

Access Log

1 Contents

1	Managing Access Log Transaction Codes	2
1.1	Configuring Data Source.....	2
1.2	Enabling Transaction Codes	4
1.3	Creating Custom Transaction Codes	6
2	Writing to Access Log	8
2.1	Enabling Access Log for Repository.....	8
2.2	Writing to Access Log from Script	9
2.2.1	C# example	9
2.2.2	QCL example	10
3	Reading Access Log.....	11
3.1	Access Log Database Structure	11
3.2	Example of Log after Calling Test Scripts.....	11
3.2.1	C# Example.....	11
3.2.2	QCL Example.....	11

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.

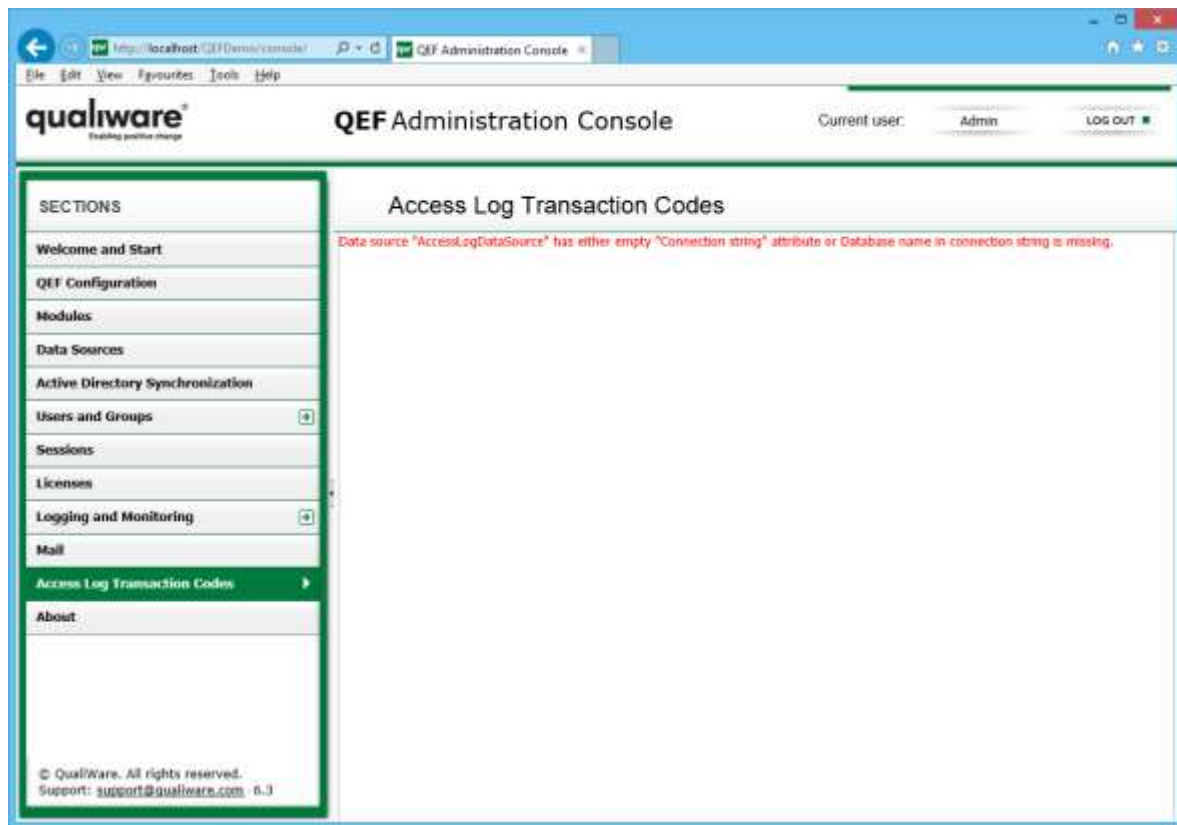
Key elements in Access Log are Transaction codes. Each unique event, either built-in or custom, has its own description in the system. Each description has unique name used as identifier in core products and inside solutions.



