Toromiro User Guide

Contents

1. Getting Started
    1.1. What is Toromiro?
    1.2. Installation
    1.3. First Steps
    1.4. Tutorial
        1.4.1. Repository Connections
            1.4.1.1. Connecting to a Jackrabbit Repository
            1.4.1.2. Custom JCR Adapters
        1.4.2. Repository Browser View
        1.4.3. Repository Search View
            1.4.3.1. Query Languages
            1.4.3.2. Modify Nodes by Script
        1.4.4. Node Editor
        1.4.5. Child Nodes View
        1.4.6. Repository Node Types View
        1.4.7. Node Type Editor
            1.4.7.1. Jackrabbit Node Type Editing Support
            1.4.7.2. Custom JCR Adapter Node Type Editing Support
        1.4.8. Node Type Diagram Editor
        1.4.9. Repository Watcher View
        1.4.10. Preferences
    1.5. Tips And Tricks
    1.6. Reporting Bugs
2. Reference
    2.1. Toromiro Legal

Toromiro User Guide

This is the **Toromiro User Guide** which will help you install, setup and work with **Toromiro**.

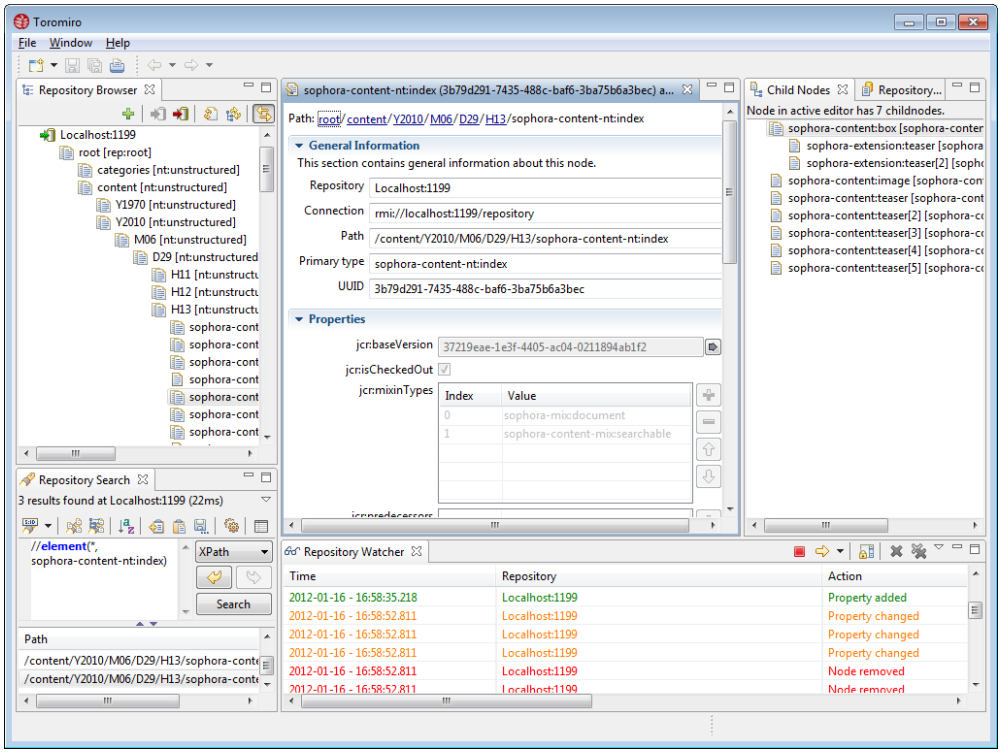
1.1. What is Toromiro?

Toromiro is a tool for searching and editing the contents of a Java Content Repository (JCR). The main features are:

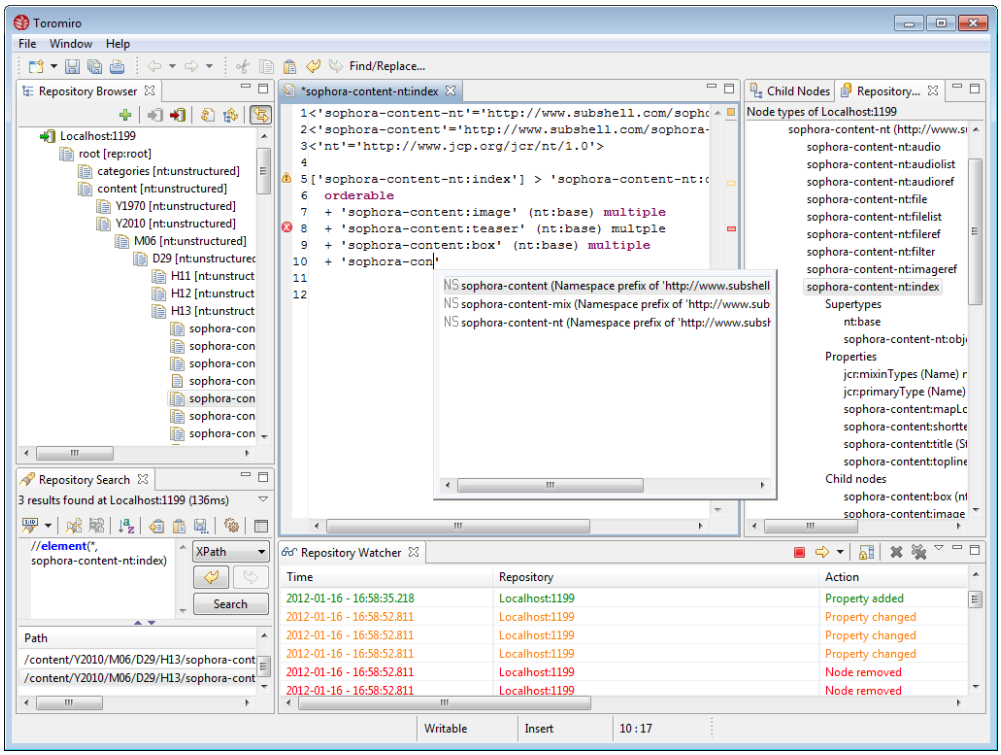
- Connecting to more than one repository at the same time
- Searching by query or UUID
- Searching by references to nodes
- Adding, deleting, moving, and renaming of nodes
- Adding, removing, and editing properties of nodes
- Adding and removing mixins on nodes
- Live monitoring of content changes in repositories
- Node type editing with syntax validation, syntax highlighting, auto completion and context assist
- Graphical display of node type hierarchies

Screenshots

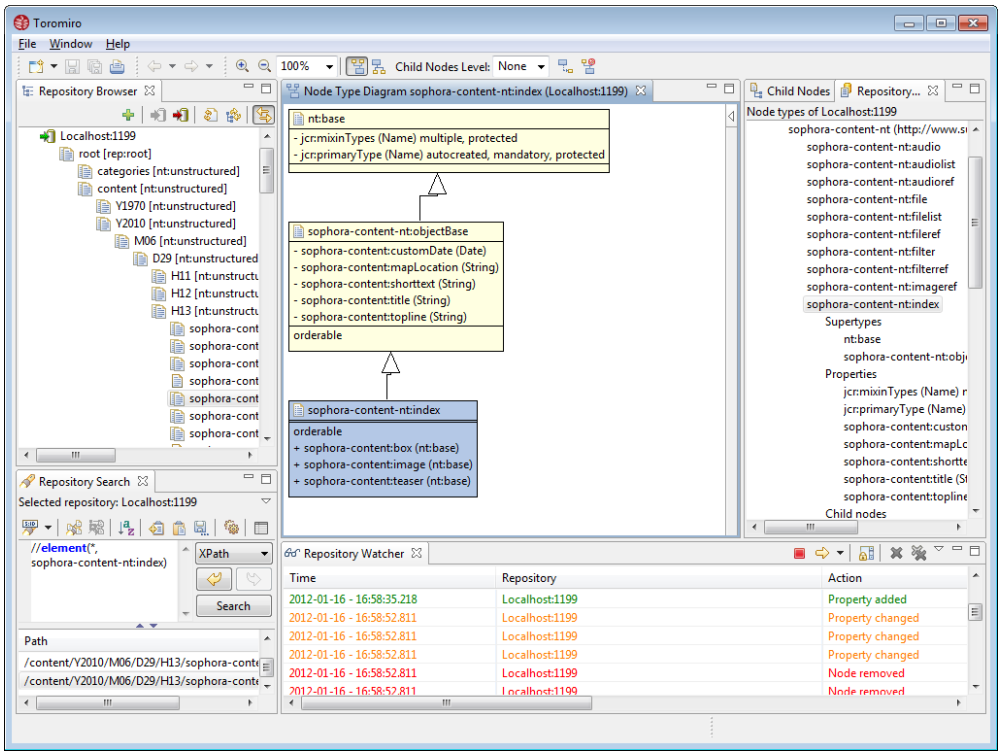
Toromiro - Node Editor



Toromiro - Node Type Editor



Toromiro - Node Type Diagram Editor



1.2. Installation

The installation of **Toromiro** is pretty simple. All you have to do is just to download and extract the archive file of **Toromiro** for your platform. You can download the files here: https://www.subshell.com/en/sophora/modules/toromiro/.

Supported Systems and Architectures

- Linux GTK 64bit
- macOS 64bit
- Windows 64bit

Minimum System Requirements

- Java Runtime Environment 11
- At least 64 MB of main memory

Starting Toromiro

Start the **Toromiro** executable, depending on your operating system:

- On Windows, run **toromiro.exe**.

- On macOS, run **toromiro.app**.
- On Linux, run **toromiro**.

> Note:
> On some operating systems like Linux or macOS you may have to set execution permissions explicitly for running the application.

Supported Repositories

Toromiro is based on version 2.0 of the Content Repository for Java Technology API (JCR) specified by the Java Specification Request 170 (JSR 170) and Java Specification Request 283 (JSR 283). By default it uses the Apache Jackrabbit implementation, a fully conforming implementation of JCR 2.0. The connection to a content repository works directly on local repositories or over Java Remote Method Invocation (Java RMI). It is also possible to define custom JCR adapters for connecting to other JCR repository implementations.

Updates

Toromiro can look for new versions each time when started and update automatically. To enable/disable automatic updates, these may be switched on/off in the Preferences within the **Install/Update > Automatic Updates** section. A manual update can be started using **Help > Check for Updates** from the main menu.

Proxy Settings

A Proxy can be configured in the Preferences within the **Network Connections** section. Restarting the application afterwards is recommended.

Add-ons

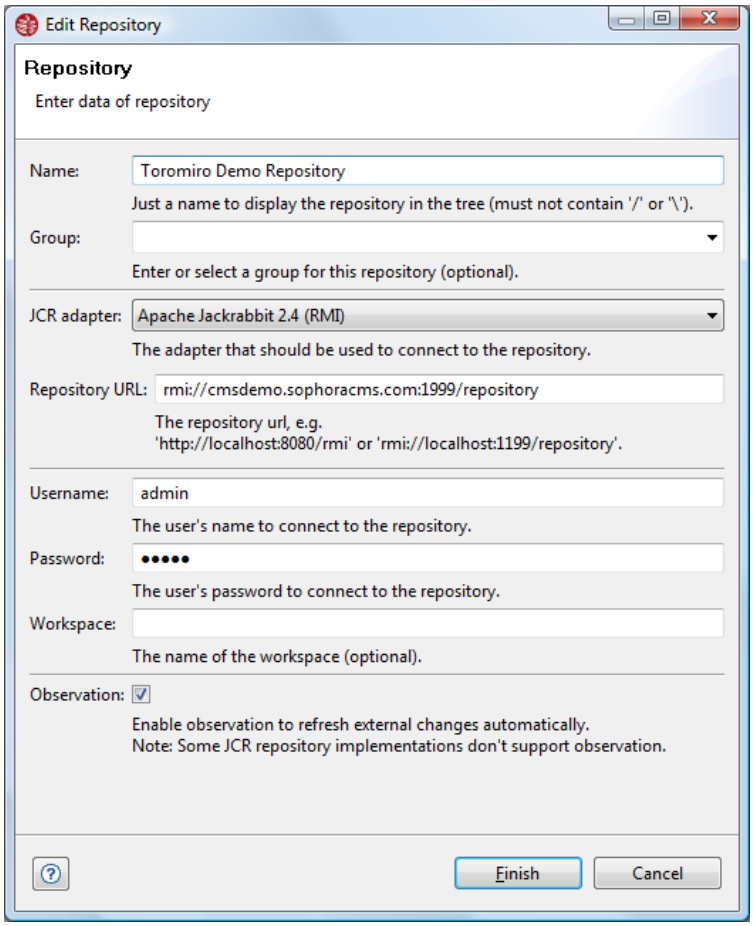
There are several add-ons available for **Toromiro**, like a special **Sophora Feature** for users of the Sophora content management system. This add-on adds search support for documents via Sophora ID to the Repository Search View. To install add-ons use **Help > Install New Software...** from the main menu and select the **Toromiro Extensions Update Site**.

