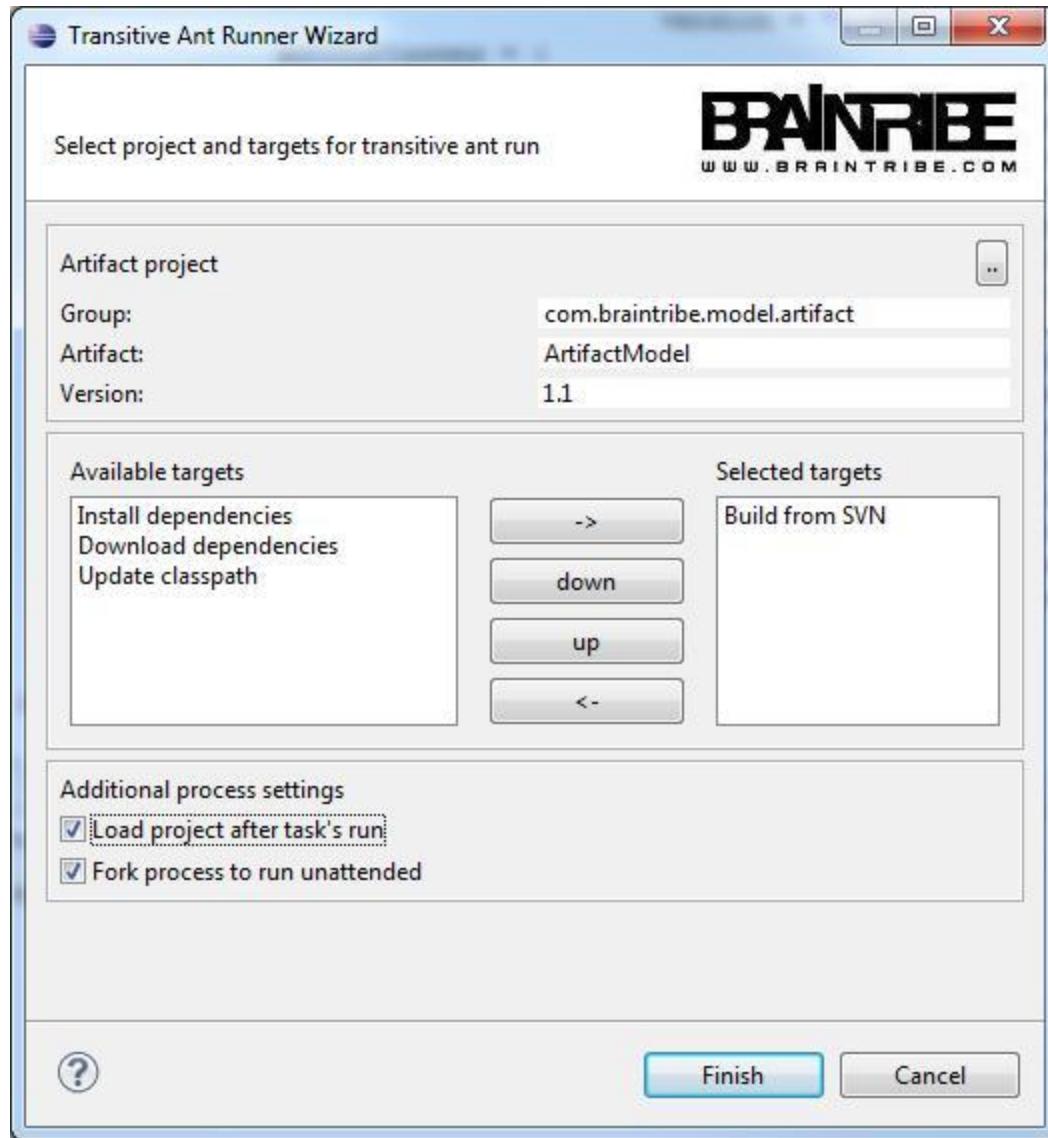
As said above, it will fork the synchronize processes - so you might only see a flashing dialog box. Check the progress view if you want to make sure the synchronizing process runs.

Transitive builds

Currently not active in AC 2.4.0, probably will return later in Panther's roll out.

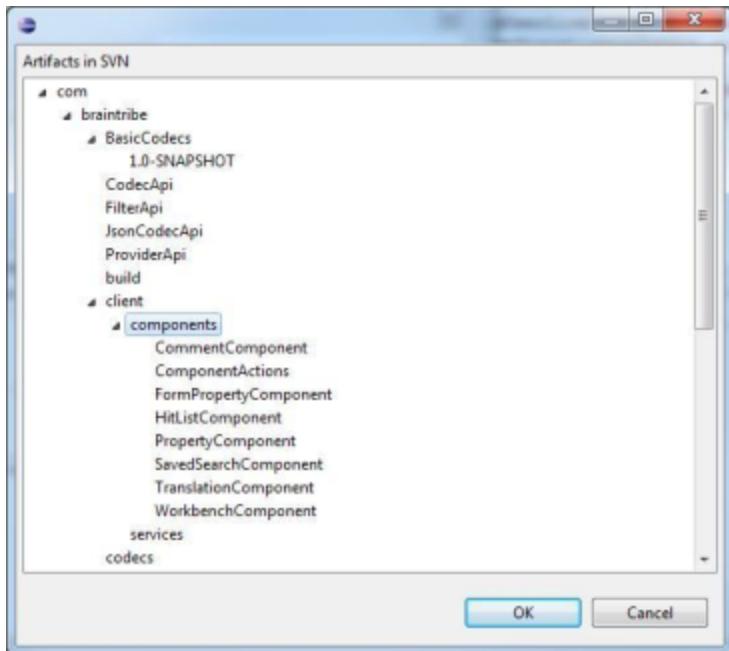
Select one or several builds from the left side, order them on the right side and then hit return.. If the project you specify is NOT in the workspace, you can have it loaded after the target's been run.

If a project's selected in the Package Explorer (and it is an artifact obviously) then the appropriate fields in the dialog are automatically filled. Otherwise, you must specify the data yourself.



You can add any number of transitive builds via the appropriate preferences section of AC.

You can specify whether you want the project loaded after the ant tasks (depending of course whether that's possible) and you can also specify that you want to have the task run as a parallel job. You'll still get the output in a console, but no dialog's shown.
The browse button top left opens the repository browser.



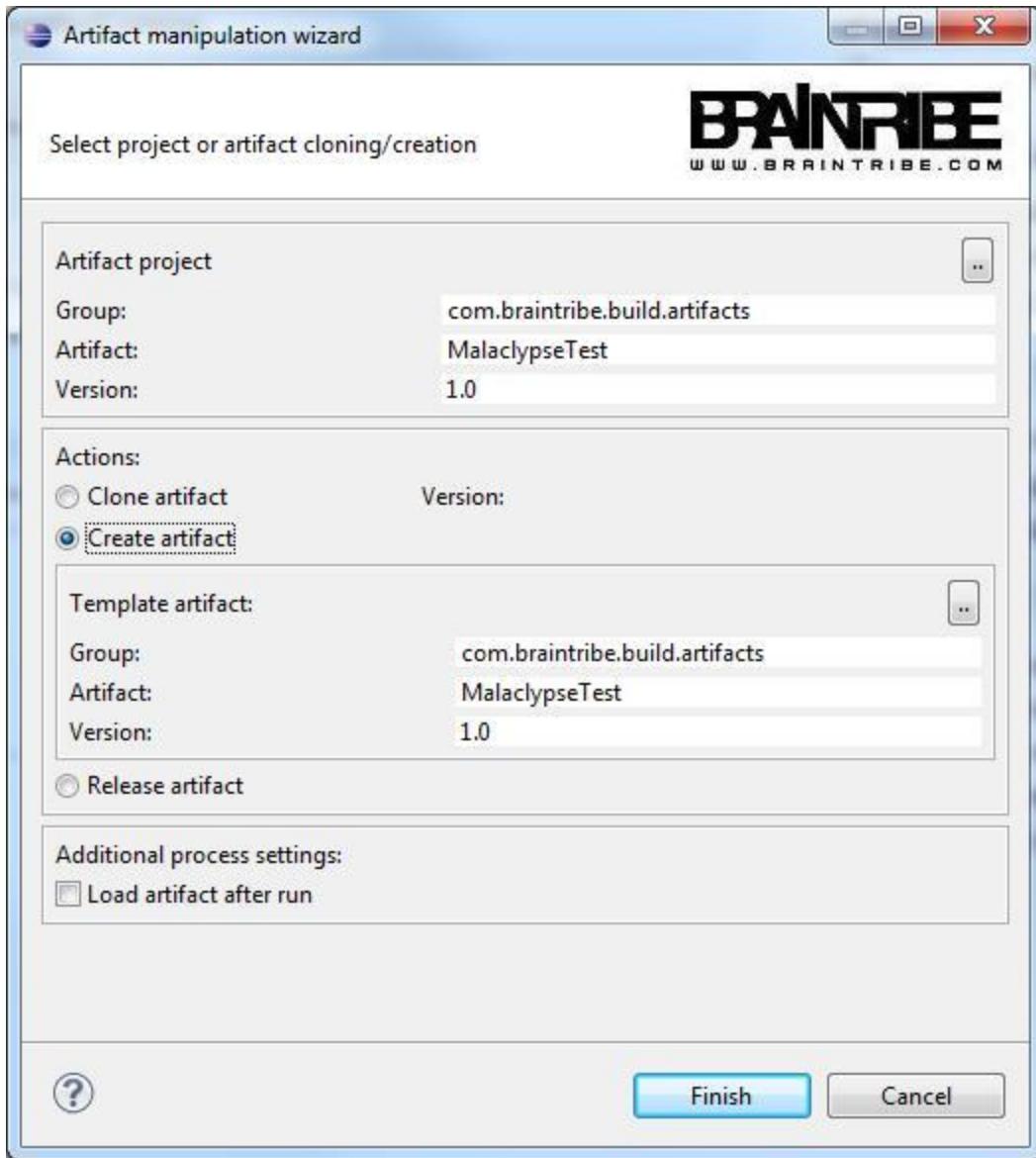
You can navigate the remove SVN repository just like you do in the Tortoise explorer. If you point to a directory which includes a pom.xml, then this pom is used to determine the artifact you want.

Artifact related svn tasks

Quite similarly to the feature above you can use Malaclypse svn integration to manipulate artifacts.

Again, the following applies:

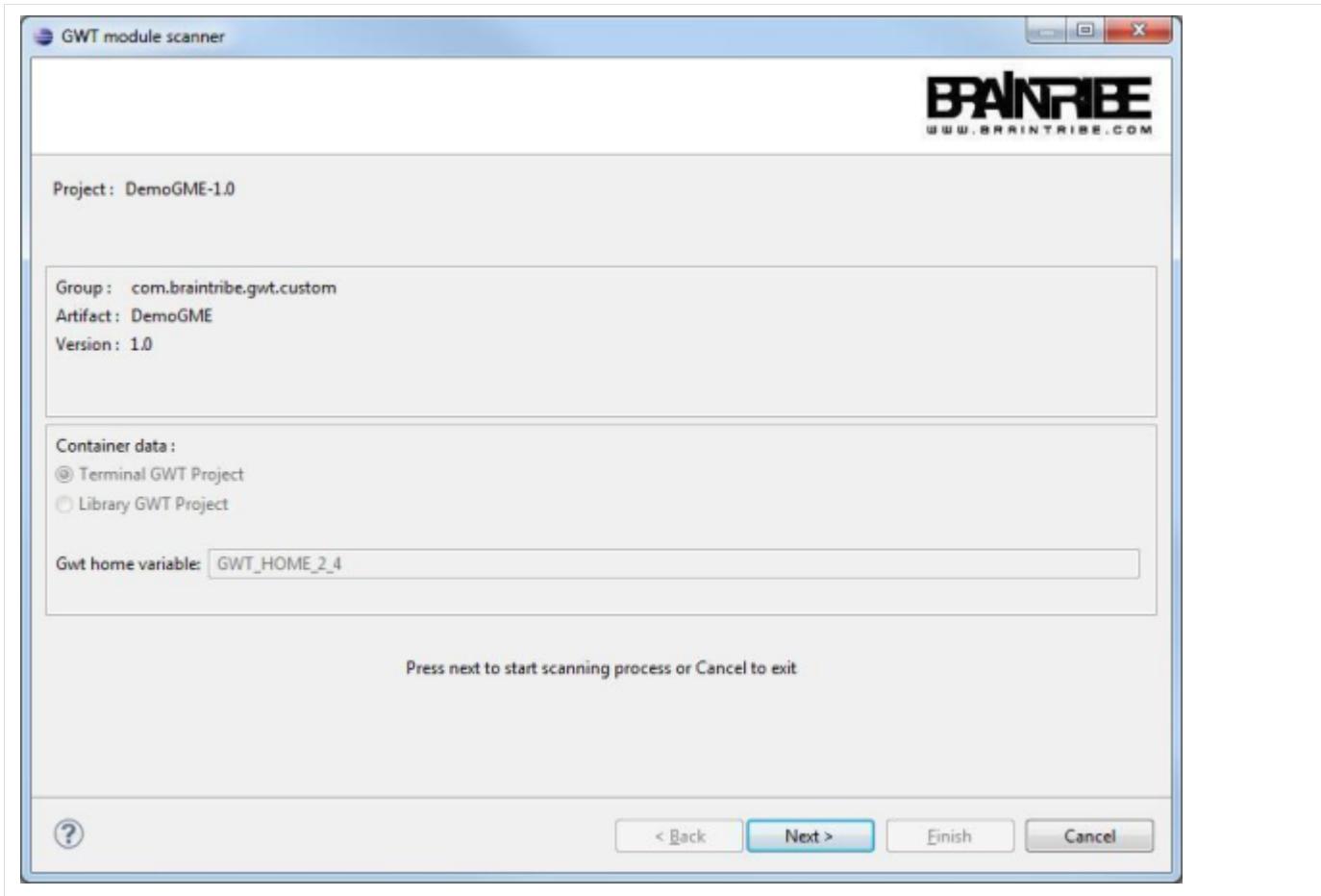
- If the project resulting from you specify is NOT in the workspace, you can have it loaded after the target's been run.
- If a project's selected in the Package Explorer (and it is an artifact obviously) then the appropriate fields in the dialog are automatically filled. Otherwise, you must specify the data yourself.



Again, the browse button will open the SVN parser dialog, just as in the case explained above.
 You have two places where you can select an artifact. If you are creating one, you can select the base artifact, i.e. the artifact whose pom.xml, build.xml, .project and .svnignore are to be taken.

