qualiware®
Enabling positive change

# Repository Administrator

# 1 Contents

# 1 Introduction

Both repositories and metamodels are managed from the web module Repository Administrator (RA) which is a Web application. The link to this application can be found under the Modules section of the QualiWare Execution Framework (QEF) console.

# 2 Repositories and Metamodels

QualiWare keeps its data in repositories and the structure of that data in metamodels. Repositories and metamodels are related in the way that one repository uses one collection of metamodels, simply referred to as “the metamodel”. The same metamodel can be, and usually is, used for several repositories.

Both repositories and metamodels are managed from the web module Repository Administrator (RA) which is a web application. The link to this application can be found under the Modules section of the QualiWare Execution Framework (QEF) console.

## 2.1 What is a Repository

A repository is a container for data objects. The repository handles revisioning, language variances, permissions etc. for the objects. It is stored in a database.

QualiWare can handle several repositories at a time, but a user always connects to one main repository at a time. The user can access data from other repositories in that connections, if these are linked to the primary repository as so called “remote repositories”.

## 2.2 What is a Metamodel

A metamodel contains information on how to interpret and handle data in objects. All objects have a type named a “template”. Definitions for templates are the central information in the metamodel. Usually the metamodel also has code, image data and other data to support the definitions.

For structural reasons, one metamodel only contains definitions for a set of templates. However metamodels refer to each other, so several metamodels are used as a larger set. The entire set of metamodels is referred to by an initialization metamodel, which then loads other referred metamodels. Which initialization metamodel to use is defined by the specific QualiWare product purchased and is subject to licensing terms.

Currently metamodels for QualiWare are used by both the QualiWare Integration Server (QIS) and QualiWare Lifecycle Manager (QLM). They are distributed with the QLM installer, and are located in the subfolder “Models” in the QLM installation.

### 2.2.1 Metamodels in QLM

When QLM is installed it has all the files it needs for the metamodels. QLM simply loads the information from the files.

### 2.2.2 Metamodels in QIS

QIS has two ways of using metamodels, which can be chosen independently for each repository:

- Development mode
- Production mode

#### 2.2.2.1 Development Mode

QIS can load the metamodel directly from the files in the Models folder. This is known as "Development mode". The benefit of this mode is that development and customization of the metamodel can be done rapidly. Any change to a file is automatically detected by QIS, and the new information is automatically loaded. The drawback is the same. Any accidental or incorrect change to a file will become a problem for users immediately.
Note: Since QLM currently uses the information directly from the files, it is always in development mode.

#### 2.2.2.2 Production Mode

Alternatively QIS can use a repository for storing the metamodel. This is known as "Production Mode". The benefit of this mode is that files can be changed and tested (in another repository in development mode) without affecting users. This mode also allows for revisioning of the metamodel in much the same way as objects in repositories. The drawback is more management, and the fact that QLM is currently not protected by production mode.

### 2.2.3 Recommendation

Since QLM is currently always in development mode, QualiWare recommends using the development mode for QIS as well. Having QIS repositories in production mode can result in QLM and QIS using different versions of the metamodel.
QualiWare plans to expand production mode to also work with QLM. Once this work is finished, modes can be changed with little effort and no loss of previous work.

## 2.3 Repository and Metamodel Basics in RA

Repositories are created and managed inside RA. A repository needs some basic setup to work:

- A database
- A pointer to a metamodel

If the repository is in development mode, the metamodel pointer is simply a path to the Models folder in the QLM installation, as well as the name of the initialization metamodel to be used for QIS and QLM.
If the repository is in production mode, the metamodel pointer is a selection of a created metamodel repository, as well as the name of the initialization metamodel to be used for QIS and QLM.

Metamodels are also created and managed inside RA. If all repositories are in development mode, no metamodel repositories need to be created. If one or more repositories are in production mode, the metamodel repository (-ies) needs to be created. Then the information from the files distributed with QLM needs to be loaded into the repository. This process is known as "Deployment". If changes are made in the files these can be redeployed once they are tested and accepted for production.

Please refer to the following sections for precise information on how to create repositories and metamodels, and how to migrate data.

# 3 Creation of Repositories

## 3.1 Preface

This document describes the process of creation and configuration of repository.

## 3.2 Prerequisites

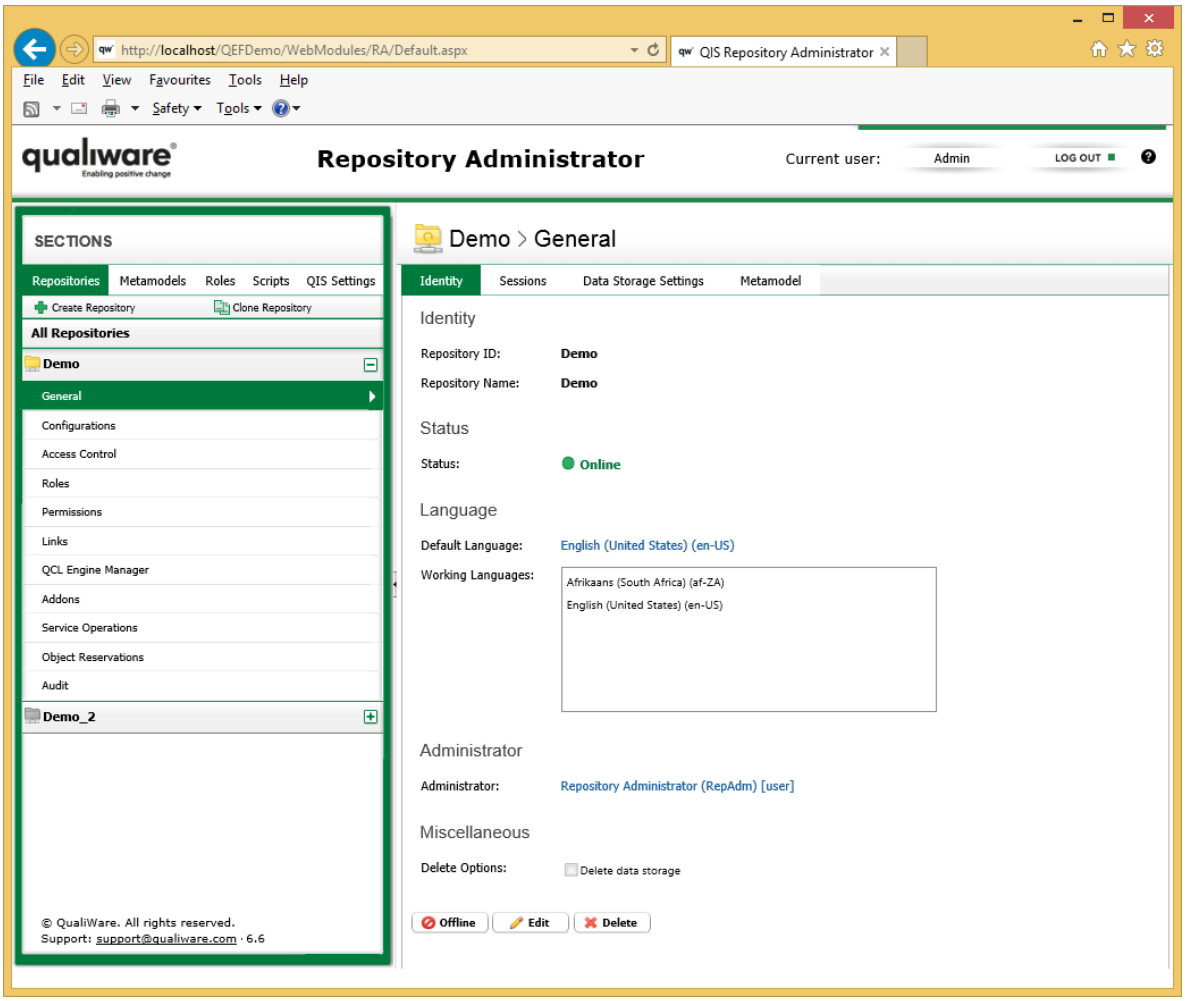
QualiWare Execution Framework (hereinafter QEF) and Admin Console are installed and running.

QualiWare Integration Server (hereinafter QIS) and Repository Administrator (hereinafter RA) modules are installed and running.

User is assigned QIS Administrator role. Some operations may require user be assigned QEF Administrator role.

## 3.3 Creation of Repository

1. Go to RA -> Repositories.
2. Click Create Repository.
3. Enter name of repository. By default name and ID of repository are the same.
4. Select default and working languages.
5. Define Administrator of repository
6. Click OK.




Notes:

- Change of ID of repository after it is created and it contains data, may break remote repository object links if such exist.
- Default language cannot be changed after repository is created. However it is possible to replace it (see Service Operations).

## 3.4 Administrator of Repository

Administrator of repository (QEF user or QEF group of users) needs to perform maintenance job (change settings etc.) with exclusive priority. That's why he has specific rights:

1. If repository is in Read-Only mode, only Administrator is able to create, edit and save objects,
2. Administrator can switch repository to offline mode (when only one session is allowed) to perform maintenance jobs. In this case sessions of other users (including QCL Engine sessions) are killed. Only session of Administrator of repository remains.

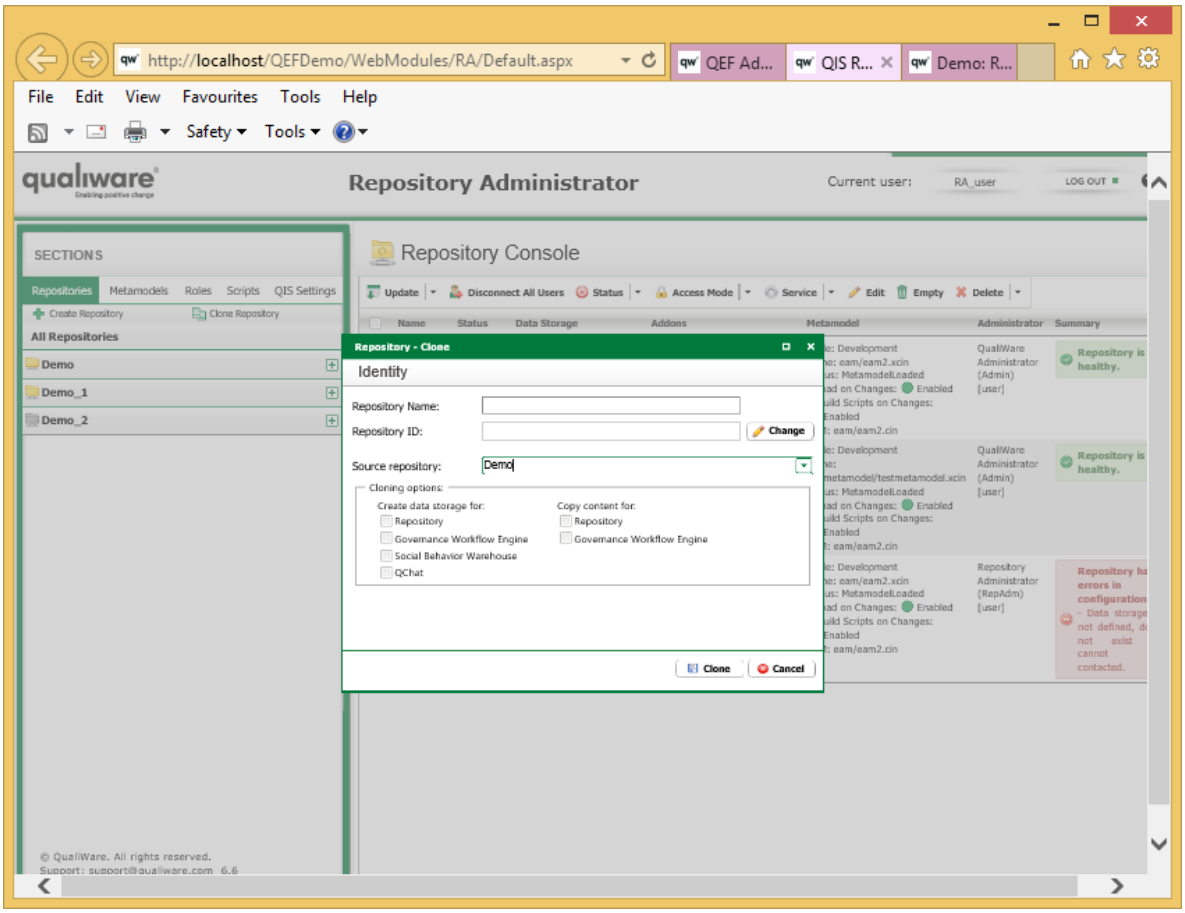
Assignment of Administrator can be done on **Identity** page or on Repository Panel (**All Repositories** page) for group of selected repositories

## 3.5 Cloning of Repository

1. Go to **RA -> Repositories.**
2. Click **Clone Repository**.
3. Enter name of repository. By default name and ID of repository are the same.
4. Select if data storage of repository and addons should be created and content copied.
5. Click **Clone.**

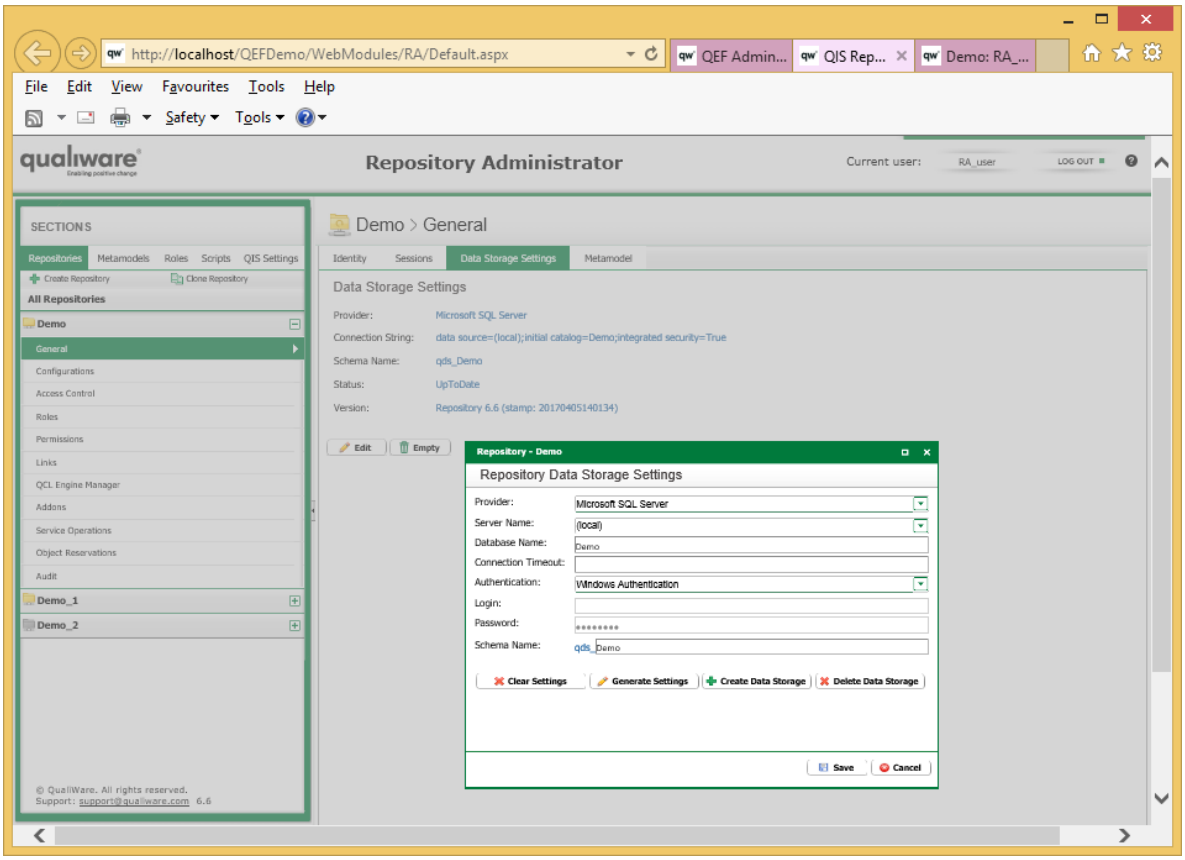
Cloned repository receives following features of source repository:

- datastorage settings, addons and content (according to user's choice),
- roles,
- permissions,
- links with remote repositories,
- Administrator of repository,
- Metamodel,
- Access Mode settings,
- QCL Engine Configuration.




## 3.6 Creation of Data Storage

1. Go to RA -> Repositories -> {Repository} -> General -> Data Storage Settings.
2. Click Generate Settings or enter settings manually. (Generated settings are taken from QIS Settings (RepositoryDataStorageTemplate))
3. Click Create Data Storage.
4. Click Save.




Notes:

- More than one repository data storage can be created in a single database. Each repository data storage must use unique schema name.

## 3.7 Set Roles and Permissions

Permissions are defined on the following levels:

- Repository permissions.
- Initial object ACL.
- Object permissions.

### 3.7.1 Repository Permissions

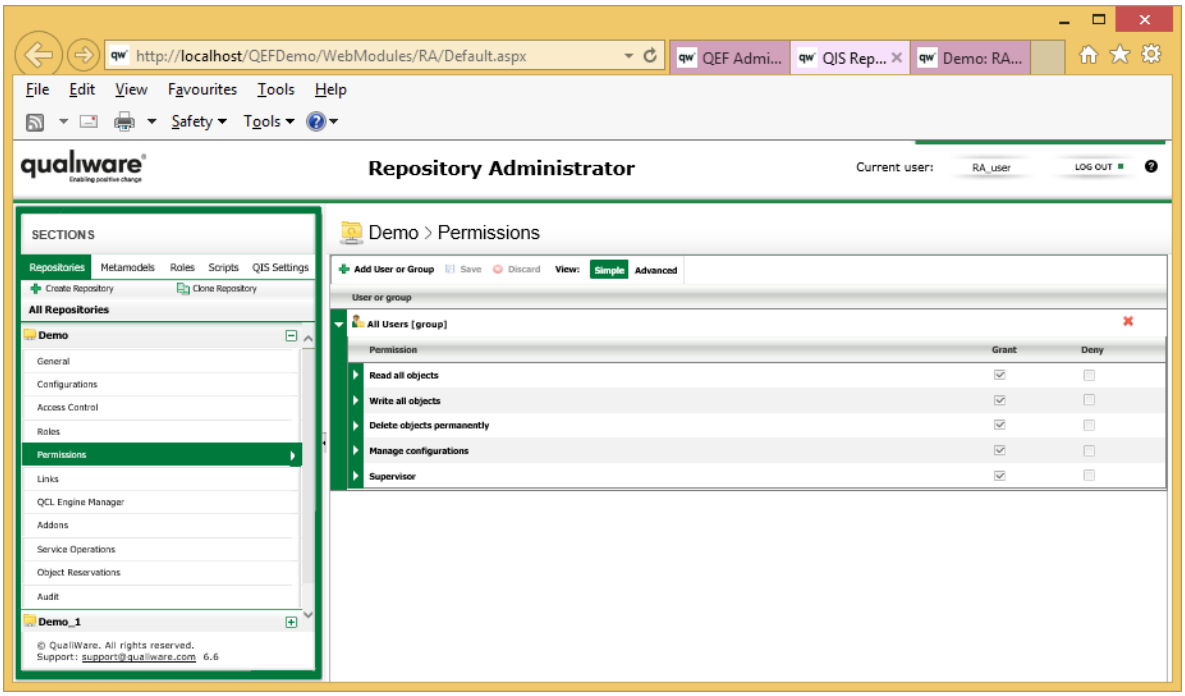
Repository permissions are not assigned to objects directly, but calculated when user connects to repository. Because they are not assigned to objects directly, it is possible to change repository permissions after objects are created and changes will have effect on existing and future objects. To change repository permissions:

1. Go to RA -> Repositories -> {Repository} -> Permissions.
2. Expand group or user node to modify existing assignments or click Add User of Group to add new assignment.

3. Modify permissions as required.

   Note: Deny permission always wins over grant.

4. Click **Save**.



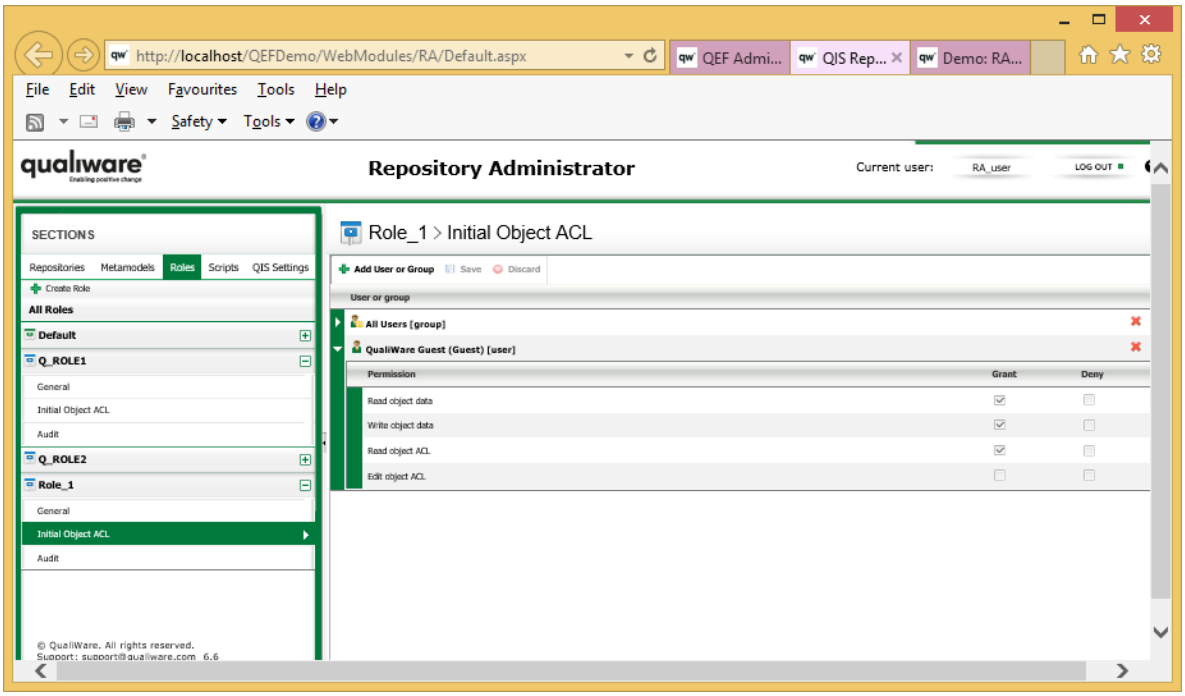

### 3.7.2 Initial Object ACL

Initial object ACL are assigned to object at the moment when it is created. These permissions overwrite repository permissions. Because they are assigned to objects directly, any change to initial object ACLs after objects are created will not have effect on them, but only on future objects. Existing objects will need to have ACLs re-assigned accordingly using QIS API or QLM. To change initial object ACLS:

1. Go to **RA -> Roles -> {Role} -> Initial Object ACL**.
2. Expand group or user node to modify existing assignments or click **Add User of Group** to add new assignment.
3. Modify permissions as required.