1.3. First Steps

After installing and starting Toromiro you need to create a Repository Connection to an existing JCR repository. The following section describes how to connect to the Toromiro Demo Repository.

- Select **File > New > Repository** from the main menu.
- The Repository Connection dialog will pop up where you'll have to enter the following connection details:
  - **Name:** Toromiro Demo Repository
  - **Group:** Demo
  - **JCR adapter:** Apache Jackrabbit 2.20 (RMI)
  - **Repository URL:** rmi://cmsdemo.sophoracms.com:1999/repository
  - **Username:** admin
  - **Password:** admin
  - **Workspace:** <leave empty>
  - **Observation:** Checked
- To connect to the Toromiro demo repository, double-click the new entry "Toromiro Demo Repository" in the Repository Browser view.

This image shows the settings for connecting to the demo repository:



Once connected to the demo repository, try the following:

- You can browse its contents in the Repository Browser.

- You can search for documents in the Repository Search using a query. E.g. enter the following XPath query:

  `//element(*, sophora-content-nt:story)`

- You can search for documents by UUID by clicking on the UUID button in the Repository Search view. E.g. search for the UUID

  `43c945f0-97d3-42fe-9c9e-e2c92f70d9d7.`

- View node type definitions as diagrams or CND in the Repository Node Types view.

There is also help on connecting to an Apache Jackrabbit Repository.

1.4. Tutorial

This tutorial describes how to use **Toromiro**.

1.4.1. Repository Connections

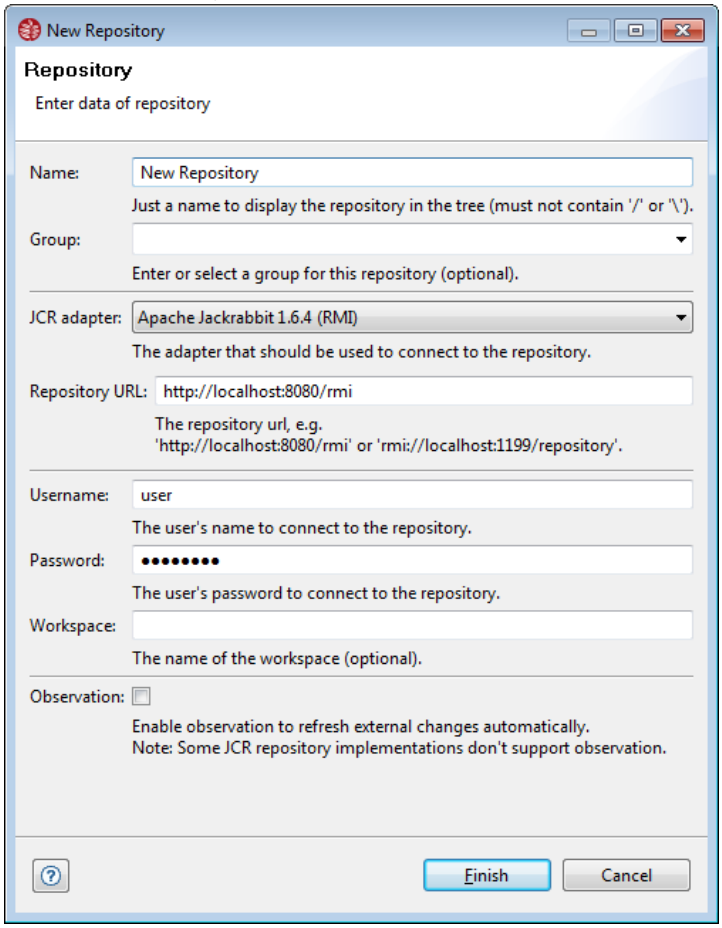
This section describes how to manage repository connections, and how to connect to/disconnect from repositories.

Creating a Repository Connection

A repository connection can be created using any of the following actions:

- In the main menu, use **File > New > Repository**.
- In the main toolbar, use **New > Repository**.
- In the Repository Browser view's toolbar, use **Add Repository** ( ).
- In the Repository Browser view's context menu, use **Add Repository** ( ).

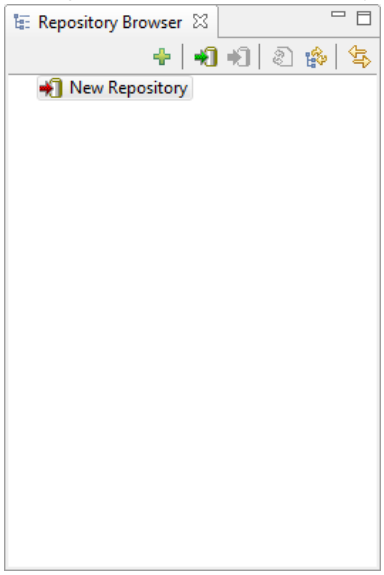
A wizard will open where you have to enter the connection's name and parameters:



Attention:
When connecting to a local repository, be sure to use the right JCR adapter and version corresponding to your JCR repository.
While some releases of JCR implementations are backwards-compatible with their earlier releases, others may contain changes that makes it impossible to downgrade to an earlier release after a repository has been upgraded to a newer version.
A full backup of the repository is recommended before connecting with Toromiro.

Note:
If the repository supports observation and you check the **Observation** checkbox all repository changes (external and local) will appear immediately in the Repository Browser View, the Node Editor and the Repository Watcher View. Otherwise you will see only the changes done by yourself and you have to refresh the Repository Browser View manually to see changes done by others.

Finishing the wizard adds the new repository connection to the Repository Browser View:



> Note:
> At this point the connection parameters, like repository URL, user name, and password, have not been checked. They will be checked each time you connect to that repository.

Import/Export Repository Connections

Repository connections can be imported and exported by using **File > Import... > Repository** or **File > Export... > Repository** or by context menu if a repository connection is selected.

Editing a Repository Connection

An existing repository connection may be edited by selecting it in the Repository Browser View and either

- clicking the **Edit Repository** button ( ) in the view's toolbar, or
- opening the context menu and selecting **Edit Repository** ( ).

The **Repository Connection Wizard** opens where you can modify the connection parameters.

Connecting to a Repository

To connect to a repository select the corresponding connection in the Repository Browser View and either

- click the **Connect to Repository** button ( ) in the view's toolbar, or
- open the context menu and select **Connect to Repository** ( ), or
- double click the repository connection.

Disconnecting from a Repository

To disconnect from a repository select it (or a content node of it) in the Repository Browser View and either

- click the **Disconnect from Repository** button ( ) in the view's toolbar, or
- open the context menu and select **Disconnect from Repository** ( ).

All opened editors relating to that repository will be closed, too.

Aborted Connections

An established repository connection may abort, e.g. if the server has been shutdown. In that case the connection state of the repository connection changes to aborted ( ) in the Repository Browser View and no more activities are possible on that repository. In the view's toolbar and context menu the **Reconnect to Repository** ( ) and **Disconnect from Repository** ( ) options are available.

Duplicating a Repository Connection

To simplify adding a new repository connection it is possible to duplicate an existing one. Select an existing connection in the Repository Browser View and either

- click the **Duplicate Repository** button ( ) in the view's toolbar, or
- open the context menu and select **Duplicate Repository** ( ).

The **Repository Connection Wizard** will open to let you modify the parameters of the new connection. Clicking the **Finish** button will create the new repository connection.

Removing a Repository Connection

To remove an existing repository connection, select it in the Repository Browser View and either

- click the **Remove Repository** button ( ) in the view's toolbar, or
- open the context menu and select **Remove Repository** ( ).

1.4.1.1. Connecting to a Jackrabbit Repository

The following sections describe how to set up and connect to an Apache Jackrabbit JCR repository.

- Connecting to a Remote Jackrabbit Repository by RMI
- Connecting to a Remote Jackrabbit Repository by WebDAV

- Connecting to a Local Jackrabbit Repository

Connecting to a Remote Jackrabbit Repository by RMI

This example shows how to connect to a remote Jackrabbit repository using RMI. In this example, we'll connect to the Jackrabbit standalone server, which can be installed as follows:

- Download the Jackrabbit Standalone jar from the Apache software foundation. This example uses the version 2.20.2.
- Rename it to jackrabbit-standalone.jar
- Run it using:

```
java -jar jackrabbit-standalone.jar
```

In Toromiro, create a repository connection with the following details:

- **Name:** Jackrabbit Standalone RMI
- **Group:** Jackrabbit
- **JCR adapter:** Apache Jackrabbit 2.20 (RMI)
- **Repository URL:** http://localhost:8080/rmi
- **Username:** admin
- **Password:** admin
- **Workspace:** <leave empty>
- **Observation:** Checked

Connecting to a Remote Jackrabbit Repository by WebDAV

This example shows how to connect to a remote Jackrabbit repository using WebDAV. In this example, we'll connect to the Jackrabbit standalone server, which can be installed as follows:

- Download the Jackrabbit Standalone jar from the Apache software foundation. This example uses the version 2.20.2.
- Rename it to jackrabbit-standalone.jar
- Run it using:

```
java -jar jackrabbit-standalone.jar
```

In Toromiro, create a repository connection with the following details:

- **Name:** Jackrabbit Standalone WebDAV
- **Group:** Jackrabbit
- **JCR adapter:** Apache Jackrabbit 2.20 (WebDAV)
- **Repository URL:** http://localhost:8080/server/
- **Username:** admin
- **Password:** admin
- **Workspace:** <leave empty>
- **Observation:** Checked

Connecting to a Local Jackrabbit Repository

Toromiro can also be used to create and edit a local Jackrabbit repository directly. Instead of configuring an URL to connect to, the directory where the repository is located / will be created has to be given, as well as the path to a *repository.xml* configuration file.

Create a directory for the repository in your file system, e.g. *C:\temp\repository*. Copy this repository.xml into the directory. In Toromiro, create a repository connection with the following details:

- **Name:** Jackrabbit Local
- **Group:** Jackrabbit
- **JCR adapter:** Apache Jackrabbit 2.20 (Local, Derby 10.14.2.0)
- **Home directory:** C:\temp\repository
- **Configuration file:** C:\temp\repository\repository.xml
- **Username:** <anything>
- **Password:** <anything>
- **Workspace:** <leave empty>
- **Observation:** Checked

When you connect to the repository for the first time, it will be created with the default Jackrabbit content in the given directory.

Database

The example above uses a Derby persistence manager which is supported by Toromiro out of the box. If your repository uses another persistence manager, e.g. a h2 database, you have to configure the JDBC driver in the *toromiro.ini* file. To do so set the property *JDBC_DRIVER* to the path of your driver jar file. Afterwards the "Apache Jackrabbit <version> (Local, Derby <version>)" JCR adapters will work with the configured JDBC driver, too.

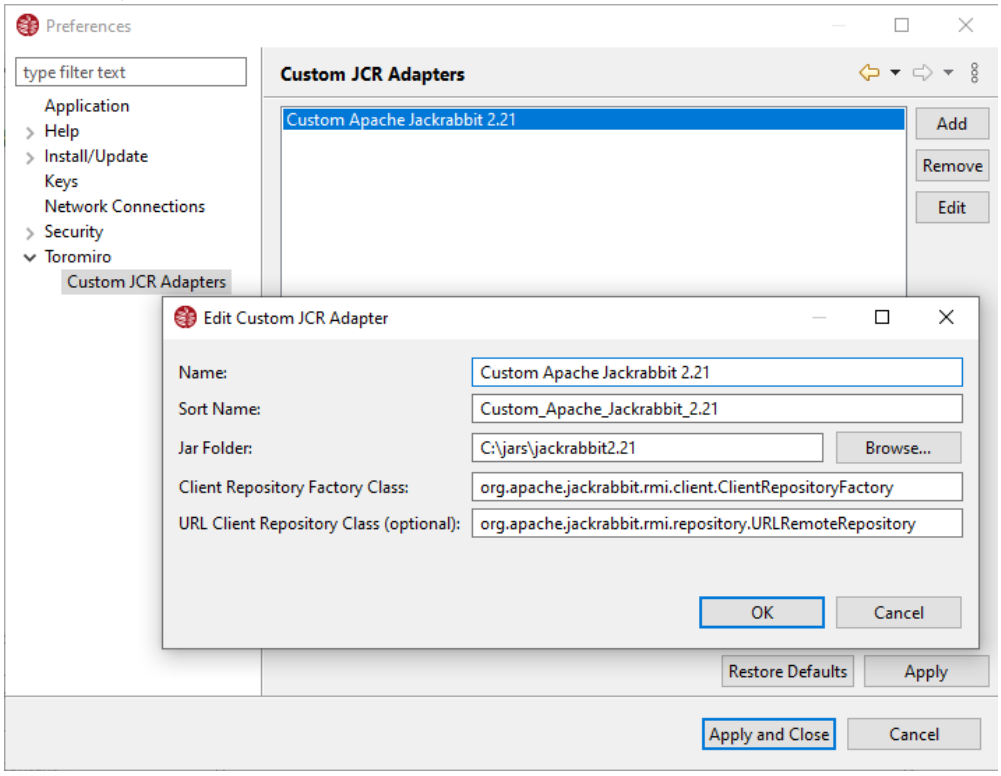
Here is an example of the *toromiro.ini* file (Windows 64 bit, h2 database). Note that *JDBC_DRIVER* is a system property and must be set after *-vmargs*. Moreover the *toromiro.ini* file must end with an empty line.

```
-startup
plugins/org.eclipse.equinox.launcher_1.5.500.v20190715-1310.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.1100.v20190907-0426
-vmargs
-Xmx1024m
-DJDBC_DRIVER=C:/path/to/h2driver/h2-1.4.180.jar
```

1.4.1.2. Custom JCR Adapters

Custom JCR Adapters can be defined for connecting to JCR repository implementations that are not supported by default.
To configure your own JCR adapter, you need the library files of the JCR implementation that are required to connect to the repository in a separate directory of your file system. Then switch to the **Custom JCR Adapters** preference page and configure a new adapter (see below). Each adapter configured on that page can be selected in

the **Repository Connection** wizard.



