The formatting and layout of this draft does not reflect its final form when published as a web specification with embedded hyperlinking.

MoReq2010 v0.92

Model Requirements for records systems

1 Core Requirements

1.1 Important information

1.1.1 Identifier

be2af786-e754-430b-b7ce-7062e301bf13

1.1.2 Title

MoReq2010 - Core Requirements

1.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the core requirements for all compliant records systems. The core requirements are mandatory.

1.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

1.1.5 Intellectual Property Rights

1.1.6 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

1.1.7 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

1.2 Background

1.2.1 MoReq

The first MoReq specification was published in 2001 as a result of close cooperation between the DLM Forum and the European Commission. MoReq provided a new pan-European specification for computer systems that manage electronic records. Prior to its publication there were only a few countries in Europe with their own national standards for records management.

Even from its earliest publication, MoReq has always had the following characteristics:

- **Universal in scope and application** – MoReq is an international specification and has been used and adopted across a large number of countries, including many outside Europe;
- **Available in many languages** – MoReq and its successor MoReq2 have been translated in full into around a dozen European, and some non-European, languages; and
- ***De facto* standardisation** – although originally conceived as a specification, MoReq is today widely recognised as a *de facto* industry standard because of its universal appeal, availability and adoption.

"MoReq" stands for "Model Requirements" and it was originally envisaged that the specification would provide an exemplar that would then be modifiable to meet local needs. The first edition even contained guidance for how to add, edit and delete chapters and requirements and manage issues such as cross-referencing within the specification.

1.2.2 MoReq2

In 2005, the DLM Forum completed a scoping study aimed at updating and extending the original MoReq specification. The result of this review was the publication of MoReq2 in early 2008.

A key feature of MoReq2 was the inclusion for the first time of a testing and certification regime. Suppliers could now receive independent certification of their products as MoReq2 compliant. In addition to its metadata model, MoReq2 also introduced an XML schema that was intended to define a common import/export format across different products and implementations.

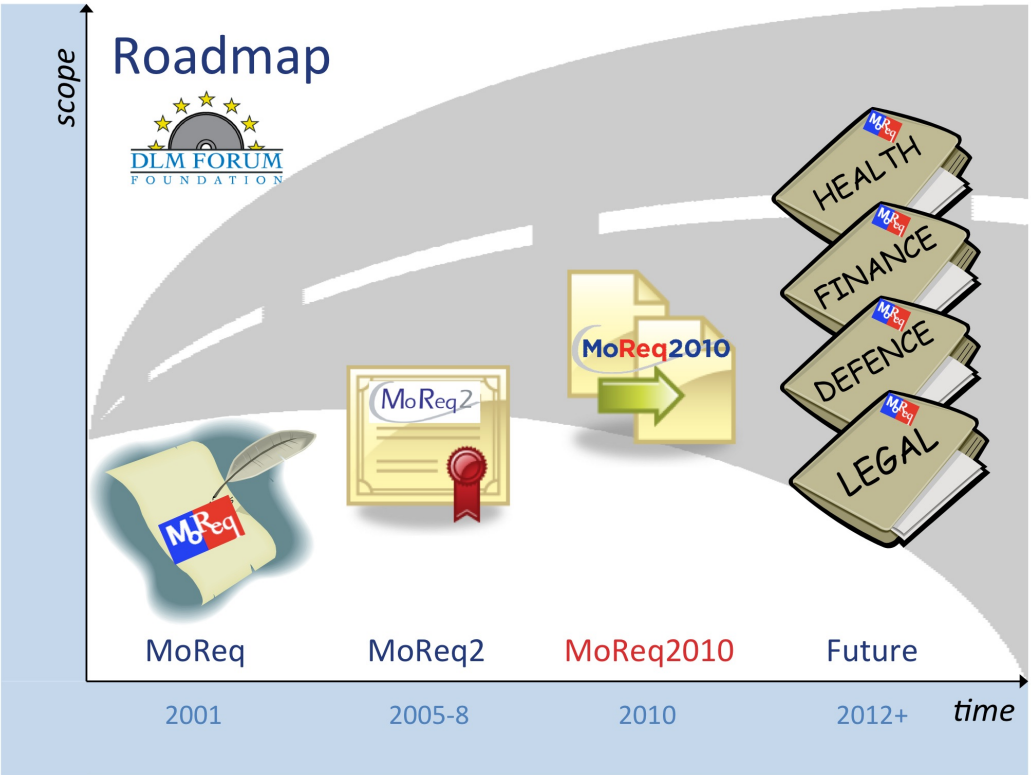
In December 2008, the DLM Forum appointed a permanent sub-committee dubbed the MoReq Governance Board. The role of the governance board is to manage all aspects of the MoReq specification including:

- Provide for ongoing maintenance, publish a roadmap for MoReq, and plan for future upgrade of the specification;
- Manage the translation programme, arrange for the validation of accepted translations, and give guidance to MoReq translators;

- Grant accreditation to recognised test centres to undertake software testing against the specification;
- Oversee the testing and certification of software products against MoReq by accredited test centres;
- Run a parallel education programme including providing workshops and training, issuing supplementary guidance and educational materials; and
- Actively market the specification, collect case studies, and encourage its adoption while simultaneously protecting the MoReq brand.

1.2.3 MoReq Roadmap

In 2009, the MoReq Governance Board produced a roadmap for MoReq, which was subsequently adopted by DLM Forum resolution at a members' meeting in Härnösand, Sweden in November of that year.



The roadmap clearly identified that while the MoReq specification was widely viewed by industry as addressing the requirements for mainstream records management in traditional domains such as Electronic Document and Records Management Systems (EDRMS) and Enterprise Content Management (ECM), it was also seen as less applicable to specialised adoption in areas like health, legal and financial services. These industry sectors were typically governed by legislation and regulation particular to themselves and as a result tended to invent and adopt their own records management criteria.

Another emerging trend was that of heterogeneity within records system design. MoReq was originally conceptually based on a central repository model where an organisation's standalone records system would capture records from a variety of external sources, including users and other business systems. The governance board roadmap recognised the increasing adoption of alternative architectures. One such architecture is a dedicated records system that is entirely built into a business system and manages only the narrow set of records held by that particular business system. Another emerging model is that of the storage-less records system that manages records *in situ* within the business systems in which they originate, rather than duplicating them into its own repository.

Finally, the MoReq roadmap identified the need for flexible and scalable requirements that were equally applicable to both large and small records management applications. MoReq2 had already more than doubled the number of functional requirements and nearly tripled the number of pages of its predecessor. It was clear that each successive edition of MoReq could not continue to similarly grow in size and complexity or it would eventually provide an insurmountable barrier to adoption, particularly by smaller, local and niche suppliers. For all these reasons the 2009 roadmap called for two future phases of MoReq evolution:

- In the short-term in 2010 to launch a refactoring project that would reorganise the specification along modular lines, simplify it where possible, and introduce support for alternative records systems architectures; and
- In the longer-term by 2012 with the assistance of specialised industry experts, and based on the flexibility afforded by a more modular approach, to broaden the applicability of MoReq into all fields of human endeavour where the sound management of records is an essential prerequisite.

The underlying objective of the MoReq roadmap was clear: to significantly increase the universal applicability and adaptability of the specification.

1.2.4 MoReq2010

In early 2010 the DLM Forum launched the MoReq2010 project. It was not just the content of the specification itself that was intended to be innovative. It was decided to publish MoReq2010 as an entirely web-based specification, making each requirement individually URL addressable and allowing immediate worldwide access to the latest version of the specification available from anywhere on the Internet.

The DLM Forum also decided to undertake two fully public consultation phases as part of developing the new specification while the European Commission appointed an Experts' Review Group made up of a cross-section of industry experts to provide advice to the project. The resulting specification has therefore been developed and discussed with an unprecedented level of collaborative input for a project of its nature and it is hoped that this is reflected in the quality of the final product.