   Note: Deny permission always wins over grant.

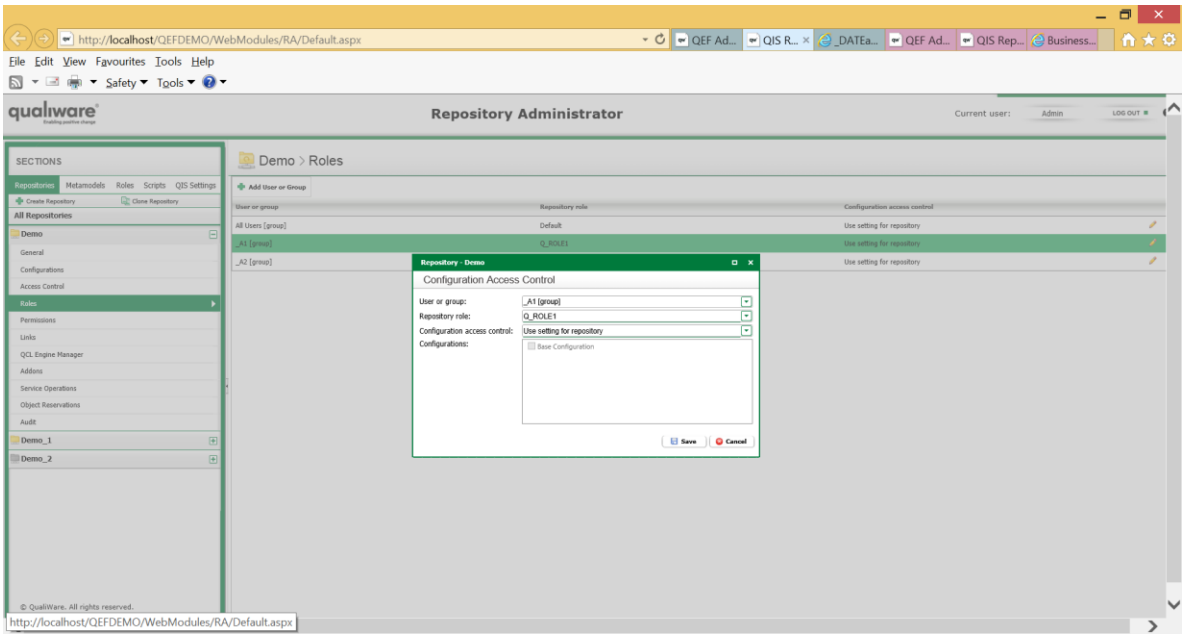
4. Click **Save**.




### 3.7.3 Assignment of Roles to Users and Groups

Roles need to be explicitly assigned to user or group for users be able to connect with the specified roles. To assign roles to users and groups, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> Roles.
2. Click Add User or Group.
3. Select user or group.
4. Select role.
5. Click Save.




### 3.7.4 Object Permissions

Object permissions are assigned to objects directly by means of QIS API or QLM.

## 3.8 Repository Status

Repository can be in Online or Offline state.

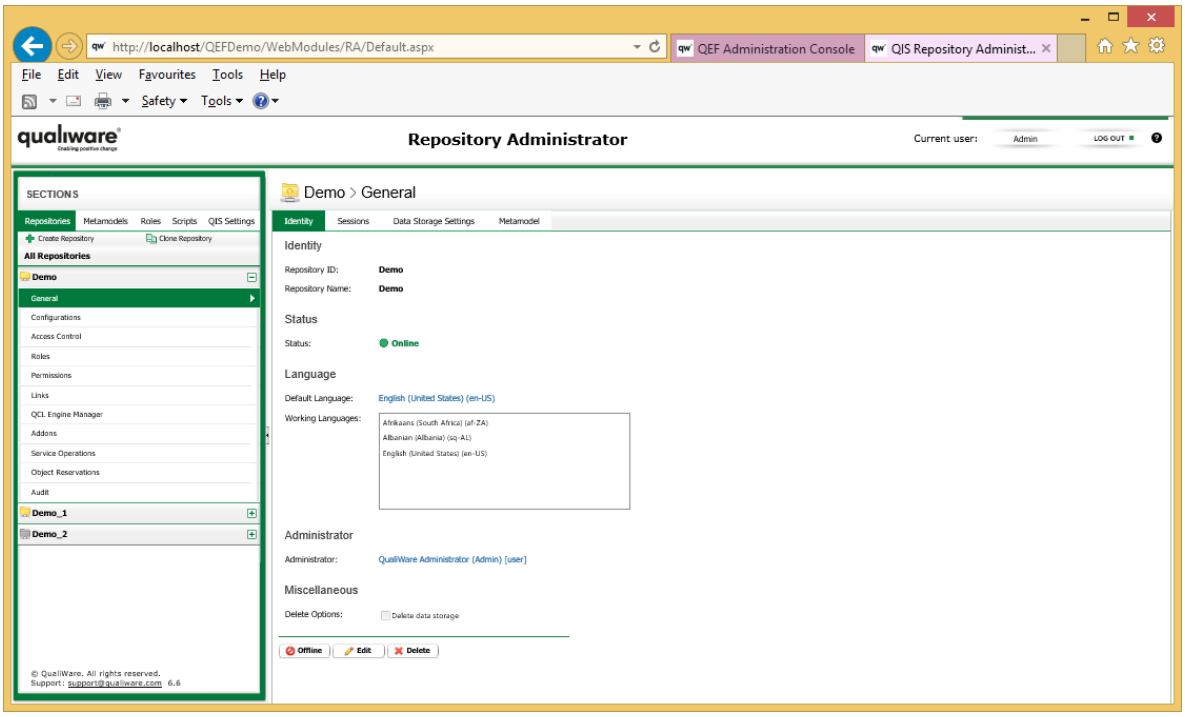
Online state allows multiple users to connect.

Offline state allows single user to connect. Offline state is designed for maintenance tasks like defining data storage and metamodel, dump loading and others.

To change repository state, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> General -> Identity.
2. Click Offline or Online, depending on the current state.

Repository state for several selected repositories can be changed on Repository panel.




## 3.9 Assignment of Metamodel to Repository

### 3.9.1 QIS and QLM Metamodels

QIS and QLM metamodels are defined separately and used by QIS and QLM sole and respectively. Both metamodels are in different formats, having some similarities. Both metamodels are generated by Casemaker at the same time.

Despite the fact that QIS and QLM metamodels are two separate metamodels, it is possible to configure QLM to use its metamodel through QIS. This configuration is not default and should explicitly be turned on. To configure QLM to use its metamodel through QIS, follow the steps:

1. Open **models\openrep.qcl** file.
2. Find **UseQisMetamodel** setting and set its value to one of the following:

- 0 (default) - Use QLM metamodel.
- 1 - Use QIS metamodel.

### 3.9.2 QIS Metamodel

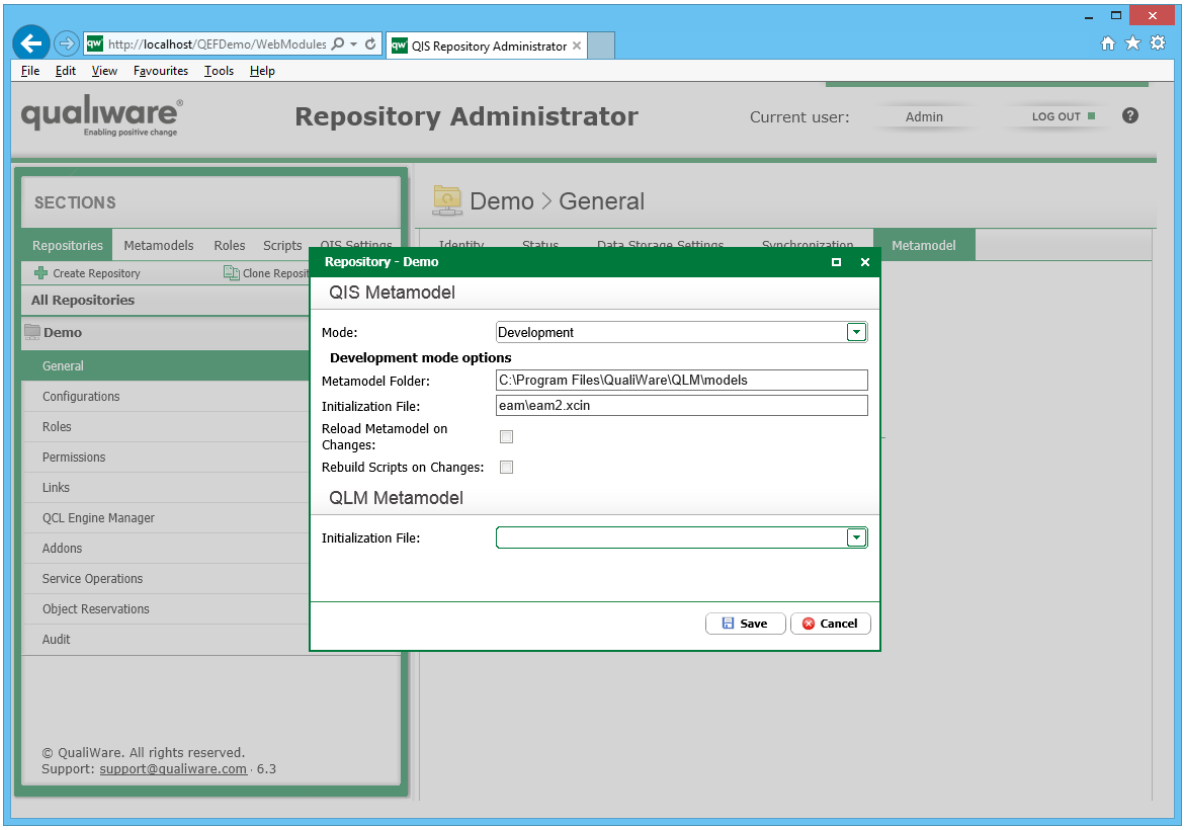
QIS metamodel supports two modes:

- **Development mode**. It takes metamodel from Models folder in file system.
- **Production mode**. It takes metamodel from database, where it must preliminary be deployed.

#### 3.9.2.1 Development Mode

To configure repository to use metamodel in development mode, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> General -> Metamodel.
2. Click Edit.
3. Select Development mode.
4. Enter full path to Models folder.
5. Enter relative path to .xcin file under Models folder.
6. Click Save.




#### 3.9.2.2 Production Mode

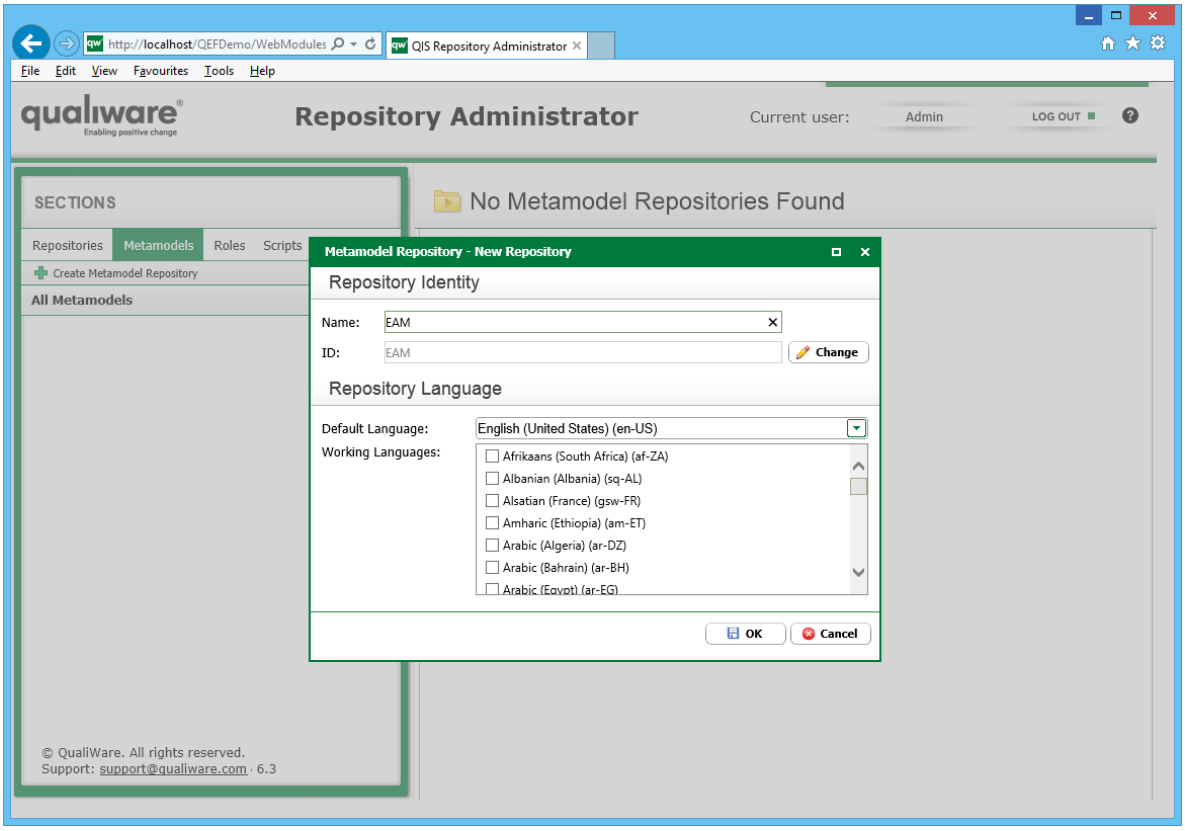
Configuration of repository to use metamodel in production mode includes the following steps:

1. Creation of metamodel repository.
2. Deployment of metamodel.
3. Assignment of metamodel.

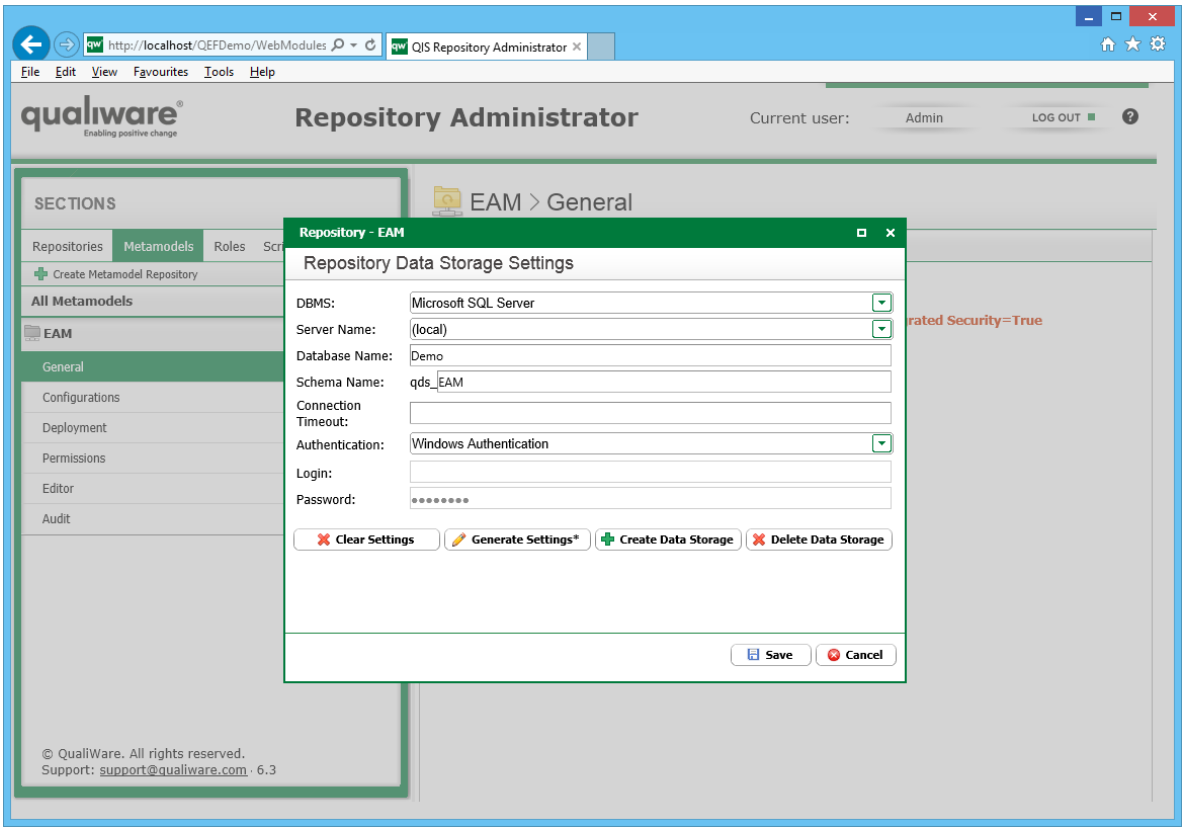
##### 3.9.2.2.1 Creation of Metamodel Repository

1. Go to RA -> Metamodels.

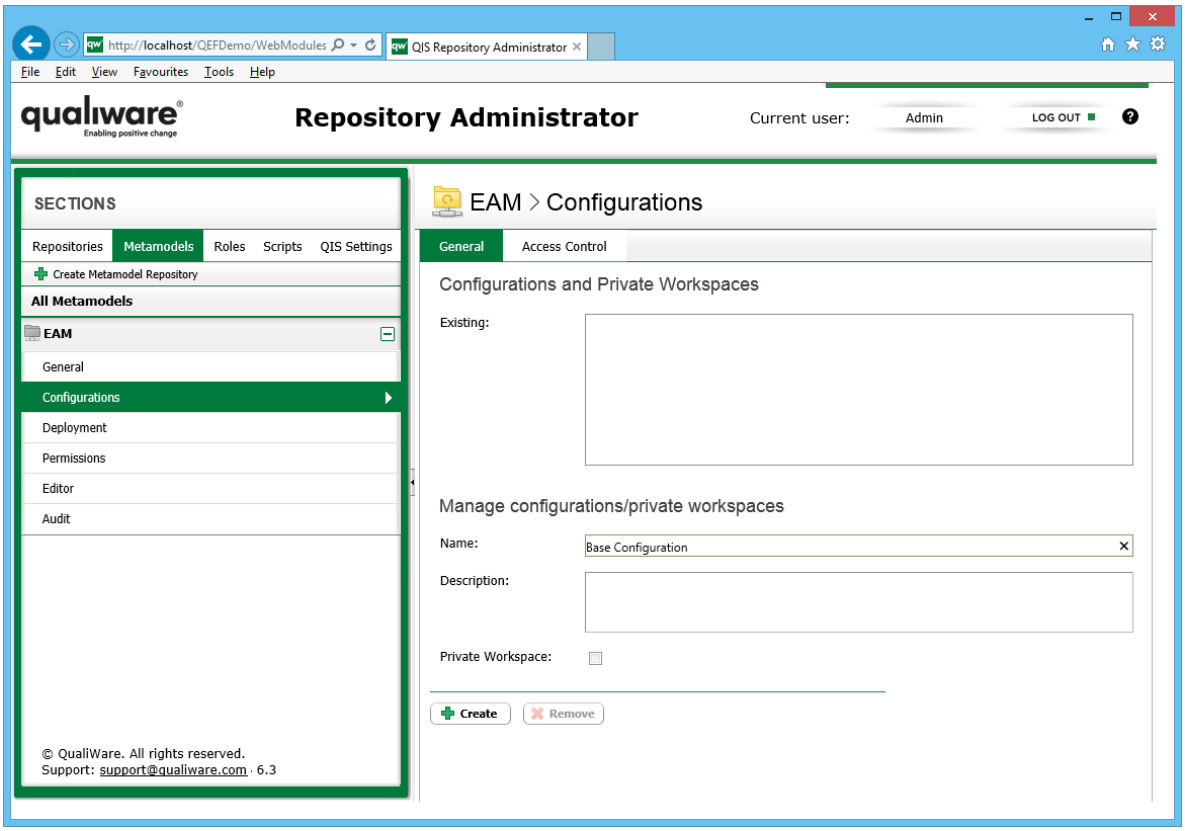
2. Click **Create Metamodel Repository**.
3. Enter name of repository.
4. Select default and working languages.
5. Click **OK**.



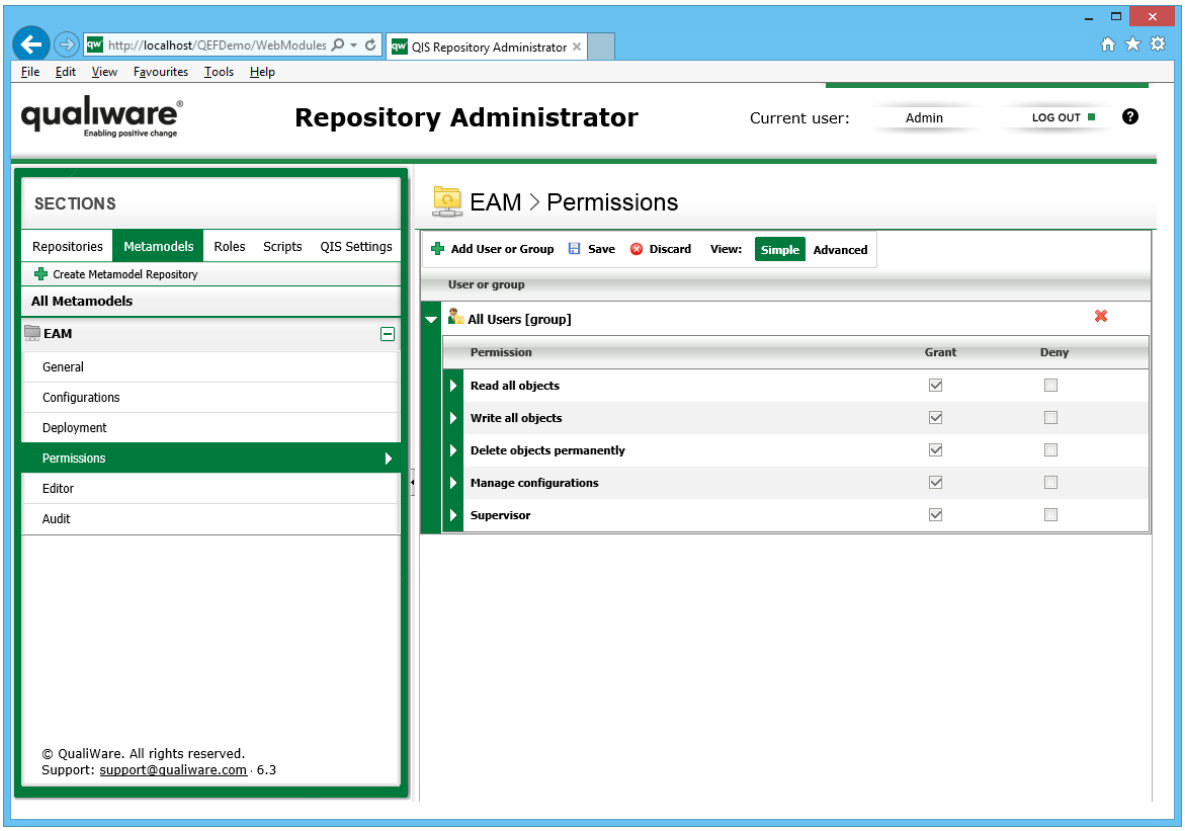

6. Go to **RA -> Metamodels -> {Repository} -> General -> Data Storage Settings**.
7. Click **Edit**.
8. Click **Generate Settings** or enter settings manually.
9. Click **Create Data Storage**.
10. Click **Save**.