The following settings can be configured for an adapter:

Name:

The name of the adapter. This is the name shown in the **Repository Connection** dialog.

Sort Name:

The internal name of the adapter that is used to sort multiple adapters e.g. in the dropdown of the **Repository Connection** dialog.

Jar Folder:

The jars found in this folder will be added to the classpath of the adapter.

Client Repository Factory Class:

This is the fully qualified name of the class that will be used to create an instance of a `javax.jcr.Repository` for connecting to a JCR repository using this adapter. The class must declare the following method:

```
Repository getRepository(String url) throws Exception
```

URL Client Repository Class:

This is an optional setting. If the connection to the repository uses the HTTP protocol, this class will be used to connect to the repository instead of the *Client Repository Factory Class*. The class given in this setting must be derived from `javax.jcr.Repository` and have a constructor with the URL as a `String` parameter.

Example

Jackrabbit 2.21

Example settings for connecting to an Apache Jackrabbit 2.21 repository:

Name:

Custom Apache Jackrabbit 2.21

Sort Name:

Custom_Apache_Jackrabbit_2.21

Jar Folder:

C:\jars\jackrabbit2.21
This folder must contain the following jars:

- commons-codec-1.11.jar
- commons-collections-3.2.2.jar
- commons-dbcp-1.4.jar
- commons-io-2.6.jar
- commons-logging-1.2.jar
- commons-pool-1.5.4.jar
- concurrent-1.3.4.jar
- derby-10.14.2.0.jar
- httpclient-4.5.11.jar
- httpcore-4.4.13.jar
- httpmime-4.5.11.jar
- jackrabbit-core-2.21.0.jar
- jackrabbit-data-2.21.0.jar
- jackrabbit-jcr2spi-2.21.0.jar
- jackrabbit-jcr-commons-2.21.0.jar
- jackrabbit-jcr-rmi-2.21.0.jar
- jackrabbit-spi-2.21.0.jar
- jackrabbit-spi2dav-2.21.0.jar

- jackrabbit-spi-commons-2.21.0.jar
- jackrabbit-webdav-2.21.0.jar
- jcl-over-slf4j-1.7.29.jar
- jcr-2.0.jar
- lucene-core-3.6.0.jar
- oak-jackrabbit-api-1.24.0.jar
- slf4j-api-1.7.29.jar
- tika-core-1.23.jar

Client Repository Factory Class:

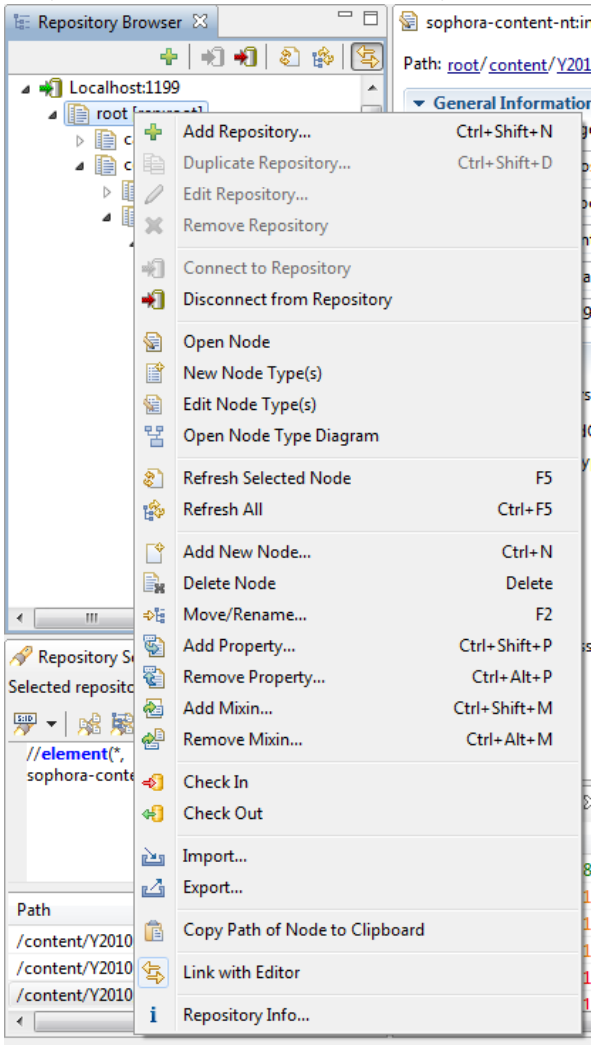
org.apache.jackrabbit.rmi.client.ClientRepositoryFactory

URL Client Repository Class:

org.apache.jackrabbit.rmi.repository.URLRemoteRepository

1.4.2. Repository Browser View

The **Repository Browser** view displays all Repository Connections. Here you may connect to a repository, browse, and edit its contents. The contents of a repository is displayed in tree form, where the root is the repository connection, and the first (and only) child is the root node of the repository.



Operations

The **Repository Browser** view has several operations to manage repository connections, connecting to/disconnecting from repositories, refreshing the node tree, and editing nodes (see above screenshot).

Repository Connections

Managing repository connections and how to connect to/disconnect from repositories is described in the Repository Connections section.

Opening Nodes

To see the properties of a node and editing them, double click on a node in the tree. The Node Editor will be opened, and the Child Nodes View will show all children of the opened node. If you have opened more than one node at the same time and the **Link with Editor** button ( ) is activated, then the tree will always select (and scroll to) the active editor's node.

To view or edit a node's type(s) open the Node Type Editor using **Edit Node Type(s)** ( ). To view a node's type hierarchy rather than editing it, open the Node Type Diagram Editor using **Open Node Type Diagram** ( ).

Editing Nodes

New nodes may be created ( ) or existing nodes deleted ( ) within the **Repository Browser** view. New nodes will be created as children of the selected node in the tree. You may also add new properties ( ) or mixins ( ) to the selected node, or remove them ,). Nodes may also be renamed or moved ( ).

Refreshing the Tree

To refresh part of the node tree, select a node and use **Refresh Selected Node** ( ). To refresh the whole repository view use **Refresh All** ( ). In most cases these operations are not necessary.

Copying A Node's Path to the Clipboard

A node's path may copied to the clipboard using **Copy Path of Node to Clipboard** ( ).

Versions of Nodes

Versions of nodes can be created by using the **Check In** ( ) and **Check Out** ( ) actions.

Import/Export

To import a node (and it's child nodes) from a XML file in the file system select the (new) parent node of the node to import and use **Import** ( ).
To export a node (and it's child nodes) as a XML file in the file system select the node you want to export and use **Export** ( ). Each operation opens a wizard that guides you through the import/export process.

The import offers several ways to handle the UUIDs of incoming nodes:

- **Always create new nodes:** Incoming referenceable nodes are added in the same way that a new node is added. That is, they are assigned newly created UUIDs, so UUID collisions will not occur.
- **Remove existing nodes with same UUIDs:** If an incoming referenceable node has the same UUID as a node already existing in the workspace then the already existing node (and its subtree) is removed from wherever it may be in the workspace before the incoming node is added. Note that this can result in nodes "disappearing" from locations in the workspace that are remote from the location to which the incoming subtree is being written.
- **Replace existing nodes with same UUIDs:** If an incoming referenceable node has the same UUID as a node already existing in the workspace, then the already existing node is replaced by the incoming node in the same position as the existing node. Note that this may result in the incoming subtree being disaggregated and "spread around" to different locations in the workspace. In the most extreme case this behavior may result in no node at all being added as child of the parent node. This will occur if the topmost element of the incoming XML has the same UUID as an existing node elsewhere in the workspace.
- **Do nothing (show error message):** If an incoming referenceable node has the same UUID as a node already existing in the workspace then an error occurs.

The export provides two different XML formats:

- **System View:** Provides a complete serialization of workspace content to XML without loss of information.
- **Document View:** More human-readable than the system view, though it achieves this at the expense of completeness.

See Java Specification Request 170 (JSR 170), Chapter 6.4 "XML Mappings" for further information.

Getting Repository Information

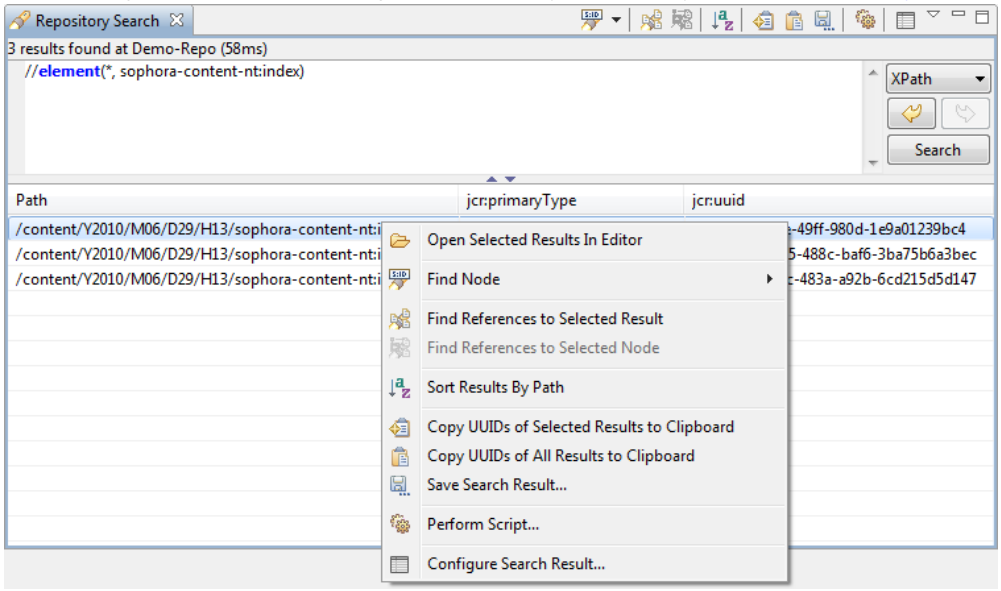
To get some information about the repository use **Repository Info...** (i)

View Configuration

The **Repository Browser** view may be configured in the Preferences. Repository URL display may be switched on or off, as well as automatic sorting of child nodes.

1.4.3. Repository Search View

The **Repository Search** view offers searching within the repository that is currently selected in the Repository Browser View or Child Nodes View.



Searching

There are different ways to search for nodes in a repository:

- Search by query
- Search by UUID
- Search by reference
- Extended searches

Search By Query

The most flexible way to search for nodes is to enter a query in the search text field and hit **CTRL + Enter** or click the **Search** button. Several query languages are supported, dependent on the underlying repository. See Query Languages for further information. Each query will be added to the history to allow searching with previous queries. The size of the query history may be changed in the Preferences.

The search text field displays the entered query with syntax highlighting (dependent on the selected query language) and auto completes bracket pairs when typing an opening bracket.
An error annotation in the upper left corner of the query text field signals whether the entered query has an invalid syntax. The tooltip of the annotation shows further information about the reason.
Instead of typing the whole query literals like "element" or "SELECT" you can enter only the first characters and then press **CTRL + Space** to open content assist. The assist makes several proposals that you can choose from. Moreover the content assist works for name space names, node type names, property names and child node names of the selected repository, too. The proposals are not case sensitive. You may also use "*" (to match multiple characters) or "?" (to match a single character).

Example:
Imagine you want to use the content assist for "jcr:nodeTypes". You can type...

- "jcr:nodetypes" + **CTRL + Space** (case insensitive)
- "jcr:nodet" + **CTRL + Space** (prefix only)
- "jc*types" + **CTRL + Space** (using *)
- "jcr:n???types" + **CTRL + Space** (using **?**)

Search By UUID

If you already know the UUID of a node you may search for that node by entering its UUID in the **Find Node By UUID** operation ( ). Each UUID will be added to the history to allow searching with a previous UUIDs. The size of the UUID history may be changed in the Preferences.