Records management today can be a complex undertaking and MoReq2010, with its modular and extensible architecture, seeks to provide an easy way to navigate the

pitfalls of specifying a records system implementation for suppliers, practitioners and consumers alike. The now established programme of testing and certification of software products against the specification negates the value of basing an implementation on MoReq but then going on to further modify and customise requirements individually. Fortunately, there is no longer any need to do this as MoReq2010's modular approach allows consumers to more easily specify a comprehensive and cohesive set of organisational requirements simply by selecting a suitable combination of modules that corresponds with their organisational needs.

Suppliers too will benefit from the refactoring of MoReq2010. While the set of additional extension modules will continue to grow and embody more and more specialised applications, the core set of requirements is correspondingly reduced in comparison to MoReq2010's predecessors. These are the only requirements that all records systems must have in common and show compliance with. The implementation of, and certification against, other modules then depends on the particular focus of a given product, its target sector and its degree of specialisation or generalisation. It is not expected that any supplier will ever seek to build a product that implements every module of MoReq2010 as there is no known environment that requires this degree of universality.

The other group that will benefit directly from the approach taken by MoReq2010 is comprised of the practitioners themselves. MoReq2010 seeks in every aspect to directly tie leading records management theory and best practice back into the specification. Professionals will find that concepts, terms and models adopted by MoReq2010 are more closely linked to those used in other international standards, and propounded by leading experts, than in any previous edition. In addition to its practical application, the specification therefore forms a sound learning and educational platform for those new to records management as it is practised today.

MoReq2010 at its time of publication marks the 15th anniversary of the DLM Forum and the 10th anniversary of MoReq. It is hoped that this next generation version of the specification will act as both a catalyst and a springboard for the betterment of the recognition, understanding and adoption of good records management throughout Europe and internationally.

1.3 Purpose

This specification may be used,

By businesses:

- As an aid for the organisational procurement of a records system;
- As a practical tool in helping organisations configure records systems to meet their business and legal obligations; and
- As a guide to the audit of an existing records system implementation.

By experts:

- As a reference document for training courses and the preparation of course material;
- As a teaching resource for academic institutions; and
- As an example of how traditional records management approaches and archival science can be applied to modern systems requirements.

By industry:

- To guide the development of records systems by suppliers;
- To integrate records systems with other business systems; and
- As the authoritative source when undertaking the testing and certification of compliant solutions by accredited test centres.

By users:

- As a user-centric and easily understandable resource and primer on implementing records systems;
- As the original for all translations; and
- As a reference glossary for guidance on records management terms and their meanings.

MoReq2010 is best used within consumer organisations as part of an overarching records management policy within a well developed strategic framework. Educating users, fostering a culture of good organisational practice, raising awareness of organisational policy and strategy, and putting in place sound manual procedures are just as important as automation and the integration of a records system into the business environment.

ISO 15489, and the emerging ISO 30300, are solid internationally recognised resources for organisations to refer to when planning their wider business objectives around MoReq2010 compliant records systems.

1.4 Key concepts

1.4.1 Architecture

This section describes the architecture underlying MoReq2010. As there are a number of new concepts, definitions and structures it is **highly recommended** that you read this section before going on to the requirements themselves.

The new specification builds on the solid foundation of its predecessors but introduces two significant new concepts:

- Modularity, and
- Interoperability.

These and other important aspects of the MoReq2010 architecture are discussed below.

1.4.2 Modularity and versioning

The MoReq2010 specification is separated into different modules of functionality that surround a reduced set of core requirements. By selecting different combinations of modules quite different types of records systems can be specified. Additional modules can and will be added to the specification in the future by the DLM Forum without an adverse impact on those parts of the specification that have already been published. This approach not only provides great flexibility to the types of records systems that can fit within the specification, it also allows for evolutionary revision and upgrade and provides a degree of future proofing, enabling MoReq2010 to adapt to new innovations and new practices in the field of records management.