11. Go to **RA -> Metamodels -> {Repository} -> Configurations.**
12. Enter name of configuration.
13. Click **Create**.




14. Go to **RA -> Metamodels -> {Repository} -> Permissions.**
15. Select **All Users** group.
16. Select **Supervisor** permission,
17. Click **Save**.

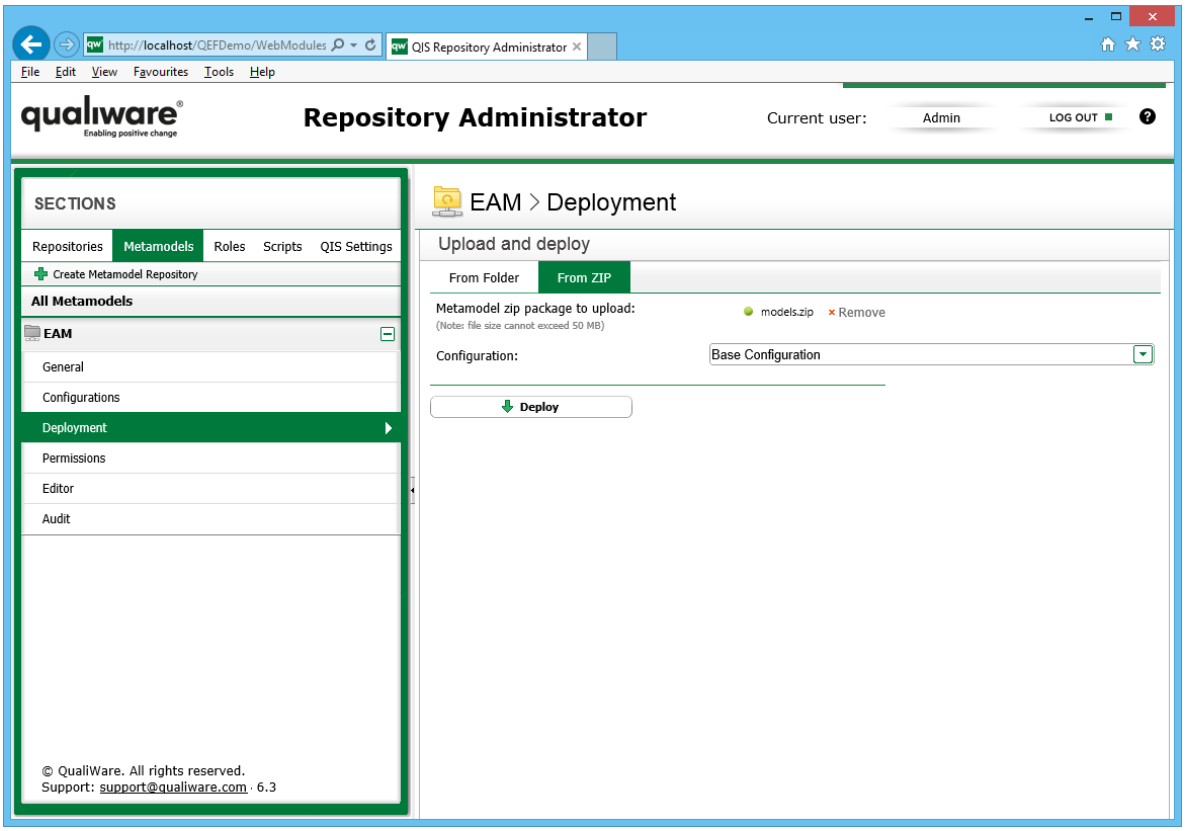



#### 3.9.2.2.2 Deployment of Metamodel

Deployment can be done either from zip archive or directly from Models folder.

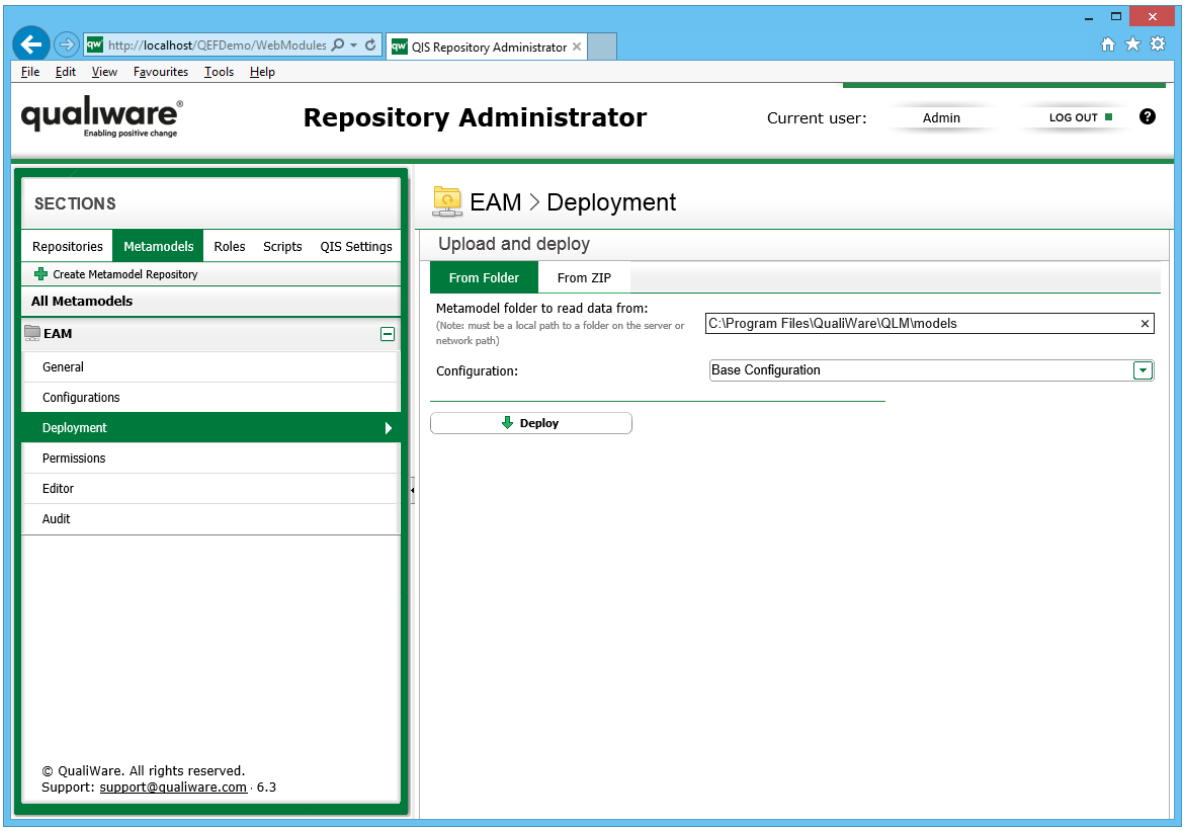
##### *3.9.2.2.2.1 Deployment of Metamodel from Zip Archive*

1. Go to RA -> Metamodels -> {Repository} -> Deployment -> From ZIP.
2. Click Browse to select zip archive to deploy.
3. Select configuration.
4. Click Deploy.




#### *3.9.2.2.2.2 Deployment of Metamodel from Models Folder*

1. Go to RA -> Metamodels -> {Repository} -> Deployment -> From Folder.
2. Enter full path to Models folder.
3. Select configuration.
4. Click Deploy.

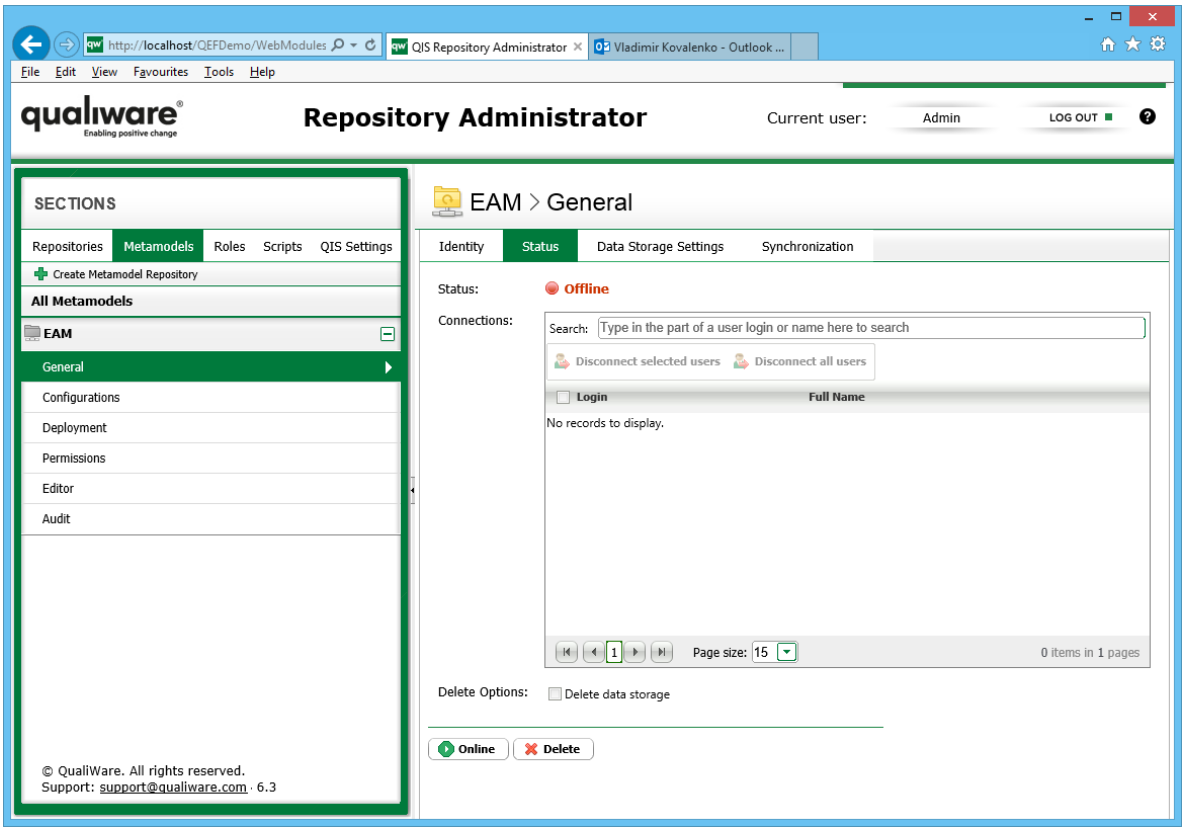



Notes:

- All metamodels found in Models folder will be loaded. With this regard clean it from old and development files (scripts, dialogs, queries etc.).

#### 3.9.2.2.3 Bring Metamodel Repository Online

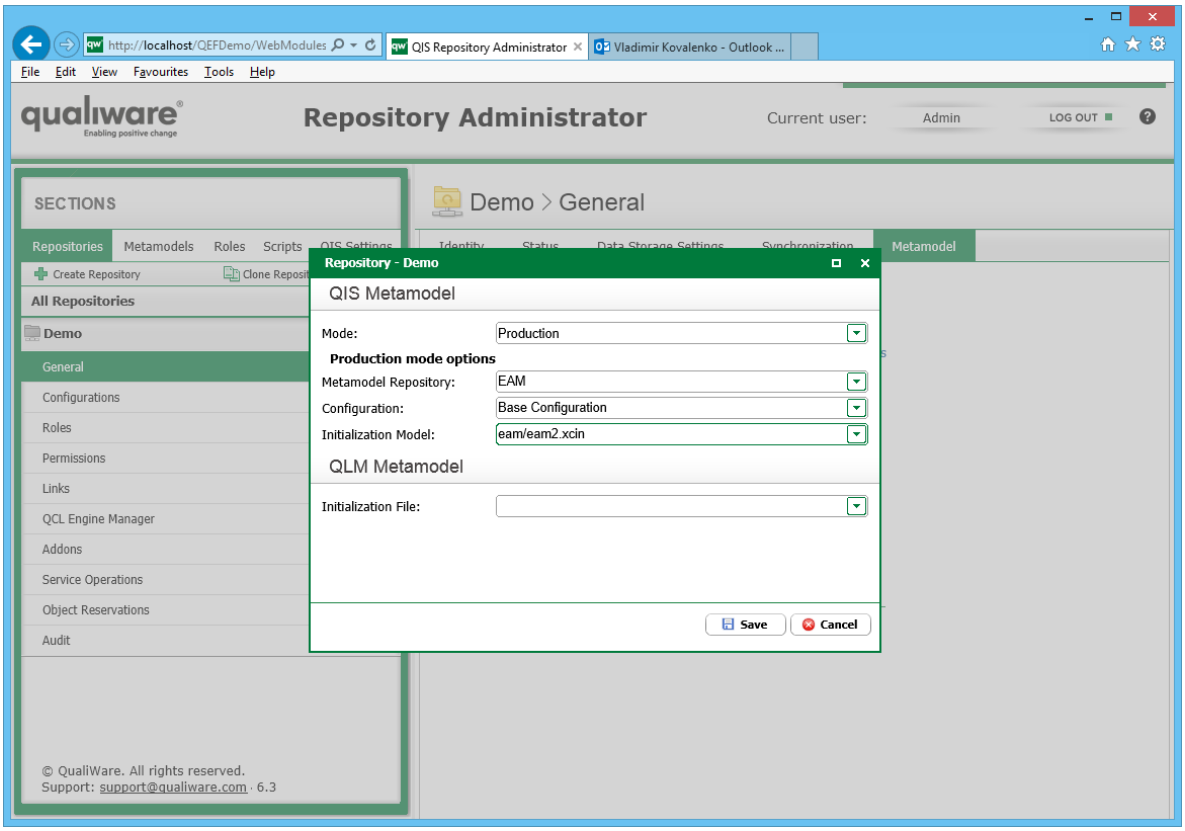
1. Go to RA -> Metamodels -> {Repository} -> General -> Status.
2. Click Online.




#### 3.9.2.2.4 Assignment of Metamodel

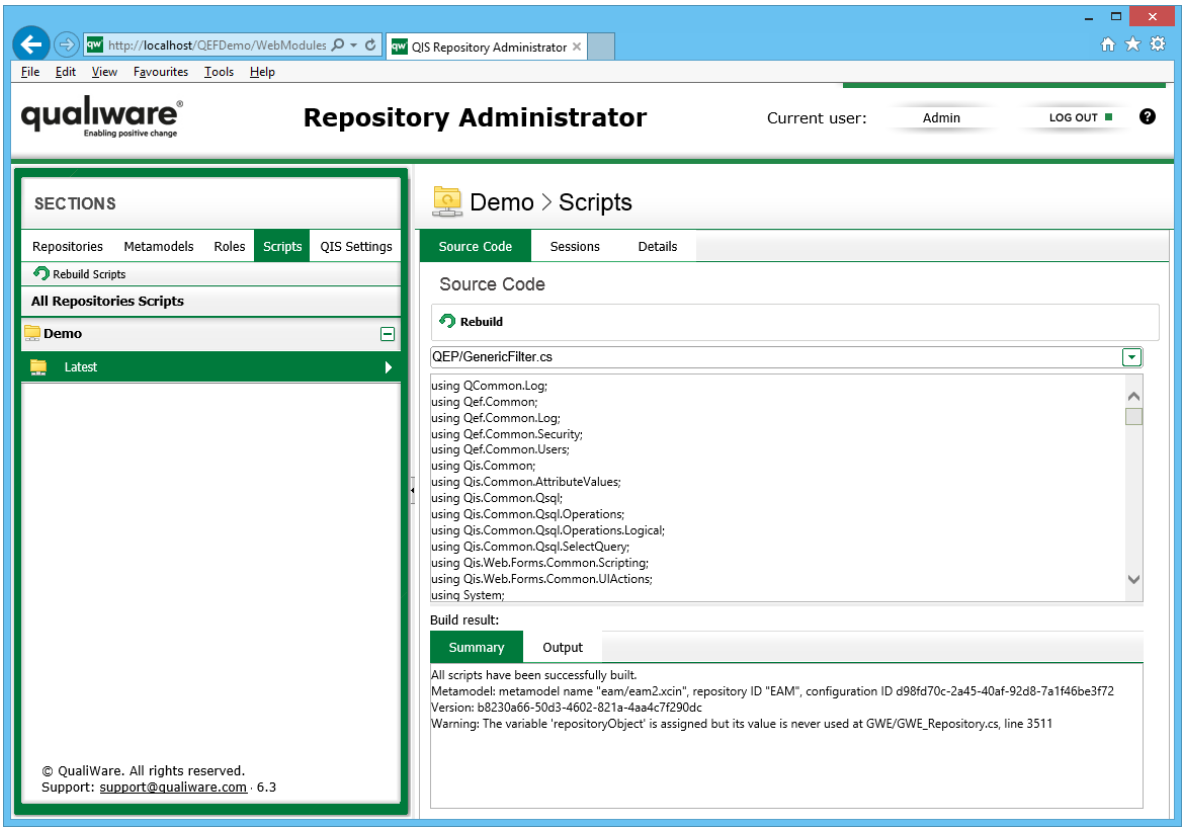
To configure repository to use metamodel in production mode, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> General -> Metamodel.
2. Click **Edit**.
3. Select **Production** mode.
4. Select metamodel repository.
5. Select metamodel configuration.
6. Select .xcin file.
7. Click **Save**.




#### 3.9.2.3 C# Scripts

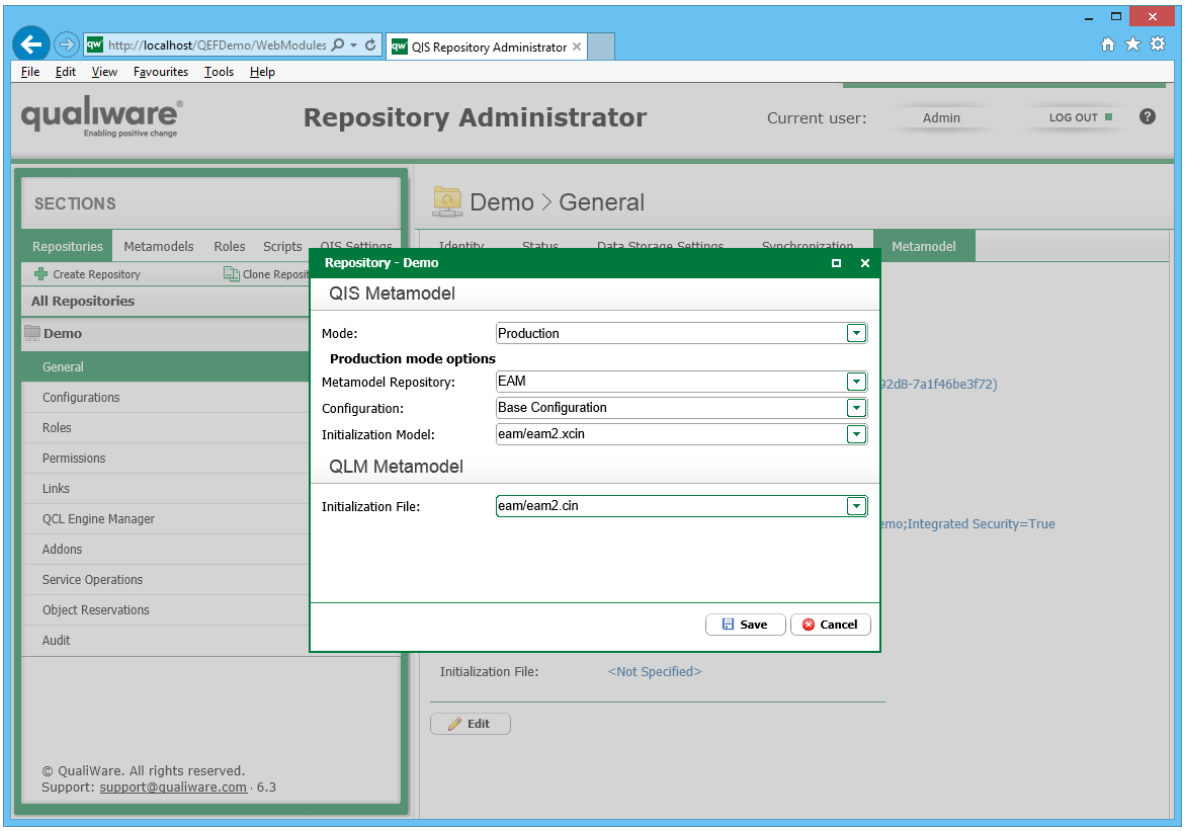
C# Scripts are a part of metamodel. Each repository has its own copy of scripts. Repository scripts can be found at the following location: RA -> Scripts. Provided scripts need to be rebuilt, click Rebuild Scripts.




