

MoReq2010®

Modular Requirements for Records Systems

Test Framework

Test Module 0: Familiarisation and training

Version 0.1 BETA

Copyright © 2011 DLM Forum Foundation, all rights reserved

# 0.0 Scope

## 0.0.1 Introduction

Test Module 0 is not like the remaining test modules because its primary purpose is to serve as a preliminary familiarisation and training exercise between the supplier and the test centre, prior to undertaking MoReq2010® testing. During this test module the supplier is able to demonstrate the features of its MCRS solution while the test centre is allowed the opportunity to explore the test system more broadly than is possible through the test modules that follow. Note that test centre representatives, not supplier representatives, should take the roles of test system operators for this and all subsequent test modules.

## 0.0.2 Learning objectives

Because MoReq2010® encourages MCRS solutions to be developed and submitted for testing in many different forms it is important that the test centre is able to understand quickly how the particular solution works and how important concepts, such as services, entities and functions are realised and implemented within the supplier’s solution.

To accomplish this, Test Module 0 sets learning objectives for each test case, rather than formal post-conditions. The test system operators should not move on to the next test case until they are confident that they have achieved the learning objectives. During this test module the supplier and test centre representatives are encouraged to collaborate together to facilitate the initial exploration of the depth and breadth of the test system. Later test modules are more specific and structured in their approach, and each test case is either passed or failed.

## 0.0.3 Interface requirements and test scripts

This test module, however, this is not just an exercise in familiarisation. The test module is also designed to ensure that the test centre checks and signs off each of the interface functional requirements for the solution. The requirements covered by the test case are from the following sections of the MoReq2010® specification:

* **101. Graphical User Interface**
* **102. Application Programming Interface**

The test module is divided into two sections each with its own individual script. Part A covers GUI interfaces and Part B covers API interfaces. The test centre must work through the scripts in one or both of these sections depending on whether the services and modules to be tested implement a GUI or an API, or a mixture of both.

* **0.2 Test script A** is specifically for graphical user interfaces, and
* **0.3 Test script B** is specifically for application programming interfaces.

## 0.0.4 When to use this test case

This test module should be used as a preliminary module to all types of testing. Whether it is the core services of MoReq2010® are being tested or extra plug-in or extension modules this test module should always be the first test module to be tested.

Additional notes are provided to test centres for clarity throughout. From time-to-time the test scripts will be updated; be sure to check that you are using the most up-to-date version of the testing materials.

## 0.0.5 Documentation of significant behaviours

From time to time situations will occur during familiarisation, training and testing which deviate from the expected behaviour of the system in particular circumstances. This may include aspects of the system as observed by the test centre that are not clear from the information provided in the pre-qualification questionnaire (PQQ).

Any such significant or special behaviour that is observed, including but not limited to observed inconsistencies or omissions, should be fully documented by the test centre and included in the test report in section **0.1.5 Description and screenshots of any significant/special behaviour**. Test centres are encouraged to include screenshots where possible to add clarity to the observation.



# TEST MODULE 0: FAMILIARISATION AND TRAINING

(To be completed by the test centre)

# Observations

To be completed by the test centre

## 0.1.1 Identification

Supplier name

Product name and version

Test centre name

Date of test

Test location

Supplier representative(s) present during testing

Test centre representative(s) present during testing

DLM Forum observer(s) present during testing

Test system technical architecture, hardware, software and operating system(s)

Purpose of familiarisation

This test module was completed prior to

🞏 Testing of core services (with extra modules)

OR

🞏 Testing of extra module(s) only

Additional remarks or observations prior to commencing testing

## 0.1.2 Learning objectives checklist

Fill in with 🗹 or 🗷 to ensure all learning objectives are met

Test system operator(s) for these test cases

Learning objectives accomplished

🞏 Understand how users are authenticated and how they access the test system

🞏 Understand how the test system represents services within its architecture

🞏 Understand how the test system implements the concept of “browsing”