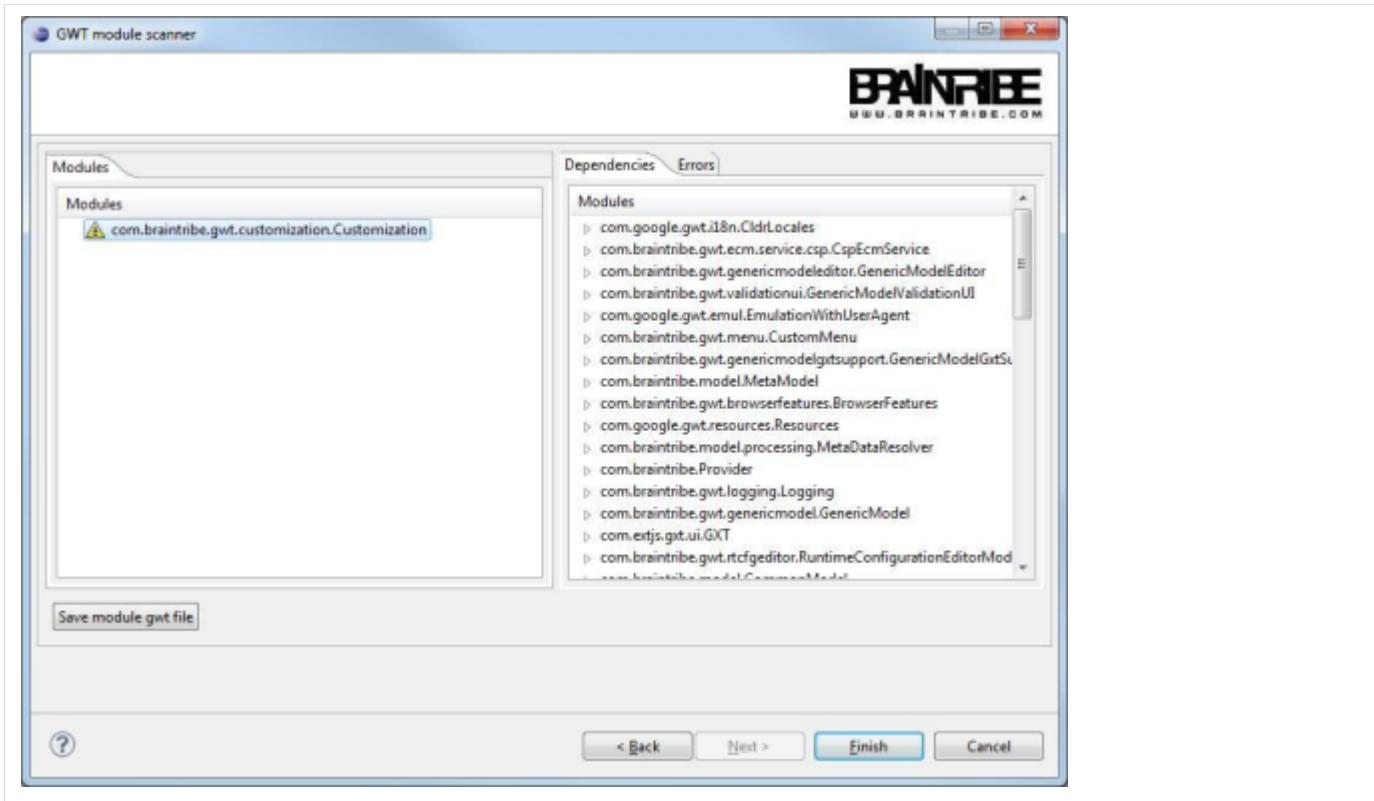
GWT features

Version 2.1.0 introduced new GWT specific features: If you select a project, you can now start the GWT checker wizard. The following screen appears, showing information about the select project:

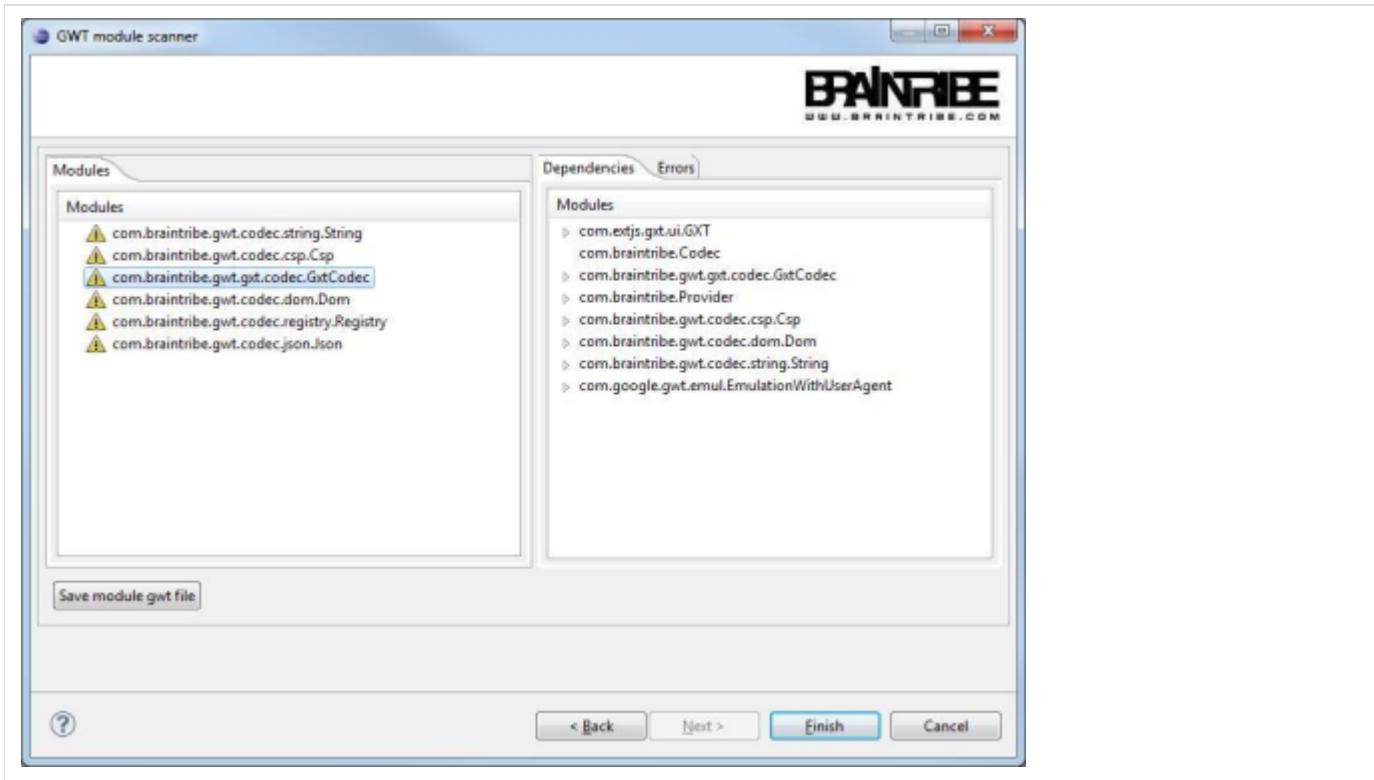


Basically, the page has no other purpose than to give the visual feedback via a progress monitor. Press next and scanning process starts. The following page is divided into two parts.

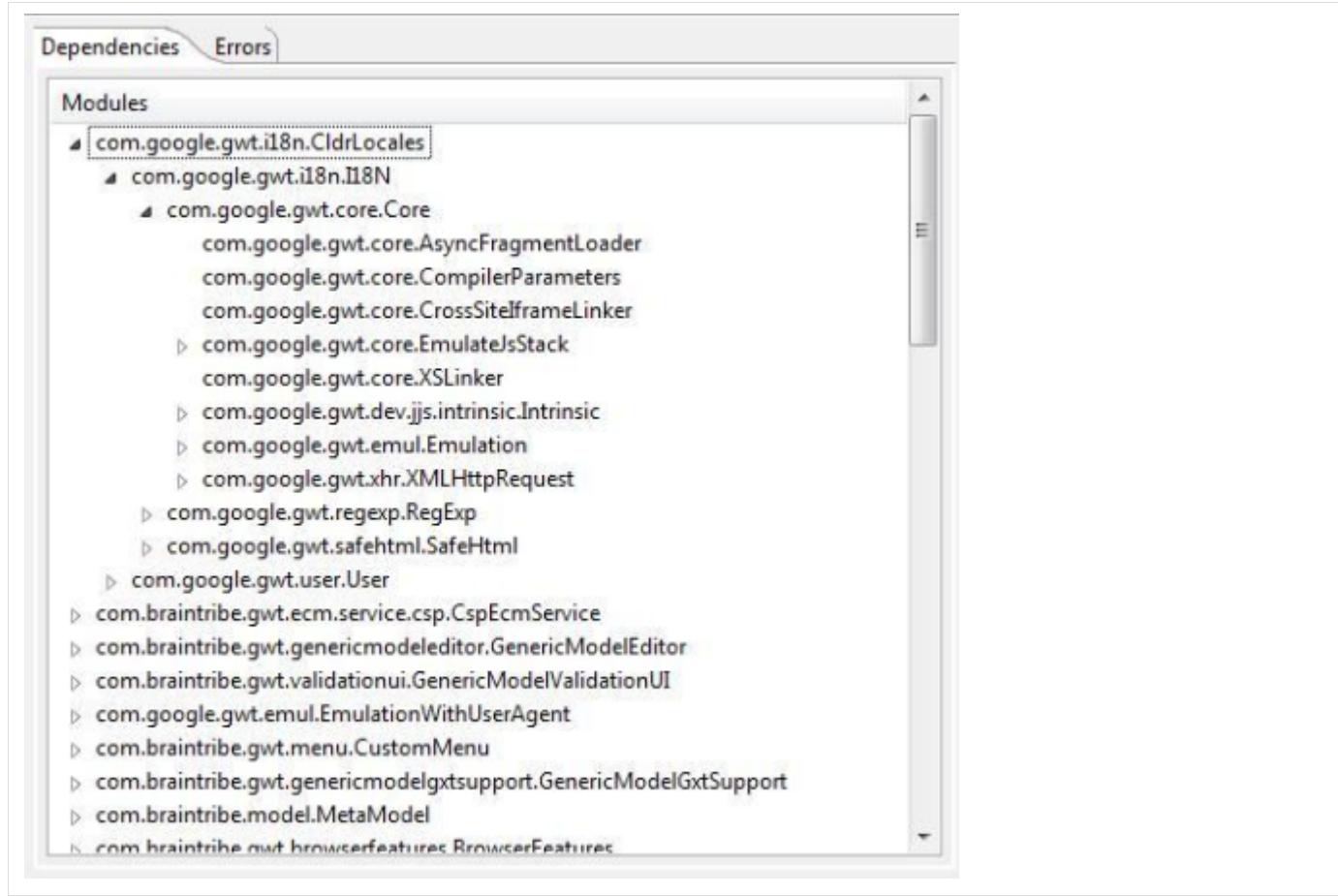
On the left side, you see the master panel, where you see the modules that are defined in your project - at least one should be shown. The example below has only one module.



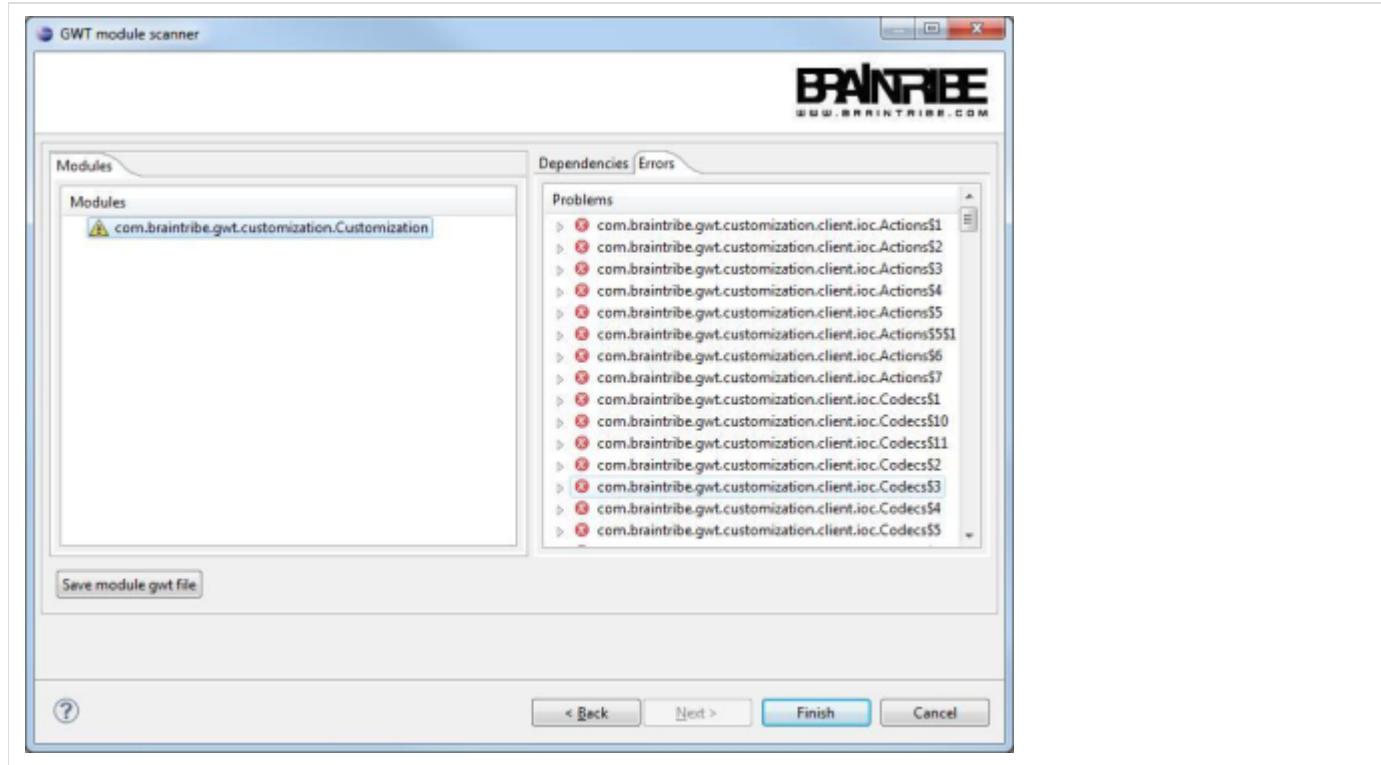
This example shows the master panel of a project with several modules.