Search By Reference

For each node in the Repository Browser View, the Child Nodes View or in the result list you can search for nodes referencing that node. To do that select the node in the Repository Browser View or Child Nodes View and use **Find References to Selected Node** ( ), or select a previously found node in the result list and use **Find References to Selected Result** ( ).

Extended Searches

The **Repository Search** view is extendable by third party plugins to allow for more specialized searches. These searches appear next to **Find Node By UUID**.

Result List

The search result is located below the search text field. Here you may sort the result by path ( ), or open nodes ( ) in the Node Editor by double clicking them.
By default the result list displays only the path of the found nodes. With **Configure Search Result** ( ) you are able to add additional properties of these nodes to display them in the search result directly.
To save the results use **Save Search Result** ( ). The results will be saved as CSV. A dialog appears to configure the format and target.
If you are only interested in the UUIDs of the nodes, you may use **Copy UUIDs of Selected Results to Clipboard** ( ) or **Copy UUIDs of All Results to Clipboard** ( ) to copy them to the clipboard.

Modify Nodes

To manipulate the nodes of a search result see Modify Nodes by Script.

1.4.3.1. Query Languages

Dependent on the underlying repository the different query languages XPath, SQL and JCR-SQL2 (since JCR 2.0) are supported. Supposed the repository contains a node type for stories like the following and each of its nodes has the JCR mixin 'mix:referenceable', here are some example queries.

Example Node Type

```
<'nt'='http://www.jcp.org/jcr/nt/1.0'>
<'my-content'='http://www.subshell.com/toromiro/my-content/1.0'>
<'my-content-nt'='http://www.subshell.com/toromiro/my-content-nt/1.0'>

['my-content-nt:story'] > nt:base
  - 'my-content:author' (string)
  - 'my-content:title' (string)
  - 'my-content:text' (string)
  - 'my-content:relatedStories' (reference) multiple
  - 'my-content:creationDate' (date)
  - 'my-content:link' (string)
```

Example Queries

Query	XPath	JCR-SQL2 (since JCR 2.0)
Get all nodes	//*	SELECT * FROM [nt:base] SELECT * FROM [rep:root] or
Get the JCR root node (/)	/jcr:root	SELECT * FROM [nt:base] WHERE ISSAMENODE('/') or SELECT * FROM [nt:base] AS node WHERE ISSAMENODE(node, '/')
Get the node /content	/jcr:root/content	SELECT * FROM [nt:base] WHERE ISSAMENODE('/content') or SELECT * FROM [nt:base] AS node WHERE ISSAMENODE(node, '/content') or if and only if at most one node named 'content' exists in repository SELECT * FROM [nt:base] WHERE NAME()='content'
Find all story nodes in /content	/jcr:root/content/element(*, my-content-nt:story)	SELECT * FROM [my-content-nt:story] WHERE ISCHILDNODE('/content') or SELECT * FROM [my-content-nt:story] AS story WHERE ISCHILDNODE(story, '/content')
Find all story nodes in /content and all its child nodes	/jcr:root/content//element(*, my-content-nt:story)	SELECT * FROM [my-content-nt:story] WHERE ISDESCENDANTNODE('/content') or SELECT * FROM [my-content-nt:story] AS story WHERE ISDESCENDANTNODE(story, '/content')
Get all nodes named X in /content and all its child nodes	/jcr:root/content//X	SELECT * FROM [nt:base] WHERE ISDESCENDANTNODE('/content') AND NAME()='X'
Find all story nodes	//element(*, my-content-nt:story)	SELECT * FROM [my-content-nt:story]

Query	XPath	JCR-SQL2 (since JCR 2.0)
Find all story nodes created by 'Homer'	//element(*, my-content-nt:story)[@my-content:author='Homer']	SELECT * FROM [my-content-nt:story] WHERE [my-content-nt:story].'my-content:author'='Homer' or SELECT * FROM [my-content-nt:story] AS stories WHERE stories.'my-content:author'='Homer'
Find all nodes created by 'Homer'	//element(*)[@my-content:author='Homer']	SELECT * FROM [nt:base] WHERE [nt:base].'my-content:author'='Homer' or SELECT * FROM [nt:base] AS nodes WHERE nodes.'my-content:author'='Homer'
Find all story nodes created by 'Homer' and Title is 'Good News'	//element(*, my-content-nt:story)[@my-content:author='Homer' and @my-content:title='Good News']	SELECT * FROM [my-content-nt:story] WHERE [my-content-nt:story].'my-content:author'='Homer' AND [my-content-nt:story].'my-content:title'='Good News' or SELECT * FROM [my-content-nt:story] AS stories WHERE stories.'my-content:author'='Homer' AND stories.'my-content:title'='Good News'
Find all story nodes created by 'Homer' or Title is 'Good News'	//element(*, my-content-nt:story)[@my-content:author='Homer' or @my-content:title='Good News']	SELECT * FROM [my-content-nt:story] WHERE [my-content-nt:story].'my-content:author'='Homer' OR [my-content-nt:story].'my-content:title'='Good News' or SELECT * FROM [my-content-nt:story] AS stories WHERE stories.'my-content:author'='Homer' OR stories.'my-content:title'='Good News'
Find all nodes with a title	//*[@my-content:title]	SELECT * FROM [nt:base] WHERE [nt:base].'my-content:title' IS NOT NULL or SELECT * FROM [nt:base] AS nodes WHERE nodes.'my-content:title' IS NOT NULL
Find all story nodes with a title	//element(*, my-content-nt:story)[@my-content:title]	SELECT * FROM [my-content-nt:story] WHERE [my-content-nt:story].'my-content:title' IS NOT NULL or SELECT * FROM [my-content-nt:story] AS stories WHERE stories.'my-content:title' IS NOT NULL
Find all nodes with UUID 'ff554f42-3a36-41d8-aea2-87b4b74561d6'	//*[@jcr:uuid='ff554f42-3a36-41d8-aea2-87b4b74561d6']	SELECT * FROM [nt:base] WHERE [nt:base].'jcr:uuid'='ff554f42-3a36-41d8-aea2-87b4b74561d6' or SELECT * FROM [nt:base] AS nodes WHERE nodes.'jcr:uuid'='ff554f42-3a36-41d8-aea2-87b4b74561d6'
Find all story nodes that refer to the story with UUID 'ff554f42-3a36-41d8-aea2-87b4b74561d6'	//element(*, my-content-nt:story)[@my-content:relatedStories='ff554f42-3a36-41d8-aea2-87b4b74561d6']	SELECT * FROM [my-content-nt:story] WHERE [my-content-nt:story].'my-content:relatedStories'='ff554f42-3a36-41d8-aea2-87b4b74561d6' or SELECT * FROM [my-content-nt:story] AS stories WHERE stories.'my-content:relatedStories'='ff554f42-3a36-41d8-aea2-87b4b74561d6'
Find all story nodes, in descending title order	//element(*, my-content-nt:story) order by @my-content:title descending	SELECT * FROM [my-content-nt:story] ORDER BY 'my-content:title' desc
Find all nodes with a text containing the value 'News'	//*[jcr:contains(@my-content:text, 'News')]	SELECT * FROM [nt:base] WHERE CONTAINS('my-content:text', 'News')
Find all nodes with a link to 'http://www.mypage.com/'	//*[jcr:like(@my-content:link,'http://www.mypage.com/%')]	SELECT * FROM [nt:base] AS nodes WHERE nodes.'my-content:link' LIKE 'http://www.mypage.com/%'
Find all story nodes created after 2011-07-10 11:00:00	//element(*, my-content-nt:story)[@my-content:creationDate > xs:dateTime('2011-07-10T11:00:00.000Z')]	SELECT * FROM [my-content-nt:story] AS stories WHERE stories.'my-content:creationDate' > cast('+2011-07-10T11:00:00.000Z' as date)

1.4.3.2. Modify Nodes by Script

The tool bar and context menu of the Repository Search View provides a small but very powerful entry: **Perform Script** ( ).
Using this option the nodes of a search result can be modified by a Groovy script. The script has to contain a class that implements the interface *com.subshell.toromiro.scripts.IToromiroScript* or just extends its default implementation *com.subshell.toromiro.scripts.ToromiroScript* which holds the JCR session and script context, e.g. for logging. The implemented class and the script file (**.groovy*) must have the same name. To use external libraries within the script you can select a directory that contains all required **.jar* files in the **Perform Script** dialog.

> Attention:
> Running an incorrect script may destroy your repository. Make a full backup of the repository before performing the script and always know what you do!

Example Script

The following (full specified) example script sets a property of each node to a preset value.

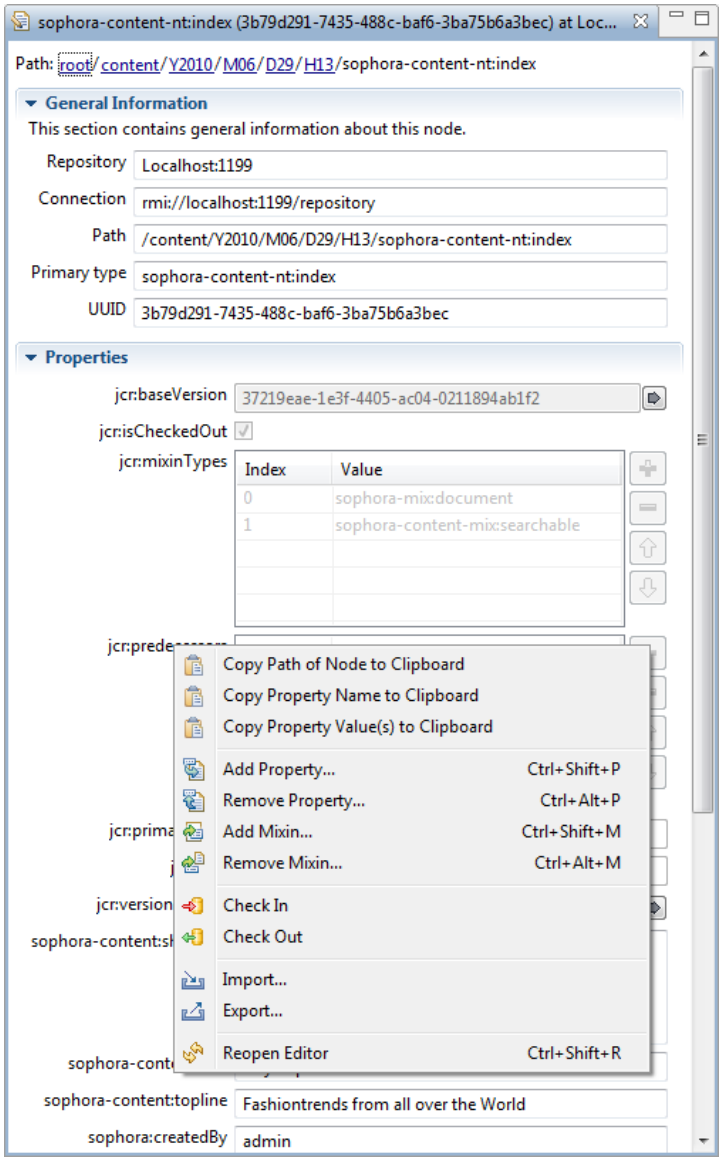
ExampleToromiroScript.groovy:

```
import javax.jcr.*
import com.subshell.toromiro.scripts.*

class ExampleToromiroScript extends ToromiroScript {

        /**
         * Initializes this script.
         * This method is called once before this script is performed
         * on each node.
         * Implementations should hold the given Session and
         * IScriptContext and may initialize global requirements
```

```
	 * of this script. Never close the given session in this script.
	 * @param session the session
	 * @param context the context of this script
	 * @see #perform(Node)
	 */
	public void initialize(Session session, IScriptContext context) {
		super.initialize(session, context) // Holds the given session and script context and logs the method call only

		// Initialize additional requirements of this script
	}

	/**
	 * Performs this script on the given Node.
	 * This method is called for each node after this script has
	 * been initialized.
	 * If the implementation consists of several long taking operations it has
	 * to ask the IScriptContext whether the script has been canceled
	 * by the use or not.
	 * @param node the node
	 * @see #initialize(Session)
	 */
	public void perform(Node node) throws Exception {
		super.perform(node) // Logs the method call for the given node only

		// Example code snippet:

		// How to get the cancel state:
		IScriptContext context = getContext()
		if (!context.isCanceled()) {
			// How to log a message:
			context.log("Setting property 'my-content:title'")

			// How to modify the given node:
			Value value = getSession().getValueFactory().createValue("This is an example")
			node.setProperty("my-content:title", value)

			// How to log an exception:
			context.log(new Exception("Logging an exception"))
		}
	}

	/**
	 * Finishes this script.
	 * This method is called once after this script has been performed
	 * on all nodes or when it has been cancelled by the user.
	 * Implementations have to ask the IScriptContext whether
	 * the script has been canceled or not to save the session and/or
	 * clean up.
	 * @see #perform(Node)
	 */
	public void done() throws Exception {
		super.done() // Logs the method call only

		IScriptContext context = getContext()
		if (context.isCanceled()) {
			// May revert changes
		} else {
			// Save the session
			session.save()
		}

		// Clean up additional requirements of this script
	}

}
```

1.4.4. Node Editor

The **Node Editor** shows all properties of a node. Here you can edit the values of its properties. You may also add and remove properties and mixins, or copy names and values of properties to the clipboard for other purposes.