🞏 Understand how entities of different entity types are represented in the test system

🞏 Understand the relationship between entities and services in the test system

🞏 Know how to access the metadata, access controls and event history of an entity

🞏 Understand how to perform functions on individual entities and where possible collectively on entities

🞏 Understand how the interface provides feedback on functions that are performed

🞏 Understand the relationship between entities, access controls and functions

🞏 Understand how active entities are differentiated from residual entities in the test system

🞏 Understand how entities of different entity types are created and added to the test system

🞏 Understand how entities are destroyed

🞏 Understand how the metadata of different entities is modified

🞏 Know how to use the test system’s search interface

🞏 Understand how the interface allows or disallows functions to be performed, depending on access controls and context

🞏 Understand how the test system prevents the modification of read only metadata and restricts the entry of metadata to an appropriate datatype

🞏 Follow what happens when an error occurs during a function and the feedback provided to the user

🞏 Know how extended error information can be accessed by the end user as well as how it is recorded in the error log

🞏 Understand the additional features and functionality provided by the test system, such as:

* In a GUI specialised interfaces intended to facilitate disposal and export
* In an API asynchronous support for multiple simultaneous users

🞏 Know how to access on line help and assistance, including help with specific functions

🞏 Know how to access documentation and user guides for the test system

Additional remarks or observations on learning objectives

## 0.1.3 Test Results for Test Script A

Fill in if **0.2 Test Script A** was completed

Test system operator(s) for these test cases

GUI checklist

🞏 The test system implements a GUI compliant with **R2.4.6** and **R101.4.1**

During testing…

🞏 …the GUI displayed entities consistently compliant with **R101.4.2**

🞏 …only active entities were displayed by default compliant with **R101.4.3**

🞏 …the GUI allowed new entities, including records, to be created with **R101.4.4**

🞏 …the GUI presented the metadata of entities when they were inspected and facilitated the modification of metadata values compliant with **R101.4.5**

🞏 …the GUI automated the entry of metadata values compliant with **R101.4.6**

🞏 …the GUI prevented modification of read only metadata compliant with **R101.4.7**

🞏 …the GUI facilitated access and update of entity’s ACLs compliant with **R101.4.8**

🞏 …the GUI presented visually and logically the relationships between entities and allowed the user to browse these relationships compliant with **R101.4.9**

🞏 …the user was able to see what operations could be performed on entities and which were unavailable by visual inspection compliant with **R101.4.10**

🞏 …the user was able to access search from any view or screen compliant with **R101.4.11**

🞏 …the user was able to graphically construct search queries compliant with **R101.4.12**

🞏 …the user was able to reorder and configure search results compliant with **R101.4.13**

🞏 …the user was able to save sets of search results as records compliant with **R101.4.14**

🞏 …the user was able to inspect and perform operations on entities directly from search results compliant with **R101.4.15**

🞏 …the user was able to make shortcuts to entities compliant with **R101.4.16**

🞏 …the GUI allowed selection of sets of entities and bulk operations compliant with **R101.4.17**

🞏 …the GUI provided a specialised interface for engaging with the disposal process compliant with **R101.4.18**

🞏 …the GUI provided a specialised interface for export compliant with **R101.4.19**

🞏 …the test system provided user feedback on the progress of long operations compliant with **R101.4.20**

🞏 …the test system provided useful error messages compliant with **R101.4.21**

🞏 …the GUI provided easily accessible help consistent with **R101.4.22**

Additional remarks or observations on test script A

## 0.1.4 Test Results for Test Script B

Fill in if **0.2 Test Script B** was completed

Test system operators for these test cases

API checklist

🞏 The test system implements an API compliant with **R2.4.6** and **R102.4.1**

During testing…

🞏 …the API could be used with multiple users simultaneously compliant with **R102.4.2** and **R102.4.3**

🞏 …the API returned an error code and provided extended error information compliant with **R102.4.4** and **R102.4.5**

🞏 …the API was able to indicate the allowable operations the user could make in respect of any entity compliant with **R102.4.6**

Additional remarks or observations on test script B

## 0.1.5 Description and screenshots of any significant/special behaviour

Use this section to document any observed behaviour of the test system that is not fully documented in the PQQ submitted by the supplier and which may aid interpretation of the utility of the solution. Include screenshots to aid clarity.