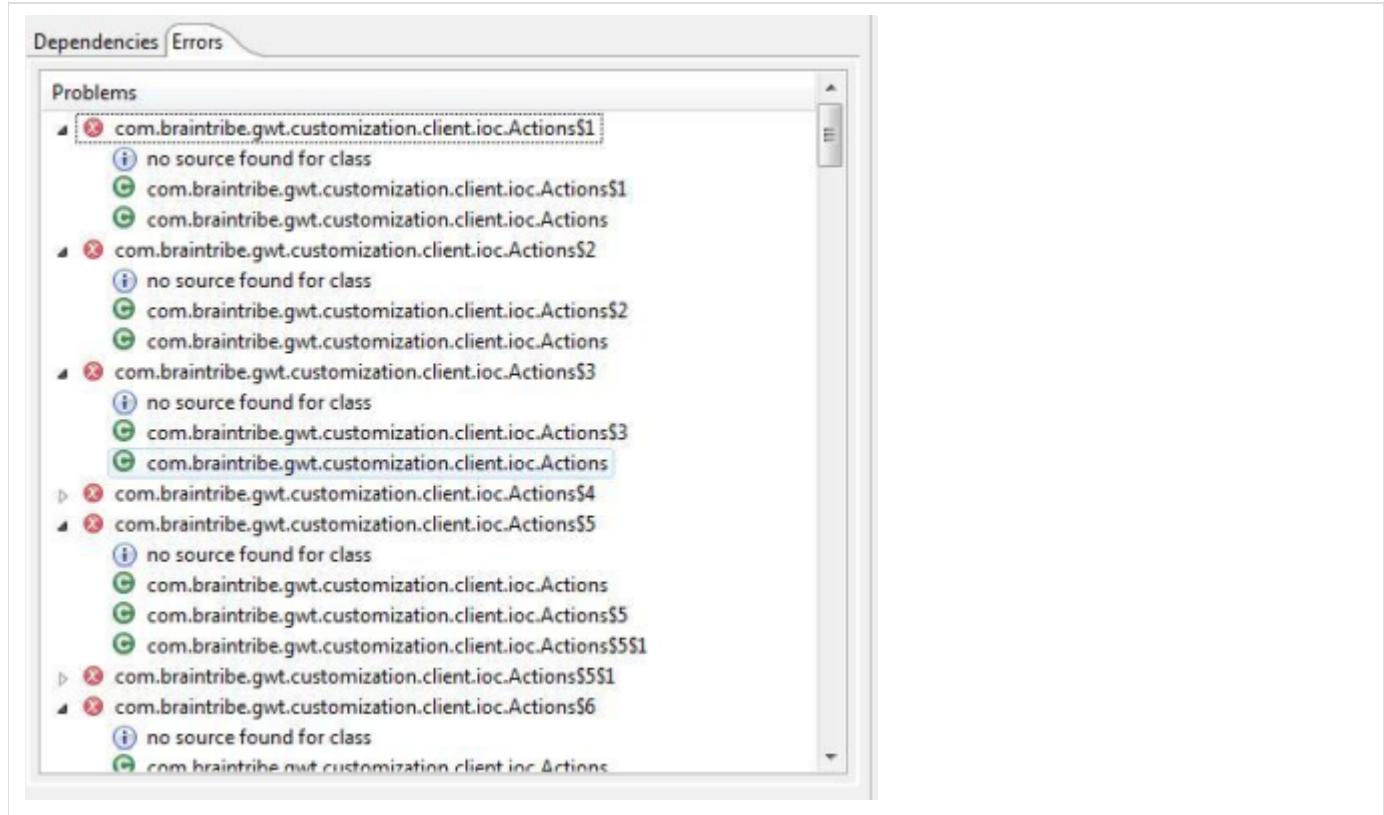
The master panel on the left and the detail panel on the right are linked by selecting the module in the master panel. The selection triggers the further processing (which may take some time again) and eventually shows the detailed data in the two detail panels. The dependencies detail panel shows the dependencies of the selected module and their dependencies in turn.



If a project in the master panel shows the warning sign left to its name, then there are entries in the error panel. Basically, it shows a list of all errors in the modules GWT.



Selecting an error, i.e. expanding it, shows further information about it.



Following is shown for each error:

- The error reason (offending class)
- The error message
- All classes that reference the offending class

Saving a gwt.xml file

If you press this button, the gwt.xml file is written to disk. A backup file is created, just in case.

TFG-BTAntTasks

 Work in Progress

Table of Contents

▼ Expand / collapse

- Introduction
- Pre-Requisites, Download and Installation
- Targets
- Eclipse
- TODO

Introduction

The BtAntTasks provide a collection of tools that are used for handling our artifacts.



Please also refer to this article for getting an overview about our dependency management:

[Dependency Management - Controlling the Uncontrollable](#)

Pre-Requisites, Download and Installation

An installation of Apache Ant is required:

<http://ant.apache.org/>

Download the BtAntTasks via following link:

[BtAntTasks-1.8.1-libs.zip](#)

Unpack the zip file and copy the contained .jar-files into the lib-folder of your Ant-installation.



This is described in more details in this article:

[Quick Start Guide for Java/GWT developers](#)

Targets

Target	Description	Globally / locally available	Example
install	Compiles the artifact from the working copy (your eclipse project) and puts it into the local repository as an artifact.	global (indirectly)	
install-all			
dist	Builds a distribution of the artifact.	local	
deploy	same as install, but also uploads the artifacts into the remote repository (archiva).	local	
assemble	same as install, but also creates a "deployable installation" e.g., BTAntTask: a zip file; Servlets: a war file. Note that not every model has the assemble task.	local	
download-deps			

install-deps	Installs all dependencies as artifacts into the local repository.	global	
update-wc	synchronises all dependencies of the artifact with svn	global	
build	depends: install-deps, assemble	global	
buildFromSvn	depends: update-wc, build	global	
prepareDeveloping	depends: update-wc, install-deps, download-deps	global	
antZargoToArtifact deprecated	reads an ARGO file and creates an artifact from it. If the artifact existst, the src directory is backed-up before a new src directory is written. There is no synchronization, any changes done in the artifact and not reflected in the ARGO file is lost (or rather is moved to the back-up directory).		ant -f antZargoToArtifact -DgroupId="com.braintribe.mode l" -DartifactId="InheritanceTestMod el" -Dversion="1.0" -DworkingCopy="c:/works/svn/artifacts"
antArtifactToZargo deprecated	reads an artifact, extracts all relevant generic entities that make up the meta model and creates an ARGO file that contains the model.		ant -f antArtifactToZargo.xml -DpomFile=c:/works/svn/artifacts /com/braintribe/model/Inheritanc eTestModel/1.0/pom.xml -DbuildDirectory=c:/works/svn/ar tifacts/com/braintribe/model/Inhe ritanceTestModel/1.0/build -DexclusionDependency=com.br aintribe.build.platform:CSP#4.0. ^ -DgroupId=com.braintribe.model -DartifactId=InheritanceTestMod el -Dversion=1.0
artifactToZargo deprecated ?	Compound ant tasks - These tasks are used to do the following: <ul style="list-style-type: none">• determine the correct dependencies• extract a meta model• convert it to an XMI and wrap it into a ZARGO file.		
zargoToArtifact deprecated ?	Compound ant tasks - These tasks are used to do the following steps: <ul style="list-style-type: none">• convert an XMI wrapped in a ZARGO file to a meta model• generate an artifact from the meta model		
mmg	sources-to-model Reads the project and extracts the model. The model file is always named as the model, with an added extension "xml".	local	
m2z	model-to-zargo Reads the model file and creates the zargo. The zargo file is always named as the model, with an added extension "zargo".	local	

z2m	zargo-to-model Reads the zargo file and extracts the model file. The model file is always named as the model, with an added extension "xml".	local	
m2s	model-to-source Reads the model file and creates the sources in your working copy. Any existing sources are backed-up.	local	
s2z	sources-to-zargo Reads your project and automatically creates a zargo file from it. It combines sources-to-model and model-to-zargo.	local	
z2s	zargo-to-sources Reads your zargo and automatically creates the sources from it. It combines zargo-to-model and model-to-sources.	local	

Eclipse

A simple way of using the BtAntTasks with Eclipse is described here:

Under "Run" --> "External Tools Configuration" use the following expression for "Buildfile":

`${workspace_loc:${project_path}/build.xml}`

If a project is selected in the PackageExplorer Eclipse evaluates this expression and is able to find the project's build file. You are then able to select the required Target at the tab "Targets". For each required target you can make a copy of this configuration and when a project is selected you can choose the respective Target from the toolbar.



1. If no project is selected in the PackageExplorer Eclipse will generate an error message.
2. You have to take care that the required target exists for the particular project. Check the "build.xml".

TODO

▼ Expand / collapse

- General description for how to use the AntTasks (Syntax, paths, build files, etc.)
- Description artifact BuildCommands
- More detailed descriptions for the Targets
- Examples for the Targets
- Check for additional Targets that may be available
- Check for deprecated Targets

TFG-Greyface

Derives from
<p>▼ show/hide</p> <p> BTT-2460 - GM: Greyface Documentation (Closed)</p>

Table of Contents
<ul style="list-style-type: none"> • 1 Introduction • 2 Settings • 3 Importing • 4 The import process is structured into 4 steps: <ul style="list-style-type: none"> • 4.1 Parametrize the Interrogation Process <ul style="list-style-type: none"> • 4.1.1 Dependency • 4.1.2 Version ranges • 4.2 Select the Artifacts to import • 4.3 Import the selected Artifacts • 4.4 View the Results • 4.5 Importing from Local File System

Introduction

Greyface is an attempt to import artifacts in a controlled manner from remote maven-compatible repositories.

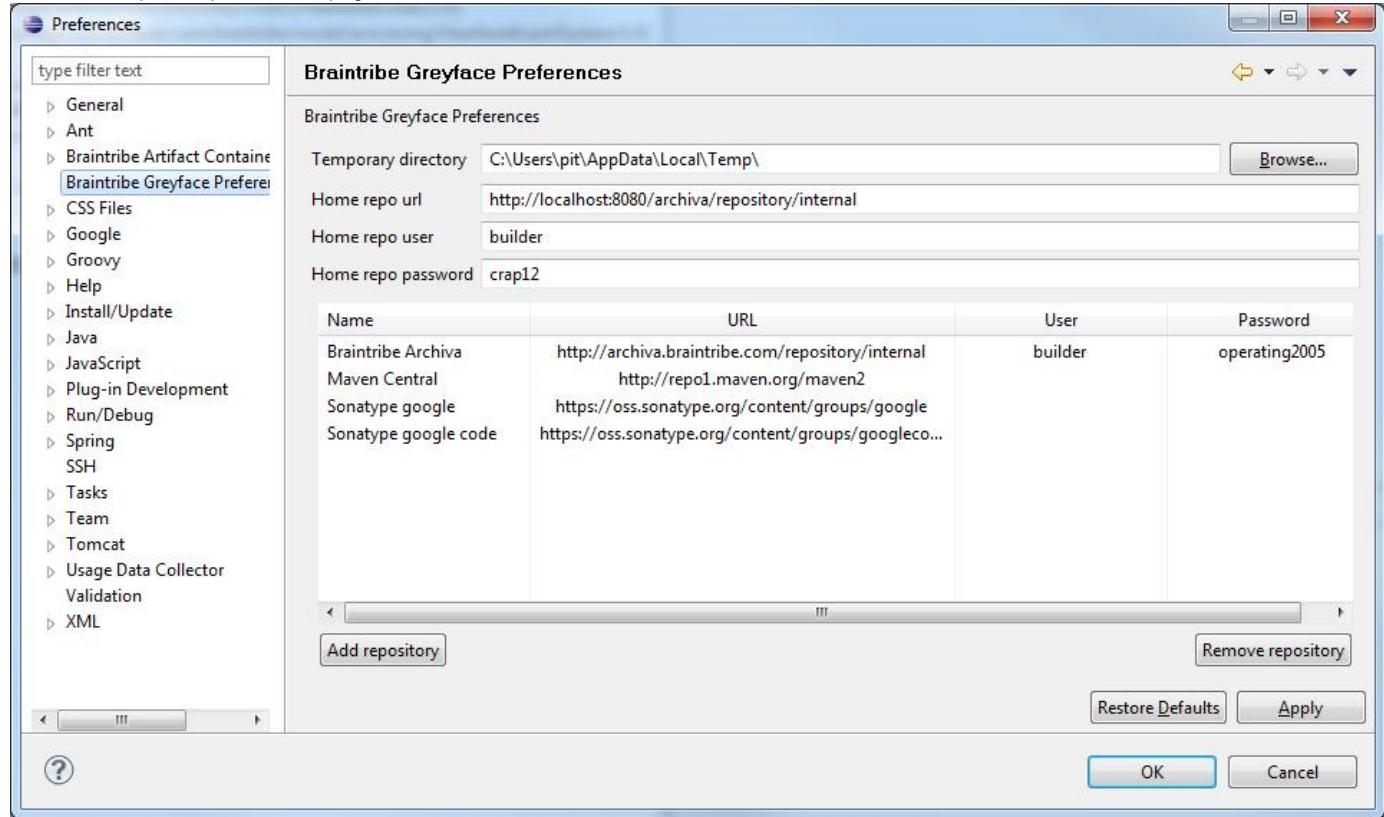
Why does Greyface exist?
<p>▼ Click to expand</p> <p>One issue is that Maven repositories typically do not store all artifacts, but are linked to proxy repositories. So if a repository doesn't find a specified artifact, it in turn asks the repositories it is linked to. Some (the Maven people) say that is a boon, some others (the Ivy people) say that this is a disadvantage. Dirk and I agree with the Ivy guys (treat yourself to their concise notes about running a closed enterprise repository on their project's website, here at http://ant.apache.org/ivy/history/latest-milestone/bestpractices.html), and we believe that our remote repository should contain everything we need and should not import what others think is needed.</p> <p>The other issue is that Maven repositories are based on the assumption that you invariably know what you're needing, and that you are never interested in what is actually available. This is exactly contrary to our approach. Even if Maven supports version ranges (i.e. an open or closed interval of versions), the repositories are not made to deal with it.</p> <p>We want to use version ranges to deal with hot-fixes - we don't want to adapt our dependencies just because a simple hot-fix of an artifact has come available, but we still want to have the hot-fix identifiable by a distinct version. So we're interested in what is available in the repository.</p> <p>In the remote repository, we can access that through the meta-data file, but this file is not updated reliably enough. On a proxied repository, this is not possible at all (unless nothing whatsoever exists in the remote repository - then we get the data from the first proxy that has some information).</p> <p>So what we are planning to do is to cut-off our repository from the proxies and instead have a controlled import process that is run when we need an external artifact. The artifact - and its dependencies - is then imported into our repository and is forthwith available for our builds.</p> <p>This process is currently in the works (none of the repositories we know of implement that out of the box), but will take some time to be properly done.</p> <p>Now that time has come and Greyface is here. It has been tested with the following products: Archiva, Artifactory and Sonatype. So if you know of any other Maven-compatible mess, let me know.</p> <p>Greyface's UI is integrated into Eclipse (that may not remain that way, as a servlet based version was envisaged as well, but for now, Eclipse will do. Furthermore, you're mostly in Eclipse when you realise that a blery artifact's missing).</p> <p>It uses plain http protocol to query the repositories, to download and upload the files. In our case, the repository is not anonymously accessible, and Greyface allows for other repositories to require that as well.</p>

Settings

As you might imagine, Greyface needs to know about the remote repositories you want to import from and obviously the repository you want to

import to.

Therefore, it sports a preferences page:



- **Temporary directory:** initially set by the JAVA TEMP variable, but any location you want the temporary files put.
- **Home repo url:** the URL of the repository you want to import the files to. To avoid any strange ramifications due to specified, uncontrolled and by any means useless Maven behaviour, make sure that the repository has no proxies attached.
- **Home repo user:** the user that may modify the repository, i.e. browse and upload files.
- **Home repo password:** the password for the user.
- **Remote repositories:**
 - **Name:** a name so you can figure out which one you're importing from
 - **URL:** the repositories main URL
 - **User:** if necessary, the user you want Greyface to use.
 - **Password:** it's accompanying password.