### 3.9.3 QLM Metamodel

To configure repository to use metamodel in production mode, follow the steps:

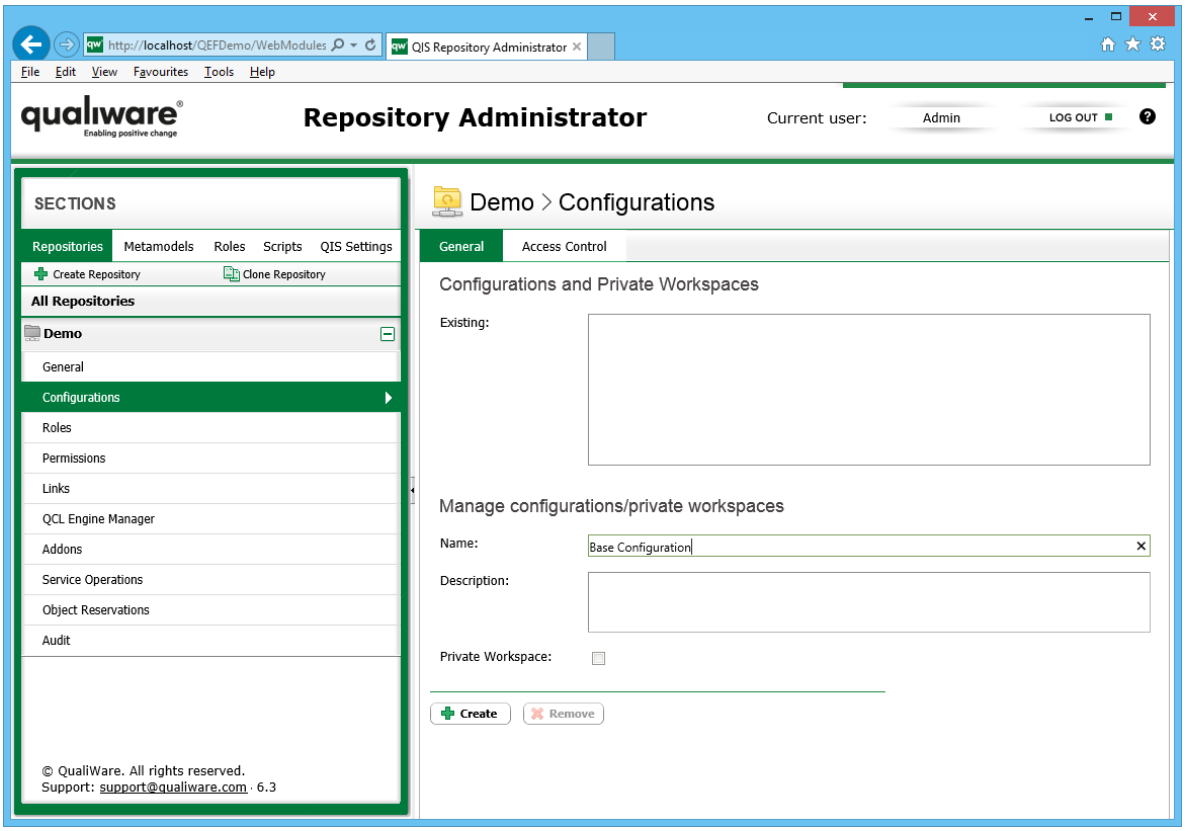
1. Go to RA -> Repositories -> {Repository} -> General -> Metamodel.
2. Click Edit.
3. Select or enter initialization file.
4. Click Save.




## 3.10 Creation of Configurations

Newly created repository does not contain configurations. Base configuration is created automatically when first connection to repository is made. Or it can be created explicitly from RA. To create configuration, follow the steps:

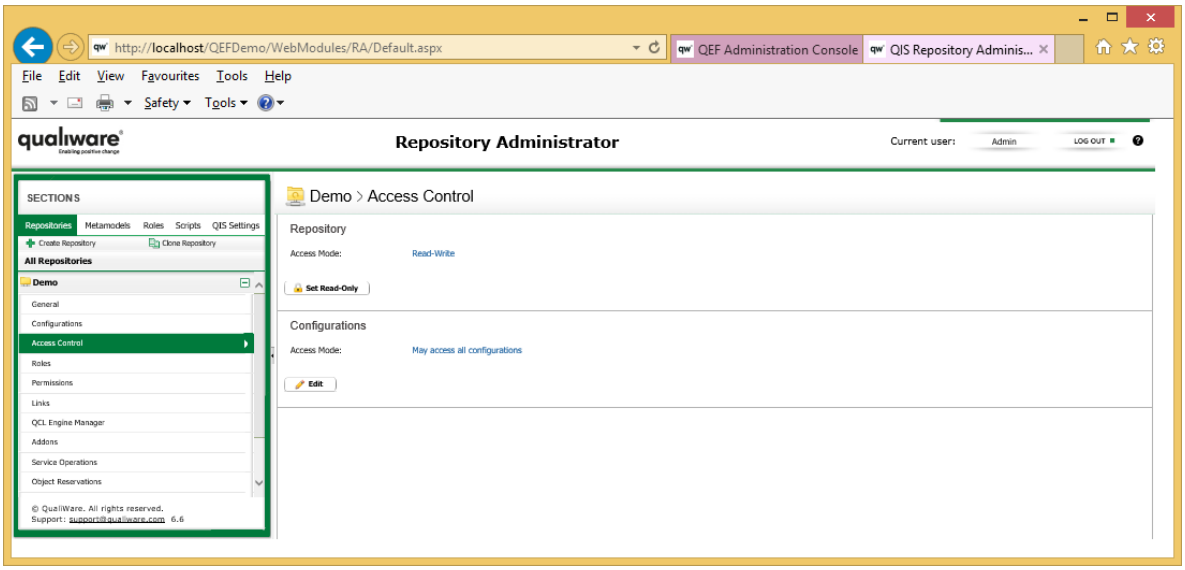
1. Go to RA -> Repositories -> {Repository} -> Configurations.
2. Enter name and description of configuration.
3. Select if new configuration is a normal configuration or private workspace.
   Note: Private workspace can only be created, provided any configuration exists and selected.
4. Click Create.




Notes:

- If dump is planned to be loaded in a newly created repository, all existing configurations will be deleted.

## 3.11 Access Control



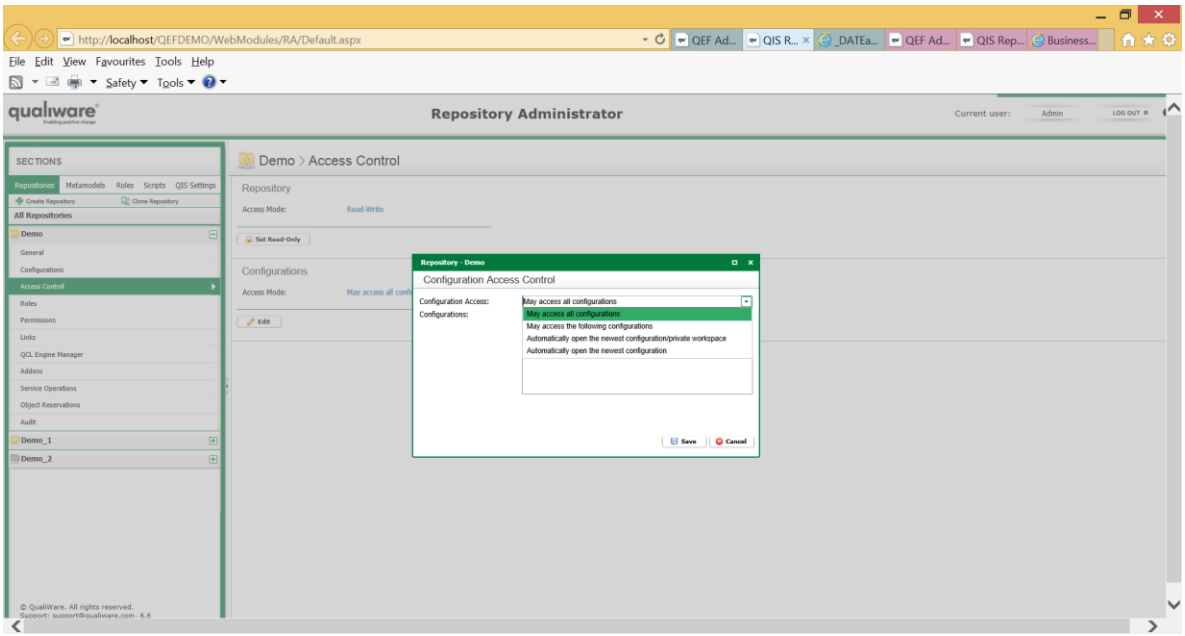

Repository can be in Read-Write mode or Read-only

By default all users have access to all configurations in repository. Access can be refined by access control settings. Access control settings can be defined in order of priority on the following levels: repository, role and remote repository link. There are the following options for access control settings:

- Use settings for repository (only available on role level). Inherit settings from repository level.
- Use settings for role (only available on remote repository link). Inherit settings from role level.
- May access all configurations. User can access all existing configurations.
- May access the following configurations. User can access only selected configurations.
- Automatically open the newest configuration/private workspace. User is automatically connected the newest configuration or private workspace.
- Automatically open the newest configuration. User is automatically connected to the newest configuration.

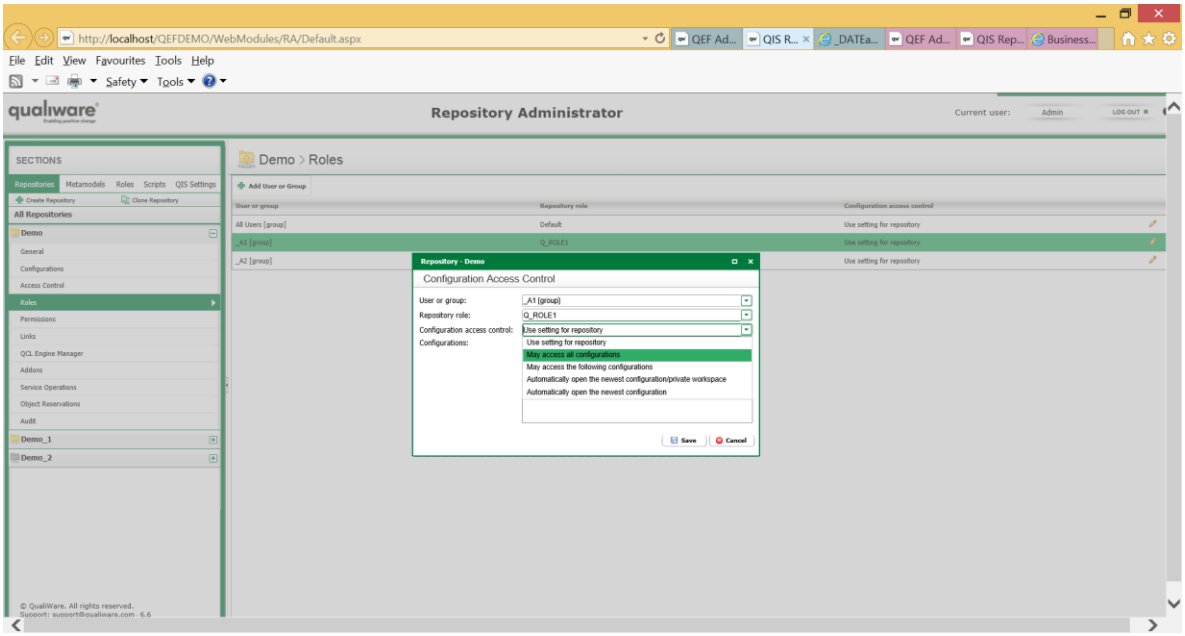
To change access role settings on repository level, follow the steps:

1. Go to **RA -> Repositories -> {Repository} -> Access Control**.
2. Click **Edit**.
3. Select one of configuration access control settings.
4. If **May access the following configurations** setting is selected, select configurations that can be accessed.
5. Click **Save**.



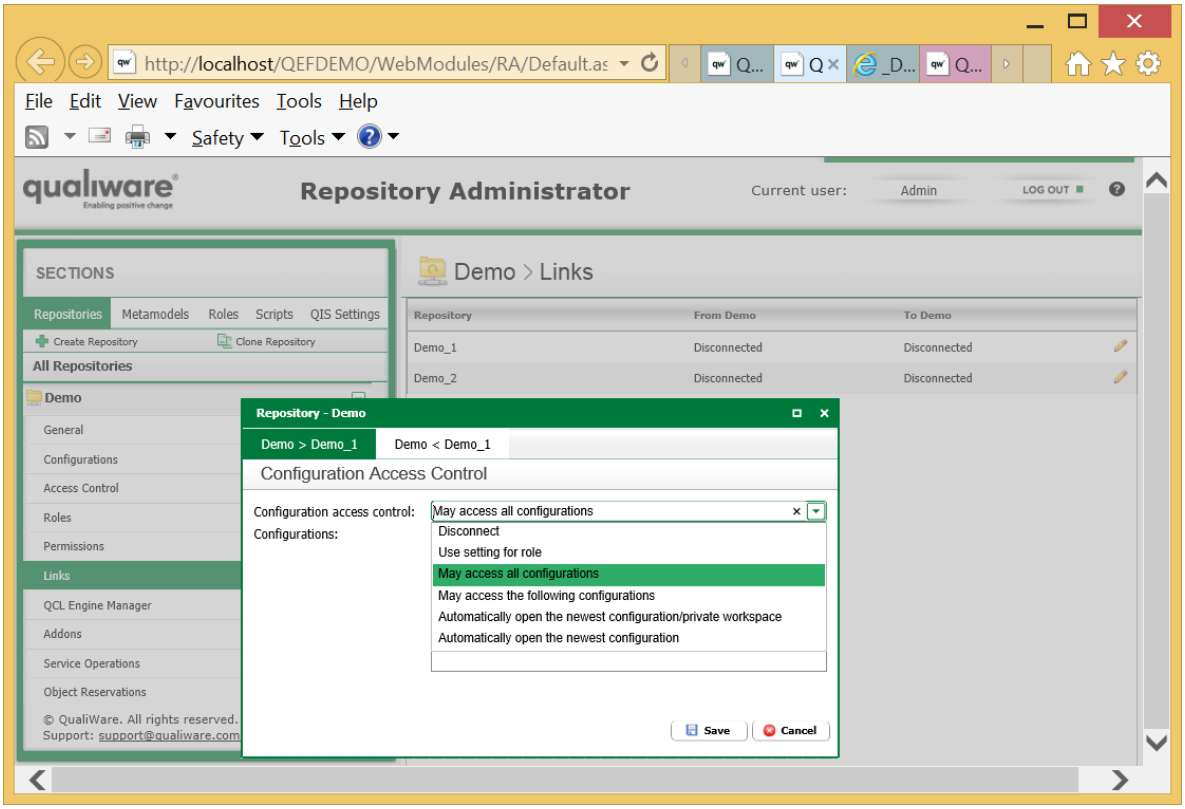

To change access role settings on role level, follow the steps:

6. Go to **RA -> Repositories -> {Repository} -> Roles**.
7. Click **Edit** in front of the role to configure.
8. Select one of configuration access control settings.
9. If **May access the following configurations** setting is selected, select configurations that can be accessed.
10. Click **Save**.



To change access role settings on remote repository link level, follow the steps:

11. Go to **RA -> Repositories -> {Repository} -> Links**.
12. Click **Edit** in front of the remote repository link to configure.
13. Select one of configuration access control settings.
14. If **May access the following configurations** setting is selected, select configurations that can be accessed.
15. Click **Save**.

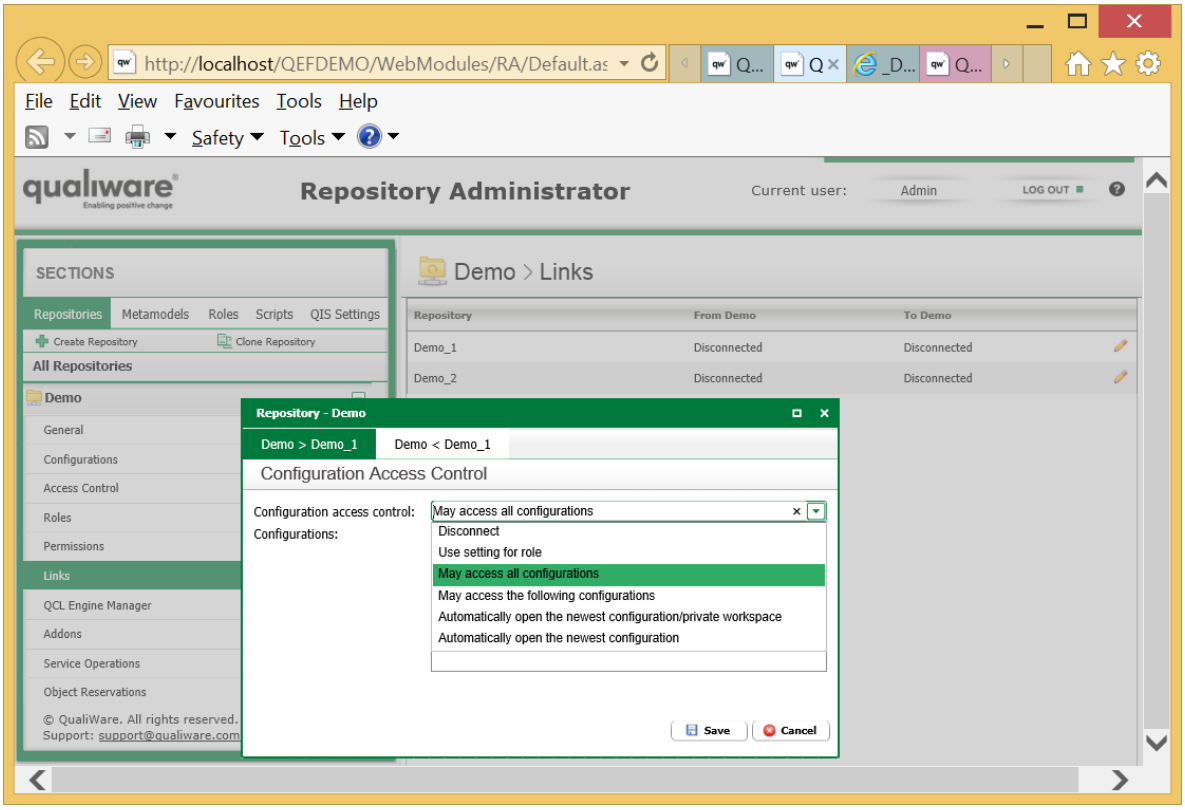



## 3.12 Remote Repositories

By default objects in one repository can only have links to other objects in the same repository. Links to objects in other repositories must be allowed explicitly. To do so, repositories must be linked and connection established. To link repositories, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> Links.
2. Click Edit in front of the repository to link.
3. Select one of configuration access control settings for forward and backward link.
4. If May access the following configurations setting is selected, select configurations that can be accessed.

5. Click **Save**.




Provided repositories are linked, remote connection can be established using Web Forms, QLM or QIS API and links to objects from other repository can be inserted or opened.

## 3.13 QCL Engine Manager

QCL Engine Manager is enabled by installing QEM module in Admin Console.

For detailed documentation on installation, updating and configuration of QEF, see the following chapters::

One time configurations inside QEF - Modules installation & updating - QCLE.docx

Repositories and metamodels - One time configuration of repositories - Enabling and configure QCLE.docx

## 3.14 Addons

Addons are extensions to the basic functionality provided by QIS. There are the following addons exist: Access Log, Governance Workflow Engine, Social Behavior Warehouse, QChat.

For detailed documentation on configuration of each of the addons, see the following documents:

Repositories and metamodels - One time configuration of repositories - Access Log.docx

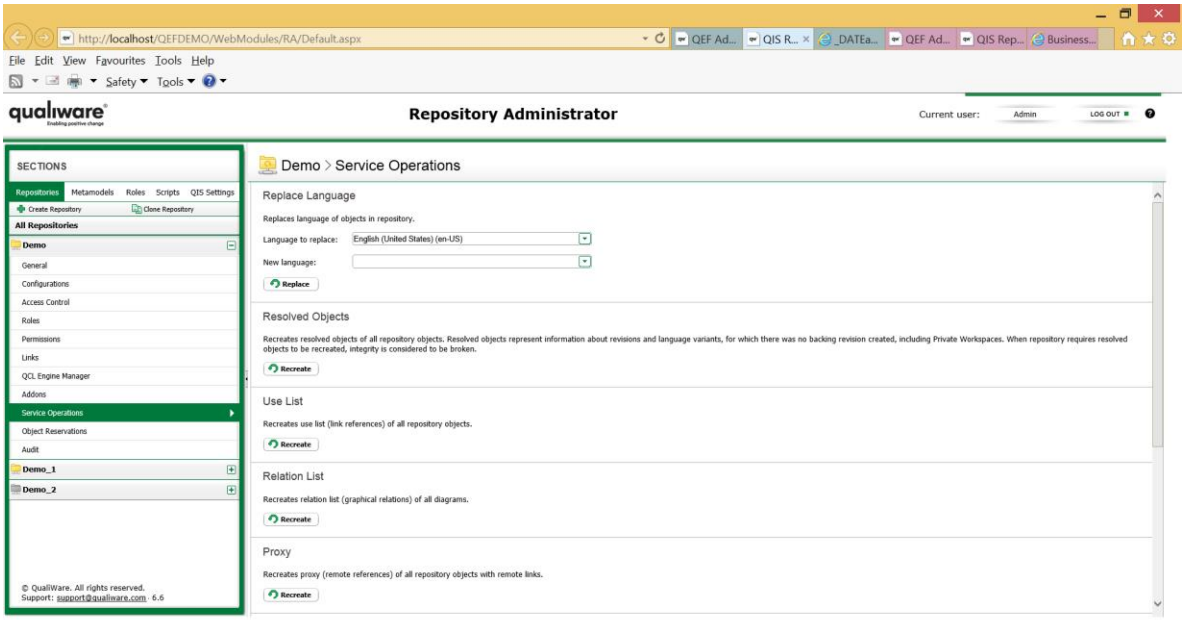
Repositories and metamodels - One time configuration of repositories - Enabling and configure GWE.docx

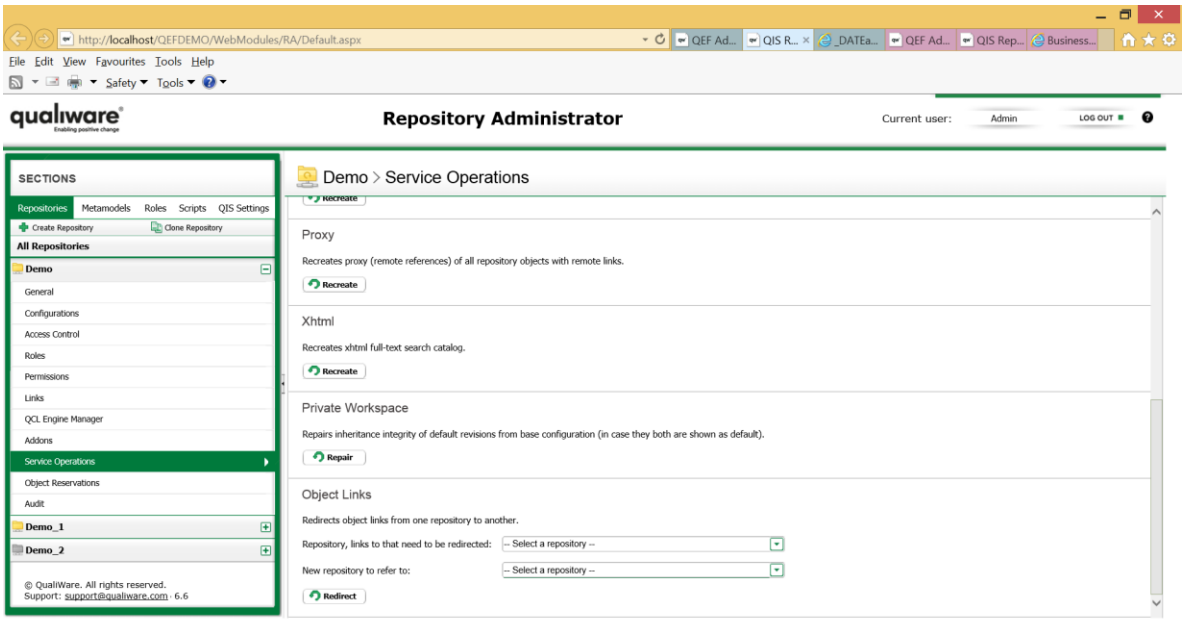
Repositories and metamodels - One time configuration of repositories - Enabling and configure SBW.docx

## 3.15 Service Operations

Service operations are operations designed to fix migration issues and maintenance issues not directly supported by the application. Service operations are available at the following location: RA -> Repositories -> {Repository} -> Service Operations. Below is a list of service operations with descriptions:

1. Replace Language. It is used to replace language of existing objects. Because change of default language is not supported, it can be worked around with replacing languages.
2. Use List and Relation List. It recreates object and graphical links respectively.
3. Proxy. Recreates proxy objects in selected repository. Proxy objects are created in local repository when a link to object in remote repository is created.
4. Xhtml. Rebuilds index used for full text search in XHTML text.
5. Private Workspace. Fixes issue with inheritance of objects from base configuration to private workspace.
6. Object Links. Redirect links to another repository. It is used, for example, in case of merging repositories.