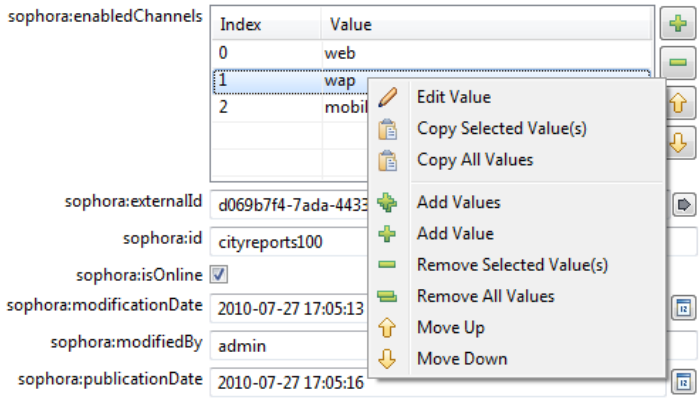
Note:
Adding or removing of properties or mixins will immediately save the node. This is in contrast to changing the value of a property, where you need to save your changes manually.

Grouping Properties

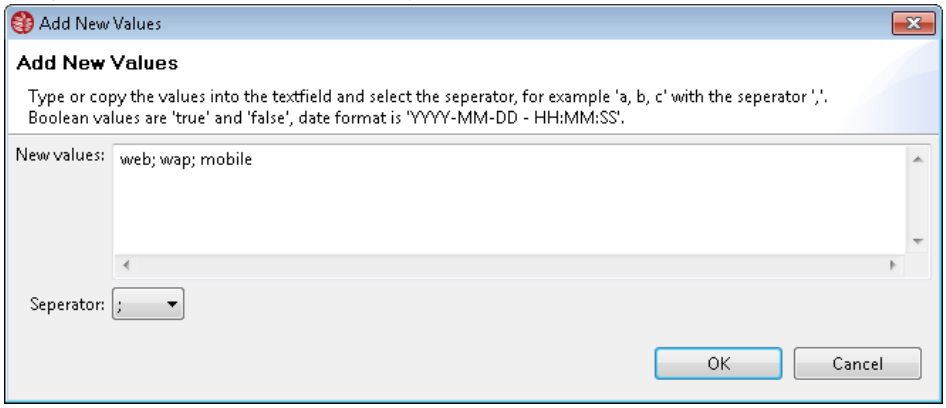
The **Node Editor** displays some general information about the node in a group on top of the editor and all its properties in another group below. This default behavior can be changed in the Preferences to group the properties of the node e.g. by namespace or declaring node type. The General Information group will always be placed on top of the editor.

Editing Properties

Depending on the property type there is another input field in the **Node Editor**. There are several input fields for single properties, like simple text fields, check boxes or date fields. Multiple properties are shown in a table that offers functions for editing, sorting, adding and removing one or more values of the property.



Adding more than one value to a multiple property at the same time:



Handling External Node Changes

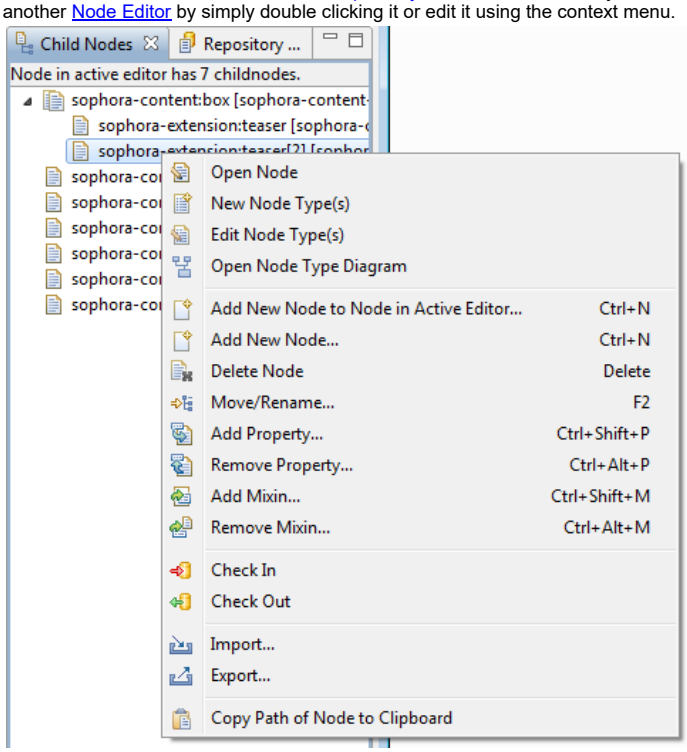
If a node is opened in an editor and this node is modified outside of this editor, the editor will become read-only. This behavior can be changed in the Preferences to automatically reopen or close the editor when a node is modified externally.

Editor Tab Configuration

The editor tab shows the primary name of the node, its UUID, and the name of the repository connection the node belongs to by default. The name of the repository connection may be switched off in the Preferences. You may also change the property whose value should appear in the tab. **Make sure to only configure properties that will always contain short strings.**

1.4.5. Child Nodes View

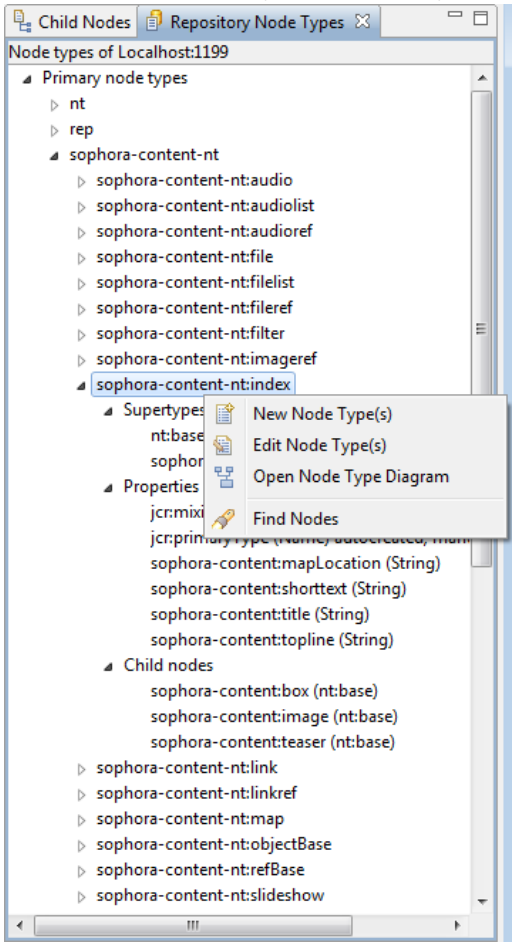
The **Child Nodes** view is similar to the Repository Browser View, but only shows all children of the node in the active Node Editor. Here you can open a child node in another Node Editor by simply double clicking it or edit it using the context menu.



1.4.6. Repository Node Types View

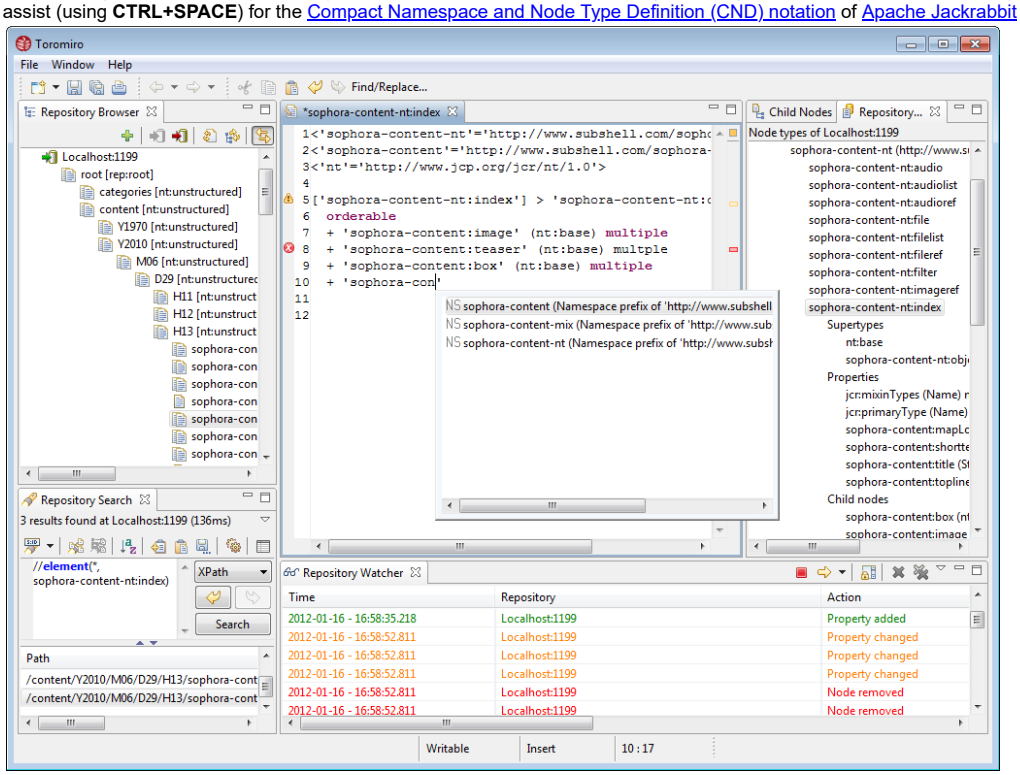
The **Repository Node Types** view displays all node types of the repository selected in the Repository Browser View. The node types are grouped by primary types, mixin types, and their name spaces. Within a node type you can see all its declared supertypes, properties and child nodes. By context menu you can open the Node Type Editor or Node Type Diagram Editor. Double clicking **Primary Node Types**, **Mixin Types**, a name space, or a node type will open the Node Type Diagram Editor, containing the selected node type(s).

To find all nodes of a specified type select the node type and use **Find Nodes** ( ) from context menu. The result will be shown in Repository Search View.



1.4.7. Node Type Editor

The **Node Type Editor** displays node types in a text representation for creating or editing. It supports syntax validation, syntax highlighting, auto completion and context assist (using **CTRL+SPACE**) for the Compact Namespace and Node Type Definition (CND) notation of Apache Jackrabbit JCR repositories.



Note:
Not every JCR adapter supports creating or editing existing node types. See overview below.

Opening Node Type Editor

To create or edit node types go to the context menu of Repository Browser View, Repository Node Types View or Node Type Diagram Editor and use **New Node Type(s)** ( ) or **Edit Node Type(s)** ( )

1.4.7.1. Jackrabbit Node Type Editing Support Overview

The Jackrabbit adapters for **Toromiro** support node type creation and editing in the following ways:

JCR Adapter	View Existing Node Types	Create New Node Types	Edit Existing Node Types
Apache Jackrabbit 2.14 (Local, Derby 10.12.1.1)	✚	✚	✚
Apache Jackrabbit 2.14 (RMI)	✖	✖	✖
Apache Jackrabbit 2.14 (WebDAV)	✚	✚	✚
Apache Jackrabbit 2.16 (Local, Derby 10.14.2.0)	✚	✚	✚
Apache Jackrabbit 2.16 (RMI)	✖	✖	✖
Apache Jackrabbit 2.16 (WebDAV)	✚	✚	✚
Apache Jackrabbit 2.18 (Local, Derby 10.14.2.0)	✚	✚	✚
Apache Jackrabbit 2.18 (RMI)	✖	✖	✖
Apache Jackrabbit 2.18 (WebDAV)	✚	✚	✚
Apache Jackrabbit 2.20 (Local, Derby 10.14.2.0)	✚	✚	✚
Apache Jackrabbit 2.20 (RMI)	✖	✖	✖
Apache Jackrabbit 2.20 (WebDAV)	✚	✚	✚

✚ Supported
⚠ Partially supported (only on HTTP connections)
✖ Not supported

1.4.7.2. Custom JCR Adapter Node Type Editing Support Overview

The custom JCR adapters for **Toromiro** support node type creation and editing in the following ways:

JCR Adapter	View Existing Node Types	Create New Node Types	Edit Existing Node Types
Any Custom JCR Adapter	✖	✖	✖

✚ Supported
⚠ Partially supported (only on HTTP connections)
✖ Not supported

1.4.8. Node Type Diagram Editor

The **Node Type Diagram Editor** displays node types in an UML-style diagram. Here you can see the hierarchical structure of node types with supertypes and subtypes, the types of child nodes and referenced nodes.