Greyface has the following repositories packed internally (what you see in the image is the test version that was using my own local archiva, a 1.42-M2):

The home repository is as follows:

Name	URL	User	Password
Braintripe	http://archiva.braintripe.com/repository/standalone	builder	operating2005

The remote repositories are as follows:

Name	URL	User	Password
Maven Central	http://repo1.maven.org/maven2		
Google	https://oss.sonatype.org/content/groups/google		

Google Code	https://oss.sonatype.org/content/groups/googlecode		
-------------	---	--	--

You can add as many repositories to the list as you want. You'll be able to select the repositories you want to interrogate later.

Importing

The import process is structured into 4 steps:

1. Parametrize the interrogation process.
2. Select the artifacts to import.
3. Import the selected artifacts.
4. Check the results.

Parametrize the Interrogation Process

Once you started it, Greyface greets you with the initial parameter page. Basically, you enter the dependency you want Greyface to resolve, in which repositories it should look for it and how it should traverse the dependencies of the found solutions.

Artifact import wizard

Set the parameters for the interrogation run

Dependency to import

Group: com.google.gwt
Artifact: gwt-user
Version Range: 2.^

Import from remote repositories Import existing file from file system

Repositories to interrogate

- Maven Central
- Sonatype google
- Sonatype google code
- Braintribe Archiva

Scan file for dependencies

Exclusions

- Activate exclusion control
- Only allow runtime/compile scopes

Miscellaneous settings

- Overwrite existing solutions

? < Back Next > Finish Cancel

Obviously enough, you can parametrize the import process in several ways:

Dependency

- Group: the artifact's group
- Artifact: the artifact's id
- Version Range: the range you want the artifact's version to be in.

Version ranges

A version range is basically an interval. It has a lower boundary and - obviously - an upper boundary. It can be open or closed. And the boundaries may collapse to a single version. The notation is basically Maven standard (they couldn't do much wrong here), but has a smart addition in form of a "dacherl" (the Ivy guys also thought about that, but understandably use a plus sign for the same purpose).



"Dacherl" = Circumflex: ^

So these are all valid ranges:

Expression	Matching versions
(1.0,2.0)	1.0 > version < 2.0
(1.0, 2.0]	1.0 > version <= 2.0
[1.0,2.0)	1.0 >= version < 2.0
[1.0,2.0]	1.0 >= version <= 2.0
2.3	2.3
2.1.^	2.1.0 >= version < 2.2
2.^	2.0 >= version < 3.0
^.	ANY

Actually, beginning from version 1.0.1 on . you can use three different delimiters. Malaclypse supports them as we found them out in the wild:

Character	Description
.	dot
-	dash
—	underline

All three are interchangeable. I don't suggest that you use any other than the common dot, but as there are such strange people in the Maven world (sic), Malaclypse must support it.

Furthermore, several common version suffixes are implemented, they are listed in the table below sorted by their intrinsic numeric value (first is the lowest):

Suffix	Description
M	Milestone
RC	Release Candidate
GA	General Availability
RELEASE	Stands for Release

i 1.0.M-1 is lower than 1.0.M-2, which is less than 1.0.RC-1, which is less as 1.0.RC-2 and so on.

i GA and RELEASE are currently interchangeable, so 1.0.GA and 1.0-RELEASE would be the same.

i Malaclypse is parametrize-able when it comes to that list. If you need other values, let us know.

Repositories to interrogate

This is a list of check boxes generated from the repositories you have listed in the preferences. Any of those checked will be interrogated in turn. Obviously, if you know that a certain repository will not have what you seek, or if you loathe the repository and will never accept anything from it, simply remove it from the list by un-checking it in the list.

Exclusions

Under this tag, two different forms of exclusions are combined:

Exclusion per Exclusion Tag

Activating this check box leads to Greyface heeding the exclusions tag in the POM file. All dependencies whose groups are mentioned in these tags are excluded from the walk. The exclusions are passed downwards, i.e. all subsequent tests inherit the exclusion set on a higher level.

Exclusion per Scope

This simply means what is written on the parameter page: Any dependency that sports another scope than "runtime" or "compile" is excluded.

Once parametrized and started via the next button, the interrogation process will in turn look in the active repositories for all solutions matching your dependency. This might take a while of course, depending on how precise your version range is and how the Maven repositories feel today (it may even be that they don't react at all today. Try tomorrow). So you can always abort the process by pressing the abort button - that square red thing right to the progress bar does the trick.

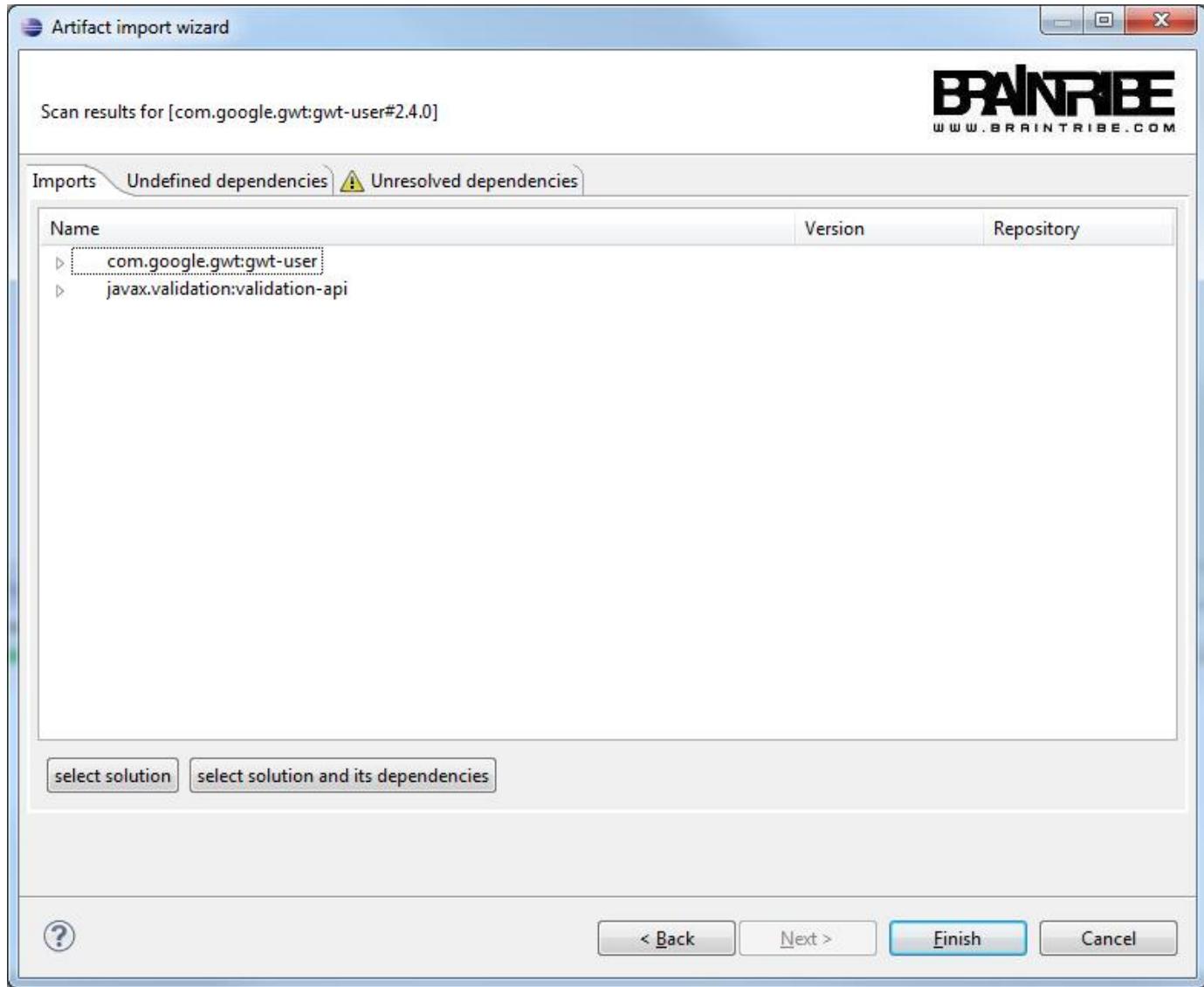
Select the Artifacts to import

Once the interrogation process's done, you are presented with the results.

The display page is structured into three tabs:

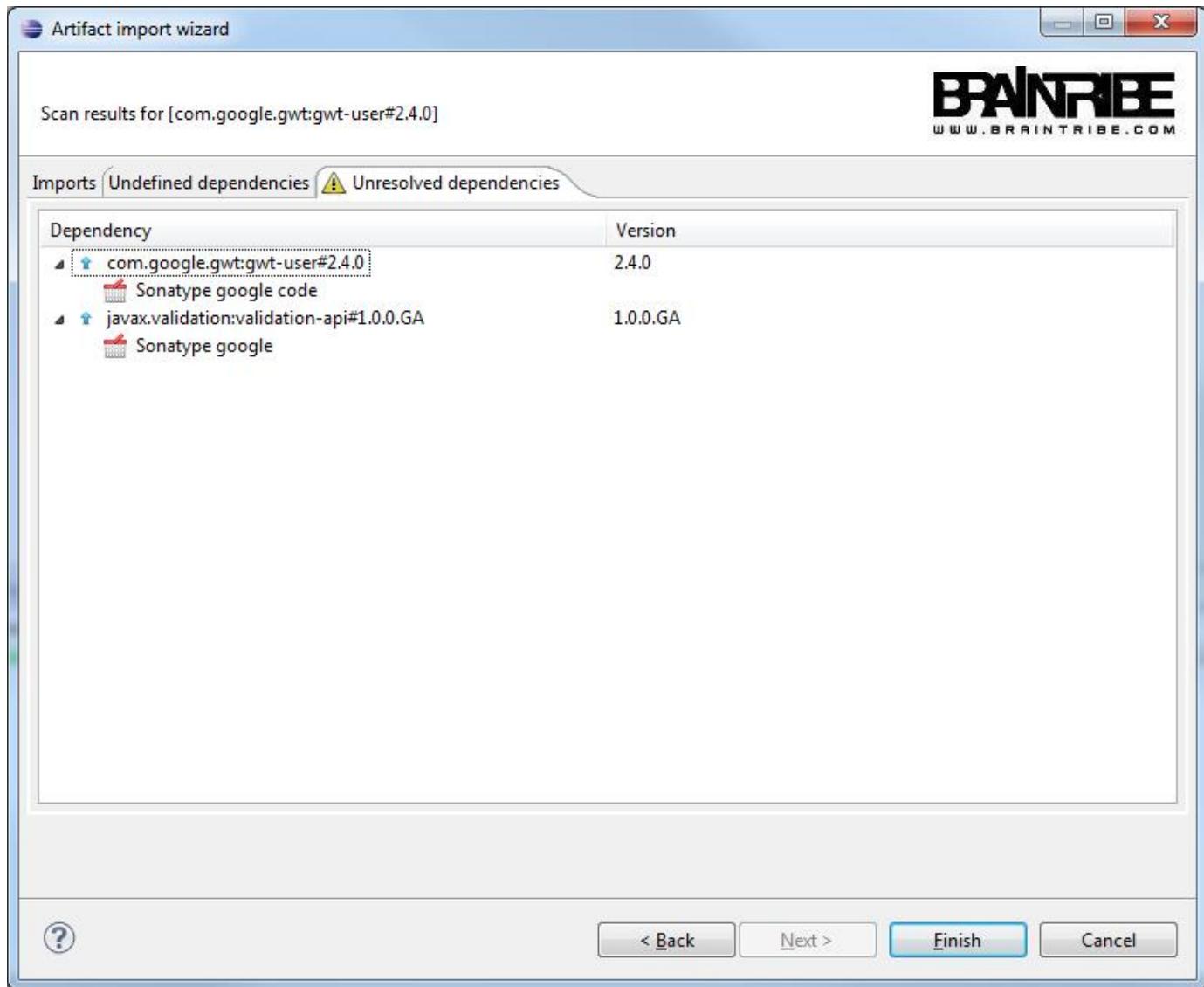
Tab name	Tab purpose
Imports	Shows the solutions that were found and their dependencies and lets you select what you want to import
Undefined dependencies	Shows dependencies that had no valid version range assigned
Unresolved dependencies	Shows dependencies that could not be found

If the tabs for the undefined or unresolved dependencies have any content, a warning sign is added to their title.



As you can see in this example, a scan for com.google.gwt:gwt-user#2.4.0 yielded two resulting solutions. There were no undefined dependencies, yet some unresolved.

Let's have a look at the unresolved dependencies.



We see that the repository "Sonatype google code" didn't contain any solution for com.google.gwt-user#2.4.0. Makes sense, as it's a repository for open source code that's hosted by google.

We also see that javax.validation:validation-api#1.0.0.GA wasn't in Sonatype google, which also makes sense, as this repository is only for code that has been released by google.

On the import tab, we see the solutions, their dependencies and where they are coming from.

Artifact import wizard

Scan results for [com.google.gwt:gwt-user#2.4.0]

BRAINTRIBE
www.BRAINTRIBE.COM

Imports Undefined dependencies Unresolved dependencies

Name	Version	Repository
com.google.gwt:gwt-user	2.4.0	Maven Central
com.google.gwt:gwt-user	1.0.0.GA	Maven Central
javax.validation:validation-api-sources	1.0.0.GA	Maven Central
javax.validation:validation-api	1.0.0.GA	Maven Central
com.google.gwt:gwt-user	2.4.0	Braintribe Archiva
com.google.gwt:gwt-user	2.4.0	Sonatype google
javax.validation:validation-api	1.0.0.GA	Sonatype google
javax.validation:validation-api-sources	1.0.0.GA	Sonatype google
javax.validation:validation-api	1.0.0.GA	Maven Central
javax.validation:validation-api	2.4.0	Maven Central
com.google.gwt:gwt-user		

select solution **select solution and its dependencies**

? < Back Next > **Finish** Cancel

What we can see here is that there was a solution to com.google.gwt:gwt-user#2.4.0 in three repositories (remember, the fourth didn't have it).

Interestingly enough, the dependencies differ, depending which repository defines them. "Maven Central" and "Sonatype google" have javax.validation:validation-api#1.0.0.GA included, the solution found in our "Braintribe Archiva" doesn't.

The javax.validation:validation-api#1.0.0.GA is only found in Maven Central (remember it has been marked as unresolved for "Sonatype google" - Why doesn't it show as missing in "Braintribe Archiva"? Because it's not referenced by any dependency of that repository).

If a version already exists in your home repository, it is specially marked like here:

com.google.gwt:gwt-user	2.4.0	Sonatype google
com.google.gwt:gwt-user	2.1-SNAPSHOT	Sonatype google
com.google.gwt:gwt-user	2.1.0	Sonatype google

How does Greyface determine whether you have that artifact already? It looks for the pom file, i.e. the two hashes of the file, the md5 and the sha1 hashes and compares them to the ones of the found solution's pom file. If they match, you obviously already have the artifact.