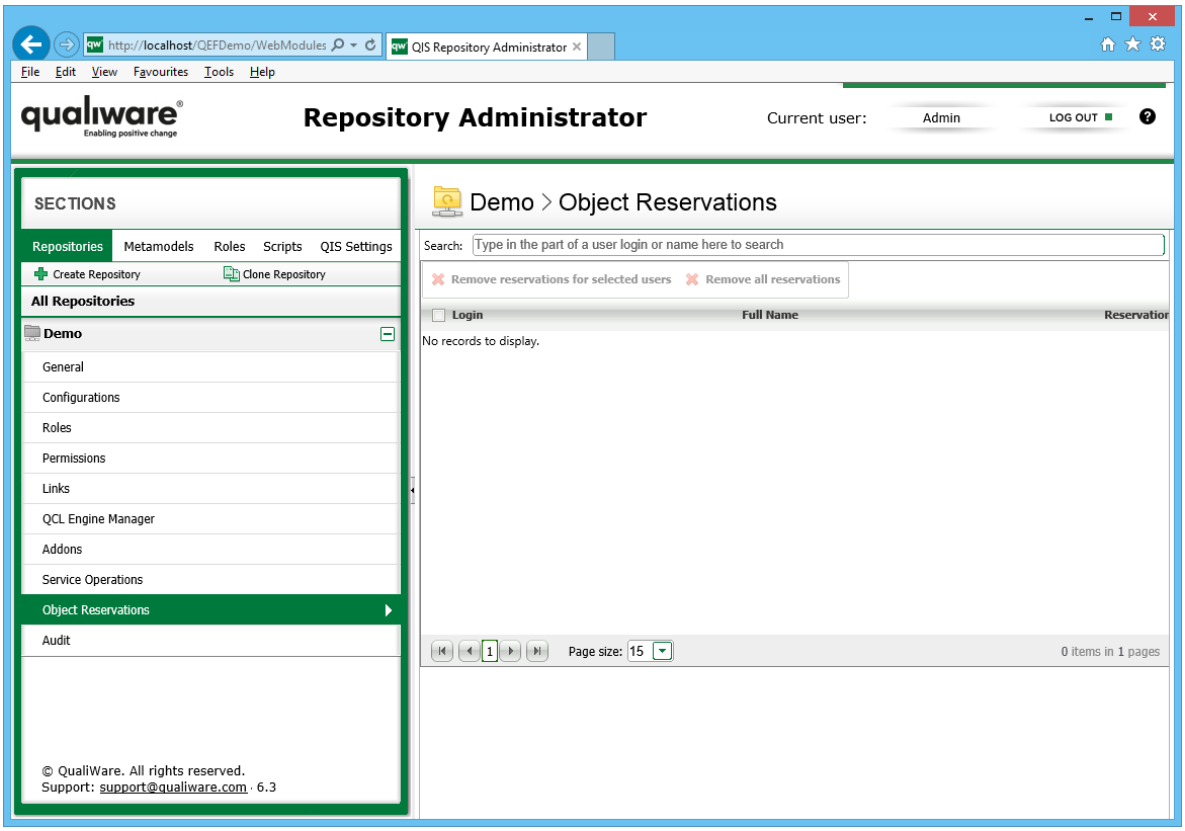





## 3.16 Object Reservations

Objects in repository are protected from concurrent editing by means of locking and reservation. Object locks are bound to session and are stored in memory on the server. When session terminates, all locks bound to session are released. Object reservations are not bound to session, but user, and are stored in database. When session terminates, reservations are not released and exist until they are explicitly released by user. There are situations when it is needed to forcibly release user locks. To do so, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> Object Reservations.
2. Select user whose reservations to release.
3. Click Remove reservations for selected users.



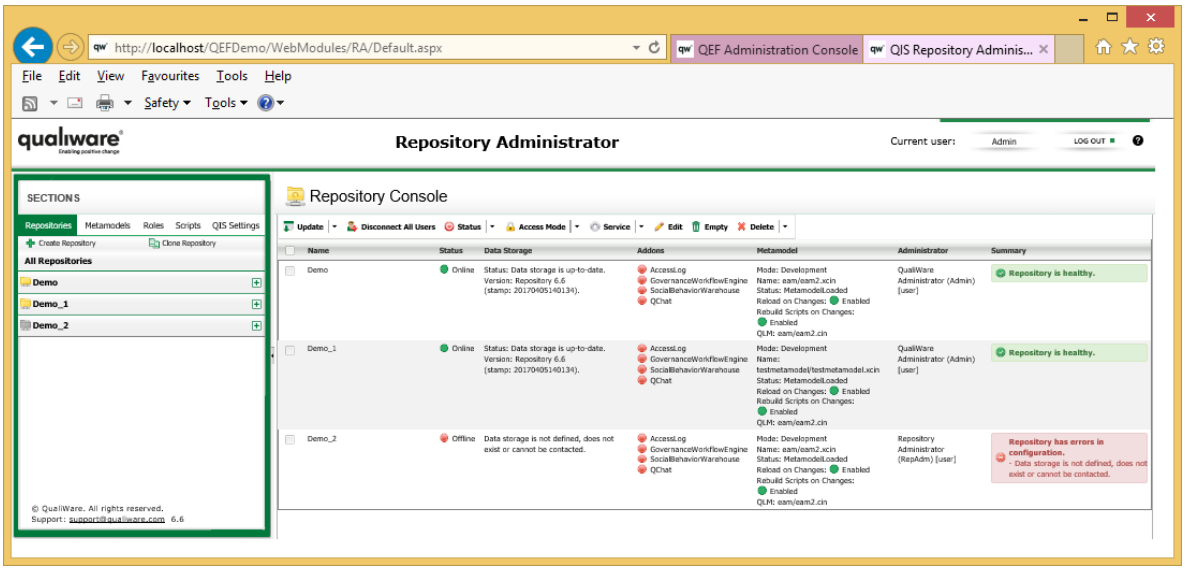

# 4 Repository Panel

Click All Repositories . List of all repositories with main settings, statuses and summary are shown.

Repository Panel provide possibility for massive operations upon repositories:

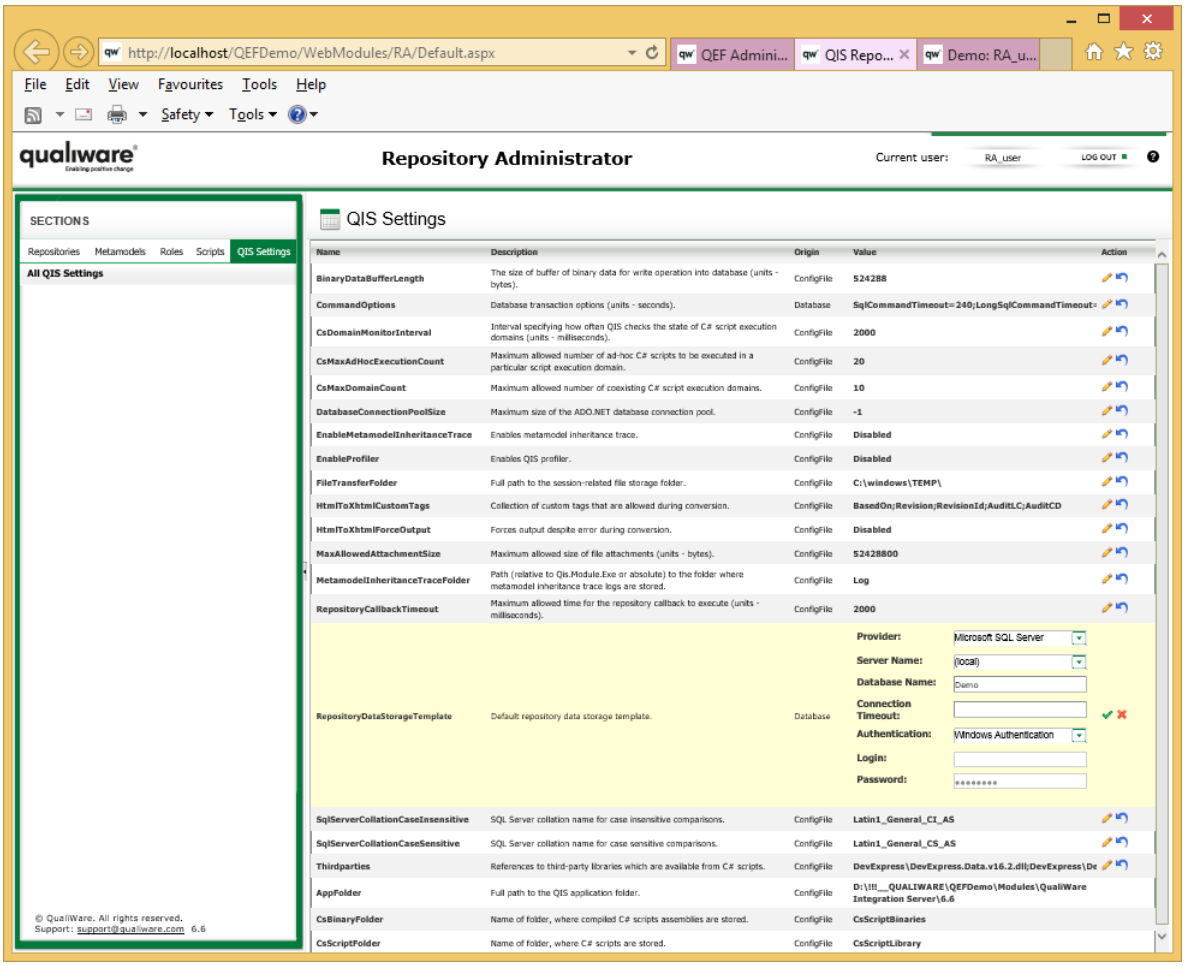
- updating datastorage versions (after installing newer version),
- disconnecting users,
- changing status (Online\Offline),
- switching mode (Read-Only\Read-Write).
- emptying and deleting repositories,
- assigning Administrator of repository,
- service operations.

Select repositories and choose operations to be performed.




# 5 QIS Settings

Default QIS Settings can be changed manually. Click tab QIS Settings.




# 6 Access Log

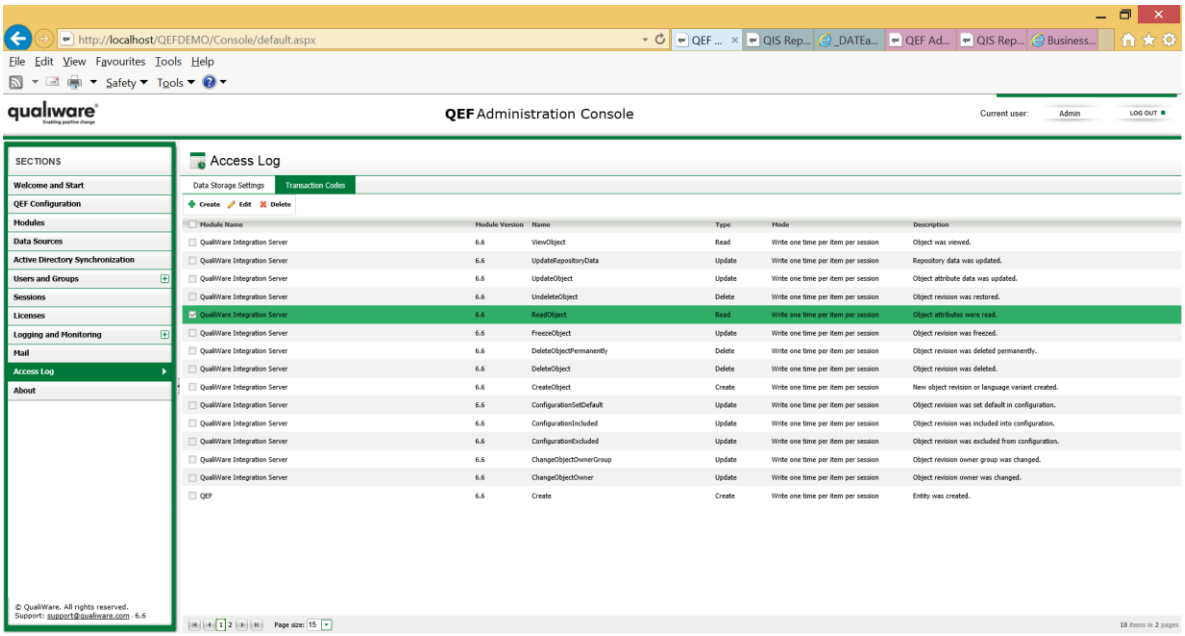
## 6.1 Managing Access Log Transaction Codes

Have your customers ever wanted to know how often their employees read documentation? Or maybe customer wanted to know how often some button in the client application was pressed?

The main idea behind Access Log is recording of events happening during user session. Bundle of built-in events exists for tracing repository objects lifecycle, such as creating, updating or deleting repository objects. Such events are recorded automatically by QIS, and just need to be enabled.

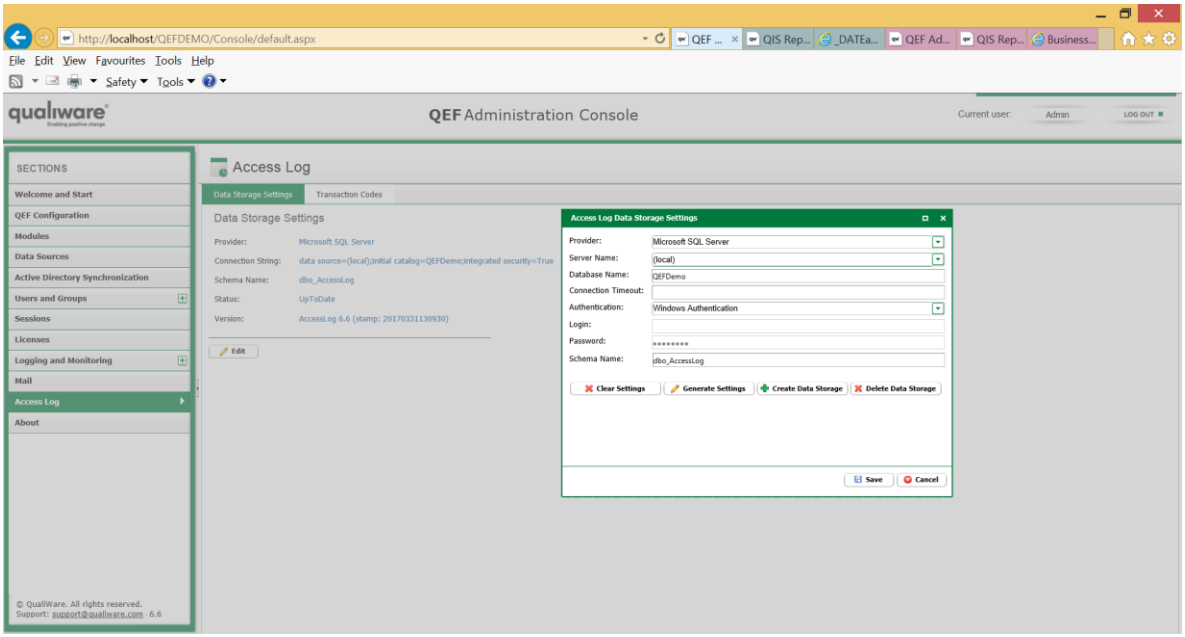
Key elements in Access Log are Transaction codes. Each unique event, either built-in or custom, has its own description in the system. Each description has unique name used as identifier in core products and inside solutions.

Start using Access Log by open QEF Admin Console and pressing Access Log /Transaction Codes.




### 6.1.1 Configuring Data Storage

1. Go to Administration Console -> Access Log ->Data Storage Settings.
2. Click Generate Settings or enter settings manually
3. Click Create Data Storage.
4. Click Save.

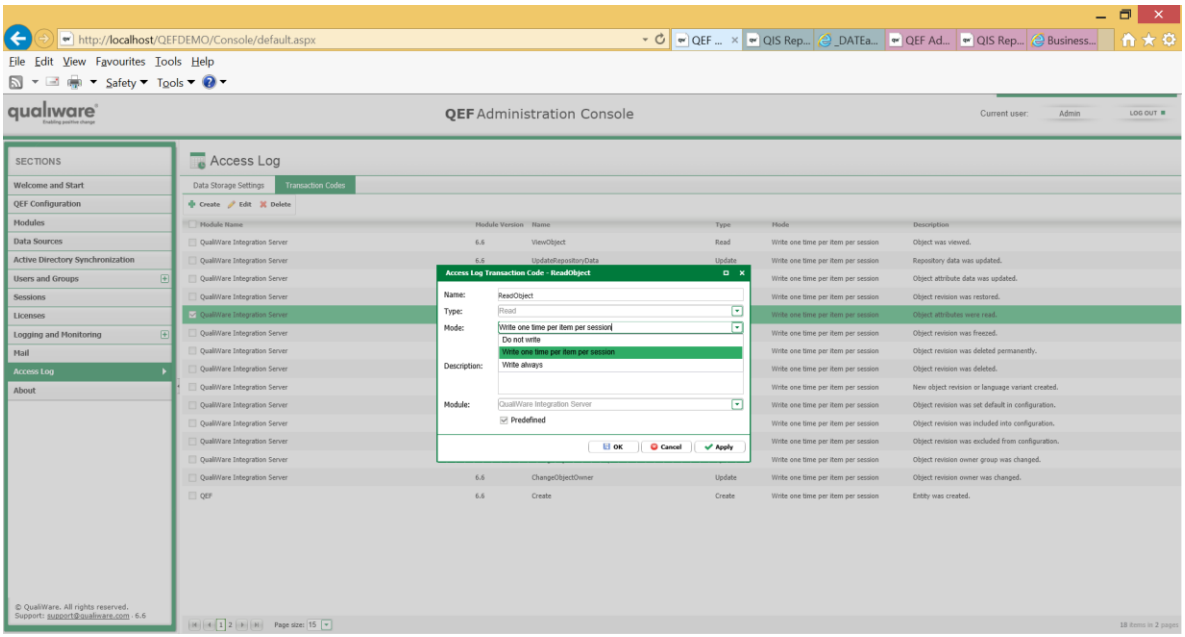



After pressing OK, we can return to Access Log Transaction Codes section for

### 6.1.2 Enabling Transaction Codes

At first, only predefined transaction codes exist. Those are created by QEF and each installed module. Separation by module allows to have two transaction code with same name for different purposes in different modules.
Predefined codes cannot be edited, just viewed, and they cannot be deleted.

Transaction code mode can be changed though.



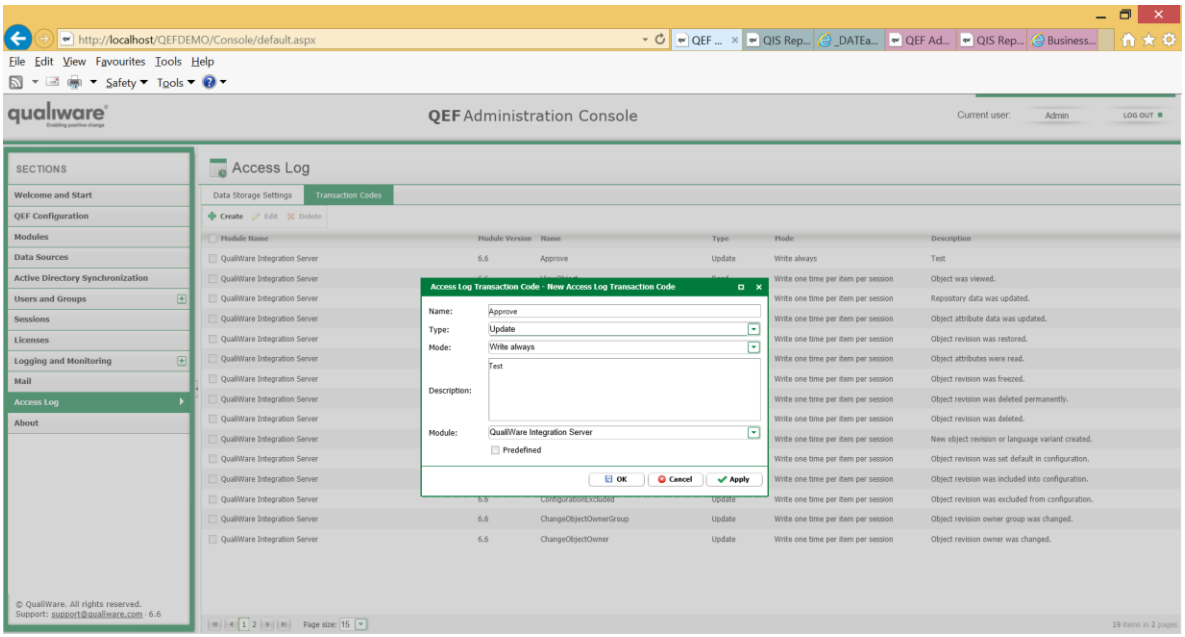

By default transaction code mode is “Write one time per item per session”, what means that only one record will be created for same modified object in same user session. And there is no reason to change it, unless you want to stop recording log for specific transaction code.

Switching off the “ReadObject” transaction code, for example, is possible unless customer needs security audit about who, when and how read repository object attributes.

“Write always” will always log the access code. This mode improves performance a little bit, as the system doesn’t need to verify if the event is already logged, but it can quickly generate huge log. Usually it is not important to know how many times during session user pressed Save button, which is why this mode is not default.

### 6.1.3 Creating Custom Transaction Codes

Sometimes solution developers have to extend the above-mentioned list with their own transaction code.




Procedure:

- Select code name unique for selected module
- Type should correspond meaning of code, as it will be later used in aggregating statistics.
- Description is nice to have for clarity.