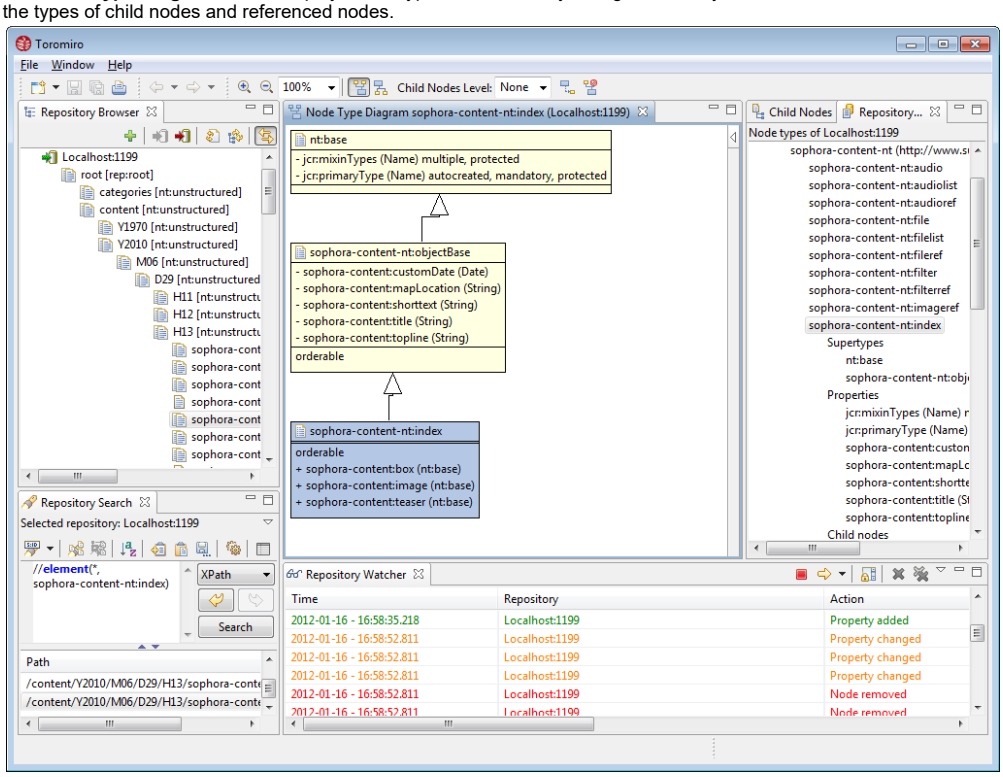


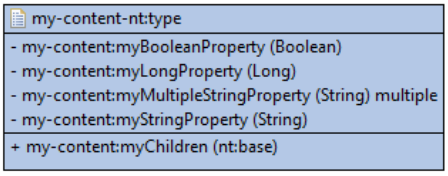
Diagram Types

There are two kinds of how to create a node type diagram: Based on **node types** and based on **nodes**.
If the diagram has been opened on a node type, e.g. out of the Repository Node Types View, the definition of that type is the base of the diagram. That means all declared supertypes, subtypes, reference properties and child nodes are analysed to create the diagram.
On the other hand, if the diagram has been opened on an existing node, e.g. from Repository Browser View, that node is the base of the diagram. In this case the primary type of the node, the existing referenced nodes and existing child nodes are shown in the diagram. The mixin types of the selected node are also displayed.

Diagram Configuration

The **Node Type Diagram Editor** displays only the selected node type(s) by default and can be configured to show other related types. The node type(s) the diagram has opened for are marked in blue, while the related types are marked in yellow. Each node type is displayed as rectangle containing its name, properties and child node declaration.

Example: Node Type



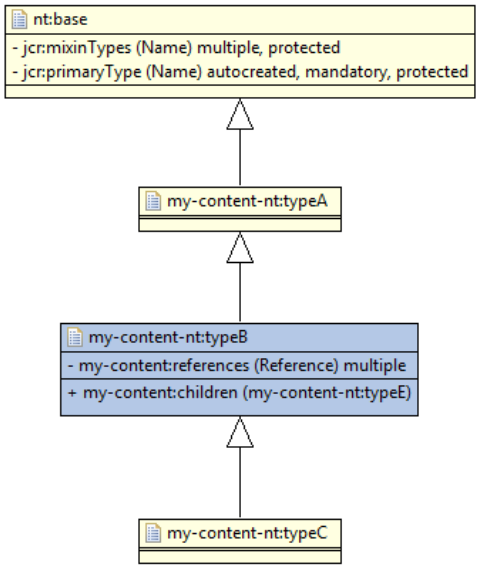
The following view settings are dependent on which editor currently has focus.

Hierarchical Structure

To see the hierarchical structure, you may switch to

- supertypes by clicking the **Show Supertype(s)** button in the main toolbar ( ), or
- subtypes by clicking the **Show Subtype(s)** button in the main toolbar ( ).

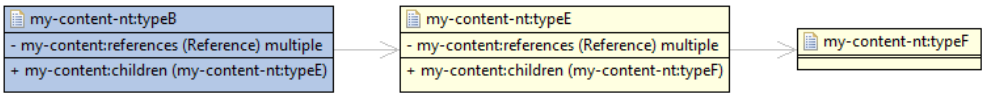
Example: Supertypes and Subtypes



Child Node Types

If you are interested in the types of the child nodes configure the deep of child nodes to respect in the **Child Nodes Level** drop down menu of the toolbar. Next to the predefined levels (**None, 1, 2, 3, All**) it is possible to enter custom levels, like **5**.

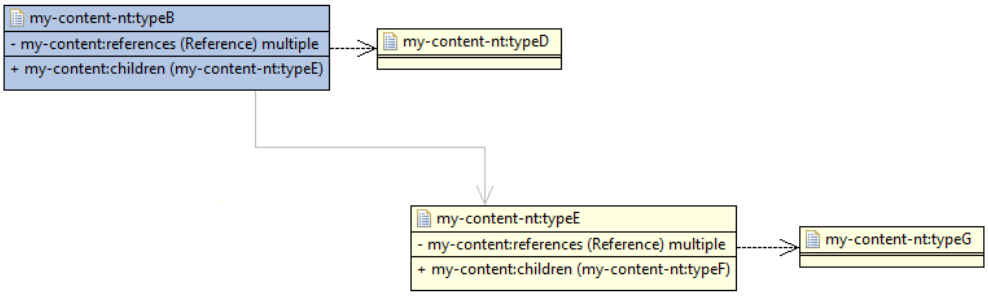
Example: Child Nodes Level = 2



Referenced Types

The types referenced by the selected types (and their childnodes) can be displayed using the **Show Referenced Types** button in the main toolbar ( ).

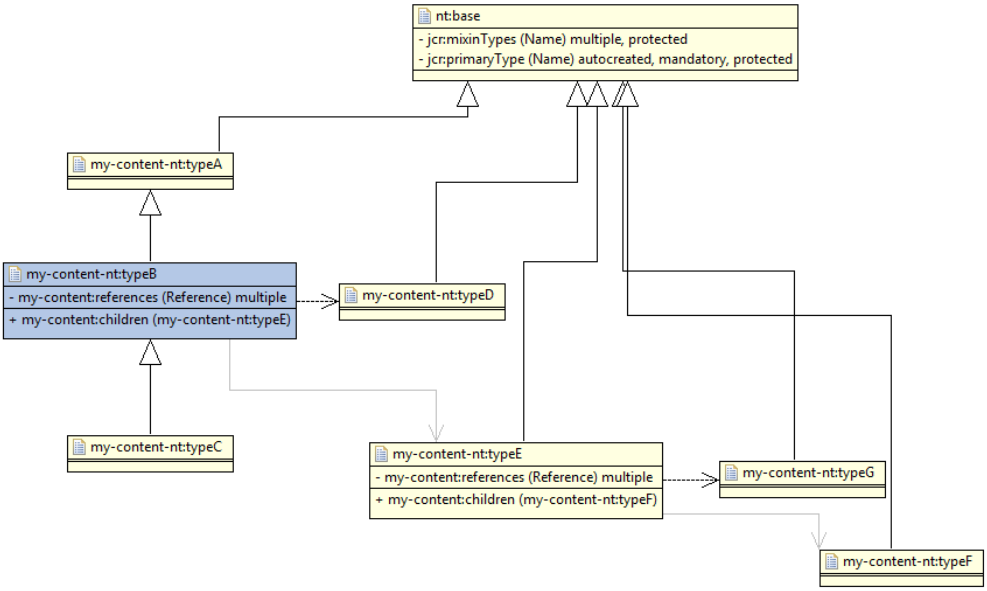
Example: Referenced Types (with Child Nodes Level = 1)



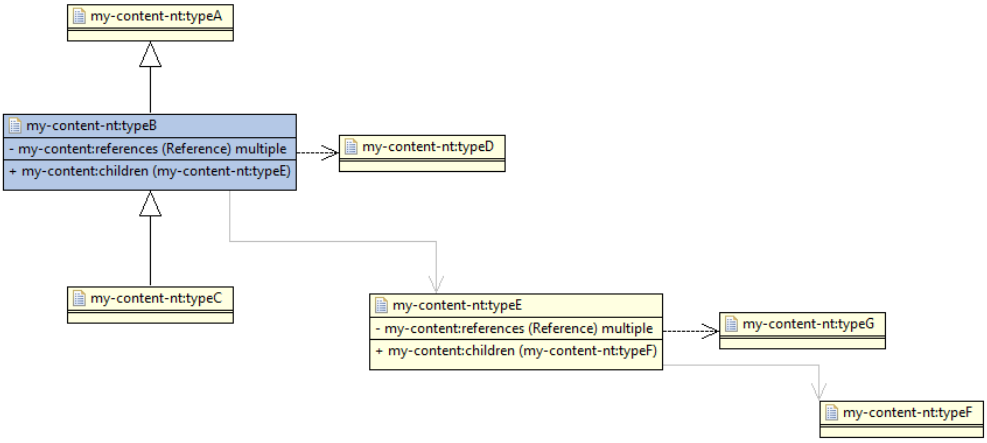
Type Filter

To ignore certain types in the diagram a type filter can be configured by **Configure Type Filter** in the toolbar ( ). The filtered types and all types only related to the filtered types will not appear in the diagram. This is useful if the diagram is confusing based on too many unnecessary connections to the same set of types. E.g. if the hierachy of node types has to be shown in the diagram but in the end all node types inherit from the same well known type "nt:base". In this case the type "nt:base" can be filtered to hide it in the diagram.

Example: No Filter



Example: Filter for "nt:base"



Connections

The routing of the connections can be changed in the editor's context menu. The following connection routers are available:

- **Direct:** Routes the connections straight from one node type to another
- **Orthogonal:** Provides connections with an orthogonal route between the node types
- **Shortest Path:** Routes the connections straight from one node type to another, avoiding other node types

Zooming

The **Node Type Diagram Editor** supports zooming the view. To zoom, either click the buttons in the main toolbar, use the editor's context menu, or use **Ctrl+Mouse Wheel**.

Screenshots

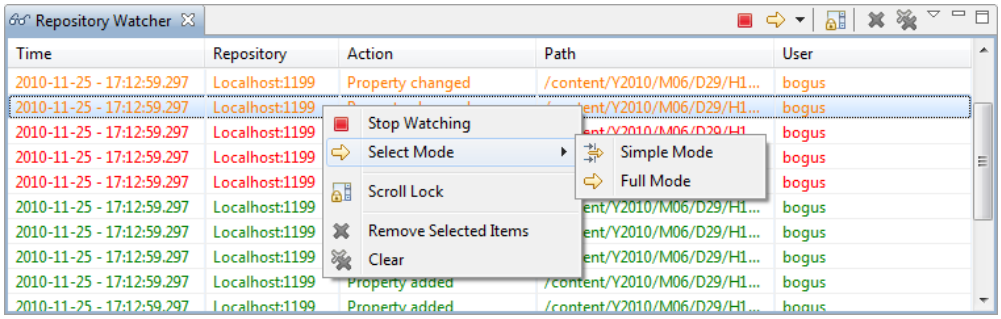
To export a node type diagram as an image file use **File > Save As...** and select an image file extension like **.bmp*, **.jpg* or **.png*.

Export as XML

To export a node type diagram as an XML file that describes the graph use **File > Save As...** and select the **.xgml* file extension. The resulting **.xgml* file can be opened by other applications like yEd Graph Editor.

1.4.9. Repository Watcher View

The **Repository Watcher** view shows all node changes in each repository connected to. This allows for monitoring which nodes are added or removed and which properties are modified.



Note:
The repositories have to support observation and the Repository Connections must have observation enabled to see external changes.

Starting And Stopping Watching

To start watching, connect to at least one repository in the Repository Browser View. Then click the **Start Watching** button ( ). Each event from the connected repositories will be displayed in the view. To stop watching, click the **Stop Watching** button ( ).