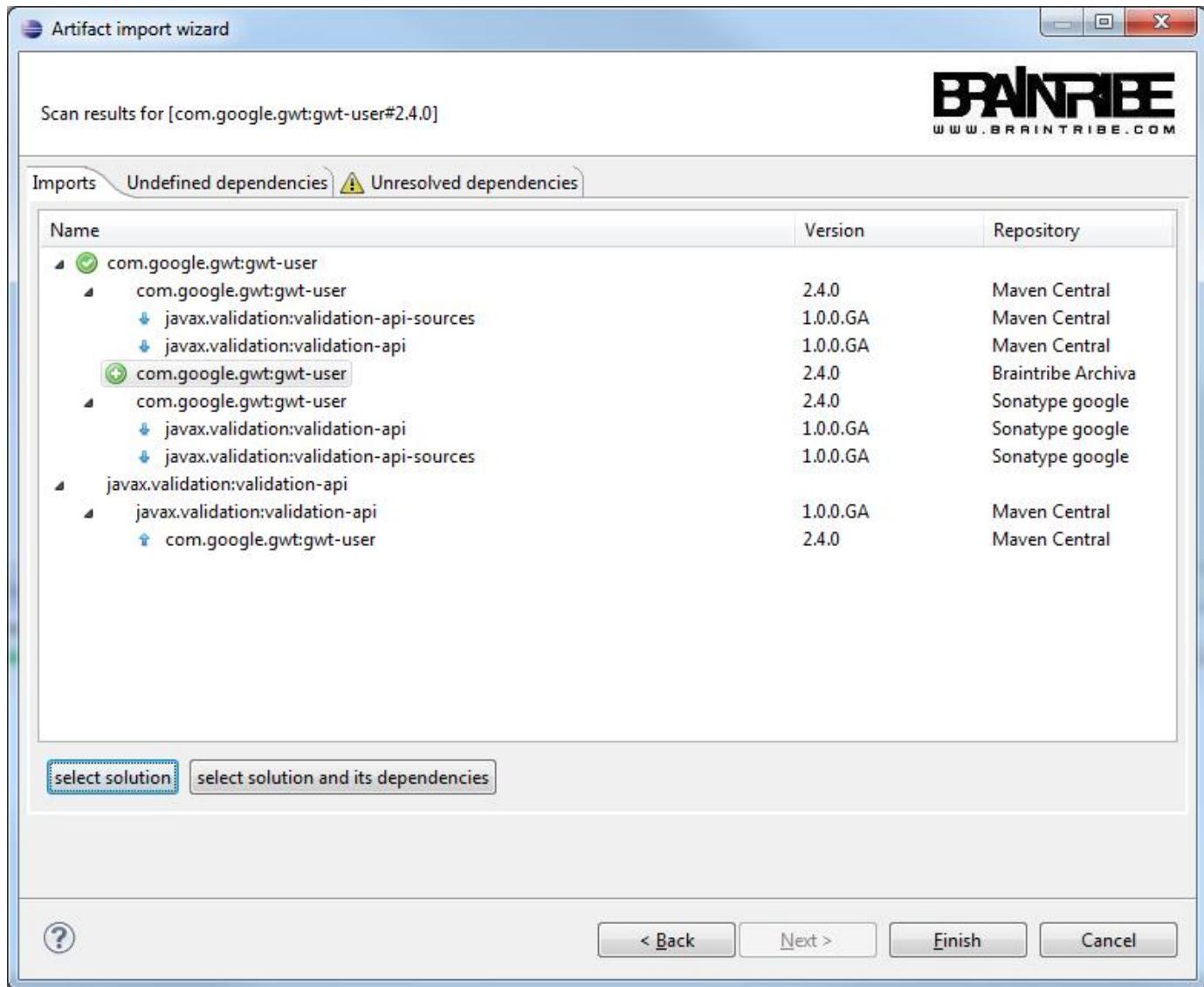


Note

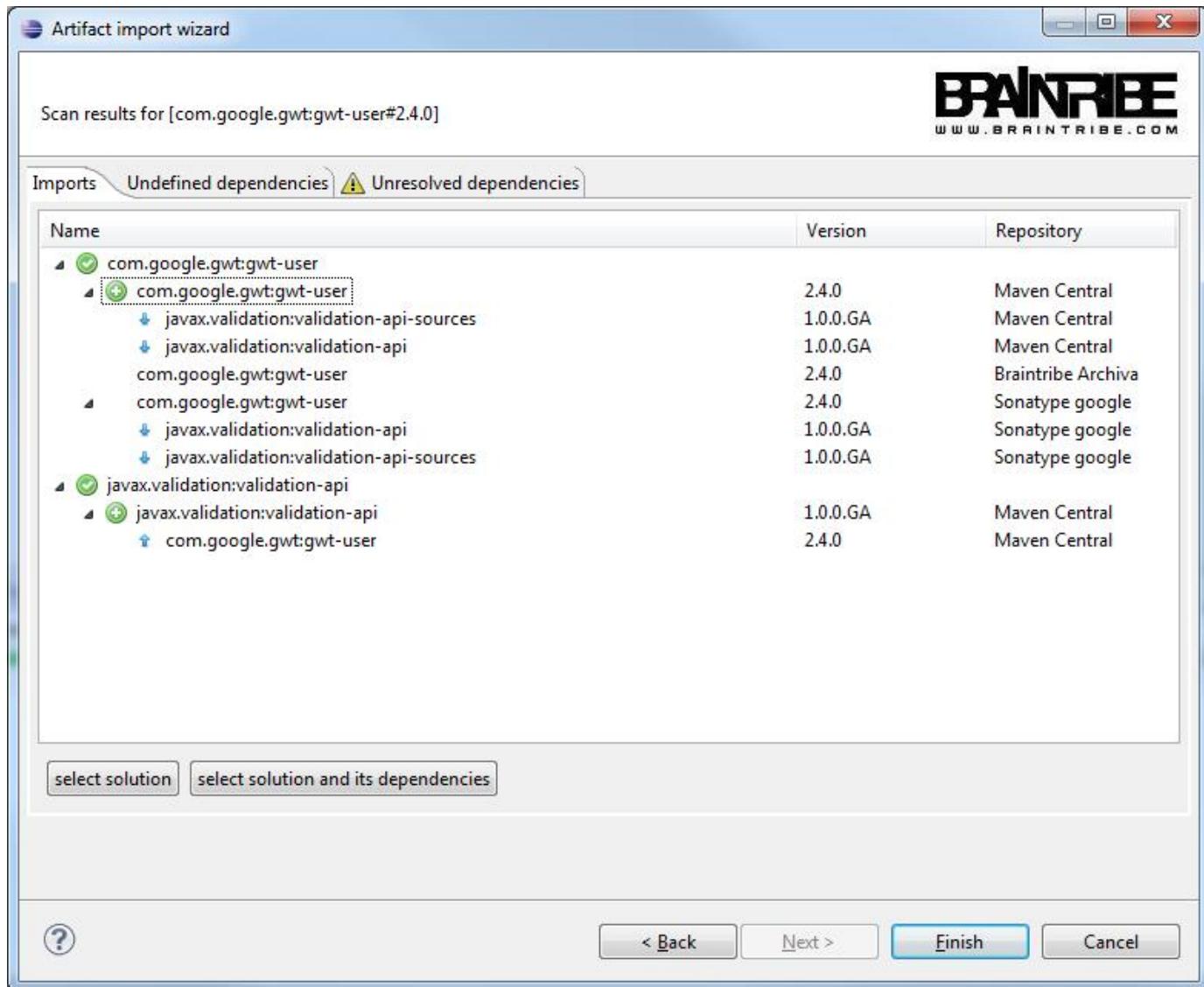
If your repository only contains the pom file and its hashes, but not all other files like the jar, you're screwed. In this case, you must manually remove the files in order to get Greyface to import the artifact. But in this case, your repository's bugged anyhow.

You have two ways to select the solutions you want to import. You can either select the solutions interactively, specifying for each selected

solution from which repository you want it imported from, or you can select a complete dependency tree of a single repository.



Selecting a solution in the tree and then clicking on the button "select solution" or double-clicking the solution will select it for the import. The solution will show a plus sign to the left and the family of the solution will get a check-sign.



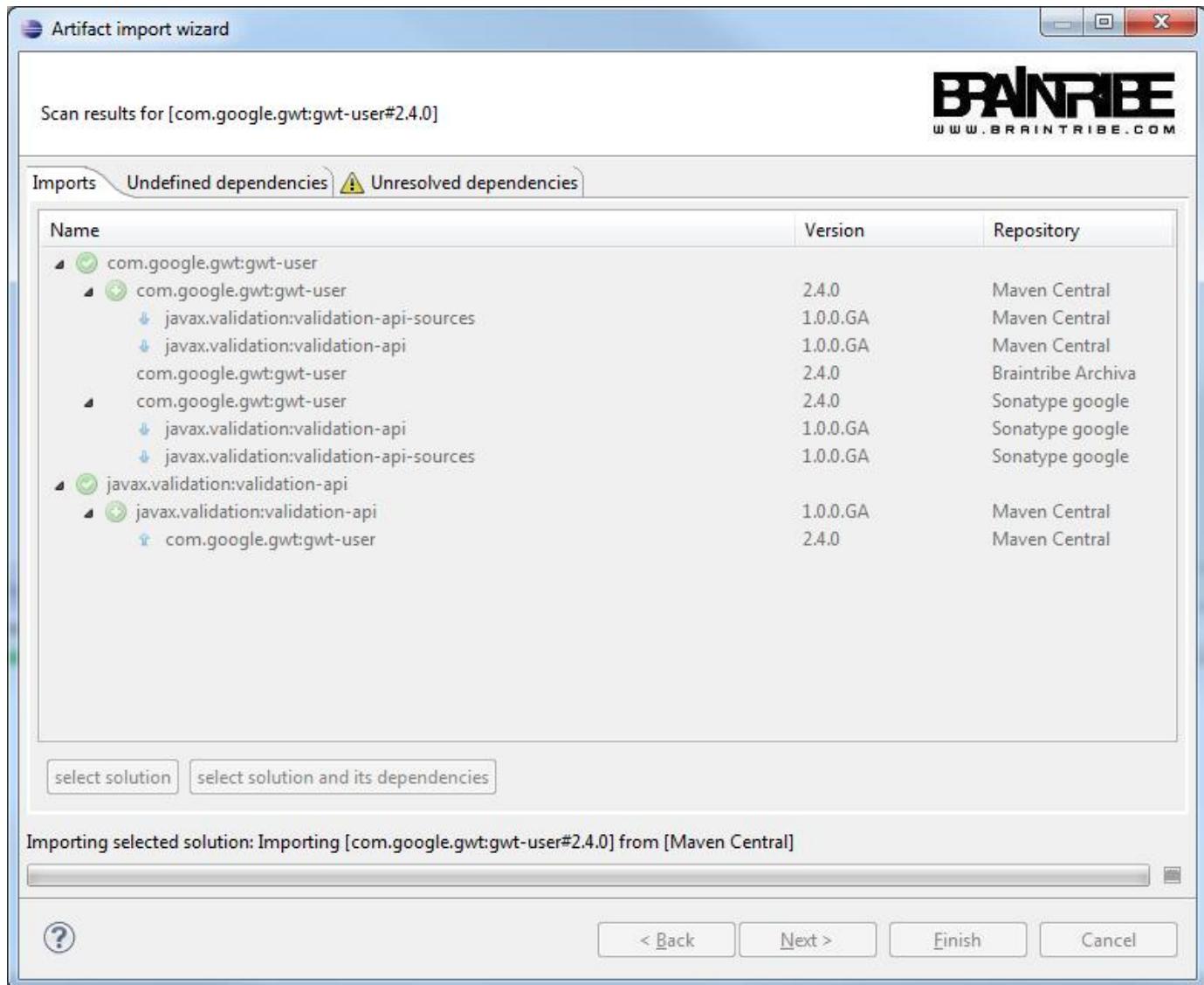
Selecting a solution in the tree and then clicking on the button “select solution and it’s dependency” or double-clicking while holding the CTRL-key the solution will select it and its dependencies from the same repository for the import.

Once you’ve done your selection, you can press next and let Greyface do its importing.

Import the selected Artifacts

Greyface will now in turn access the respective repository, download all files of the selected artifact to the temporary location and upload them to your home repository. If needs be, it will calculate the hashes and create the files for the two hashes.

Furthermore, it will write the meta-data file expected by Maven.



Again, during the import, you can abort the process anytime. If you wanted to. Of course.

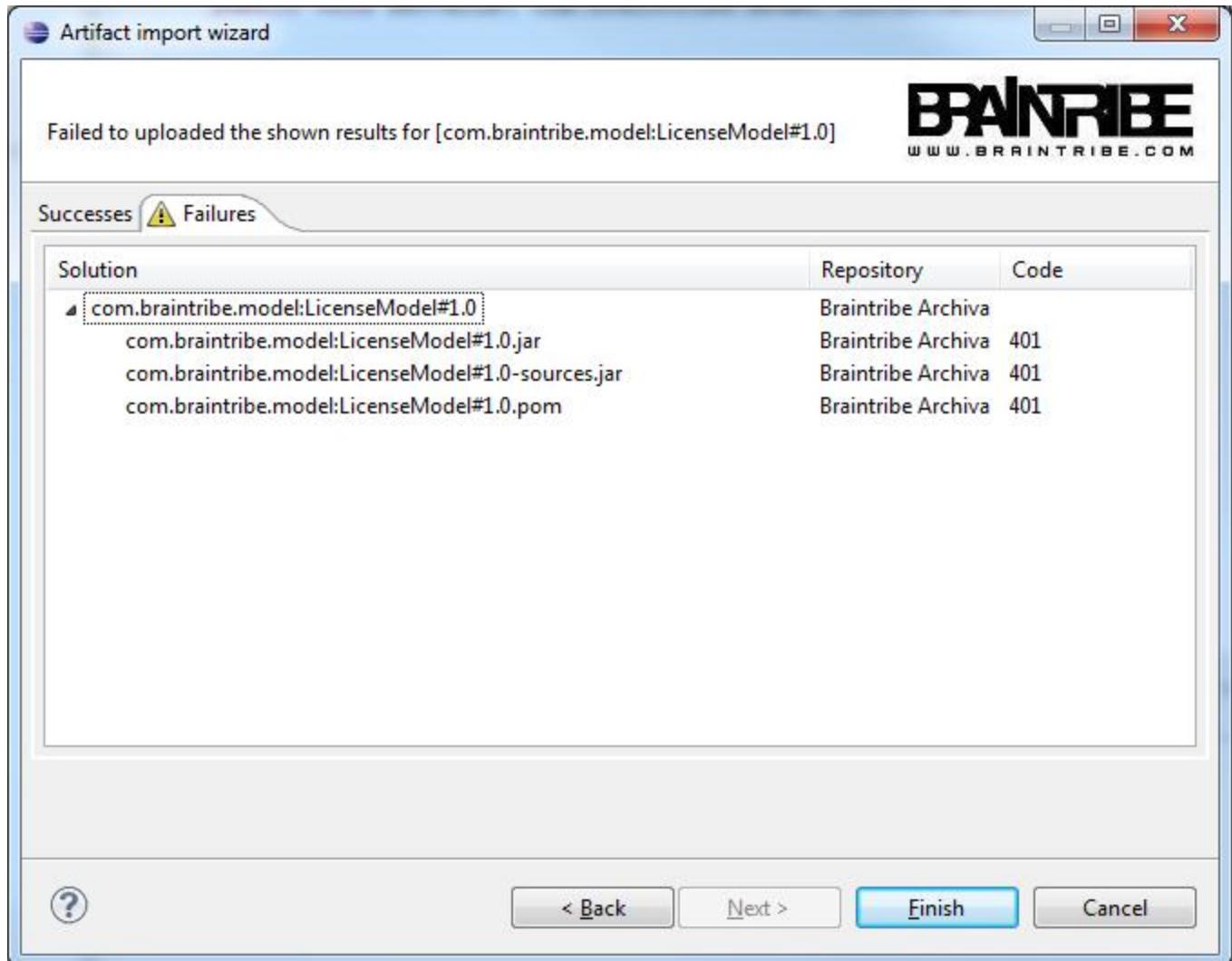
View the Results

Once the import's done, the final page appears.