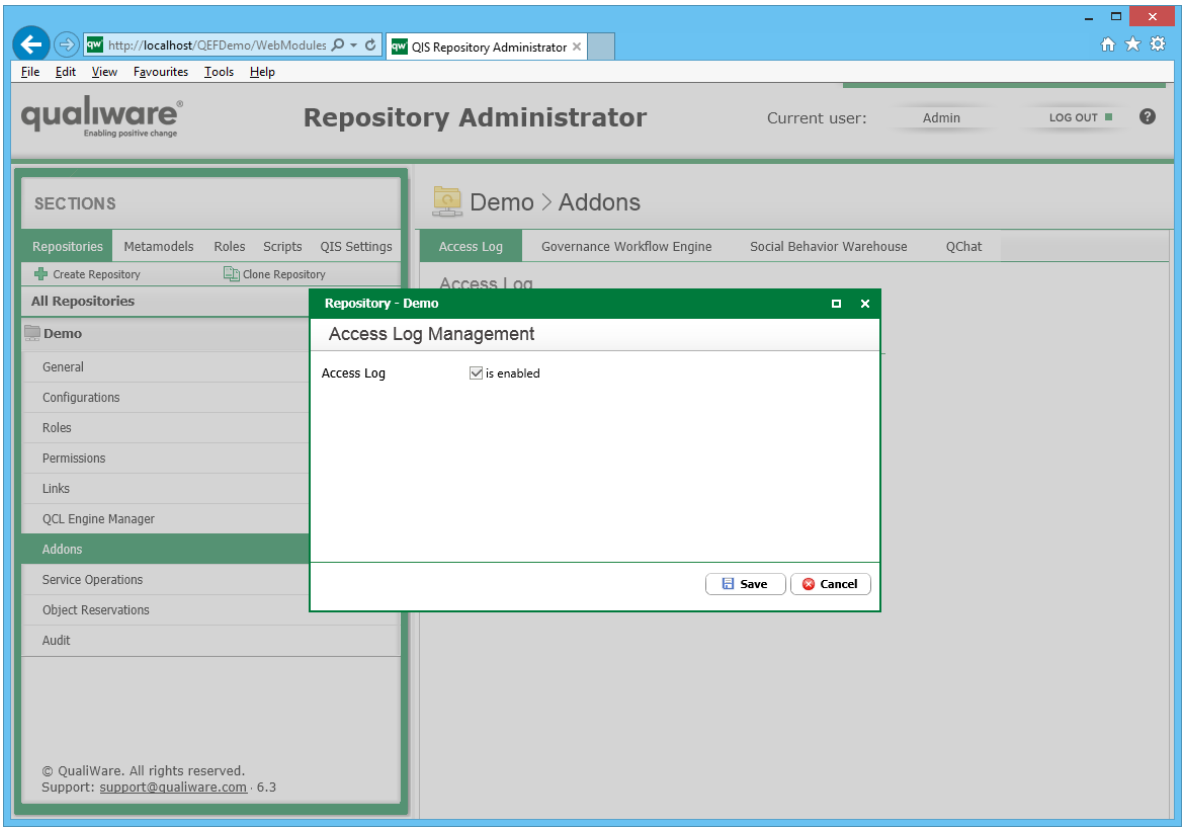
Custom transaction codes are editable and can be deleted. However deleted code remains in database for consistency and can be used in further analysis.

## 6.2 Writing to Access Log

Once Access Log is configured, there is great temptation to use it. Even though Access Log was created as universal tool that can be used in any QEF module, now it is used mostly in QIS. So at first

### 6.2.1 Enabling Access Log for Repository

Please go to repository addons in QIS RA and enable Access Log there.




Enabling/disabling addon also leaves traces in Access Log for security reasons. So that users do not switch off logging without control.

Now all built-in QIS module transaction codes are activated for that particular repository and SBW solution can use it for analysis. However, log record can also be written from script.

### 6.2.2 Writing to Access Log from Script

#### 6.2.2.1 C# example

```
using System.Collections.Generic;
```

```
using Qef.Common.AccessLog;
using Qis.Common;
using Qis.Common.AccessLog; // Add this namespace, if you want to use extension
methods
using Qis.Common.AttributeValues;
using Qis.Common.Scripting.Events.EventHandlerArguments;
using Qis.Common.Scripting.Events.EventHandlerAttributes;

namespace Qis.Module.Scripts
{
    public class TestAccessLog
    {
        #region Fields

        private const string c_descriptionField = "Description";
        private static readonly TransactionCodeId s_descriptionChanged =
            new TransactionCodeId("DescriptionChanged");

        #endregion

        [ObjectChanged("BrowserDiagram")]
        public static void LogIfDescriptionIsChanged(ObjectChangedEventArgs args)
        {
            var savedObj = args.Configuration.FindObject(args.ObjectId);
            if (savedObj == null)
            {
                return;
            }

            if (args.OldData.Attributes[c_descriptionField]
                .Value.GetContent() == savedObj.Attributes[c_descriptionField]
                .Value.GetContent())
            {
                return;
            }

            // During saving log record,
            // you can easily store additional information in key-value way
            // Here object name and description are put into dictionary
            var extraInfo = new Dictionary<string, string>()
            {
                { "Name", savedObj.Attributes["Name"].Value.GetContent() },
                {
                    c_descriptionField,
                    savedObj.Attributes[c_descriptionField].Value.GetContent()
                }
            };

            // Sometimes it is convenient to pass object instance
            args.Qis.LogAccess(savedObj, s_descriptionChanged);

            // And it is possible to add extra information we created before
            args.Qis.LogAccess(savedObj, s_descriptionChanged, extraInfo);

            // However if you don't have object instance
```

```
            args.Qis.LogAccess(
                args.RepositoryId,
                args.ConfigurationId,
                args.ObjectId,
                s_descriptionChanged);

            // Or if you need absolutely unique record
            args.Qis.LogAccess(
                AccessSource.Create("InTheMiddleOfNowhere"),
                AccessEntryId.Create(args.ObjectId.Revision.ToString()),
                s_descriptionChanged);
        }
    }
}
```

#### 6.2.2.2 QCL example

For generic log record:

```
local source = 'AlmostQLM';
local entryId = '0001';
local transactionCodeId = 'FromQLM';
local extraData = "key1\tvalue1\nkey2\tvalue2";
WriteToAccessLog(source, entryId, transactionCodeId, extraData);
```

For object log record:

```
local transactionCodeId = 'FromQLM';
local extraData = "key1\tvalue1\nkey2\tvalue2";
WriteToAccessLogForRepository("Audit Report", transactionCodeId, extraData);
```

## 6.3 Reading Access Log

Sometimes it is necessary to do a manual analysis of the Access Logs. The Social Behavior Warehouse Module does a good job at providing analysis but this is how you should process if you want to conduct a manual analysis:

### 6.3.1 Access Log Database Structure

Access log database contains:

1. Log table – actual log records
2. LogData table – contains key-value extra information.
3. Modules table – contains information about module, from which log record was created.
4. TransactionCodes table – enumerates transaction codes that are used in existing log.

### 6.3.2 Example of Log after Calling Test Scripts

#### 6.3.2.1 C# Example

When a new transaction code was is defined and used in script, record appears in TransactionCodes table:

| | Id | ModuleId | CodeId | CodeDescription | ValidFrom | ValidTo | IsPredefined | CodeType | CodeMode |
|---|---|---|---|---|---|---|---|---|---|
| 25 | 25 | 3 | DescriptionChanged | No description added yet. | 2014-08-05 12:41:46.477 | NULL | 0 | -1 | -1 |

Actual log records are:

| | Id | ModuleId | UserLo... | Source | EntryId | TransactionCod... | DateTime | Session | Recurrence | EntryVersion |
|---|---|---|---|---|---|---|---|---|---|---|
| 235 | 235 | 3 | q | A/db515458-8019-460a-94fe-467fab268f0c | e56ff080-8240-43b5-8cbd-4164f3aaf3f8/7c9bb2dc-edb2-... | 25 | 2014-08-08 11:13:17.253 | B629E415-B86B-4150-8956-C86772C16774 | 3 | 0x0100000000000000 |
| 236 | 236 | 3 | q | InTheMiddleOfNowhere | 7c9bb2dc-edb2-df77-47d1-f5ef3f45b46b | 25 | 2014-08-08 11:13:17.260 | B629E415-B86B-4150-8956-C86772C16774 | 1 | 0x0100000000000000 |

As you can see, these two records have transaction code 25, which matches DescriptionChange.
And first record has recurrence 3, which matches amount of non-custom calls in C# script.

As we added special value for log record, it can be found in LogData table:

| | LogId | Key | Value |
|---|---|---|---|
| 687 | 235 | Description | Test |
| 688 | 235 | Name | Audit Planning |

#### 6.3.2.2 QCL Example

When a new transaction code was defined and used in script, in TransactionCodes table appeared record:

| | Id | ModuleId | CodeId | CodeDescription | ValidFrom | ValidTo | IsPredefined | CodeType | CodeMode |
|---|---|---|---|---|---|---|---|---|---|
| 26 | 26 | 3 | FromQLM | No description added yet. | 2014-08-05 12:53:17.273 | NULL | 0 | -1 | -1 |

Actual log records are:

| | Id | ModuleId | UserLo... | Source | EntryId | TransactionCod... | DateTime | Session | Recurren... | EntryVersion |
|---|---|---|---|---|---|---|---|---|---|---|
| 251 | 251 | 3 | q | AlmostQLM | 0001 | 26 | 2014-08-08 13:29:36.680 | C0D966D6-E239-4AD3-804B-CBF0425095... | 1 | 0x0100000000000000 |
| 252 | 252 | 3 | q | A/db515458-8019-460a-94fe-467fab268f0c | b55bb3e7-4c30-49bc-96d8-6757071f1df6 | 26 | 2014-08-08 13:31:36.000 | C0D966D6-E239-4AD3-804B-CBF0425095... | 1 | 0x0100000000000000 |

As you can see, these two records have transaction code 26, which matches FromQLM.

As we added special value for log record, it can be found in LogData table:

| | LogId | Key | Value |
|---|---|---|---|
| 731 | 251 | key1 | value1 |
| 732 | 251 | key2 | value2 |

# 7 Enabling and Configure GWE

## 7.1 Preface

This document describes installation of the standard solution Governance Workflow Engine (hereinafter, GWE).

## 7.2 Pre-Installation checklist

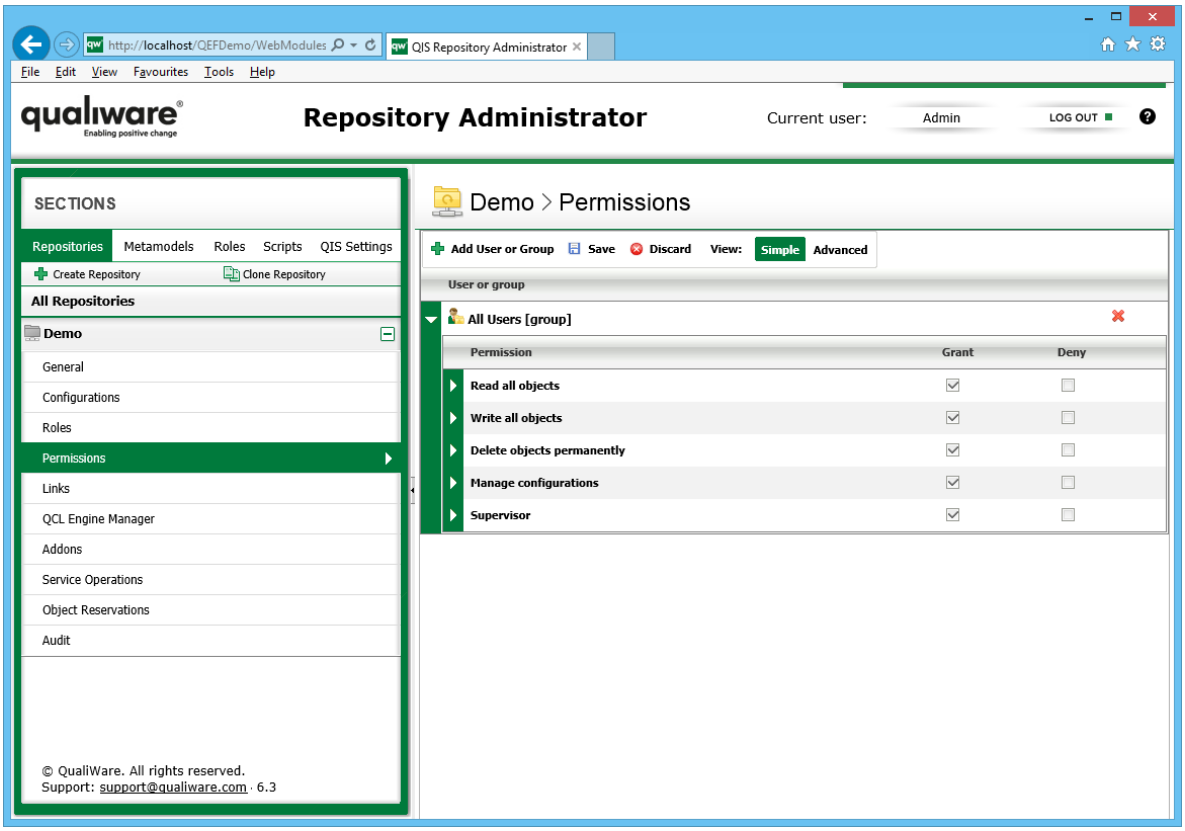
The document assumes the reader has installed QEF, QIS, Web Forms, RA and QLM.

Before GWE is installed, it is advised that repository is created and configured (see Creation of repositories.docx and One time configuration of repositories.docx) and QEP is installed (see Publishing.docx).

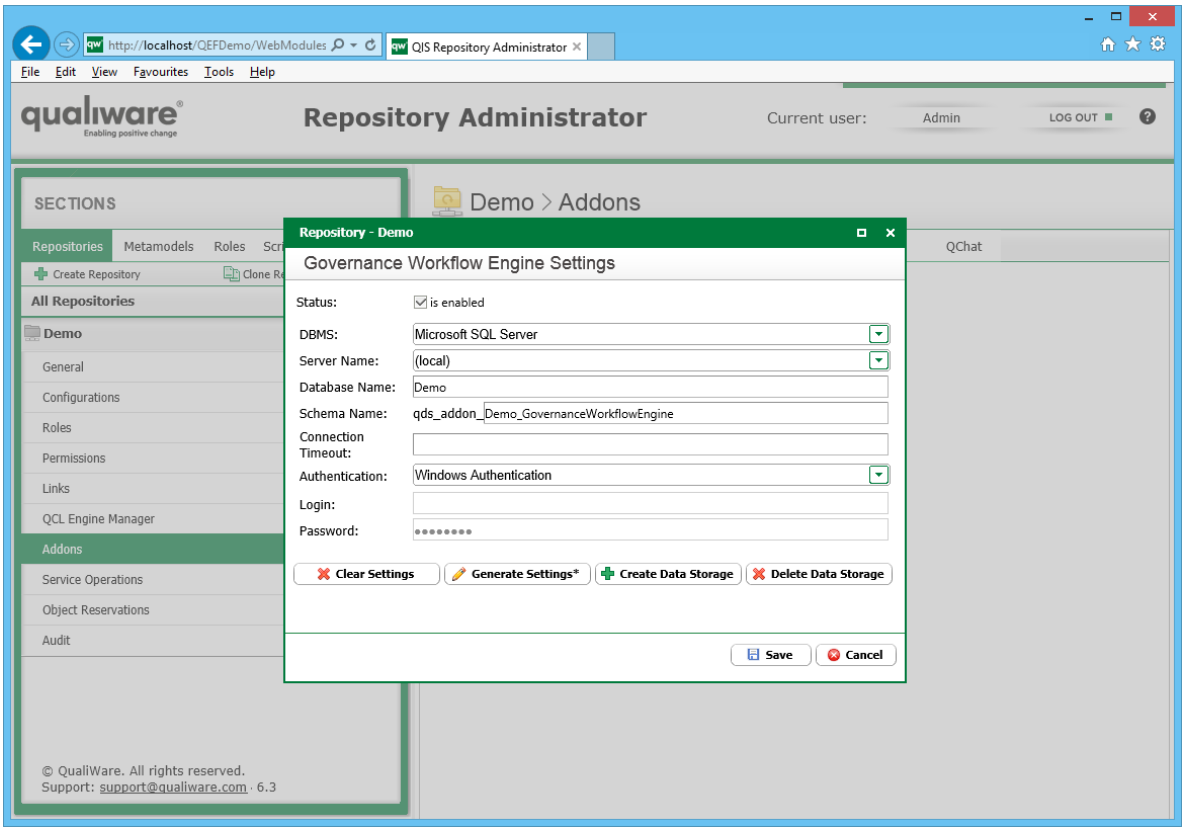
User account used to perform the installation must be QEF and QIS administrator.

## 7.3 Installation of Governance Workflow Engine

1. Go to RA.
2. Select existing repository or create a new one.
3. Go to the Permissions page.
4. Assign the Supervisor permission to user or group, which will be used to install the solution from QLM.




1. Go to the **Addons** page.
2. Select **Governance Workflow Engine**.
3. Click **Edit**.
4. Check **Status** so that it is enabled.
5. Click **Generate Settings** and then modify the settings according to the needs or enter all settings manually.




6. Click Create Data Storage.
7. Click OK.
8. Start QLM and connect to the repository with user from the step 4.
9. Click Yes when asked to import GWE objects.

# 8 Enabling and Configure SBW

## 8.1 Preface

This document describes the main purpose, structure of Social Behavior Warehouse (hereinafter, SBW), and its installation and configuration procedure.

## 8.2 General Information

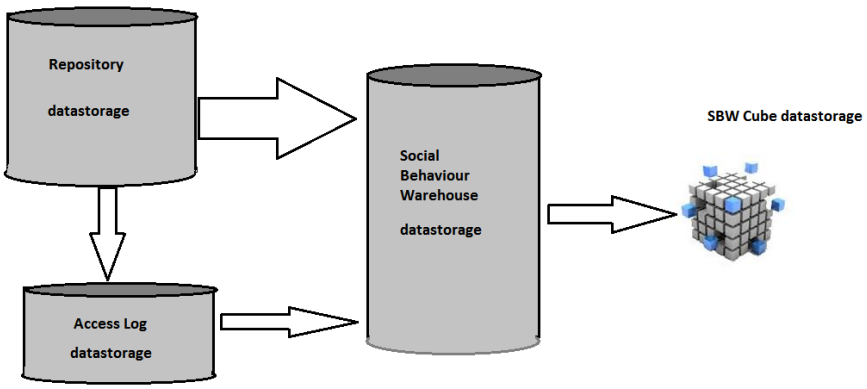
SBW is designed to collect and report activities performed against data stored in repositories. Collected data is analyzed, extended and made available by means of charts and reports, part of Analytics metamodel.

SBW helps to answer the following questions:

- What parts of the enterprise knowledge is used by organization?
- Where is the knowledge used? Web, mobile, thick client?
- How active are different parts of organization?

- How often do employees use core process models to guide them in their day to day operations?
- How much time does it take for a business model to be approved and penetrate the business?
- And other.

SBW is represented as a QIS add-on, which gets its data from repository and Access Log. Configuration of SBW includes configuring and enabling of Access Log and SBW. Below is a general illustration of how repository, Access Log and SBW data storages are connected.




## 8.3 Pre-Installation Checklist

The document assumes the reader has installed QEF, QIS, Web Forms, RA, QTS and QLM.

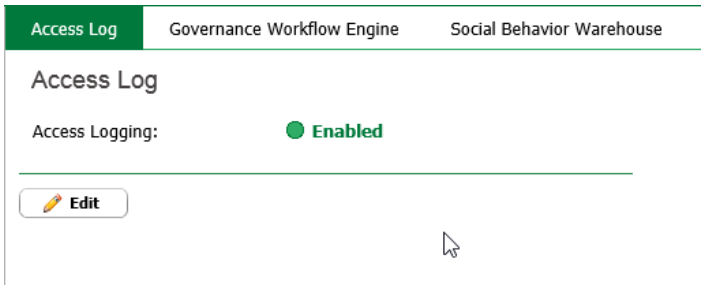
Before SBW is installed, the following must be performed:

- SQL Server Analysis Services is installed.
- Access Log is created and configured (see Repositories and metamodels - One time configuration of repositories - Access Log.docx).
- Repository is created and configured (see Creation of repositories.docx and One time configuration of repositories.docx).
- Repository is set to use Analytics metamodel.

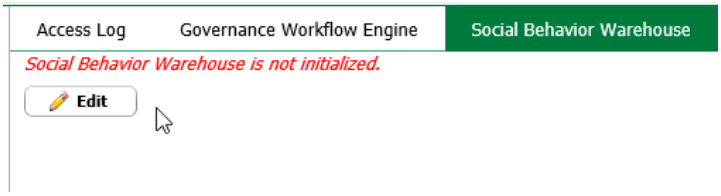
User account used to perform the installation must be QEF and QIS administrator.

## 8.4 Installation of Social Behavior Warehouse

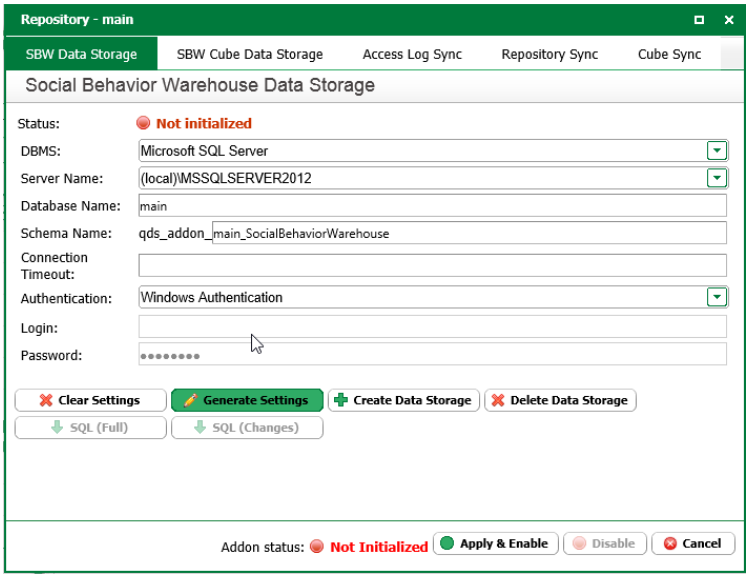
5. Go to QIS RA.
6. Select existing repository or create a new one.
7. Go to Addons -> Access Log.
8. Click Edit.
9. Check is enabled.
10. Click OK.




11. Go to **Addons -> Social Behavior Warehouse**.
12. Click **Edit**.
    *NOTE. If "Analytics is not supported by the current metamodel." message is displayed, please check whether Analytics metamodel is included into the current repository metamodel.*

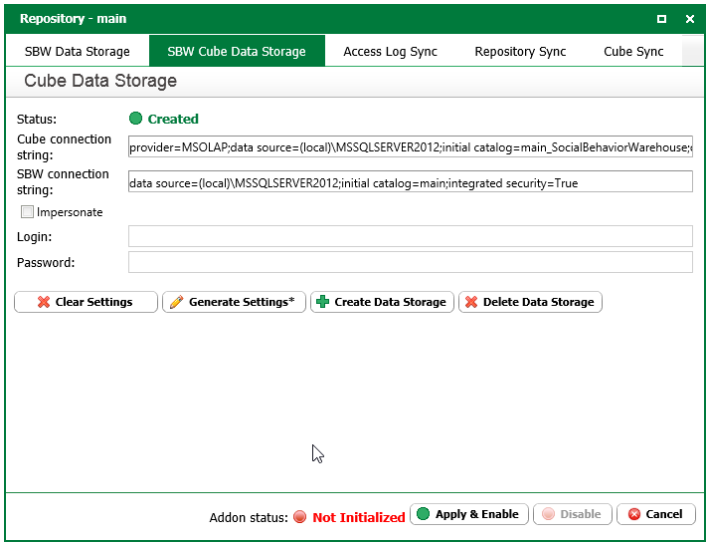



13. Go to **SBW Data Storage** tab.
14. Click **Generate Settings** and then modify the settings according to the needs or enter all settings manually.