# 0.2 Test script A – Graphical User Interface

Complete the test cases in order and answer the evaluation questionnaire at the completion of all test cases.

## T0.2.1 – Test case

### 0.2.1.1 Test case description

**Service discovery**

### 0.2.1.2 Test case preconditions

* Test system is operational
* Test user log on accounts are available

### 0.2.1.3 Test case steps (instructions)

* Practice logging into and out of the test system as a user under **R3.4.1**
* Browse to different services under **R2.4.3** and examine their metadata under **R2.4.2**

### 0.2.1.4 Test case post-conditions (learning objectives)

* Understand how users are authenticated and how they access the test system
* Understand how the test system implements services within its architecture
* Understand how the test system implements the concept of “browsing”

### 0.2.1.5 Notes to test centre

* Under **R3.4.1** all users of an MCRS must be authenticated (the precise mechanism to be used for authentication is not specified)
* Under **R2.4.1** and **R101.4.1** all of the core services of an MCRS must be represented in the test system
* In some test systems the different services will be differentiated
* In other test systems all services will be bundled together

## T0.2.2 – Test case

### 0.2.2.1 Test case description

**Entity discovery**

### 0.2.2.2 Test case preconditions

* Test system is operational
* Test system operators are logged in and have completed the previous test case

### 0.2.2.3 Test case steps (instructions)

* Learn how to browse from different services to the different types of entity managed by each service under **R2.4.9**
* Recognise how entities of each type are represented by the interface, where they may be found, and the different views that are available (for example, list views and tree views)
* Practice inspecting entities and examining their:
	+ Metadata,
	+ Access controls, and
	+ Event history

### 0.2.2.4 Test case post-conditions (learning objectives)

* Understand how entities of different entity types are represented in the test system
* Understand the relationship between entities and services in the test system
* Know how to access the metadata, access controls and event history of an entity

### 0.2.2.5 Notes to test centre

* Under **R2.4.9**, the core services collectively manage entities belonging to some 16 different entity types (including the services themselves):
	+ Aggregation (**E14.2.1**)
	+ Class (**E14.2.2**)
	+ Component (**E14.2.3**)
	+ Contextual Metadata Element Definition (**E14.2.4**)
	+ Disposal Hold (**E14.2.5**)
	+ Disposal Schedule (**E14.2.6**)
	+ Entity Type (**E14.2.7**)
	+ Event (**E14.2.8**)
	+ Function Definition (**E14.2.9**)
	+ Group (**E14.2.10**)
	+ Metadata Element Definition (**E14.2.11**)
	+ Record (**E14.2.12**)
	+ Role (**E14.2.13**)
	+ Service (**E14.2.14**)
	+ Template (**E14.2.15**)
	+ User (**E14.2.16**)
* **R101.4.2** states that entities of different entity types should be recognisable and displayed consistently throughout the interface
* **R101.4.9** states that the interface must visually show the relationships between entities and allow users to browse these relationships
* Depending on whether or not the test system implements the model role service it may have different access controls for its services and entities
* Under **R101.4.8** the interface must provide access to the access control list (or equivalent) of an entity for both inspecting and modifying
* All entities should have an accessible event history

## T0.2.3 – Test case

### 0.2.3.1 Test case description

**Performing functions**

### 0.2.3.2 Test case preconditions

* Test system is operational
* Test system operators are logged in and have completed the previous test case