The view is split in to two tabs - one tab shows the successfully imported parts whereas the other shows the parts that could not be imported.

The tab for the failures shows which parts weren't imported. In the rightmost column, the HTTP return code for the upload process is shown.

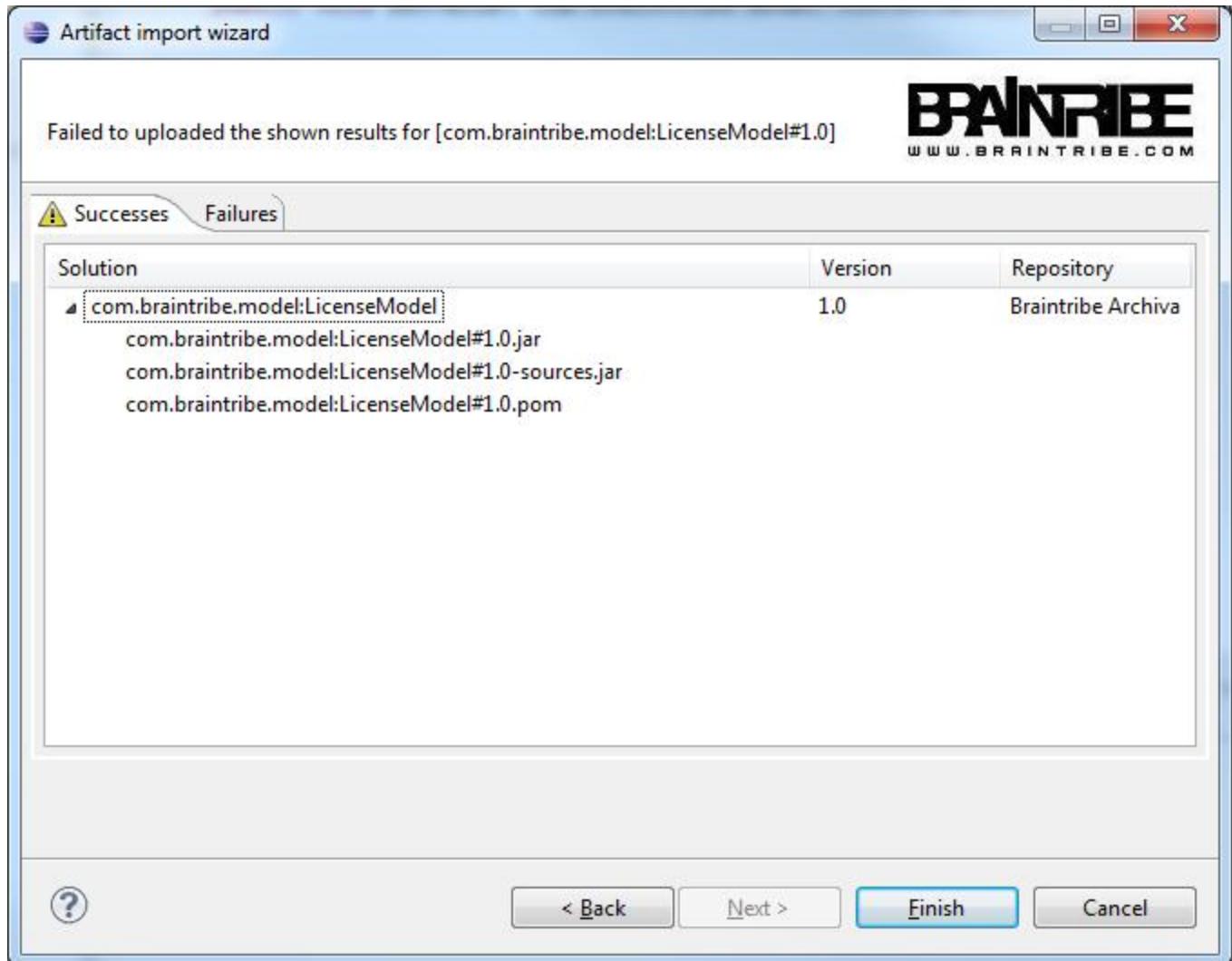
Here, the code 401 means that the access was unauthorized - the password of the user was wrong:



As you can see, if the tab's not empty, the caution sign is added to the tab's name.

So if everything went ok, like in the example below, the caution sign's missing on the "Failures" tab and appears on the "Sucesses" tab.

Again, the tab shows the successfully imported solutions, and the parts it consist of.



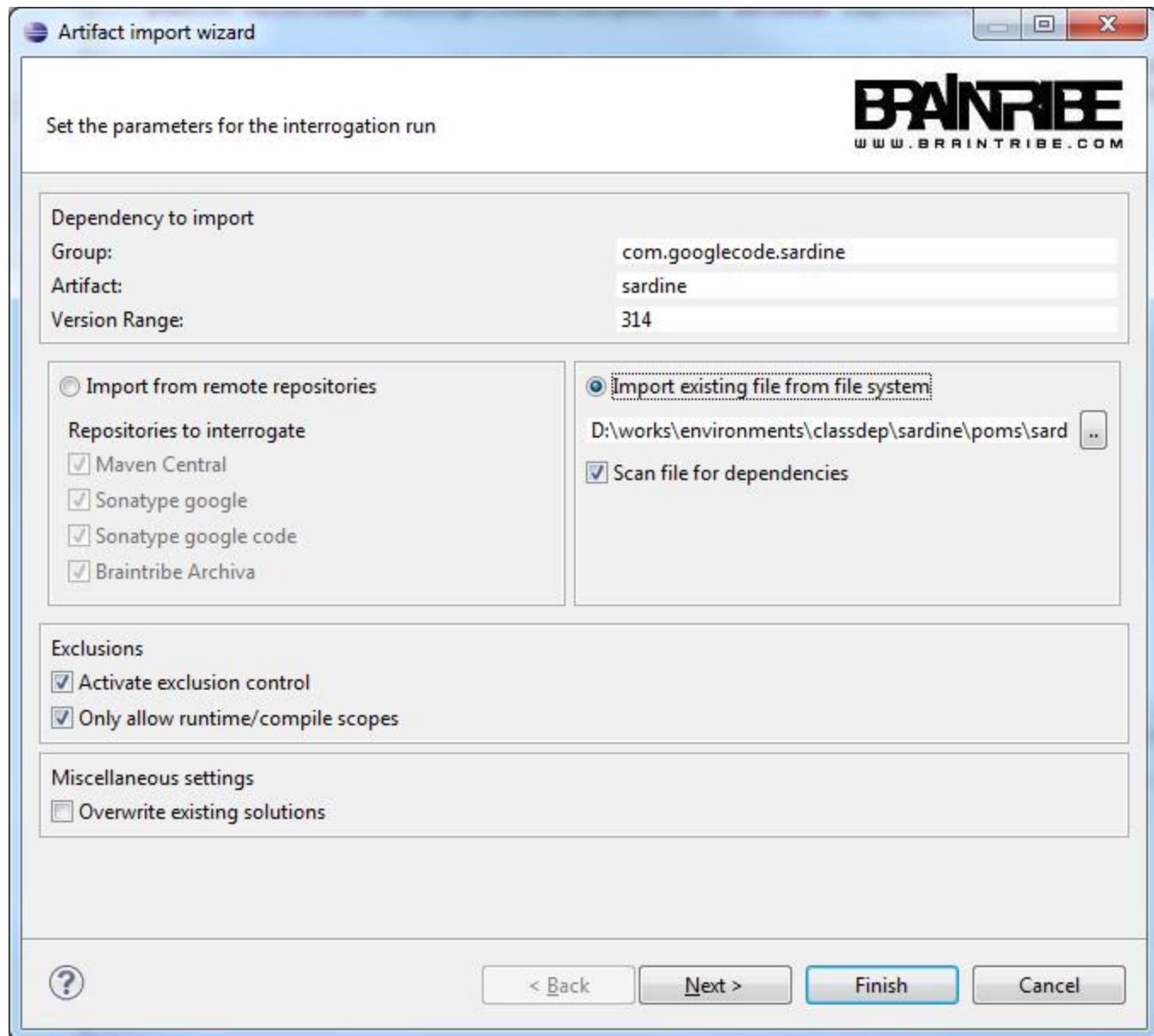
Now you can close the wizard by either pressing Finish or Cancel.

Importing from Local File System

As an alternative, there is the possibility to upload one or several files from the local file system as in some cases, you only have the file locally present (for example after creating the pom via the ClassDependencyWalker). For this case, Greyface's local import has been implemented.

When you parametrize the import, select the radio button labelled "Import existing file from file system".

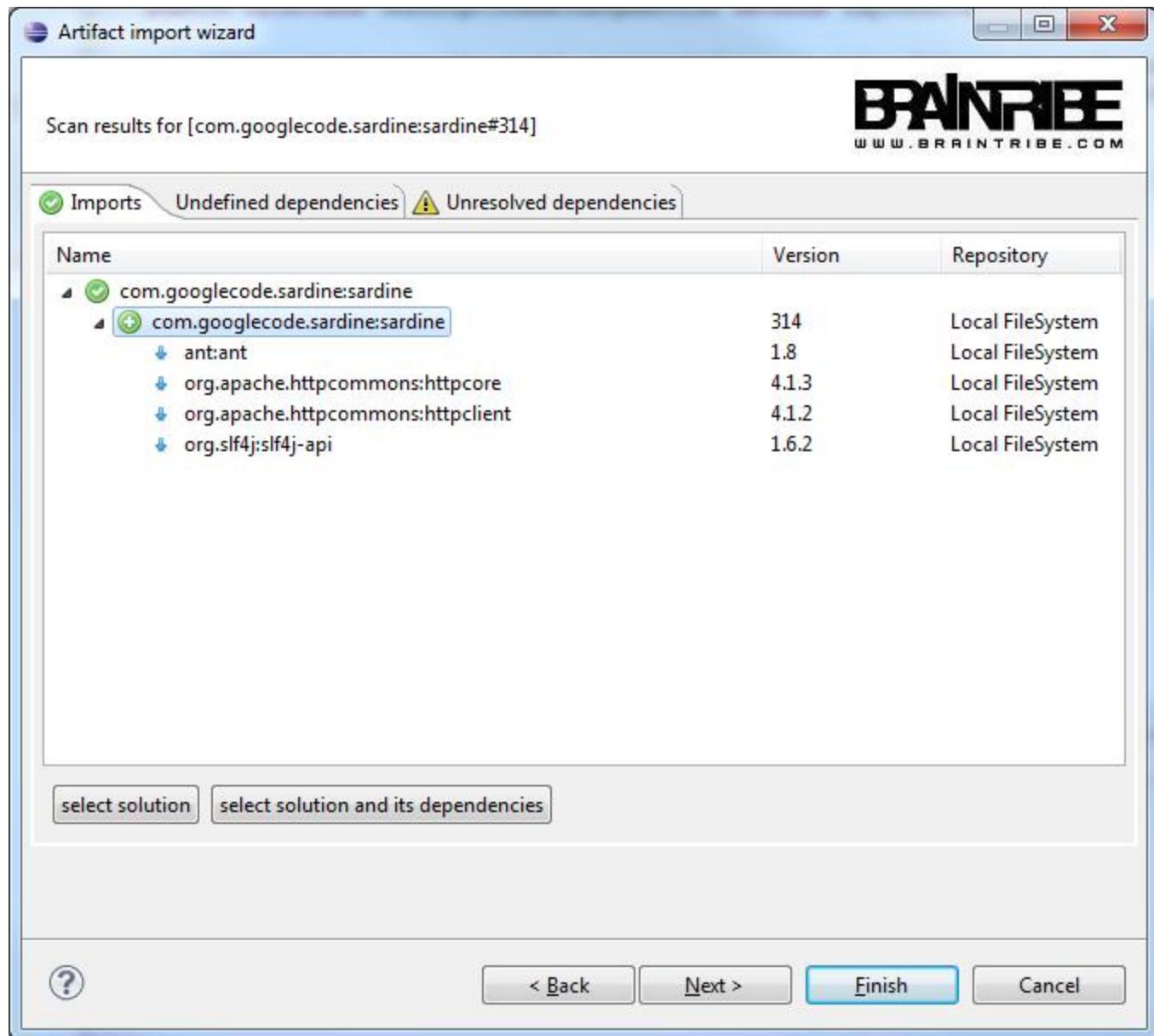
Then browse for the file you want to import.



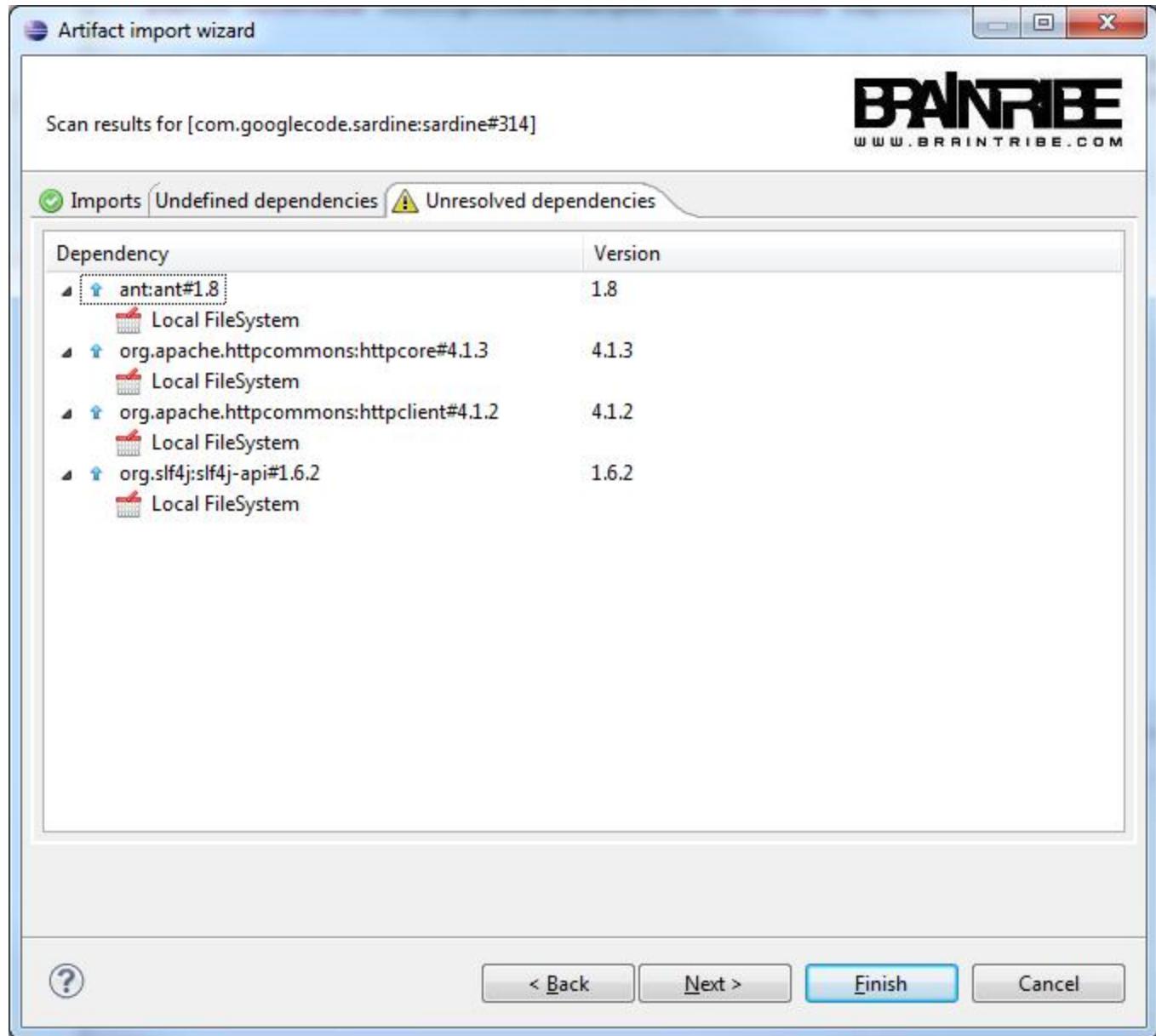
Check	Description
Scan file for dependencies	If set, Greyface will scan for the dependencies of the selected artifact, using the local directory as container, i.e. the solutions should also reside in the same directory as the selected terminal artifact. If not set, Greyface will not scan for any dependencies.