Each module in MoReq2010 is separately versioned with a major version number and a minor version number (for example, version "1.3"). Increments to the minor version number are made for purposes of clarification, amplification and error correction. In other words, a minor update does not alter any of the underlying intent of a module. A system that has been tested against version 1.3 of a module will comply with version 1.4 of the same module. By contrast the major version of a module is upgraded when either the meaning and/or the scope of the module changes. Therefore, a system that is compliant with version 1.3 of a module will not be compatible with version 2.0 of the same module.

MoReq2010 relies heavily on universally unique identifiers (UUID) to identify entities and entity type definitions. Where these are written into the specification, MoReq2010 provides the identifiers as part of the specification so that suppliers can embed them in their products. Each module of MoReq2010 is also identified with a UUID to enable systems to handshake with one another and compare which modules and which versions of MoReq2010 they support. For example, the UUID for the version of the core requirements that you are currently reading is: "be2af786-e754-430b-b7ce-7062e301bf13" and can be found under section 1.1, Important information. A new identifier will be issued to each module for every major version change to that module. The DLM Forum will not issue new identifiers for minor version changes.

1.4.3 Interoperability and XML schema

The second significant principle underlying the design of MoReq2010 is the consideration given to interoperability. The DLM Forum has a launched a separate initiative on interoperability with a much wider and more encompassing scope that will cover issues such as real time collaboration between records and archival systems, live transfer and accession. In a sense MoReq2010 prepares the way for this new initiative.

Within the bounds of MoReq2010, interoperability is confined to the XML schema that underpins the specification. This schema is used to export records and related entities from one records system and import them into another. The goal of interoperability in MoReq2010 is to transfer records between systems without a resulting loss of context. This includes the ability to start a record's lifecycle inside one system and complete it in another, something that has never been achieved by previous versions of the specification. MoReq2010 recognises that records will outlive the technology of modern records systems and may need to be transferred many times. Lossless transfer is therefore an imperative.

In order to achieve this goal of interoperability, experts will find that MoReq2010 is far more prescriptive that any previous version of the specification. It specifies not just that records management processes should be carried out, but defines how they are to be carried out, and codifies this into its functional and metadata models. Without this level of specificity one records system cannot interpret the metadata of another system. Even if two systems use metadata elements of the same name with values of the same datatype, they will not be able to understand each other's data unless they can both rely on a common usage and meaning for the values held in the metadata.

For the first time the MoReq2010 XML schema represents a true "lingua franca" between different records systems from different suppliers across different countries and market sectors. It is hoped that what may be lost by restricting developers' freedom to reinterpret the functional requirements in original ways is more than made up for by this move towards industry wide standardisation and interoperability between records systems. It should be noted that while MoReq2010 does carefully define the process and the outcome, it gives the supplier a great degree of latitude around representation and presentation and choice of technology platform, so there is little chance that all future systems will look and feel the same even though they share a common specification.

1.4.4 Records systems

Section 1.5 of the core requirements contains functional requirements for records systems. ISO 15489-1 defines a records system as an, "information system which captures, manages and provides access to records through time" (page 3). It goes on to describe the role of a records system in upholding the fundamental characteristics of a record as: authenticity, reliability, integrity and usability.

A records system that meets the detailed requirements of MoReq2010 is described throughout this specification as a **MoReq Compliant Records System** or **MCRS**. To be certified as an MCRS a records system must have been tested by a DLM Forum

accredited test centre and been issued a certificate of compliance for at least the core requirements. The DLM Forum also issues certificates of compliance against the additional modules of MoReq2010, provided an MCRS has previously (or simultaneously) been certified against the core requirements. MoReq2010 does not define these modules as "optional" as they may not be optional. In some legal jurisdictions and across some industry sectors they may be rated as highly desirable or even as required functionality.

Under section 1.5 of MoReq2010 each instance or implementation of an MCRS is regarded as an entity and given a system identifier based on a UUID as well as some other identifying metadata. It is not the brand of software but each separate site or installation that is identified in this way. This allows any record that has been transferred between systems to be traced back to the very instance in which it was created and every separate installation it has been held in subsequently.