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

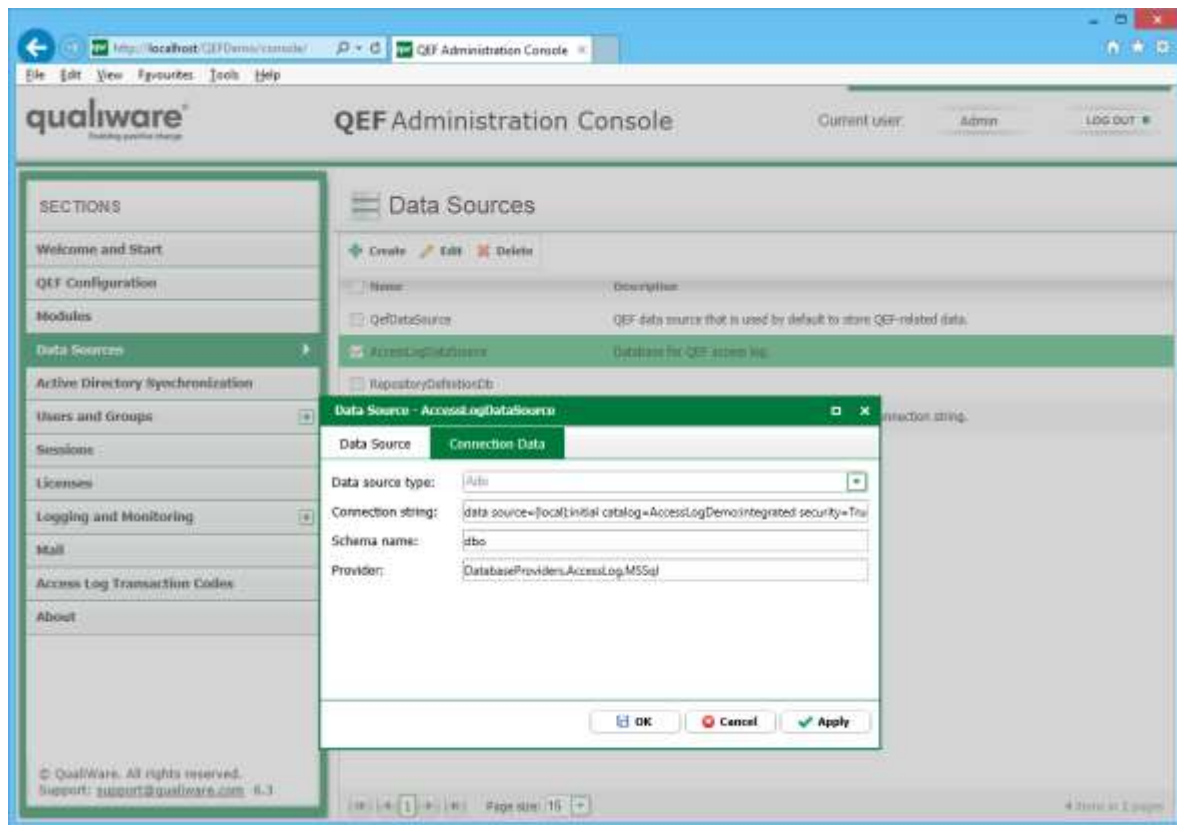
1.1 Configuring Data Source

If it is not the first usage, this part of manual can be skipped. Unless you see such message



Please go to Data Sources section, find there the AccessLogDataSource and start editing it. Fill the **Connection string** field with proper value. For example:

data source=(local);initial catalog=GlobalAccessLog;integrated security=True;



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

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.