Once you're done, press next.

Greyface will now show the exact same screen as during a standard import: it will display what it found and you can select what (if any) solution you want to import.



As, in our test case, only the terminal artifact exists, the dependencies cannot be found.



You can now proceed as you would in the remote import process. Greyface will generate the hash-files and the appropriate meta files for the any solutions you selected.

tribefire Development

tribefire Development

This section contains development-relevant documentation about tribefire.



Children pages

- Building CSP ServerDeps using the new platform-exclusion-based way
- Transition from the ECM Service to the Access Service
- Provider API
- Codecs

Building CSP ServerDeps using the new platform-exclusion-based way

Introduction

For GM 1.0, a new way of bundling the artifact dependencies for CSP is used. This new way is based on so called platform-exclusions, described here:

https://docs.google.com/a/braintribe.com/document/d/1xFJfMVIKRZyIn_ZGMfDleaJtBg2tqltLqpqkCTXUWRA/edit#

Related artifacts

Artifact	Description
com.braintribe.csp.server.deps:StandardServerDeps-4.0	Contains the <i>pom.xml</i> describing the artifact dependencies for CSP 4.0
com.braintribe.build.platform:CSP-4.0	Contains the <i>exclusions.txt</i> describing which libraries are already present in CSP 4.0 and therefore do not need to be bundled in the generated JAR

Building the jar:

```
ant -Dartifact=com.braintribe.csp.server.deps:StandardServerDeps-4.0 build
```

The exclusions are maintained in the **com.braintribe.build.platform:CSP-4.0** artifact.



After updating the *exclusions.txt*, install the artifact.

Transition from the ECM Service to the Access Service

Introduction

Part of the roadmap for GM 1.0 aka Tribefire was an overhaul of the APIs of the Accesses used to access model data and the service providing centralized access to them. Previously this part was covered by the **ECM** module in CSP. This module was superseded by the new **Access Service** - the transition and changes are briefly described in this page.

i While it is planned to host the services for accessing GM data in a J2EE environment in the future, GM 1.0 will still use a CSP Server as platform.

CSP Configuration Patches

The configuration patches for the Access Service are based on the patches of the ECM module.

Old patch base location	New patch base location
csp/cfg_patches/ecm	csp/cfg_patches/access_service

They have been drastically reduced to the current basic needs, though, and therefore only consist of these patches initially:

- base
- accessDeployment
- test

Other patches will be created or migrated from the ECM service when they are needed.

CSP Configuration

While actual configuration of the access service is similar to that of the ECM service it differs mainly in the naming of folders, files and configuration elements.

Here is a table containing a quick overview of the changes:

Type	Old	New	Description
Folder	cfg_patches/ecm	cfg_patches/access_service	Patch folder
Folder	cfg/IOC/Ecm	cfg/IOC/Access	IOC configuration root for the service
Folder	cfg/IOC/Ecm/Access	cfg/IOC/Access/Impl	IOC configuration for the Access implementations hosted by the service
Folder	cfg/Resources/Ecm	cfg/Resources/Access	Configuration root for resources used by the service and/or its Accesses
File	ecm.spring.xml	accessService.spring.xml	Configuration file of the service
Module Name	ECM	ACCESS	CSP module name of the service
Spring bean ID	ecm.implementation	accessService.implementation	ID of the implementation bean of the service
Spring bean ID	remotes.ecm	remotes.accessService	ID of the proxy bean for accessing the service in CSP server code

Naming conventions for Spring IOC beans

Previously the bean IDs used for the Accesses looked like this:

```
ecm.access.test.Access  
ecm.access.test.MetaModelAccess  
ecm.access.accessDeployment.Access  
...
```

Since these IDs were kind of bloated and would have gotten even more confusing when replacing `ecm` with `accessService`, we decided to agree on new naming conventions. The basic pattern is:

```
$modelName.Access  
$modelName.MetaAccess
```

In the basic patches, this leads to bean IDs like these:

```
testModel.Access  
testModel.MetaAccess  
testModel.MetaFileAccess  
testModel.RawAccess  
testModel.FileAccess  
testModel.tc.map  
testModel.tc.default  
testModel.tc.complexTypeCondition  
  
accessDeploymentModel.Access  
accessDeploymentModel.MetaAccess  
accessDeploymentModel.RawMetaAccess  
accessDeploymentModel.FileMetaAccess  
accessDeploymentModel.RawAccess  
accessDeploymentModel.FileAccess  
accessDeploymentModel.tc.map  
accessDeploymentModel.tc.default  
accessDeploymentModel.tc.std
```



These naming conventions are, of course, not final. In case there are issues with this convention they can be discussed. In case the description of the convention provided here is not clear enough or should contain more examples, feel free to edit this page.

Provider API

Table of Contents

▼ Expand / collapse

- Provider API
- - Provider
 - SimpleProvider
 - IndexedProvider
 - SimpleIndexedProvider
 - MapIndexedProvider
 - FallbackProvider
 - FilterBaseIndexProvider
- Receivers
 - Receiver
- Holders
 - Holder
- Compound Providers
 - CompoundProvider
 - HashSetCompoundProvider
 - ArrayListCompoundProvider
 - Example
- GWT IOC centered abstract providers

Provider API

Probably the most underrated API in the artifact code-base is obviously the ProviderAPI. If I label it obviously it because we see newly interfaced interfaces that are not only absolutely superfluous as they declare things that are already included in the ProviderAPI collection, but are also thoughtlessly used to pollute distinct areas such as server vs client, CSP code-base vs Artifact code-base.

You might find it futile (and perhaps even rude) to document such a trivial thing as the ProviderApi. And yet, it is a very central thing (as the use cases are very common) and we had to learn (the hard way) that it is not really used where it definitively should have been used.

Following is a recent example found in the CSP code-base with impact on clients:

```
public interface InterfaceA extends Provider<String> {

    public String provide() throws ProviderException;
    public void receive(String payload);
}
```

```
public class ClassA implements InterfaceA{

    String sessionID = null;

    public ClassA (String payload) {
        this.payload = payload;
    }

    @Override
    public String provide() throws ProviderException {
        return this.payload;
    }

    @Override
    public void receive(String sessionID) {
        //do nothing
    }
}
```

You shouldn't need to ask me now what is wrong with that code, but I'll tell you anyway.

The interface that was created that is absolutely superfluous.

Interfaces are a special thing that they are the connectors of two different realms. We call that encapsulation. If an interface is defined in one realm (or code-base or server) and required in another realm (or code-base or client) then we have introduced a dependency between the two realms.

File-wise that might be ok.

But if it's part of a project which has other dependencies, then we're polluting the other realm. In this actual case, the interface was defined server side and needed to be referred from the client side.

That's actually rather obvious, right? So do me a favour and don't do that anymore.

Now a much cleaner way would have been to derive from the Holder class of the Provider API

```

public class Holder<E> extends SimpleProvider<E> implements Receiver<E>{

    public Holder(E object) {
        super(object);
    }

    public Holder() {
        super(null);
    }

    public void setProperty(E object) {
        this.object = object;
    }

    public void receive(E object) throws ReceiverException {
        setProperty(object);
    }
}

```

Why is that cleaner? Because you can override (if you really need to) in the CSP code-base. It is automatically imported via the CspServerDeps library, and in the Artifact code-base, its definition exists in an artifact anyway. So there are NO introduced dependencies of the CSP code-base directly into the artifact code-base.

Again I ask you for a favour: Pretty please, let's keep our coding world clean.

Ok, you didn't know about the ProviderApi. Right. That's the reason for this paper.

It is intended to give you an overview of the functionality the ProviderAPI contains and insights of how to use them.

Remember, the ProviderAPI is a basic part of the CspServerDeps, so you can use any of them within the CSP codebase and of course in the Artifact codebase as well.

Now, please don't get me wrong. I don't mind documenting these things, but I do mind giving the impression of being a control-freak, which I am not. I like personalized code and I will always oppose strict coding guide-lines that makes a programmer feel like living in a totalitarian state.

So, as an idea, instead of introducing code-reviews, why don't we do code-previews: If you need something done in the artifact code-base or in an area of the server that impacts the artifact code-base, go to one of the (possibly) more proficient artifact coding buddies and tell them what you intend to do. He might be able to give you tips about how to do it, and where to find useful patterns and code snippets.

Provider

A simple but powerful interface is the simplest provider.

```

package com.braintribe.provider;

public interface Provider<E> {
    E provide() throws ProviderException;
}

```

Typical patterns are:

- To defer the instantiation from its actual use - you can wire it in Spring as a singleton at start-up and let it provide the required data once you need it.
- To deliver the contents of a file, like a velocity template or an XSD schema; or in general
- To provide a CSP specific setting to an Artifact without polluting the Artifact code-base.

Note the template parameter `<E>`. This means of course that you don't have to declare an interface `BlaProvider<String>` and a `BlaBlaProvider<Object>` - all your instance has to do is to implement `Provider<WhateverObjectYouNeedInYourInstance>`.

SimpleProvider

This instantiation of the `Provider<E>` interface is actually **deprecated** and only mentioned for matters of completeness. See the holder class for a replacement.

```
public class SimpleProvider<E> implements Provider<E>
```

Property	Type	Description
object	E	The object that <code>provide()</code> should return.

IndexedProvider

This interface is a bit more complex. It allows for the passing of one parameter, but otherwise works like the `Provider<T>` interface.

```
package com.braintribe.provider;

public interface IndexedProvider<I, E> {
    public E provide(I index) throws ProviderException;
}
```

Patterns are obvious, I think - anywhere you'd use a `Provider`, but need a parameter.

SimpleIndexedProvider

Again, a simple implementation of the `IndexedProvider<I,E>` interface.

```
SimpleIndexedProvider<I, E> implements IndexedProvider<I, E>;
```

The single value it returns (disregarding the index parameter I) is passed by a constructor.

MapIndexedProvider

```
public class MapIndexedProvider<I, E> implements IndexedProvider<I, E>;
```

The class MapIndexedProvider<I,E> is an special implementation of the IndexedProvider. It interprets the parameter directly as index for the value to return.

Property	Type	Description
map	Map<I,E>	A map of the corresponding types as the IndexedProvider-interface uses.

FallbackProvider

The fallback provider class parses a supplied list of providers that may return null until it finds a provider that returned a non-null value.

```
public class FallbackProvider<T> implements Provider<T>;
```

Properties are as follows:

Property	Type	Description
providers	List<Provider<? extends T>>	A list of Provider<E> where E may derive from the original parameter T.
treatNullAsFailure	boolean	If set to true, null-results are returned, if set to false, it will try all providers and throw an exception if no non-null value's found.

FilterBaseIndexProvider

```
public class FilterBasedIndexProvider<I, E> implements IndexedProvider<I, E>
```

The FilterBasedIndexProvider is again a special implementation of the IndexedProvider interface. It has a map of filters (see [com.braintribe.FilterApi](#) for more about filters) as keys to values. It filters the passed parameter and if the filter matches the parameter, it will return the appropriate value.

Property	Type	Description
filterMap	Map<Filter<I>, E>	A map linking a filter using I as parameter with a value E.

Receivers

Receiver

This very basic interface is used to declare the inverse of a provider.

```
public interface Receiver<T> {
    public void receive(T object) throws ReceiverException;
}
```

Holders

Holder

The Holder class is an instance of both the Provider and the receiver interface.

```
public class Holder<E> extends SimpleProvider<E> implements Receiver<E>;
```

It implements both provide() and receive() functions, but also treat the object it can receive and will provider as a standard property.

Property	Type	Description
property	E	the property that will provide() will return. You can also set the same property via receive(E).

Compound Providers

CompoundProvider

The CompoundProvider is an abstract implementation of a Provider that bundles multiple other providers to create a Collection of results.

```
public abstract class CompoundProvider<T extends Collection<E>, E> implements Provider<T>;
```

Generic Type	Description
T	Generic Type specifying the Collection implementation the compound provider will return, e.g. HashSet.
E	Generic Type specifying the type of elements contained in the resulting collection.

The collection of providers it uses to create its results is configurable using a setter-method. Actual implementations of this class only have to implement the method newCollectionInstance().

Currently, there are two implementations of compound providers:

HashSetCompoundProvider

A CompoundProvider implementation for java.util.HashSet results.

```
public class HashSetCompoundProvider<E> extends CompoundProvider<HashSet<E>, E>
```

ArrayListCompoundProvider

A CompoundProvider implementation for `java.util.ArrayList` results.

```
public class ArrayListCompoundProvider<E> extends CompoundProvider<ArrayList<E>, E>
```

Example

Here is an example for using a compound provider, based on the `HashSetCompoundProvider`:

```
//creating the provider
HashSetCompoundProvider<String> compoundProvider = new
HashSetCompoundProvider<String>();

//configuring the providers it uses to create its results
Provider<String> provider1 = new Holder<String>("string1");
Provider<String> provider2 = new Holder<String>("string2");
List<Provider<String>> providers = Arrays.asList(provider1, provider2);
compoundProvider.setProviders(providers);

//this result will contain the values 'string1' and 'string2'
HashSet<String> result = compoundProvider.provide();
```

GWT IOC centered abstract providers

The rest of the API is used to implement Spring-like IOC features to GWT and are therefore not of global interest. In consequence they aren't documented here.

Codecs

Introduction

Another pattern that you'll find quite often in the artifact codebase is the codec. A codec in our terms is always bidirectional, i.e. it translates from one format into another and vice versa.