Watching Modes

There are two different watching modes that you can select:

- **Full Mode** ( ): Shows each event, e.g. which properties have been changed on a node.
- **Simple Mode** ( ): Combines several events, e.g. any properties have been changed on a node.

Auto Scrolling

By default the events list is scrolled automatically down to the newest event. To stop auto scrolling use **Scroll Lock** ( ) in the view's toolbar or from the context menu.

> Note:
> The events list shows a maximum of 25 entries by default. This may be changed in the Preferences.

Cleaning Up the Events List

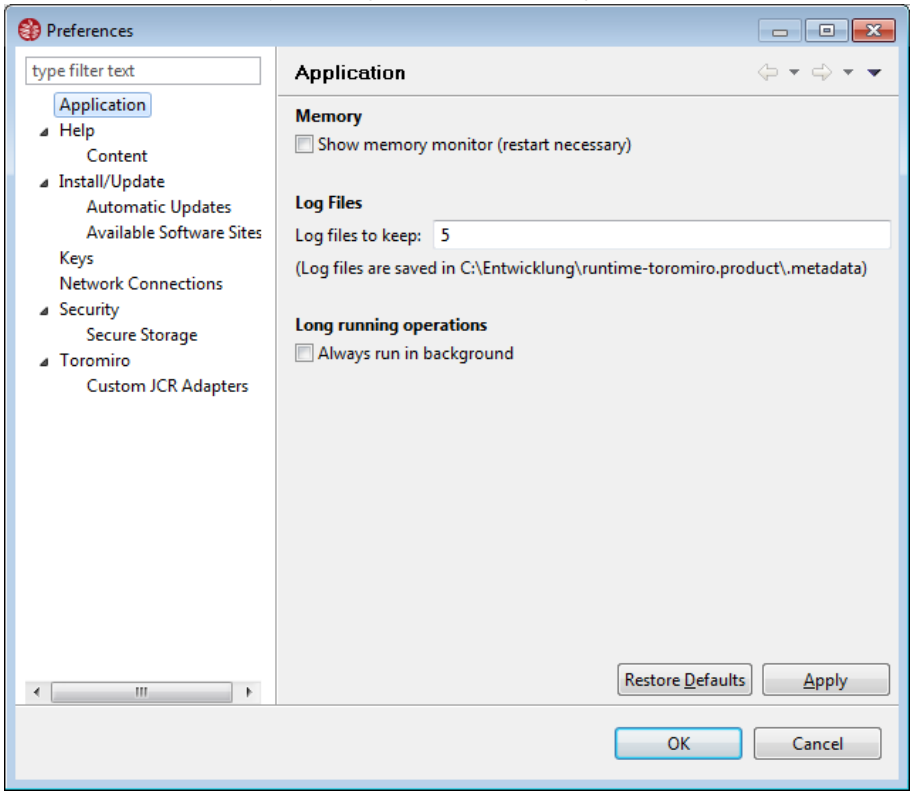
To remove unwanted events from the table, select the events and use **Remove Selected Items** ( ). To clear the whole list use **Clear** ( ) in the view's toolbar or from the context menu.

1.4.10. Preferences

The **Preferences** are available in the **File** menu of the application. Here you may change various settings that control **Toromiro**.

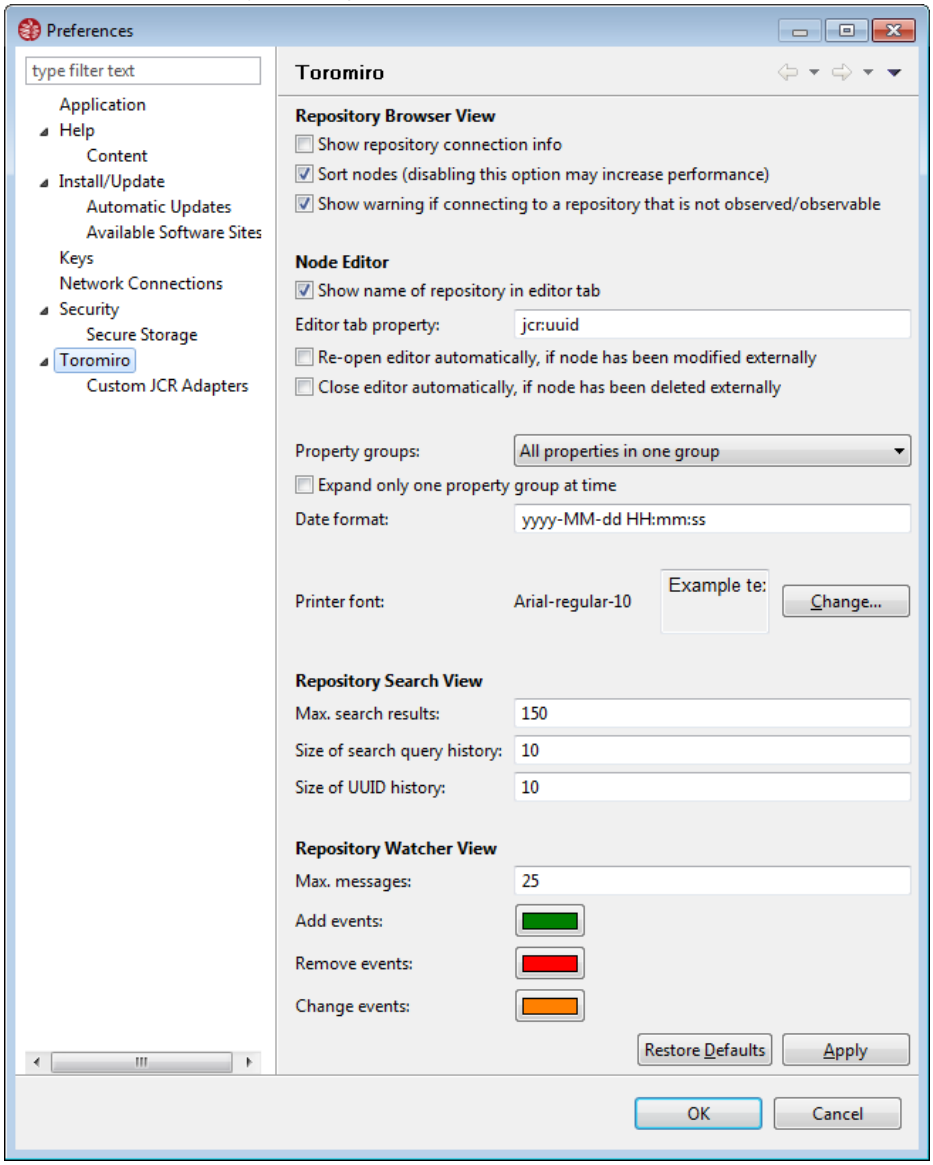
Application Preferences

The **Application** section allows you to change basic application settings.



Toromiro Preferences

The **Toromiro** section allows you to change the different views and editors of **Toromiro**.



Date and Time Patterns

Date and time formats are specified by *date and time pattern* strings. Within date and time pattern strings, unquoted letters from 'A' to 'Z' and from 'a' to 'z' are interpreted as pattern letters representing the components of a date or time string. Text can be escaped using single quotes (') to avoid interpretation. "'" represents a single quote. All other characters are not interpreted. They are simply copied into the output string during formatting or matched against the input string during parsing.

The following pattern letters are defined (all other characters from 'A' to 'Z' and from 'a' to 'z' are reserved):

Letter	Date or Time Component	Examples
G	Era designator	AD
y	Year	1996; 96
M	Month in year	July; Jul; 07
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day in week	Tuesday; Tue
a	Am/pm marker	PM
H	Hour in day (0-23)	0
k	Hour in day (1-24)	24
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978
z	Time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	-0800

Examples

The following examples show how date and time patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Date and Time Pattern	Result
`"yyyy.MM.dd G 'at' HH:mm:ss z"`	`2001.07.04 AD at 12:08:56 PDT`
`"EEE, MMM d, ''yy"`	`Wed, Jul 4, '01`
`"h:mm a"`	`12:08 PM`
`"hh 'o''clock' a, zzzz"`	`12 o'clock PM, Pacific Daylight Time`
`"K:mm a, z"`	`0:08 PM, PDT`

```
"yyyyy.MMMMM.dd GGG hh:mm aaa" 02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"   Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"                010704120856-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSZ"   2001-07-04T12:08:56.235-0700
```

1.5. Tips And Tricks

This section contains some tips and tricks that make your daily work with **Toromiro** easier.

Getting Help

In any view, editor, wizard and preference section you can get a quick help by pressing **F1**.

Duplication Repository Connections

To simplify adding a new repository connection it is possible to duplicate an existing one. Select an existing connection in the Repository Browser View and either

- click the **Duplicate Repository** button ( ) in the view's toolbar, or
- open the context menu and select **Duplicate Repository** ( ).

The **Repository Connection Wizard** will open to let you modify the parameters of the new connection. Clicking the **Finish** button will create the new repository connection.

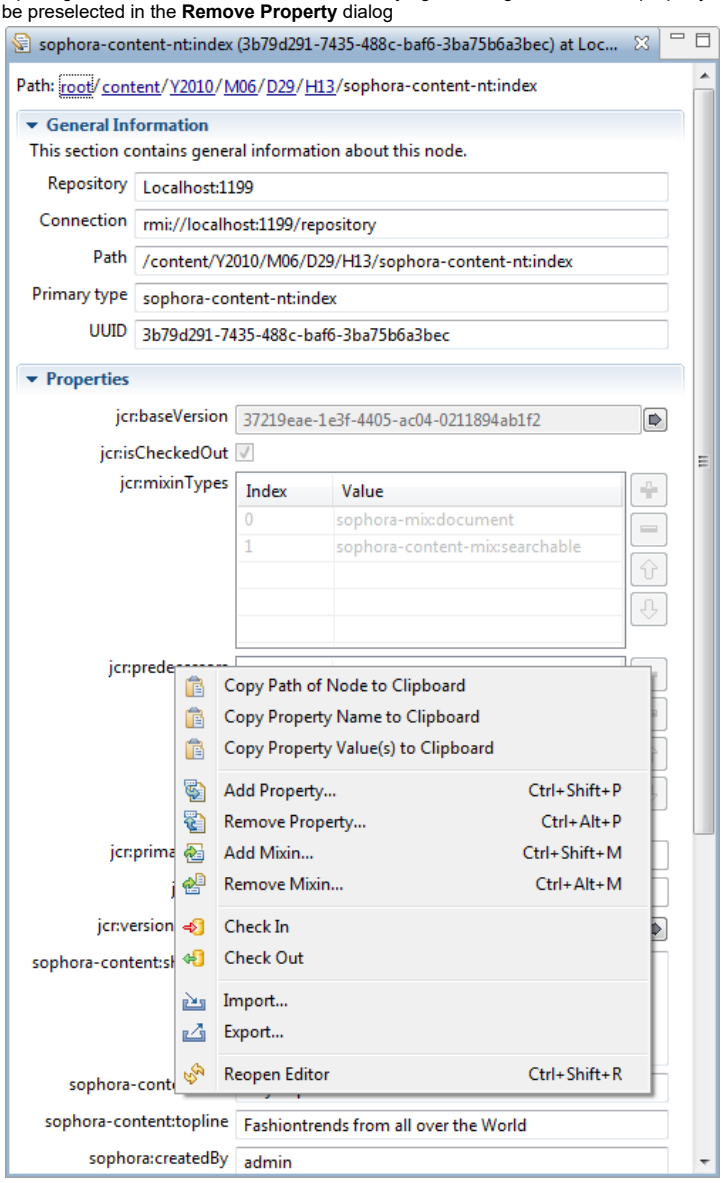
Grouping Repository Connections

To group repository connections in the Repository Browser View just edit the repository connections in the **Repository Connection Wizard**. There you can enter a new group name or select an existing one for each repository connection. See Repository Connections for further information.

Node Editor

Context Menu And Properties

Opening the context menu of the Node Editor by right clicking the name of a property allows you to copy the name and value of that property. The property name will also be preselected in the **Remove Property** dialog



Navigating Nodes

If the current node has a property whose value is a UUID, you can click on the little ▶ next to the property field to open that node in a new editor.

Reopen Node Editor

The active Node Editor can be reopened (reloaded) with the key command **Ctrl+Shift+R**.

Editor Tab Configuration

The editor tab shows the primary name of the node, its UUID, and the name of the repository connection the node belongs to by default. The name of the repository connection may be switched off in the Preferences. You may also change the property whose value should appear in the tab. **Make sure to only configure properties that will always contain short strings.**

Repository Search

Content Assist

Instead of typing the whole query literals like "element" or "SELECT" you can enter only the first characters and then press **CTRL + Space** to open content assist. The assist makes several proposals that you can choose from. Moreover the content assist works for name space names, node type names, property names and child node names of the selected repository, too. The proposals are not case sensitive. You may also use "*" (to match multiple characters) or "?" (to match a single character).