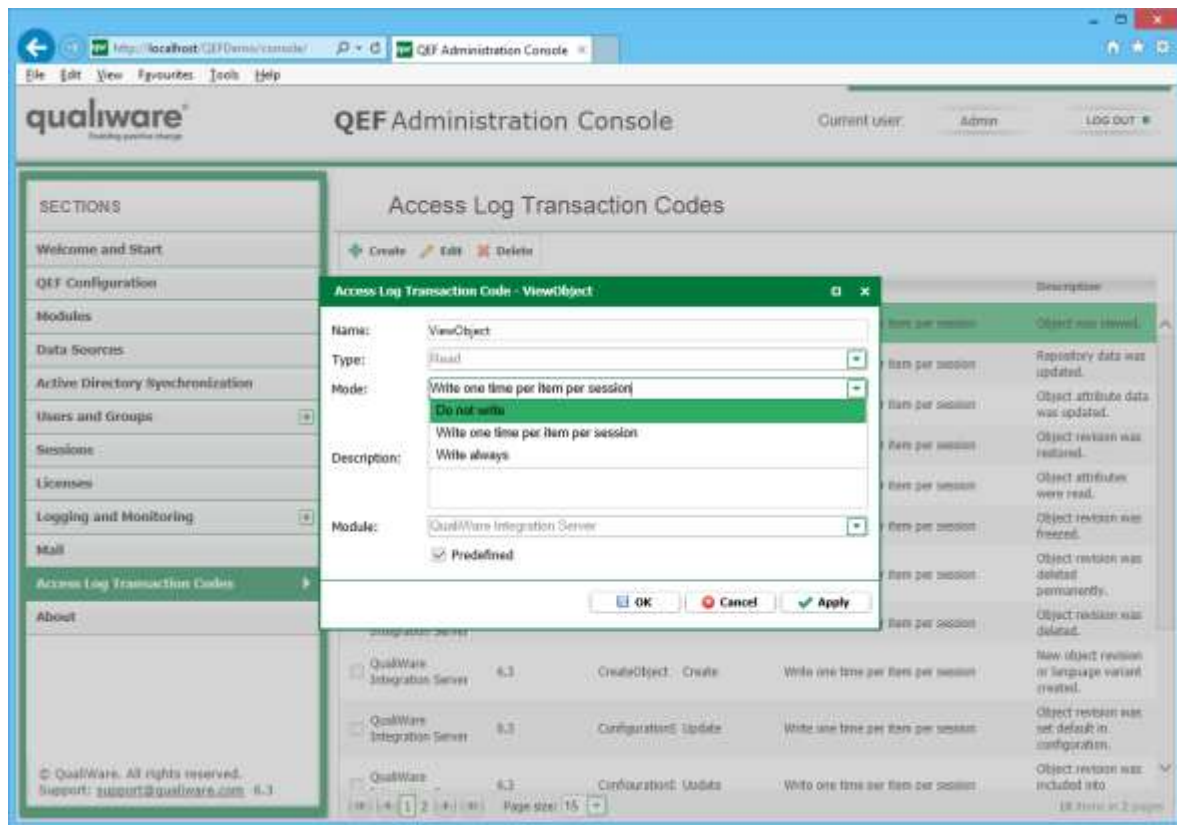
Predefined codes cannot be edited, just viewed, and they cannot be deleted.

The screenshot shows the QualiWare QEF Administration Console interface. The main content area is titled 'Access Log Transaction Codes' and contains a table with the following data:

Module Name	Module Version	Name	Type	Mode	Description
QualiWare Integration Server	6.3	ViewObject	Read	Write one time per item per session	Object was viewed.
QualiWare Integration Server	6.3	UpdateRepository	Update	Write one time per item per session	Repository data was updated.
QualiWare Integration Server	6.3	UpdateObject	Update	Write one time per item per session	Object attribute data was updated.
QualiWare Integration Server	6.3	UndelateObject	Delete	Write one time per item per session	Object revision was restored.
QualiWare Integration Server	6.3	ReadObject	Read	Write one time per item per session	Object attributes were read.
QualiWare Integration Server	6.3	FreezeObject	Update	Write one time per item per session	Object revision was frozen.
QualiWare Integration Server	6.3	DeleteObjectP	Delete	Write one time per item per session	Object revision was deleted permanently.
QualiWare Integration Server	6.3	DeleteObject	Delete	Write one time per item per session	Object revision was deleted.
QualiWare Integration Server	6.3	CreateObject	Create	Write one time per item per session	New object revision or language variant created.
QualiWare Integration Server	6.3	ConfigurationS	Update	Write one time per item per session	Object revision was set default in configuration.
QualiWare Integration Server	6.3	ConfigurationD	Update	Write one time per item per session	Object revision was included into

The interface also includes a left-hand navigation menu with sections like 'Welcome and Start', 'QEF Configuration', 'Modules', 'Data Sources', 'Active Directory Synchronization', 'Users and Groups', 'Sessions', 'Licenses', 'Logging and Monitoring', 'Mail', and 'About'. The 'Access Log Transaction Codes' section is currently selected. At the bottom of the page, there is a footer with the text: '© QualiWare. All rights reserved. Support: support@qualiware.com 6.3'.