Codecs basically come in two flavours. Generic codecs and expressive codecs. Generic codecs can transform ANY structure of GenericEntities, whereas expressive codecs even if their parameters are structures of GenericEntities will only transform a certain subset of GenericEntities into a specific representation of it.

Typical representatives for the generic kind are the GenericModelJsonCodec and the GenericModelDomCodec the ones for the expressive kind are the ContentDefinitionCodec that handles CSPs contentdefinition and hierarchy and the PainCodec that understands SWIFT's payment initiation format (pain.00x.003.02).

The basic interface, exceptions and utils are in the **com.braintribe:CodecApi1.1SNAPSHOT**, the implementing codecs are found in the group com.braintribe.codec.

The dependency looks like this :

```
<dependency>
    <groupId>com.braintribe</groupId>
    <artifactId>CodecApi</artifactId>
    <version>1.1SNAPSHOT</version>
</dependency>
```

Some important basic implementations (for instance the CodingContext, see below) are found in the artifact **com.braintribe.BasicCodecs1.0SNAPSHOT**. Its dependency is as follows:

```
<dependency>
    <groupId>com.braintribe</groupId>
    <artifactId>BasicCodecs</artifactId>
    <version>1.0SNAPSHOT</version>
</dependency>
```

It includes codecs for all simple types, but also convenient codecs for urlencoded data, and semantically important codecs like the passthroughcodec. Additionally, it contains the codec registry which can be used to smartly build a more complex codec.

Codec interface

```
package com.braintribe.codec
public interface Codec<T, E> {
    public E encode(T value) throws CodecException
    public T decode(E encodedValue) throws CodecException
    public Class<T> getValueClass()
}
```

Reflection

The function below is used to tell us about the value that the codec handles.

```
public Class<T> getValueClass()
```

This is handy when you write complex codecs which are dynamically configured with a set of “sub codecs”.

Decoding

Decoding obviously accepts the encoded form of the handled class and decodes it.

```
public T decode(E encodedValue) throws CodecException
```

How the codec does it is of course entirely left to the codec. There are however a few things to be considered, see the chapters about the coding contexts and the one about the recursivity.

Encoding

Again, this is more or less plain to see: it accepts an actual instance of the handled class and returns its encoded form.

```
public E encode(T value) throws CodecException
```

Again, some points need to be considered see below.

CodingContext

As mentioned above, the coding context is also part of the BasicCodec artifact. It is handy to transport data throughout the structure of the codec. Basically, it's used as a base class for more specific contexts, yet also as a stack for the contexts via it's static stackhandler functions.

```
try {
    EncodingContext ctx = new EncodingContext()
    ctx.setMatcher(matcher)
    ctx.setDocument(getDocumentBuilder().newDocument())
    CodingContext.push(ctx)
    try {
        ctx.pushTraversingCriterion(new RootCriterion(), entity)
        GenericModelDomCodecRegistry registry = getCodecRegistry()
        Codec<T, Element> codec = registry.getCodec(type)
        Element element = codec.encode(entity)
        return element
    } finally {
        ctx.popTraversingCriterion()
    }
} finally {
    EncodingContext.pop()
}
```

The class EncodingContext is derived from CodingContext, and used to convey additional data to the codecs stored in the codec registry. Before the inner loop starts, it is pushed to the top of the stack, and when it's no longer used, it's popped off the stack.

Typically, if encoding to XML is intended, the encoding context would contain the document, so that the actual handlers can access it to create elements within the documents.

Responsiveness

One important point to consider is the question of responsiveness when working with recursive structures. It is less of importance in the server environment, but becomes a very important topic once moved to the GWT world.

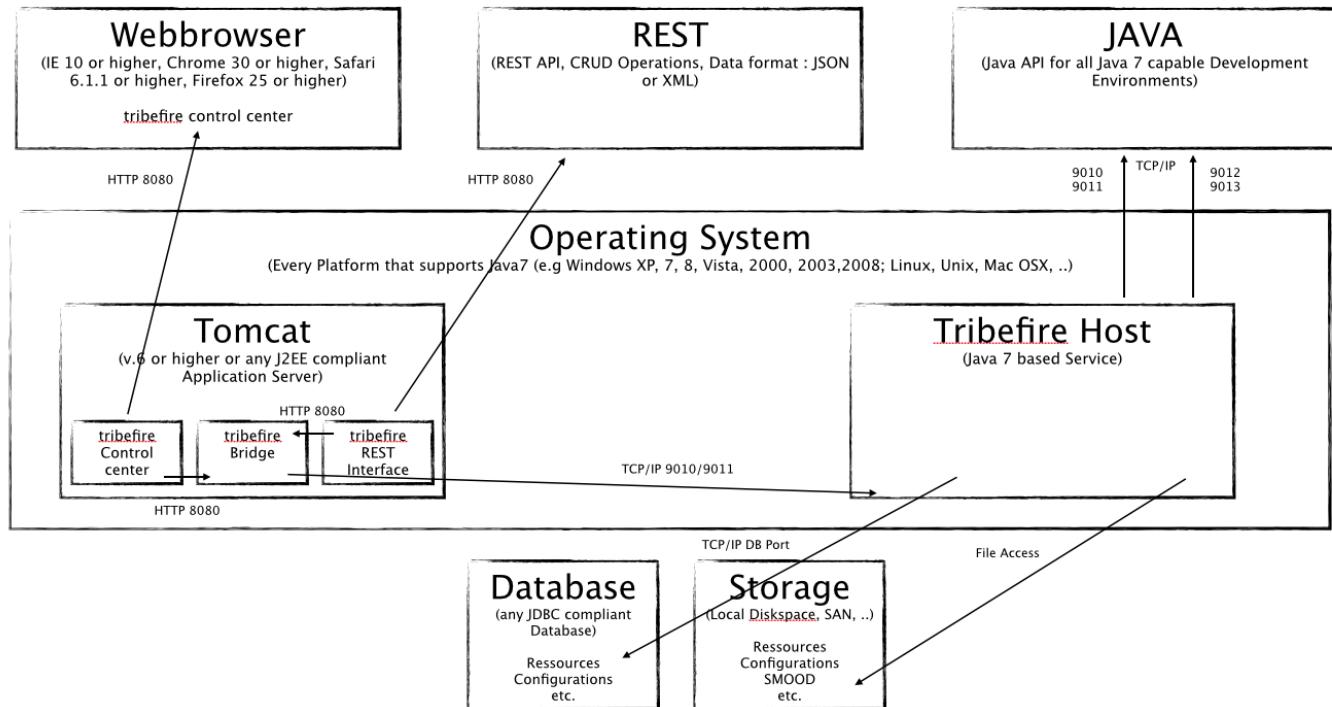
There will be a special document concerning responsive codecs or better any time consuming process.

tribefire System Requirements

API	
Java	Java API for all Java 7 capable Development Environments
REST	REST API, CRUD Operations, Data format : JSON or XML
tribefire host	
Operating systems	Every Platform that supports Java7 (e.g Windows XP, 7, 8, Vista, 2000, 2003,2008; Linux, Unix, Mac OSX, ..)
Processors	Minimum 2 Cores, recommended 4 Cores (physical CPUs or virtualized Cores)
Memory	Minimum 8 GB
Disc space	Minimum 30 GB
Network connections	TCP/IP
Database	any JDBC compliant Database
tribefire control center	
Browser	IE 10 or higher, Chrome 30 or higher, Safari 6.1.1 or higher, Firefox 25 or higher
screen resolution	Minimum 1280 x 1024
Software requirements	
Apache Tomcat or any J2EE compliant application server	v.6 or higher
Java	v.7 or higher
Port	
Usage	
8080	Default Port (e.g in case of tribefire control center is running on Tomcat)
9010/9011	Methodcalling and Streaming
9012/9013	Secure Methodcalling and Secure Streaming (TLS encrypted)

For Proof of Concepts or Pilots it is recommended to have remote access to the tribefire installation.

Port	Usage
22	SSH (Unix / Linux)
3389	Default Port for Windows Remote Desktop (in case the tribefire Host is running on Windows)



tribefire Glossary

Derived from [BTT-2442 - GIFT - Glossary in Tribefire](#) (Open) - Contribution is welcome!

Term	Meaning	Comment
tribefire cortex	Generic Model - The paradigm where everything is a model .	This will probably need a better description.
GME	Generic Model Explorer (f.k.a.: Generic Model Editor) - UI for creating, editing and using models .	
tribefire	The product derived from Generic Model .	
Model	A model is an assembly of entity types and enum types.	Better description required.
Manipulation Model	The Manipulation Model is the model describing manipulations in order to reflect any modification of a GM entity .	
MetaModel	A Meta Model allows to talk about models in a generic manner. The Meta Model is the model that models models.	
Assembly	An assembly is a set of meshed entity instances of a model. Some special case of an assembly is just one single entity instance, but mostly it's a subset of a population.	
Traversing criteria	Traversing criteria are a set of statements that tell a traversing process of an assembly where to abort the traversal. Typical use cases are deep-cloning, querying.	Should probably link that to the examples in the upcoming compendium (pit)
Types - GmTypes	Types are used to reflect the types we encounter in the real world.	
MetaData	MetaData is data that is attached to the types. They contain any form of data that gives additional information about the types.	
Constraints	Constraints are a form of MetaData that in some aspect reflect restrictions.	
Prompting	Prompting controls how entities or properties should be displayed in the UI.	
Selector	A Selector controls whether associated meta data is applicable or not, i.e. they are a kind of activation criteria. A selector in the end evaluates to either true or false, i.e. it matches or it doesn't.	
Query Model	The Query Model allows to describe a query in a generic way that can be interpreted by the actual query processors of the different persistence layers.	
Query Plan Model	A model that maps the Query Model to a more specific model that knows joins, indices and tuples.	joins, indices and tuples are specific DB terms in my understanding and should be explained or at least link to explanations
Connector	A Connector provides a basic connection to a 3rd-party system like a database or an ECM system.	

Access	Provides access to a 3rd-party system's entities and data (e.g. database tables and data).	Better description required. Aren't Accesses more than means for accessing 3rd-party systems?
Entity - GenericEntity	An entity is an instance of a type of a model .	
GmEntityType	A type instance of the Entity in the model .	
GmEnumType	An enumeration type in the model .	
GmEnumConstant	A value of an enumeration type in the model .	
GmProperty	A member property of a type within the model .	
Expert	An expert interprets a model . Models are platform independent, experts are not.	'Platform independence' might not be a term everyone is familiar with.
ArtifactContainer (AC)	Eclipse plugin to control the classpath and implement other useful features of the artifact code-base.	Currently linking to 'artifact', but maybe a separate entry for 'artifact codebase' might make sense?
Artifact Model	A model that models all classes involved in a artifact s and their dependencies.	
Malaclypse	A library that implements features centered on artifacts and their dependencies (can be used in Eclipse, ANT, Gradle etc.)	
Greyface	Greyface is an attempt to import artifacts in a controlled manner from remote maven-compatible repositories. Eclipse plugin to import artifacts into our company repository.	
AcDevloader	A class loader for Tomcat 7 and AC based projects as a replacement for the standard classloader. Improves usability in conjunction with the Sysdeo plugin.	
Ravenhurst	A servlet that accesses the database of our Archiva installation and retrieves information about the installed artifacts , such as what artifact changed after a certain time stamp or what the actual time stamp of the parts of an artifact is. It is used by Malaclypse 's global-repository expert.	
Dependency Controller (DC)	A servlet that gives access to company wide settings and features, such as release/publish procedures, check builds etc. (<i>only vaporware at this point, even if Ravenhurst is a step into this direction</i>)	
GmDeclarationEditor	An Eclipse plugin that creates model declaration for models, stored in the package-info.java file.	
Khayyam	A graphical viewer for dependency trees (<i>only vaporware at this point</i>)	
SMOOD	Simple Memory Object Oriented Database - our own database for models no matter how complex they are.	
DeploymentAccess	The main access through which other accesses can be configured and deployed.	

Prototype	Prototypes can be used when designing a new model (or adding new entity types). For example, if one wants to model existing data stored in a database, the system can automatically build prototypes based on the database tables. These can then be used to conveniently create new entity types.	
ITW	Instant Type Weaving - A technology to create entities based on a MetaModel at runtime.	
Manipulation Tracking	The recording of instantiations of entities and manipulations applied on entities during a session scope. The manipulations are reflected as instances of the ManipulationModel .	
WebReader	A GWT UI component that deals with the DocumentModel .	
Workbench	?	Actually at term solely used in the GME realm as far as I know but a short description might make sense?
Session	?	We should describe what a 'Session' in GM context means.
DocumentModel	?	Actually a concrete model but it might still make sense to have a short description here for the sake of completeness?
Artifact	An artifact encapsulates source code and/or resources. It can depend on other artifacts and 3rd party libraries - this supports reuse of code/resources and better separation of concern. Applications can be built by combining a multitude of small artifacts.	Maybe this document can help finding a better description?
Terminal Artifact	A terminal artifact is the starting point of a dependency walk, the root of the dependency tree. A artifact aimed for distribution (an application, or more generically speaking, an artifact reflecting a product or a release in a WAR or EAR or whatever) is obviously also a terminal artifact.	Is this the kind of application built from small artifacts I described in Artifact? (FPA).. not necessarily so - see text (pit)
Manipulation	A manipulation describes a modification of an entity, for example changing the value of a property.	Since we use this in some other descriptions we should define our understanding of a manipulation here.
Persistence Layer	Encapsulates the actual storage of persistent data - for example as XML or in a database.	Does this need a detailed definition here?
GWT	Google Web Toolkit, a toolkit for writing web-applications in Java. More info at http://www.gwtproject.org/overview.html .	
Synclonizing	The process of synchronizing a pure "type skeleton" meta model with a existing meta model that already has meta data attached. As the process involves both synchronization and cloning, it was dubbed "synclonizing".	
lightweightStateProcessor	Term from the Maelstrom project. Discontinued and replaced by the StateProcessing complex.	
Lazy loading	The principle of deferred loading of data, so that the access on the data triggers the retrieval of the data. Antonym to "eager loading"	

Partial representation	The principle that any representation of data can either be complete or partial. It is of utmost importance as in a world of interlinked objects, by means of recursively following the links of an object, a single object can be related to the whole world. Therefore, retrieval of objects from a persistence level for instance can only be partial.	
Template / Template Model	A template is a blue print for the creation of data. It may consist of default values and variables (or functions, i.e. methods to generate/retrieve the required values). The template model describes such templates.	
Incremental Access	<p>In order to encapsulate the different specific features of persistency implementations behind a layer of abstraction, we introduced the 'IncrementalAccess' as an implementable interface. The interface aims at the following:</p> <ul style="list-style-type: none"> • To provide the MetaModel • To apply incremental changes described by the ManipulationModel to the persistency layer • To process queries as described by the QueryModel <p>This 'IncrementalAccess' is the essential abstraction layer that allows software components like the GME to process their persistency tasks in a normalized manner. At the same time, we see it as an adaptor for the heterogene environments we are confronted with at customers' sites.</p>	
ECM Access	?	
ECI Access	?	

 Use headings for the terms to be able to cross-link them. An inner-page link can be created using the link macro (CTRL + K) - choose *Advanced* and configure #HEADING_YOU_WANT_TO_LINK as link (e.g. #Model)