### 0.2.3.3 Test case steps (instructions)

* For different types of entity practice performing functions on that entity type
* Select sets of the same entity type and perform the same function across all selected entities
* Learn to recognise from visual feedback what functions are available
* Check whether the interface provides access to functions consistently

### 0.2.3.4 Test case post-conditions (learning objectives)

* Understand how to perform functions on individual entities and where possible collectively on entities
* Understand how the interface provides feedback on functions that are performed
* Understand the relationship between entities, access controls and functions

### 0.2.3.5 Notes to test centre

* **R101.4.10** states that the interface must visually guide the user by showing what operations are available on an entity
* Under **R101.4.17** the interface must allow multiple entities of the same type to be selected and the same operation performed simultaneously across all entities

## T0.2.4 – Test case

### 0.2.4.1 Test case description

**Interface discovery**

### 0.2.4.2 Test case preconditions

* Test system is operational
* Test system operators are logged in and have completed the previous test case

### 0.2.4.3 Test case steps (instructions)

* Learn how to switch between a view of the test system which includes only active entities or one which includes both active and residual entities
* Practice creating and destroying entities of different entity types
* Modify the metadata of entities of different entity types
* Learn how to search and how to build both simple and complex search queries
* Check how search results are arranged and can be manipulated
* Know how to search across active entities only, as well as both active and residual entities

### 0.2.4.4 Test case post-conditions (learning objectives)

* Understand how active entities are differentiated from residual entities in the test system
* Understand how entities of different entity types are created and added to the test system
* Understand how entities are destroyed
* Understand how the metadata of different entities is modified
* Know how to use the test system’s search interface

### 0.2.4.5 Notes to test centre

* Under **R101.4.3** the interface must display only active entities by default but can be switched to display both active and residual entities
* Under **R101.4.4** the interface must facilitate the creation of new entities by users
* Under **R101.4.5** the interface must facilitate the inspection and modification of interface metadata by users
* **R101.4.11** states that the search interface must be available from any screen in the test system
* **R101.4.12** states that the test system must facilitate the construction of search queries by users
* Under **R101.4.13** the interface must allow users to graphically configure search results
* Under **R101.4.15** users must be able to inspect and perform operations on entities directly from search results

## T0.2.5 – Test case

### 0.2.5.1 Test case description

**Error behaviours**

### 0.2.5.2 Test case preconditions

* Test system is operational
* Test system operators are logged in and have completed the previous test case

### 0.2.5.3 Test case steps (instructions)

* Check whether it is possible to perform illegal functions on entities, for example:
	+ Changing the metadata of a residual entity,
	+ Modifying a read only metadata value,
	+ Entering in a metadata value using an incorrect datatype, or
	+ Performing a function that the user is not authorised to perform.
* Check what happens when a user attempts to log in using an unauthorised account
* Cause, or with the assistance of the supplier, simulate an error where an operation fails and check:
	+ The test system’s response to the user
	+ The extended error information in the external error log

### 0.2.5.4 Test case post-conditions (learning objectives)

* Understand how the interface allows or disallows functions to be performed, depending on access controls and context
* Understand how the test system prevents the modification of read only metadata and restricts the entry of metadata to an appropriate datatype
* Follow what happens when an error occurs during a function and the feedback provided to the user
* Know how extended error information can be accessed by the end user as well as how it is recorded in the error log

### 0.2.5.5 Notes to test centre

* Under **R101.4.7** the interface must ensure that only metadata values that are of the correct datatype can be entered by the user and that read only values cannot be changed at all
* **R2.4.7** requires that the test system maintain an external error log of all errors that occur during test system operations
* Under **R101.4.21** the interface must provide useful error messages that inform the user not just that an error has occurred but how to proceed
* The supplier should be prepared to provide assistance with “simulating” appropriate error conditions showing the range of information that is sent to the user and is accessible in the error log

## T0.2.6 – Test case

### 0.2.6.1 Test case description

**Additional features**

### 0.2.6.2 Test case preconditions

* Test system is operational
* Test system operators are logged in and have completed the previous test case