Different instances of MCRS systems are required to be able to individually report on their level of compliance with the core requirements of MoReq2010 including which modules they support, using the unique identifiers given to different versions of different modules of MoReq2010, described above. Without this feature, different MCRS solutions would not be able to interrogate each other to discover which level of functionality the other supports.

Under MoReq2010, all MCRS systems must interface in some way either to users directly through a user interface and/or with other business systems through a services interface. MoReq2010 does not specify that an MCRS needs to do both. In order to cater for these different modes of operation between user facing and back end records systems, section 1.5 allows each supplier to choose which of the MoReq2010 series 100 interface modules their products implement. For example, module 101 contains requirements for graphical user interfaces (GUI), while module 102 contains requirements for application programming interfaces (API). Significantly for compliance all MCRS software must implement at least one of the 100 series modules. Some solutions may implement and be certified against more than one of these modules.

Also in section 1.5 the concept of an "inventory" is introduced. An inventory represents the set of entities of a particular type within each MCRS, so for example all disposal schedules are defined as belonging to a common disposal schedule inventory. MoReq2010 uses this term as a convenience to describe all the entities of a particular entity type collectively and sometimes to give them common characteristics, for example administrators may have access to the disposal schedule inventory while users do not. So long as this is true, MoReq2010 does not specify how the term "inventory" should be interpreted within different MCRS products or represented to users.

MoReq2010 uses inventories to refer to not just the set of entities of a particular entity type but also to sets of type definitions. Type definitions in MoReq2010 take on the same characteristics as entities. So for example, all entity types in MoReq2010 are identified by a system identifier, have a title and description and belong to the entity

type inventory. The difference between the entity type inventory and the disposal schedule inventory is that MoReq2010 defines the full range of entity types available. In this sense the entity type inventory is not extensible, except by additional modules of MoReq2010 where new entity types are introduced. By comparison the disposal schedule inventory within each MCRS is extensible and administrators are free to create new disposal schedules and add them to the disposal schedule inventory as required.

Where MoReq2010 provides its own type definitions, such as for entity types, it also provides the system identifiers that all MCRS software must use to refer to those type definitions in the XML schema. This means that one MCRS solution can export a record and another can import it and recognise it as a record because it uses the same MoReq2010 supplied UUID to represent a record. Because MoReq2010 provides these underlying identifiers for types defined by the specification, each system is able to refer to them by different titles and descriptions. What one MCRS describes as a "Record" might be a "Dossier" in another but they will still have the same type identifier.

Finally, section 1.5 contains requirements for system alerts. Each MCRS must have a proactive system of alerts that is used by the MCRS to notify administrators of an issue that requires attention. MoReq2010 does not specify what technologies should be used by systems to raise alerts. MoReq2010 does define the types of alerts that the MCRS must be able to raise.

1.4.5 Performing functions

Section 1.6 of the core requirements covers how records systems execute the functions defined by the functional requirements of MoReq2010. Each MCRS must keep a system log as well as support the MoReq2010 functional model and entity event histories. System logs, stored outside the system itself are valuable resources when the system fails and is not operational.

Requirements for bulk operations are also covered in section 1.6. Bulk operations allow a user to select more than one entity of the same entity type at the same time and simultaneously perform the same function on the whole selection, as if the user was performing a single function with a single entity. For example, an administrator may select multiple records and move them all into the same aggregation as a single operation, or the user may select multiple aggregations and close them in a single bulk operation.

Every function described by MoReq2010 is given its own function definition with its own system identifier, title and description. Administrators may configure the MCRS and decide which functions in the function inventory when performed result in the generation of an event that is permanently added to an entity's event history. Because of this standardisation of function definitions across the metadata model introduced by MoReq2010 an entity's whole event history will be able to be read, indexed and understood even when the entity is exported from one MCRS and imported into another. In other words, under MoReq2010 each entity's event history has become a permanent and necessary part of the metadata of that entity and replaces the concept of a system wide "audit trail" where all records system operations are recorded but in a

format that is only understood by the originating system. It is not possible under MoReq2010 to export an entity without simultaneously exporting all of its metadata including its event history.