Example:
Imagine you want to use the content assist for "jcr:nodeTypes". You can type...

- "jcr:nodetypes" + **CTRL + Space** (case insensitive)
- "jcr:nodet" + **CTRL + Space** (prefix only)
- "jc*types" + **CTRL + Space** (using *****)
- "jcr:n???types" + **CTRL + Space** (using **?**)

Display Properties of Nodes in the Result List

By default the result list displays only the path of the found nodes. With **Configure Search Result** ( ) you are able to add additional properties of these nodes to display them in the search result directly.

Workbench Layout

Each view can be resized, moved and closed to your liking. To reopen a closed view use **Window > Views** in the main menu. The layout of all windows can also be modified and saved. To save a personal layout use **Window > Save personal window layout**. Restoring the personal layout can be done by using the command **Window > Restore personal window layout**. To restore the default layout use the **Window > Restore default window layout** command.

Automatic Updates

Toromiro can look for new versions each time when started and update automatically. To enable/disable automatic updates, these may be switched on/off in the Preferences within the **Install/Update > Automatic Updates** section. A manual update can be started using **Help > Check for Updates** from the main menu.

Proxy Settings

A Proxy can be configured in the Preferences within the **Network Connections** section. Restarting the application afterwards is recommended.

1.6. Reporting Bugs

If you find a bug or have requests for improvements, new features and other changes, feel free to report us at http://support.subshell.com/browse/TOR. You may use **Help > Report a Bug** from main menu to open the issue tracker, too.

There are several components, reports can be made for:

- **Core**, for basic functions and features
- **Add-ons**, for extended functions and features
- **Documentation**, for (dynamic) help and online documentation

Each report should be assigned

- a **type** ("Bug", "New feature", "Change", etc.)
- a **priority** ("Blocker", "Critical", "High", "Middle", "Low", "Trivial") and
- the **version** of Toromiro, the problem occured.

2.1. Toromiro Legal

Toromiro is developed by

subshell GmbH
Hongkongstrasse 5
20457 Hamburg

All rights reserved.
Visit www.subshell.com

Toromiro License Agreement

BEFORE YOU CLICK ON THE "ACCEPT" BUTTON AT THE END OF THIS DOCUMENT AND DOWNLOAD THE SOFTWARE, CAREFULLY READ THE TERMS AND CONDITIONS OF THIS LICENCE. BY CLICKING ON THE "ACCEPT" BUTTON AND BY DOWNLOADING THE SOFTWARE YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING THE LICENSEE TO THIS LICENCE. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS LICENCE, CLICK THE "DO NOT ACCEPT" BUTTON AND YOU WILL NOT BE ABLE TO DOWNLOAD OR USE THE SOFTWARE.

1. Licence

When you "click" the "Accept" button subshell GmbH, Hongkongstrasse 5, 20457 Hamburg, Germany ("subshell") grants you a non-exclusive, non-transferable licence to install and run Toromiro Software in object code form only subject to the following conditions:

1. You may install Toromiro Software on the hard disk of one computer only, and may make one further copy of Toromiro Software for back-up purposes only. Any other copies of the whole or any part of Toromiro Software are unlawful. You may not install Toromiro Software for use on a network or merge the whole or any part of the Toromiro Software in object code form into another software programme.
2. The associated electronic documentation provided with Toromiro Software is also protected by copyright. You may download and store one copy only of such documentation.
3. You may not transfer Toromiro Software, its documentation or this licence agreement to a third party without subshell's prior written approval, nor may you rent, lease or sublicense it.
4. This licence agreement does not grant you any rights, licence or interest in or to any improvements, modifications, enhancements or updates to Toromiro Software. Updates, if available, may be obtained by you at subshell's then current applicable rates, charges and terms.
5. You may not alter, modify or translate Toromiro Software in any way, nor may you disassemble, decompile or reverse engineer it, except that you may decompile Toromiro Software only to the extent permitted by Section 69e UrhG (German Copyright Act) where this is indispensable to obtain the information necessary to achieve the interoperability of an independently created program with Toromiro Software or with another program and such information is not readily available from the subshell or elsewhere. To that end you will notify subshell in writing prior to decompiling the Toromiro Software and will ask subshell to provide any necessary

information within an appropriate timeframe and in any event not within less than two weeks. Any information processed by subshell to that effect will be treated by you as Confidential Information.

6. You further undertake:
    1. to maintain accurate and up-to-date records of the number and location of all copies of the Software;
    2. to supervise and control use of the Toromiro Software in accordance with the terms of this licence agreement;
    3. to ensure that your employees, agents and other parties who will use the Toromiro Software are notified of this licence agreement and the terms hereof prior to such employee, agent or party using the same;
    4. to reproduce and include the copyright notice of subshell on all and any copies, whether in whole or in part, in any form, including partial copies or modifications of the Toromiro Software made herein;
    5. not to provide or otherwise make available the Toromiro Software in whole or in part (including where applicable, but not limited to program listings, object code and source program listings, object code and source code), in any form to any person other than the your employees or as specified in Section 1.6.3 above without prior written consent from subshell.
7. If you breach any of the terms of this licence agreement, your licence to use Toromiro Software is automatically terminated. This means that any further use of Toromiro Software and its accompanying documentation is unlawful, as are the copies on your hard drive and back-up copy, and you should therefore destroy all copies of Toromiro Software and its documentation within 14 days after the date of termination.

2. Updates

The license for Toromiro Software includes the right to receive updates and upgrades of the software as provided at subshell's discretion from time to time. Updates and upgrades will be provided without additional charge for one year after buying the product. If you want to receive additional updates and upgrades after that period, we may offer you a new license key which covers subsequent updates and upgrades.

3. Warranty

1. subshell does not warrant that the operation of Toromiro Software will be uninterrupted or error free.
2. subshell warrants that Toromiro Software will perform substantially as described in its published specification for a period of (12) months from the date of supply of Toromiro Software. If any material departures from the published specification are reported to subshell within (12) months of the supply of Toromiro Software, subshell will at its option attempt to correct or work round errors, supply an amended version of Toromiro Software or refund the licence fee and terminate this licence agreement. If subshell decides to correct or work round errors it will have at least (3) three attempts to do so. Any reports to subshell of material departures from the published specification outside the (12) months period mentioned above will not be covered by this warranty.
3. You acknowledge that Toromiro Software and its accompanying documentation have not been prepared to meet your individual requirements and that it is your responsibility to ensure that facilities and functions of Toromiro Software meet your requirements. Any warranties or guarantees providing rights beyond those for published specifications of the Toromiro Software, in order to be invalid and defective, require the prior written consent of the Board of Directors of subshell.

4. Indemnity

1. subshell agrees to indemnify and save harmless and defend you at its own expense from and against any and all claims of infringement of any patent, trade mark, industrial design, copyright or other proprietary right affecting the Toromiro Software PROVIDED THAT you
    1. shall not have done, permitted or suffered to be done anything which may have been or become an infringement of any such rights (including but not limited to using the Toromiro Software to perform your or any third party's applications or using the Toromiro Software in combination or merged with other software programs or devices); and
    2. shall have exercised a reasonable standard of care in protecting the same; failing which, you shall indemnify subshell against all actions, proceedings, costs, claims and expenses incurred in respect thereof.
2. You undertake that subshell shall be given prompt notice of any claim specified in 3.1 above that is made against you and subshell shall have the right to defend any such claims and make settlements thereof at its own discretion and you shall give such assistance as subshell may reasonably require to settle or oppose any such claims.
3. In the event that any such infringement occurs or may occur, subshell may at its sole option and expense:
    1. procure for you the right to continue using the Toromiro Software or infringing part thereof; or
    2. modify or amend the Toromiro Software or infringing part thereof so that the same becomes non-infringing; or
    3. replace the Toromiro Software or infringing part thereof by other software of similar capability; or
    4. repay to you the Licence Fee or balance thereof relating to the whole or the infringing part of the Toromiro Software.
4. subshell's liability under this clause shall, at subshell's option, be limited to the Licence Fee less an equitable proportion thereof as relates to the period prior to cessation owing to infringement.

5. Limitation of Liability

1. subshell shall not be liable for losses caused by the misuse of your personal data by third parties who gained access to such data through no fault of subshell's.
2. Except as expressly set forth in this licence agreement, subshell's liability for damages or for reimbursement of expenses - irrespective of the legal basis for any such claims - shall be limited to damages and/or reimbursement of expenses directly caused by the willful default or gross negligence of subshell, its employees and/or agents.
3. subshell's liability for losses caused by the gross negligence of subshell or its agents shall be limited to such amount that was reasonably foreseeable at the time this licence agreement was entered into which shall not exceed the fees actually paid by you for the Toromiro Software in the six months prior to the event which gave rise to the liability.
4. In respect of losses caused by subshell, its employees and/or agents that are neither due to the willful default or gross negligence of subshell, its employees or agents, subshell will only be liable for any losses caused by material breach of this license agreement, in which case subshell's liability shall be limited as set forth in Section 4.3.
5. subshell's liability shall be subject to any contributory negligence by you. Specifically, it is your responsibility to keep your data secure through sufficient back-ups and the use of appropriate anti-virus software.
6. Regarding any liability for death or personal injury and claims resulting from an acceptance of a guarantee or product liability, the statutory provisions shall apply.
7. To the extent subshell's liability is limited in accordance with this Section 4 the limitation of liability also applies to subshell's employees.

6. Copyright Patents, Trademarks and Other Intellectual Property Rights

You acknowledge that any and all of the copyright, trade marks, trade names, patents and other intellectual property rights subsisting in or used or in connection with the Toromiro Software including all documentation and manuals relating thereto are and remain the sole property of subshell. You shall not during or at any time after the expiry or termination of this licence agreement in any way question or dispute the ownership by subshell.

7. Confidential Information

1. All information, data, drawings, specifications, documentation, software listings, source or object code which subshell may have imparted and may from time to time impart to you relating to the Toromiro Software (other than the ideas and principles which underlie the Toromiro Software) is proprietary and confidential ("Confidential Information"). You hereby agree that you shall use the same solely in accordance with the provisions of this licence agreement and that you shall not at any time during or after expiry or termination of this licence agreement, disclose the same, whether directly or indirectly, to any third party without subshell's prior written consent.
2. Subject only to the specific, limited provisions of Section 1 above, you further agree that you shall not yourself or through any subsidiary, agent or third party use such Confidential Information to copy, reproduce, translate, adapt, vary, modify, decompile, disassemble or reverse engineer the Toromiro Software nor shall you

sell, lease, license, sub-license or otherwise deal with the Toromiro Software or any part or parts or variations, modifications, copies, releases, versions or enhancements thereof or have any software or other program written or developed for itself based on any Confidential Information supplied to you by subshell.

3. The foregoing provisions shall not prevent the disclosure or use by you of any information which is or hereafter, through no fault of you, becomes public knowledge or to the extent permitted by law.

8. Force Majeure

subshell shall be under no liability to you in respect of anything which, apart from this provision, may constitute breach of this licence agreement arising by reason of force majeure.

9. Data Protection

subshell will process any of your personal data in accordance with any applicable data protection laws and regulations and in particular in accordance with the provisions of the German data protection act (Bundesdatenschutzgesetz).

10. Termination

1. subshell may by notice in writing to you terminate this licence agreeement if you are in breach of any term, condition or provision of this licence agreement or as required by the applicable law and you fail to remedy such breach (if capable of remedy) within 30 days of having received written notice from subshell specifying such breach.
2. Upon termination, you shall pay to subshell all costs and expenses, including legal and other fees incurred and all arrears of fees, charges or other payments arising in respect of the Toromiro Software, this licence agreement or otherwise and you shall comply with your undertaking specified in Section 1.7 above.
3. Termination, howsoever or whenever occasioned shall be subject to any rights and remedies subshell may have under this licence agreement or under the applicable law.

11. Assignment

You shall not assign or otherwise transfer all or any part of the Toromiro Software or this licence agreement without the prior written consent of subshell.

12. Changes to the Contract

This licence agreement constitutes the entire agreement between the parties and any waiver, amendments or modification of any right, remedy or other term under the agreement will not be effective unless in writing and signed by an authorised person of the party against whom enforcement is sought.

13. Severability

In the event that any of these terms and conditions shall be determined by any competent authority to be invalid, unlawful or unenforceable to any extent, such term shall to that extent be severed from the remaining terms which shall continue to be valid to the fullest extent permitted by law.

14. Governing Law and Jurisdiction

1. This licence agreement is governed by German law. The Parties expressly exclude the application of the United Nations Convention on Contracts for the International Sale of Goods to this licence agreement.
2. Any waiver amendments or modification of any right, remedy or other term under the Agreement will not be effective unless in writing and signed by an authorised person of the party against whom enforcement is sought.
3. All disputes in connection with this contract shall be subject to the exclusive jurisdiction of the German courts at subshell's principal seat of business in Hamburg. subshell may, at its discretion, elect to submit to your place of jurisdiction.

Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries or both.