### 0.2.6.3 Test case steps (instructions)

* Explore the additional features provided by the test system including those mandated by MoReq2010® (see list below) and those additional which may be pointed out by the supplier

### 0.2.6.4 Test case post-conditions (learning objectives)

* Understand the additional features and functionality provided by the test system such as:
	+ The specialised interfaces intended to facilitate disposal and export, or
	+ The ability to interrupt and cancel lengthy operations

### 0.2.6.5 Notes to test centre

The following list contains additional features mandated by MoReq2010® (note that the test system may provide further additional features on top of the features listed here to enhance functionality and ease of use):

* Under **R101.4.6** the interface must where possible provide the user with automated assistance in performing certain types of task
* Under **R101.4.14** the interface must allow users to save their search results and capture them as records
* Under **R101.4.16** the interface must allow the user to make and share shortcuts to entities (for example, providing a URL to an entity that can be sent by email to another user)
* Under **R101.4.18** the test system must provide a specialised interface for accessing the disposal process
* Under **R101.4.19** the test system must provide a specialised interface for accessing the export process
* Under **R101.4.20** the interface must provide feedback, such as a progress bar, on all lengthy operations and allow the user to cancel them before they have completed

## T0.2.7 – Test case

### 0.2.7.1 Test case description

**Help, documentation and support**

### 0.2.7.2 Test case preconditions

* Test system is operational
* Test system operators are logged in and have completed the previous test case

### 0.2.7.3 Test case steps (instructions)

* Check with the supplier what help is available from each screen and dialogue in the interface

### 0.2.7.4 Test case post-conditions (learning objectives)

* Know how to access on line help and assistance, including help with specific functions
* Know how to access documentation and user guides for the test system

### 0.2.7.5 Notes to test centre

* Under **R101.4.22** the interface must provide easily accessible help from its screens, views and dialogues
* Help may include access (including by keyword) to local or remote help files, tutorials and context sensitive help

# 0.3 Test script B – Application Programming Interface

Complete the test cases in order and answer the evaluation questionnaire at the completion of all test cases.

## T0.3.1 – Test case

### 0.3.1.1 Test case description

**Service discovery**

### 0.3.1.2 Test case preconditions

* Test system is operational
* A test harness has been provided by the supplier to enable the test centre to engage with the API (see the **Test Framework: Overview and Instructions, 4.1 Providing a test harness for API interfaces**)
* Test user accounts are available

### 0.3.1.3 Test case steps (instructions)

* From the information provided on the API interface learn how users are authenticated
* Practice making authenticated method calls to the test system as a user under **R3.4.1**
* Determine how browsing is implemented using the API
* Determine how services are implemented using the API
* Use the API to browse from one service to another under **R2.4.3** and examine their metadata under **R2.4.2**

### 0.3.1.4 Test case post-conditions (learning objectives)

* Understand how users are authenticated and how they access the test system
* Understand how the test system implements services within its architecture
* Understand how the test system implements the concept of “browsing”

### 0.3.1.5 Notes to test centre

* An API is a set of methods provided in some programming language or environment, each method can be called separately
* In order to test an API the supplier must provide both the API description and tools, as well as a test harness suitable for making calls to the API
* Under **R102.4.3** each method call must be made by, and is attributable to, an individual authenticated user
* The API may support user authentication in one of two ways:
	+ Session management – where the user authenticates once and remains authenticated for a whole session
	+ Call management – where the user must provide authentication credentials or a security token with every call to the API
* Under **R3.4.1** all users of an MCRS must be authenticated (the precise mechanism to be used for authentication is not specified)
* Under **R2.4.1** and **R102.4.1** all of the core services of an MCRS must be represented in the test system
* In some test systems the different services will be differentiated
* In other test systems all services will be bundled together
* An API may provide a separate interface or end point for each service, or it may provide a common end point or interface for all services