1.4.6 Agents and roles

Section 1.7 describes the interaction between users and groups, collectively called "agents", and roles. Users and groups are both concepts that exist outside an organisation's records system. Users represent the individual staff and systems in the organisation, while groups represent its administrative and operational units. Organisations will commonly share the same users and groups across all the business systems in their network by accessing a shared directory service.

For this reason MoReq2010 specifically excludes the management of users and groups from its functional requirements. While this does not preclude an MCRS from providing its own directory services, it is relatively unlikely that the same administrators who manage the organisation's records also manage the coming and going of personnel and the formation of groups.

While the presence of authentication services and directory services within the immediate environment of the records system is assumed by the MoReq2010 specification, the long term preservation of the information they contain is not. It is equally assumed that the information held by such systems is highly volatile and that the directory service will only ever hold the most current information about a user.

Section 1.7 of the core requirements therefore requires that an MCRS regularly sample the metadata held by the directory service for users and groups, and store any changes to these entities over time as part of a MoReq2010 style event history. This regular synchronisation ensures that the records managed by the records system are stored within a rich metadata context.

By comparison with users and groups, roles are specific to the records system. In MoReq2010 roles are defined as a set of function definitions. Both users and groups can be granted roles since users will have access to roles granted to groups as a result of their group membership. This means that roles only have meaning in the context of an MCRS as the set of available functions is defined by MoReq2010.

The core requirements include only two roles. The default user role is granted to all users of the MCRS, while the administrator role provides a higher level of access to the classification scheme, aggregations, templates, disposal schedules, etc. Both the administrator and user roles defined in section 1.7 are system wide roles in the core requirements.

1.4.7 Classification

Unlike previous versions of the specification, MoReq2010 defines classification and aggregation as separate but related concepts. Classification is the subject of section 1.8.

ISO15489-2 defines classification as, "the process of identifying the category or categories of business activity and the records they generate and of grouping them, if applicable, into files to facilitate description, control, links and determination of disposition and access status" (page 16). In MoReq2010 these categories of business activity are called "classes", while the term "file" (which has a specific meaning not included in the core requirements) is replaced by the term "aggregation" (from ISO23081) in this context.

Therefore, to reinterpret the ISO definition given above in the vocabulary of MoReq2010, classification involves applying a class (representing a category of business activity) to a record, or (preferably) collectively to a whole aggregation of records.

MoReq2010 mandates that each MoReq compliant records system must implement a classification scheme that contains classes. There are several different ways of organising the classes within a classification scheme in common use. MoReq2010 does not mandate any single approach to structuring the classes within a classification scheme. Instead section 1.8 allows each supplier to choose between implementing one of the MoReq2010 series 200 classification modules. For example, module 201 contains requirements for hierarchical classification, while module 202 contains requirements for thesaurus based classification. Significantly, to achieve compliance all MCRS must implement at least one of the 200 series classification modules. Some suppliers may choose to implement more than one of these modules as this gives greater choice to consumers.

Regardless of the structure of the classification scheme that is implemented there are also requirements that are common to all forms of classification and these are detailed in section 1.8. Templates, discussed under 1.4.9 below, may be associated with classes as can disposal schedules, discussed under 1.4.10.

In previous versions of MoReq each aggregation could only ever have a single classification and each record would necessarily have the same classification as its parent aggregation. It was not possible for a single aggregation to have multiple classifications. This is now possible in MoReq2010. Aggregations and records can be given several classifications both by inheriting their parent aggregation's classes as well as by having classes applied to them directly. This is a significant change of approach.

Because a single entity can carry more than one classification under MoReq2010, the specification requires that one classification in particular must be identified as the entity's primary classification. Each aggregation and record must have at least a primary classification and may have additional secondary classifications.