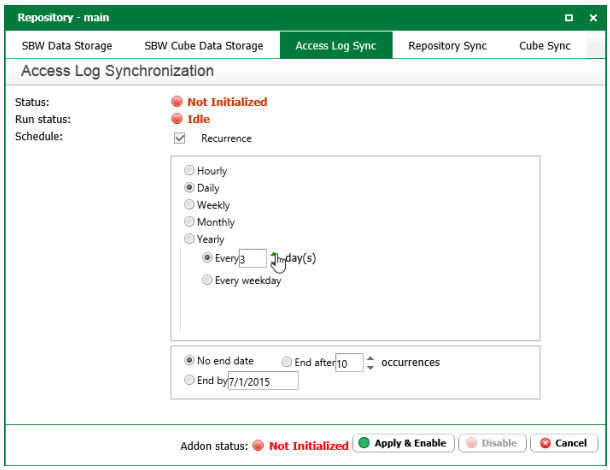



10. Click **Create Data Storage** tab. Please check that **Status** changed to **Created.**

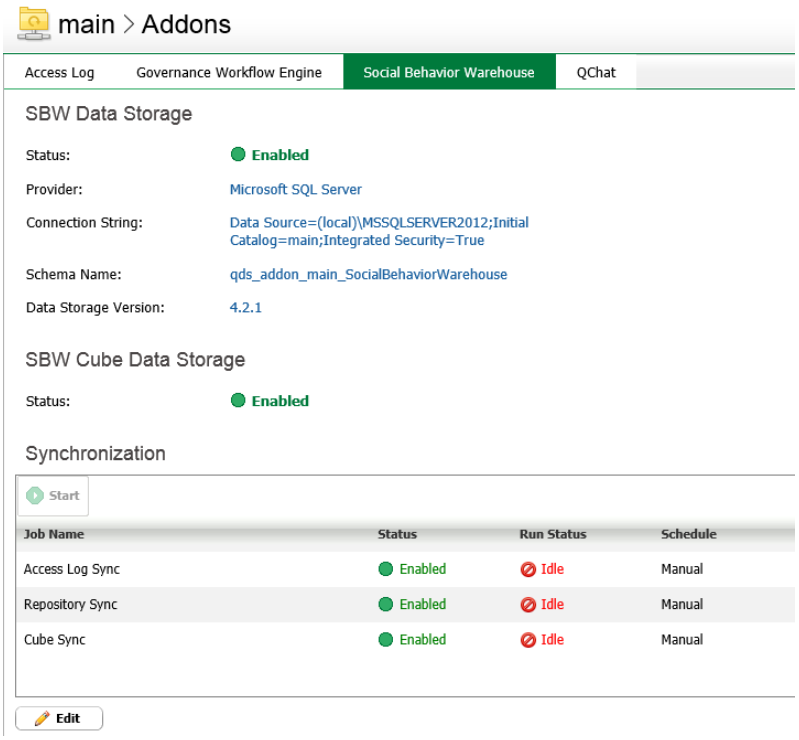
11. Go to **SBW Cube Data Storage** tab. Perform p. 10-11. For the impersonation and cube access details, refer to the **section 4 (Setting up Permissions)** of this manual. For each repository its own cube database should be defined.




12. Go to **Access Log Sync** tab.
13. Define **recurrence** schedule if necessary. Otherwise, manual synchronization will be required.

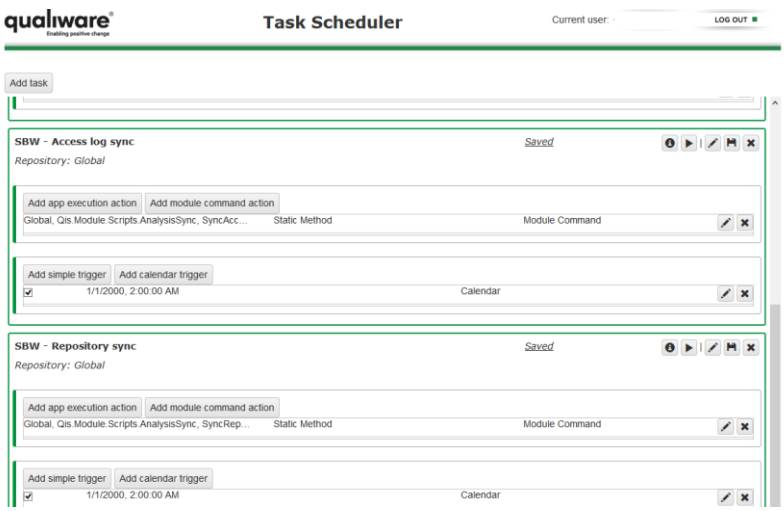



14. Go to **Repository Sync** tab. Perform p. 14.
15. Go to **Cube Sync** tab. Perform p. 14.
16. Click **Apply & Enable**. Close the dialog.
17. Check that **Addon**, **Cube data storages** and **Synchronization** tasks are all **Enabled**.




## 8.5 View Jobs Schedule (QTS)

- If synchronization jobs are defined as scheduled (not manual), they can be seen and edited in Task Scheduler manager.
- If user edits synchronization job in QTS, changes are not reflected in SBW settings.
- If user changes schedule of SBW synchronization jobs in QTS, jobs are performed according to QTS schedule.



## 8.6 Setting up Permissions

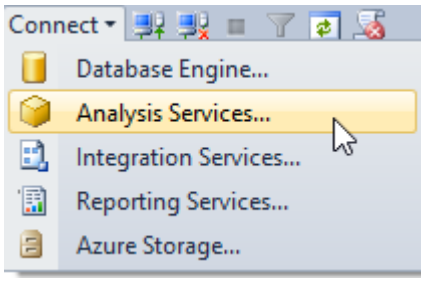
There are 3 categories of permissions to be set up:

- QIS Access to SQL Server Analysis Services.
- SQL Server Analysis Services access to Repository Data Storage.
- QIS Web Forms access to SQL Server Analysis Services.

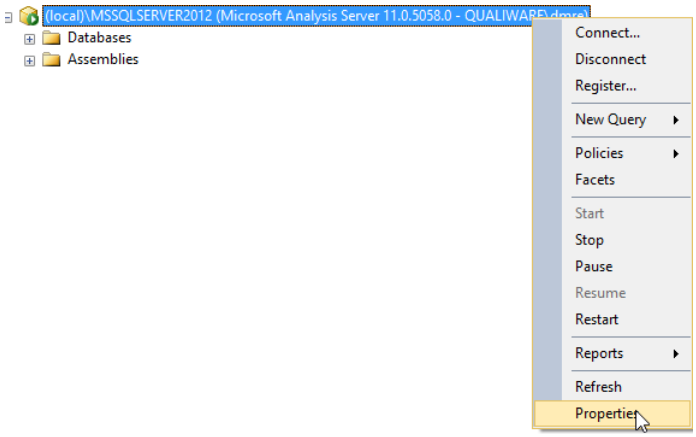
### 8.6.1 QIS Access to Sql Server Analysis Services

QIS requires permissions to create Analysis Services database (cube data storage). To provide such permissions, perform the following:

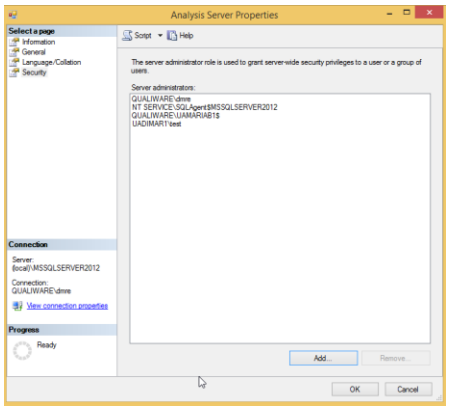
1. Start SQL Server Management Studio.
2. Connect to Analysis Services.




3. Right-click Analysis Services database and select Properties.

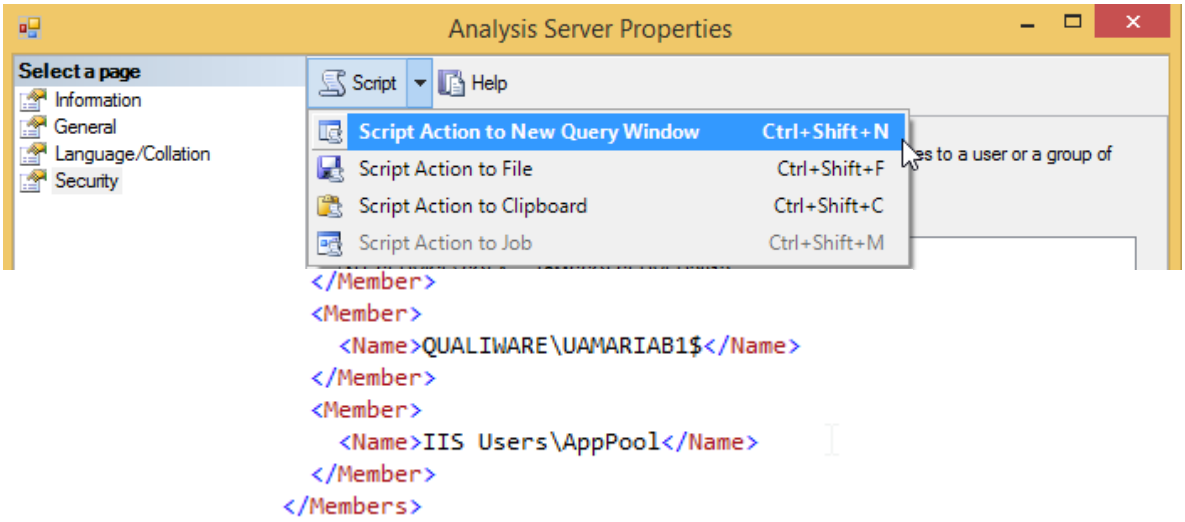



4. On Security tab add user account used to access/create Analysis Services database to server-wide administrators. This account may be specified in Cube Data Storage settings via connection string.




*NOTE: It is advised to use Integrated Security for security reasons. Thus, the account used to access Analysis Services would be QIS server account, and may be found using Task Manager.*

5. In case the user cannot be found via "**Add...**" dialog, click **Script** -> **Script Action to New Query Window** and type in the account by hand.

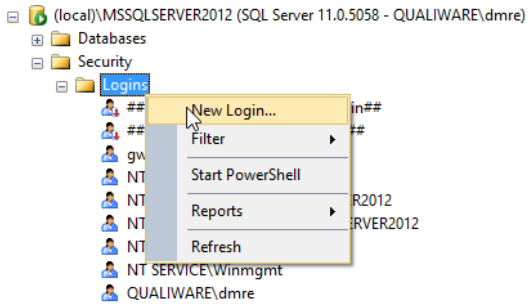



### 8.6.2 SQL Server Analysis Services access to Repository Data Storage

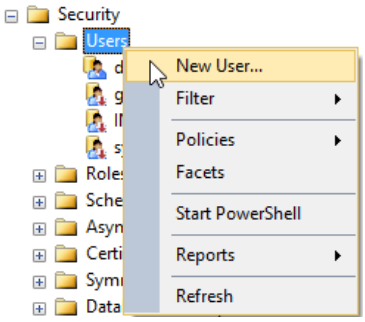
During cube processing, triggered either manually via **QIS RA**, **SQL Server Management Studio** or automatically per specified schedule, cube is filled by the data formed from the repository data storage and **SBW** data storage. For this operation to succeed, **Analysis Services** needs read-only permissions to these data storages.

To add such permissions, perform the following:

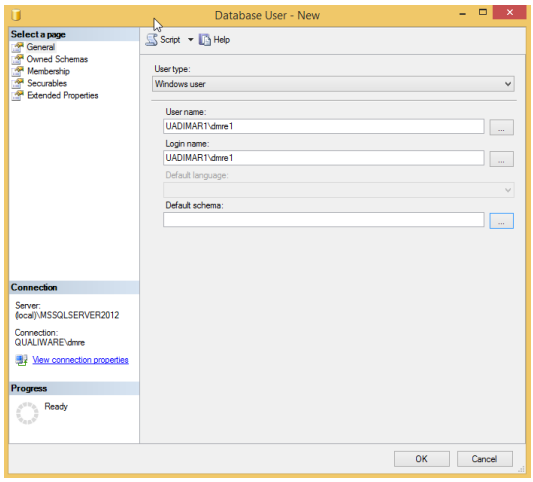
1. Start **SQL Server Management Studio**
2. Connect to **Database Engine**, where Repository and **SBW** data storages are located.
3. Expand **Security** node. Right-click **Logins**, select **New Login**.



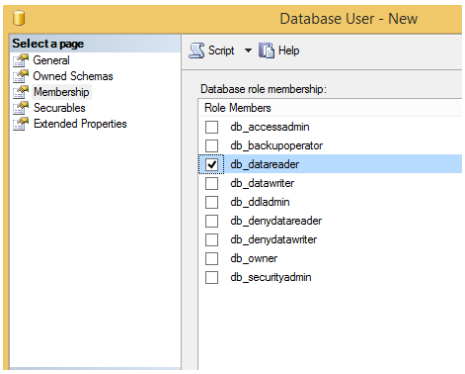

4. Add account used by analytical processing to access repository and SBW data storages.
   *NOTE: In case impersonation is not used, it is the account Analysis Services is running under. To identify this account, please go to Task Manager and see the user for msmdsrv.exe process. Otherwise, it is the account specified for impersonation (please, note that this account may be only windows account, not SQL server account).*
5. Expand repository database.
6. Expand Security node. Right-click Users and select New User.




7. Add user specified in step 4.




8. On **Membership** tab check **db_datareader** permission.



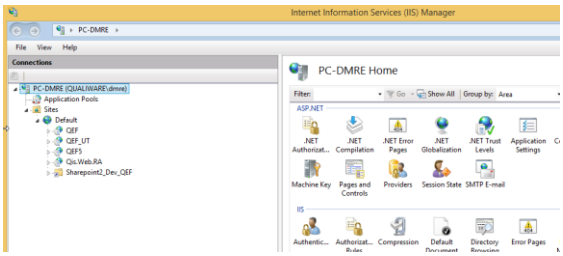

9. Click **OK**.

### 8.6.3 QIS Web Forms access to SQL Server Analysis Services

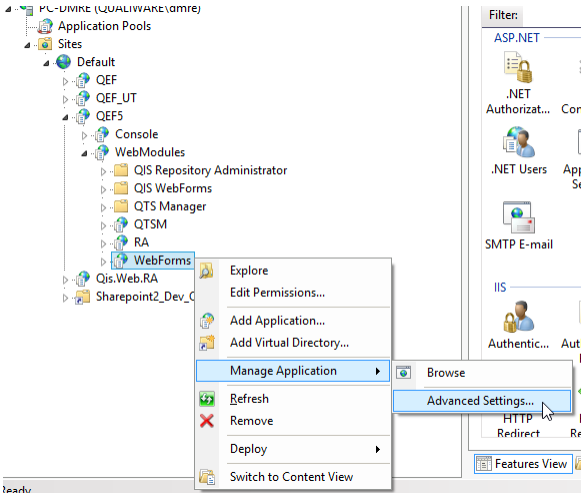
In order for **SBW** to work from solution, additional permissions need to be given to **QIS Web Forms** to access **Analysis Services**. To do this, execute the steps in **section 4.1** for **QIS Web Forms** user account.

To identify **QIS Web Forms** user account, perform the following:

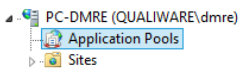
1. Start **Internet Information Services Manager**

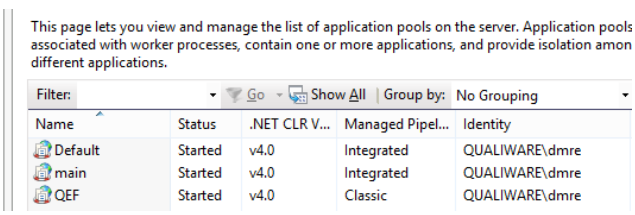



2. Expand **Sites**. Find application, related to current **QEF** instance.
3. Expand **WebModules**.
4. Right-click **WebForms** -> **Manage Application** -> **Advanced Settings...**



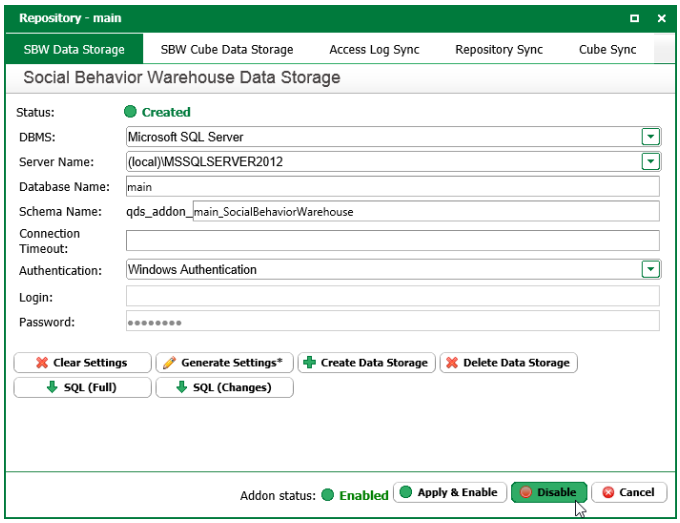

5. Check the **Application Pool** value.
6. Go to **Application Pools**.
7. The "**Identity**" set for the application pool is the user account that needs access to **Analysis Services**.



## 8.7 Uninstalling Social Behavior Warehouse

1. Open QIS RA.
2. Select existing repository.
3. Go to Addons.
4. Select Social Behavior Warehouse tab.
5. Click Edit.




6. On Access Log Sync, Repository Sync and Cube Sync tabs uncheck recurrency.
7. Click Apply & Enable.
8. Click Disable.
9. On SBW Data Storage tab, click Delete Data Storage.
10. On SBW Cube Data Storage tab, click Delete Data Storage.
11. Close the window.

# 9 Enabling and Configure QCLE

## 9.1 Preface

This document describes steps necessary to create, configure and enable QCLE instances.

## 9.2 Prerequisites

In order to setup QCLE instances, the following requirements must be met:

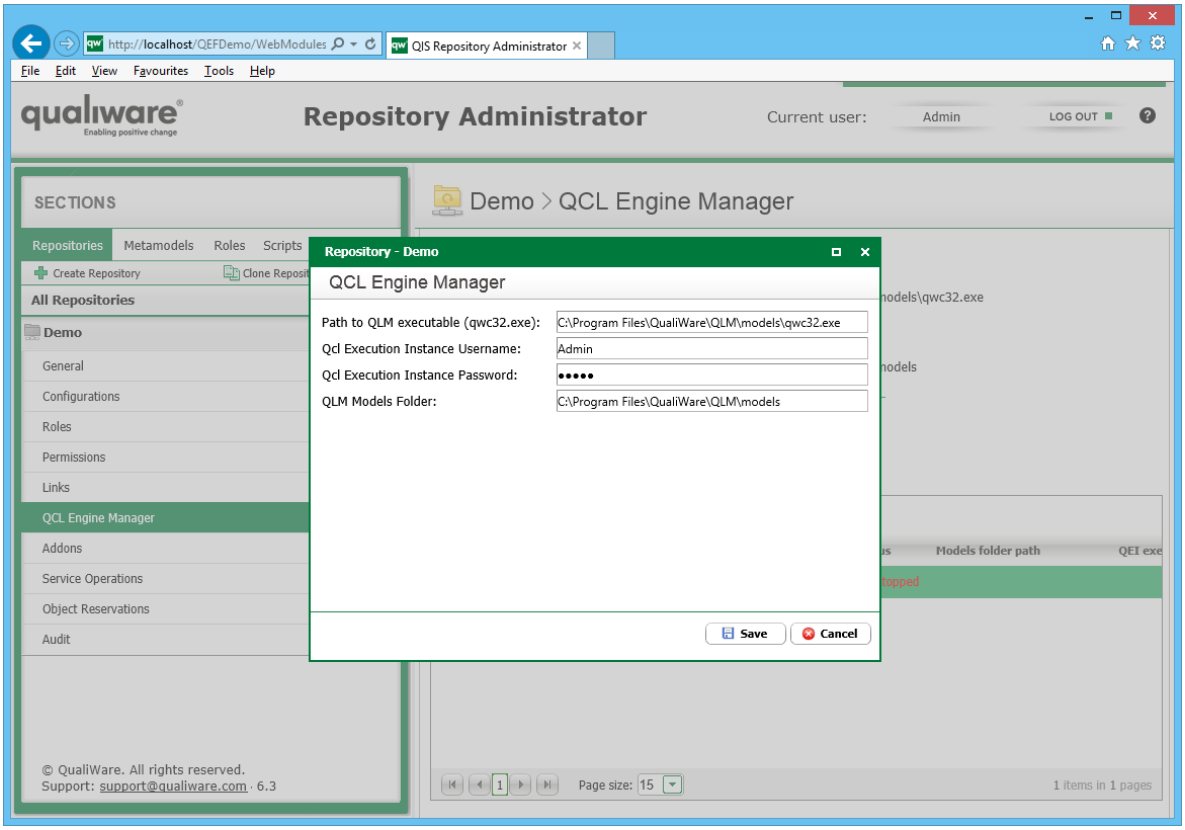
- QIS module is installed and running;
- QCLE module is installed and running;
- RA module is installed and running.

## 9.3 Configuration of QCLE Execution Pool

1. Go to RA -> Repositories -> {Repository} -> QCL Engine Manager.
2. Click Edit.
3. Enter user login and password.

   Note: It is considered best practice to create a single separate user for QEI instances with QIS Administrator role in order to use it as a part of QEI publishing process, since it is generally operated via QualiWareSetup which is user-specific.

4. Optionally, enter path to QLM models folder. It must be an absolute path on the server, where QCLE module is running. In case it is absent, default path will be used.
5. Optionally, enter path to QLM executable file. It can be absolute or relative. In case it is absent, default path will be used.
6. Click Save.