Transaction code mode can be changed though.



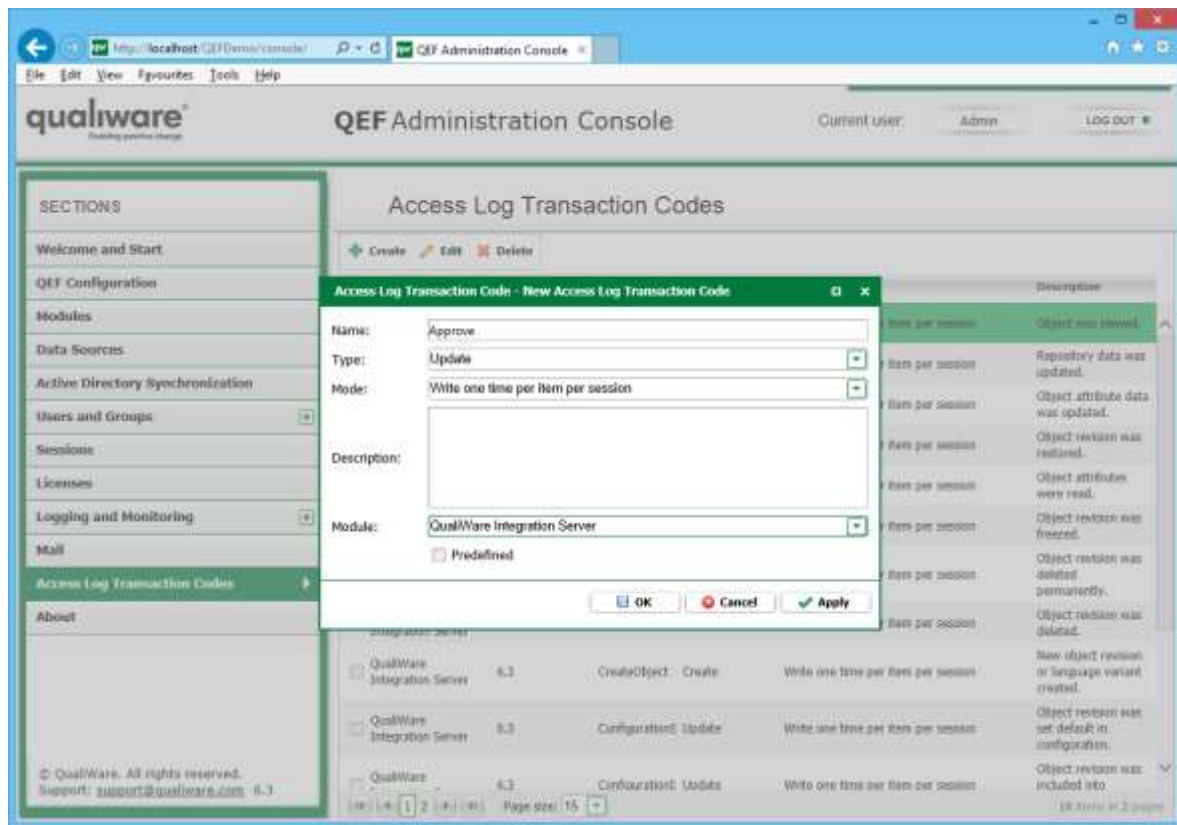
By default transaction code mode is “Write one time per item per session”, what means that only one record will be created for same modified object in same user session. And there is no reason to change it, unless you want to stop recording log for specific transaction code.

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.

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.