## T0.3.2 – Test case

### 0.3.2.1 Test case description

**Entity discovery**

### 0.3.2.2 Test case preconditions

* Test system is operational
* Test system operators are authenticated and have completed the previous test case

### 0.3.2.3 Test case steps (instructions)

* From the API documentation and test harness learn how to browse from different services to the different types of entity managed by each service under **R2.4.9**
* Recognise how metadata and content for different types of entities and services can be retrieved by the interface, and the different ways of accessing entities using API calls
* Practice inspecting entities and examining their:
	+ Metadata,
	+ Access controls, and
	+ Event history

### 0.3.2.4 Test case post-conditions (learning objectives)

* Understand how entities of different entity types are represented in the test system
* Understand the relationship between entities and services in the test system
* Know how to access the metadata, access controls and event history of an entity

### 0.3.2.5 Notes to test centre

* Browsing using an API is generally implemented by returning sufficient information from a method call to one entity to allow the user to call one or more methods on a different entity
* In other words, interrogating one entity or service can lead to discovering other entities or services
* Under **R2.4.9**, the core services collectively manage entities belonging to some 16 different entity types (including the services themselves):
	+ Aggregation (**E14.2.1**)
	+ Class (**E14.2.2**)
	+ Component (**E14.2.3**)
	+ Contextual Metadata Element Definition (**E14.2.4**)
	+ Disposal Hold (**E14.2.5**)
	+ Disposal Schedule (**E14.2.6**)
	+ Entity Type (**E14.2.7**)
	+ Event (**E14.2.8**)
	+ Function Definition (**E14.2.9**)
	+ Group (**E14.2.10**)
	+ Metadata Element Definition (**E14.2.11**)
	+ Record (**E14.2.12**)
	+ Role (**E14.2.13**)
	+ Service (**E14.2.14**)
	+ Template (**E14.2.15**)
	+ User (**E14.2.16**)
* Inspecting entities using an API has the same meaning as inspecting them visually – it means using method calls to retrieve their metadata, or the data associated with their access controls or their event history
* Depending on whether or not the test system implements the model role service it may have different access controls for its services and entities
* All entities should have an accessible event history

## T0.3.3 – Test case

### 0.3.3.1 Test case description

**Performing functions**

### 0.3.3.2 Test case preconditions

* Test system is operational
* Test system operators are authenticated and have completed the previous test case

### 0.3.3.3 Test case steps (instructions)

* For different types of entity practice performing functions on that entity type
* Select sets of the same entity type and perform the same function across all selected entities
* Use methods to determine what functions are available for particular entities
* Check whether the interface provides consistent access to functions

### 0.3.3.4 Test case post-conditions (learning objectives)

* Understand how to perform functions on individual entities and where possible collectively on entities
* Understand how the interface provides feedback on functions that are performed
* Understand the relationship between entities, access controls and functions

### 0.3.3.5 Notes to test centre

* Most APIs will provide methods that mirror the functions of MoReq2010® to the same level of granularity
* However, some APIs may map a single method to several MoReq2010® functions while other APIs require several method calls to execute a single function
* The supplier’s API documentation should provide information about how API methods map to functions while the test harness provided by the supplier should make this transition so that MoReq2010® functions can be performed and tested
* **R102.4.6** states that the interface must provide a method that returns, for any given entity, the current user’s allowable operations in respect of that entity

## T0.3.4 – Test case

### 0.3.4.1 Test case description

**Interface discovery**

### 0.3.4.2 Test case preconditions

* Test system is operational
* Test system operators are authenticated and have completed the previous test case

### 0.3.4.3 Test case steps (instructions)

* Learn how to make method calls which return only active entities and how to make method calls which return both active and residual entities
* Practice creating and destroying entities of different entity types through the API
* Modify the metadata of entities of different entity types through the API
* Learn how to construct both simple and complex search queries and launch searches through the API
* Check how data, such as search results, are returned by the API
* Conduct searches that return only active entities as well as searches that return both active and residual entities