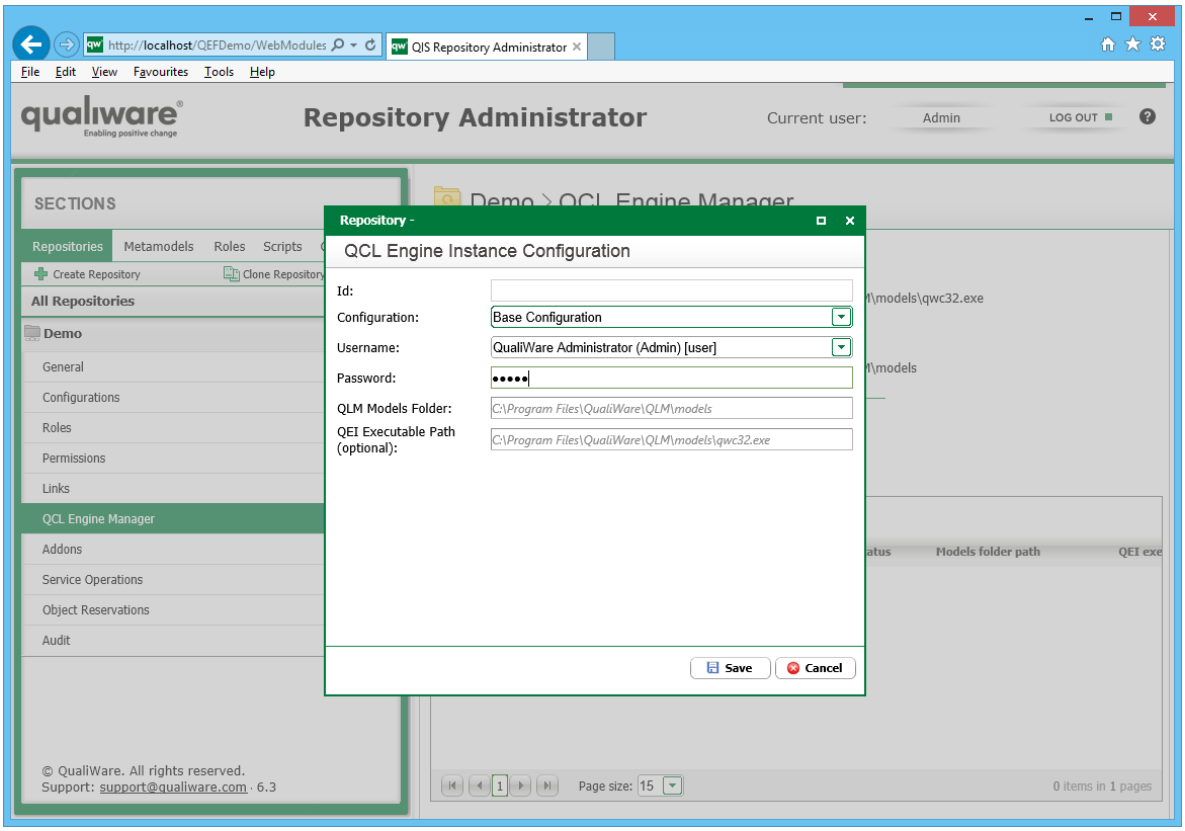

## 9.4 Configuration of QCLE Instances

In order to add or edit QCLE instance, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> QCL Engine Manager.
2. To add a new QCLE instance, click Add, or to edit existing QCLE instance, select specific instance and click Edit.
3. Select configuration to connect to.
4. Enter user login and password.

   Note: It is considered best practice to create a single separate user for QEI instances with QIS Administrator role in order to use it as a part of QEI publishing process, since it is generally operated via QualiWareSetup which is user-specific.

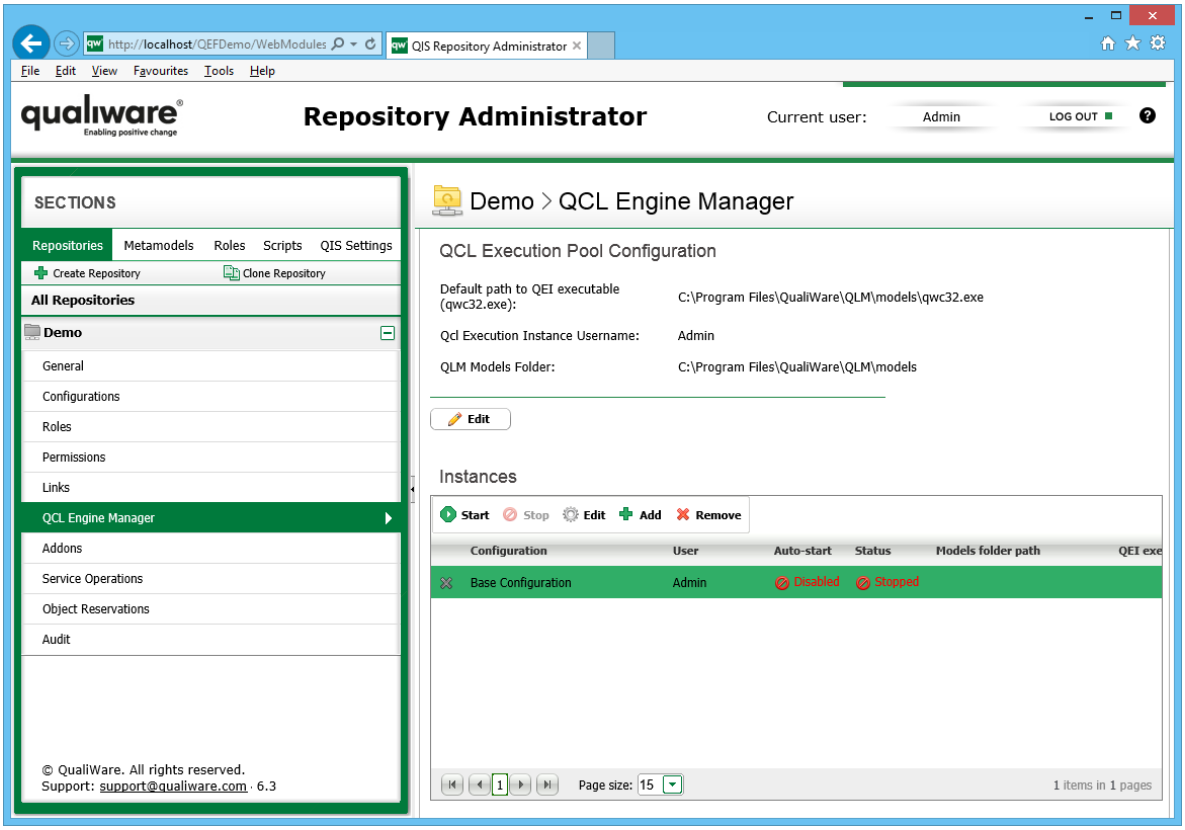
5. Optionally, enter path to QLM models folder. It must be an absolute path on the server, where QCLE module is running. In case it is absent, default path will be used.
6. Optionally, enter path to QLM executable file. It can be absolute or relative. In case it is absent, default path will be used.
7. Click Save.




In order to delete QCLE instance, follow the steps:

1. Go to RA -> Repositories -> {Repository} -> QCL Engine Manager.

2. Select specific instance and click **Remove**.




## 9.5 Working with QCLE Instances

QCLE instances may be either enabled or disabled. Whenever QCLE instance is enabled, QCLE module will automatically restart the specified instance upon unexpected crashes. Disabled instances are stopped by QCLE module and are not run.

QCLE module also monitors QCLE instances for configuration changes (e.g. user password change) and will automatically stop/start erroneous/correct instances in case they are in “enabled” state.

In order to enable QCLE instance, follow the steps:

1. Go to **RA -> Repositories -> {Repository} -> QCL Engine Manager**.
2. Select specific instance and click **Start**.

After these steps, the following is expected:

- Instance “Auto-start” property is set to “Enabled”
- Instance “Status” property is set to either “Starting” or “Running”

- After several seconds upon page refresh, instance status should say “Running”. If it is not so, see General Troubleshooting.

In order to disable QCLE instance, perform the following steps:

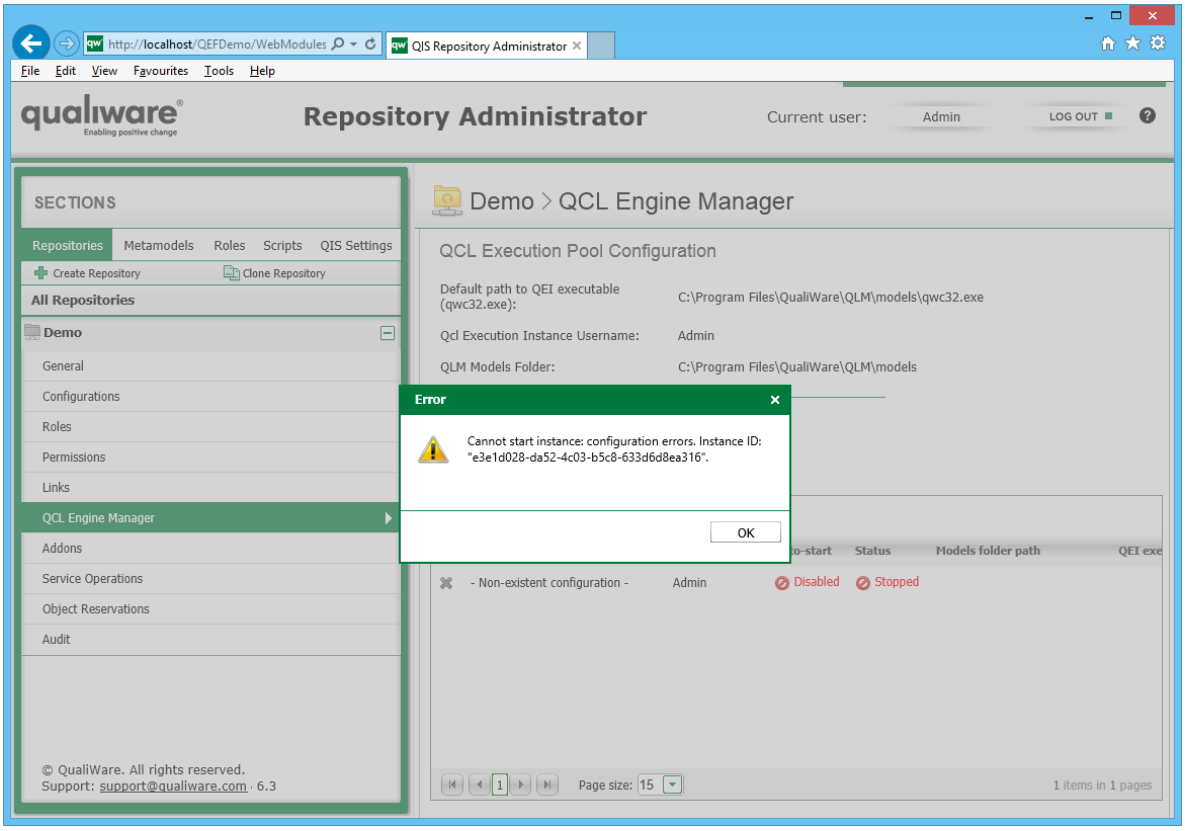
1. Go to **RA -> Repositories -> {Repository} -> QCL Engine Manager**.
2. Select specific instance and click **Stop**.

After these steps, the following is expected:

- Instance “Auto-start” property is set to “Disabled”
- Instance “Status” property is set to either “Stopping” or “Stopped”
- After several seconds upon page refresh, instance status should say “Stopped”.

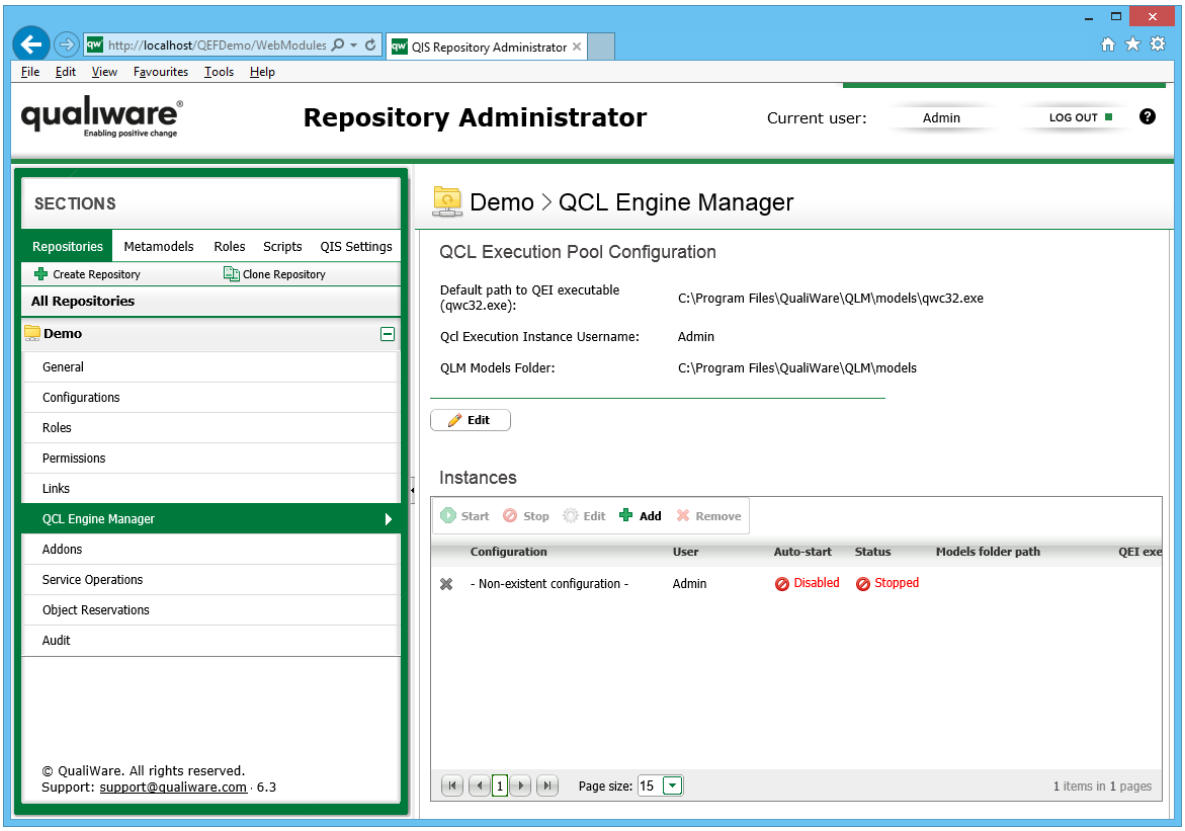
## 9.6 General Troubleshooting

Q: Whenever QCLE instance is started, it gives the following message:




Additionally, the instance icon is grey.

A: Icon to the left is used to show whether the configuration for the specific instance is correct. In case of correct configuration green mark will be seen; otherwise, it is gray:




Incorrect instance configuration may mean one of the following:

- Specified user and/or password are invalid.
- Specified configuration no longer exists in the repository.
- Current repository is in offline mode.

In order to proceed, please verify that these parameters are correct and refresh the page.

Q: Instance is always in “Starting” state.

A: This may be caused by several reasons:

- Instance has exited right after start due to misconfiguration or crash;
- Instance cannot maintain contact with either QEF, QIS or QCLE module.

  Please verify that QEF address and port are correctly configured for QCLE instance executable.

  Try to start it manually and connect to the particular repository and configuration pair using QCLE instance credentials.

  If that does not help, please check for errors in either QEF Log or Windows Event Log.

Q: QCLE module is stopped when started together with QIS.

A: QIS upon start takes some time to prepare metamodels and compile scripts; if it does not respond in 30 seconds timeframe, QCLE module shuts down. Restarting the module will fix the problem.

## 9.7 Appendix A. Error Messages

These error messages may appear in QEF log or in RA.

| Message | Meaning |
|---|---|
| "QCL Engine Instance with ID='...' did not respond in timely fashion and was stopped." | QCLE Instance couldn't connect to QCLE module. Please verify QCLE module configuration or Windows Error Log for unexpected errors. It may be also that incorrect executable is started, which doesn't maintain connection to QCLE module. |
| Error checking hanging QCL Engine Instance with ID='...' | Generic error. Please see error description for additional info. |
| Error executing operation. | Generic error when starting/stopping QCLE instances when their configuration has changed (e.g. user password was changed – QCLE instance configuration became erroneous). Please see error description for additional info. |
| Could not stop QCL Engine Instance with erroneous configuration. | QCLE module was unable to stop QCLE instance which had configuration change to erroneous. Please see error description for additional info. |
| Error starting QCL Engine Instance having correct configuration | QCLE module was unable to start QCLE instance which had configuration change to correct. Please see error description for additional info. |
| Error checking hanging QCL Engine Instance with ID='...' | Error happened when QCLE module was checking up on hanging QCLE instances. |
| Cannot start instance: repository is offline. | QCLE module is unable to start instance for offline repositories. Please set repository to online mode. |
| Instance is already running. | QCLE module can't start already running instance. If instance is not currently running, please restart QCLE module. |
| Cannot start instance: configuration errors. | QCLE instance has erroneous configuration. Please check whether repository is online, configuration exists and is accessible with specified credentials. |
| Cannot modify running instance. | QCLE instance can't be modified when it's enabled. Please, disable QCLE instance and perform the modification. |
| File path does not exist. File: ... | Specified QCLE instance executable path is incorrect or inaccessible. Please verify that path exists and is accessible by QCLE module. |
| QCL Engine Instance with id = '...' exited right after the start. Check QEF logs to see if there is anything wrong. | QCLE instance exited right after start. Please refer to General Troubleshooting section. |

| | |
|---|---|
| Couldn't auto-start QCL Engine Instance with ID = '...' | QCLE module couldn't restart some instances upon start. Please see error description for additional info. |

# 10 Data Migration

Previous sections describe how to create new metamodels and repositories. In this section we will describe how to migrate data from other QualiWare installations.

Since all user data is contained in repositories, moving repositories is the main subject of this section.

Moving repositories may be used in the following cases:

- Moving one or more repositories to a different database server.
  Note that this operation might instead be done using backup and restore options on database server in many cases.
- Updating QualiWare systems from systems not based on QualiWare Integration Server (QIS). Normally QLM 5.2 or earlier.
- Fixing incorrectly created repositories (rare cases).

Moving repositories is not needed in the following cases:

- Upgrading a QualiWare system from another QIS based system.
- Moving QEF/QIS/QLM installation to a different server.

## 10.1 Move a Repository

When a repository is to be moved, only the content is actually moved. The new repository has to be created and configured manually first. Then data is dumped from the old repository, and finally loaded into the new repository.

### 10.1.1 Collect Information

When creating the new repository it is necessary to have basic information about the old repository ready. The following information should be ready:

- Repository ID.
- Metamodels used.
- All working languages of repository
- If migrating from non QIS versions of QLM, it is important to note the meaning of the special language none/0. That is the language actually used to name and describe objects of the none language.

### 10.1.2 Create and Configure the New Repository

The new repository is created as normal in QIS RA (see section "Creation of repositories"). It is not a strict requirement that the repository ID is the same as in the old repository, but it is recommended to do so if possible.

Note that it is important to match the old repository's language configuration correctly. In QIS versions of QualiWare, languages are defined by a text string with the format language-LOCATION. Examples are en-US, en-GB, nb-NO.

Here there are three cases:

- Migrating from newer QIS versions (5.5.10 – 6.0)
- Migrating from early QIS versions (5.5.7 – 5.5.10)
- Migrating from non QIS version of QLM

#### 10.1.2.1 Migrating from Newer QIS Versions (5.5.10 – 6.0)

Please configure the languages of the new repository exactly as the languages for the old repository.

#### 10.1.2.2 Migrating from Early QIS Versions (5.5.7 – 5.5.10)

In early versions of QIS there existed some additional languages, without location. Examples are "en" and "de". These languages cannot be used with QualiWare 6.0.

Currently there exists no automatic system for resolving this case. Please contact QualiWare support for help if the languages in the old repository has this format.

These QIS versions also had support for valid languages (en-US etc.). In this case, simply continue like described in previous section.

#### 10.1.2.3 Migrating from non QIS Version of QLM

In non QIS versions of QLM, language is defined by a number. This number is automatically translated when loading dump file into QIS version. However in these versions there was a special language called "none" with the number 0, which was always default. When loading the dump, this language is translated to the default language of the new repository. This can lead to a conflict in case the chosen default language in the new repository also exists explicitly in the old repository:

| Old repository | |
|---|---|
| Language | Object content |
| **0 (None)** | English |
| 1031 (de-DE) | German |
| 3082 (es-ES) | Spanish |

Automatic mapping

| New repository | |
|---|---|
| Language | Object content |
| en-US | Empty |
| **de-DE** | Mixed |
| es-ES | Spanish |

The above illustration shows an example of such a conflict. The default languages are written in bold. Because both 1031 and none will both be mapped to de-DE in the new repository, the conflict arises and the load will fail.

The key insight here is that the default language in the new repository should be chosen as the language that none represented in the old repository (English in above example). The correct language configuration for the new repository is shown below:

| Old repository | |
|---|---|
| Language | Object content |
| 0 (None) | English |
| 1031 (de-DE) | German |
| 3082 (es-ES) | Spanish |

Automatic mapping

| New repository | |
|---|---|
| Language | Object content |
| en-US | English |
| de-DE | German |
| es-ES | Spanish |

### 10.1.3 Dump Data from the Old Repository

Finally make a dump from the old repository.

In non QIS versions of QLM this is done from inside Repository Administrator (RA):

- Select repository on the graph.
- Go to tools -> Import and Export -> Dump data from selected repository.

In QIS versions of QLM (5.5 and newer) connect to repository with QLM, then go to:

- File -> Import and Export-> Copy repository to dump file

This will create a file with the repository contents, and possibly a folder. The folder contains some special large objects, used for the QualiWare Document Manager (QDM) system.

### 10.1.4 Load Dump in the New Repository

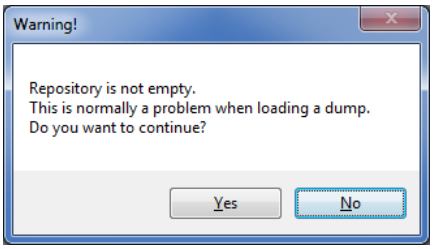
Before starting the dump load following should be checked:

- The new repository is empty
- The new repository is in offline state
- There is a user will all permissions for the repository as defined in QIS RA

To load the dump in the new repository, open the special dump load tool.

If the load process is started directly on the server where QLM is installed, the dump load tool can be started from the "QLM Dump Loader" shortcut in the QualiWare section in the start menu. If the load process is started on a different PC, please run the executable "QlmDumpLoad.exe" from the QLM models folder.

This will prompt you to login to the new repository. Once connected, the tool will check that the repository is empty. If the repository is not empty it will show a warning dialog:



It is highly recommended to only load dumps into empty repositories. If the repository is not empty, it can be emptied by deleting and creating the data storage in the Data Storage Settings for the repository in QIS RA.

The dump loader tool will ask for a dump file to load. Please choose the dumpfile and press open.

If everything is ok, the load will now run. Please note that loading dumps can be very long running operations. The dump loader tool will notify you when it is finished.