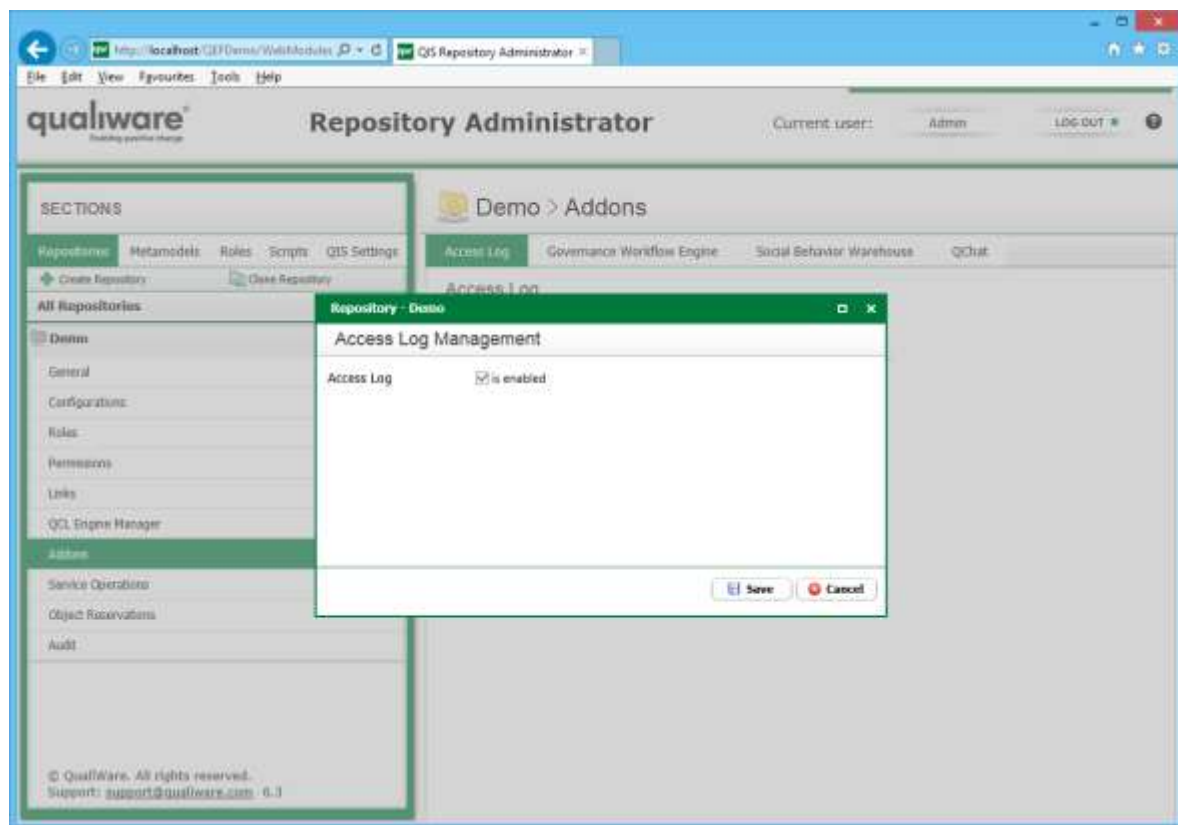
Custom transaction codes are editable and can be deleted. However deleted code remains in database for consistency and can be used in further analysis.

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

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.

2.2 Writing to Access Log from Script

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);
    }
}
}

```

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);

```

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:

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.

3.2 Example of Log after Calling Test Scripts

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	IsPredetined	CodeType	CodeMode
25	25	3	DescriptionChanged	2014-05-05 12:41:48.477	NULL	0	-1	-1

Actual log records are:

Id	ModuleId	UserId	Score	EntryId	TransactionCode	DateTime	Session	Recurrence	EntryVersion
235	235	3	5	A0813100-0018-483e-905e-4876a30000	25	2014-05-05 11:13:17.203	80208415-8080-4100-8000-C08770C18770	3	3d13300000000000
236	236	3	4	b76a3000-0018-483e-905e-4876a30000	25	2014-05-05 11:13:17.205	80208415-8080-4100-8000-C08770C18770	1	3d13300000000000

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:

EntryId	Key	Value
557	235	Description
558	235	Name
		Test
		Audit Planning

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	IsPredetined	CodeType	CodeMode
26	26	3	FromQLM	2014-08-05 12:53:17.273	NULL	0	-1	-1

Actual log records are:

Id	ModuleId	UserId	Score	EntryId	TransactionCode	DateTime	Session	Recurrence	EntryVersion
251	251	3	4	A0813100-0018-483e-905e-4876a30000	26	2014-05-05 13:37:36.883	23288606-6238-4425-0040-C0F04320E5	1	3d11000000000000
252	252	3	4	A0813100-0018-483e-905e-4876a30000	26	2014-05-05 13:37:36.885	23288606-6238-4400-0040-C0F04320E5	1	3d11000000000000

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
001	key1	value1
002	key2	value2