### 0.3.4.4 Test case post-conditions (learning objectives)

* Understand how active entities are differentiated from residual entities in the test system
* Understand how entities of different entity types are created and added to the test system
* Understand how entities are destroyed
* Understand how the metadata of different entities is modified
* Know how to use the test system’s search interface

### 0.3.4.5 Notes to test centre

* Under **R2.4.22** method calls to the API should return only references to active entities unless the inclusion of residual entities is specifically requested
* Under **R10.4.17** any search method must return by default only active entities unless the inclusion of residual entities is specifically requested

## T0.3.5 – Test case

### 0.3.5.1 Test case description

**Error behaviours**

### 0.3.5.2 Test case preconditions

* Test system is operational
* Test system operators are authenticated and have completed the previous test case

### 0.3.5.3 Test case steps (instructions)

* Check whether it is possible to call methods on entities that are illegal, for example:
	+ Changing the metadata of a residual entity,
	+ Modifying a read only metadata value,
	+ Entering in a metadata value using an incorrect datatype, or
	+ Performing a function that the user is not authorised to perform.
* Check what happens when a user attempts to call a method using an unauthorised user account, or without sufficient permissions
* Cause, or with the assistance of the supplier, simulate an error where an operation fails and check:
	+ The test system’s response to the method call
	+ The extended error information in the external error log

### 0.3.5.4 Test case post-conditions (learning objectives)

* Understand how the interface allows or disallows functions to be performed, depending on access controls and context
* Understand how the test system prevents the modification of read only metadata and restricts the entry of metadata to an appropriate datatype
* Follow what happens when an error occurs during a function and the feedback provided to the user
* Know how extended error information can be accessed by the end user as well as how it is recorded in the error log

### 0.3.5.5 Notes to test centre

* **R2.4.7** requires that the test system maintain an external error log of all errors that occur during test system operations
* Under **R102.4.4** each method in the interface must return an error code that indicate the success or failure of the call
* Under **R102.4.5** the interface must also provide a method that returns extended error information for this error code
* The supplier should be prepared to provide assistance with “simulating” appropriate error conditions showing the range of information that is sent to the user and is accessible in the error log

## T0.3.6 – Test case

### 0.3.6.1 Test case description

**Additional features**

### 0.3.6.2 Test case preconditions

* Test system is operational
* Test system operators are authenticated and have completed the previous test case

### 0.3.6.3 Test case steps (instructions)

* Explore the additional features provided by the test system including those mandated by MoReq2010® (see below) and those additional which may be pointed out by the supplier

### 0.3.6.4 Test case post-conditions (learning objectives)

* Understand the additional features and functionality provided by the test system such as asynchronous support for multiple simultaneous users

### 0.3.6.5 Notes to test centre

* When a MCRS offers an API interface then under **R102.4.2** the API must support multi-user asynchronous calls
* This means that many users can use the API at the same time to manipulate entities within the solution and any clashes will be automatically arbitrated by the system
* Asynchronous refers to the order in which method calls are made: the API should be able to reconcile authenticated calls to any method for any entity in any order at any time – it should be multi-tasking as well as multi-user

## T0.3.7 – Test case

### 0.3.7.1 Test case description

**Help, documentation and support**

### 0.3.7.2 Test case preconditions

* Test system is operational
* Test system operators are authenticated and have completed the previous test case

### 0.3.7.3 Test case steps (instructions)

* Check with the supplier what API documentation is available and how it can be accessed

### 0.3.7.4 Test case post-conditions (learning objectives)

* Know how to access on line help and assistance, including help with specific functions
* Know how to access documentation and user guides for the test system

### 0.3.7.5 Notes to test centre

* API documentation may include developer and administrator technical manuals, examples, tutorials, etc.
* For example, a Java API may be supported with accompanying javadoc