Not only must every record have a primary classification but, under MoReq2010, each entity's primary classification must carry with it an associated disposal schedule. Disposal in MoReq2010 is based on the concept that an entity's fate follows its primary classification.

1.4.8 Aggregation

As has been described above, in MoReq2010, aggregation and classification are orthogonal concepts. Aggregation, the subject of section 1.9, always follows a purely hierarchical structure down from a root aggregation, which is defined as an aggregation with no higher level parent aggregation. There is no requirement for all aggregations in an MCRS to be incorporated into a single giant aggregation so the MCRS must be able to have multiple root aggregations.

At the lowest level of aggregation, aggregations contain records. At intermediate levels of aggregation, aggregations contain other aggregations. An aggregation cannot contain both records and aggregations. An MCRS must support seven levels of aggregation including records.

Aggregations in MoReq2010 do not just cluster entities together, they also place them into an implicitly ordered time sequence. Each entity that is added to an aggregation joins the end of the sequence, but an administrator can reorder the sequence if necessary. It is also possible to split an aggregation into two aggregations at any point in the sequence.

Aggregations have an open/closed property not shared with other entity types. When an aggregation is closed then no more entities can be included in it unless it is reopened. The closing of its parent aggregation is commonly used as a disposal trigger for a record.

1.4.9 Records and components

Section 1.10 covers records and record components. ISO15489-1 states that one of the characteristics of a record is that, "the links between documents, held separately but combining to make up a record, should be present" (page 7). In MoReq2010, these separate documents are called the "components" of a record and each record must have at least one, but may be made up of several components.

MoReq2010's interpretation of a record is the same as that of an "item" in ISO23081-2, which states that it is the, "smallest discrete unit of records managed as an entity" (page 12). ISO23081-2 goes on to say that while an item can have components, "the components of the item are managed as a single entity within the system". This is true of a record in MoReq2010.

When a record is created in an MCRS each of its components must be separately identified. In some records system implementations, when a record is created its components are captured and stored in the records system itself, in other implementations the components exist externally to the records system which maintains their metadata but does not provide its own integrated storage repository.

MoReq2010 refers to these different approaches as "managed components" under module 301 and "unmanaged components" under module 302 and neither approach is preferred over the other. Instead section 1.10 allows each supplier to choose between implementing one or both of the MoReq2010 series 300 component modules. When a

component is managed directly by the records system there is less risk that it will be deleted or changed and will be kept intact over time. However, when a record is stored in its originating business system there is more likelihood that it can be read, interpreted and appropriately rendered to different devices. There are many datafile formats in everyday use and it is unlikely that any centralised MCRS with its own separate repository will recognise and be able to render more than a handful of the most commonly used document types.

Unmanaged components under MoReq2010 do not just include datafiles held in external repositories. They also include paper documents; physical entities such as CDs, tapes and discs; and so called "structured" records such as individual financial transactions held in rows and tables in databases.

In this sense, MoReq2010 does not differentiate between concepts such as "structured" and "unstructured" records or "physical" and "electronic" records, but only between "managed" and "unmanaged" components. For this reason, MoReq2010 cannot be categorised as a specification that applies solely to electronic records because it is equally applicable to all records with any type of component.

One of the characteristics of a record defined in section 1.10 is that a single record can be placed into more than one aggregation. This is done by duplicating the record at the point it is associated with the second and subsequent aggregation, including duplicating the record's metadata and event history, but with each instance sharing the same components. Each instance of the record then follows a separate lifecycle and retention path from that point on with the components themselves only being destroyed when the last instance of the record is destroyed.

1.4.10 Metadata and templates

Requirements for metadata and templates can be found in section 1.11. MoReq2010 provides definitions for each of the metadata elements described in the specification, such as each entity's system identifier, title, description, created timestamp, etc. Element definitions are of two types; they either contain data in some format (textual, numeric, date/time, etc.) or they contain a reference to another entity in the MCRS, for example, the entity's entity type, the entity's primary classification, the entity's disposal schedule, the user who created the entity, etc. For element definitions that contain data, MoReq2010 requires that they be defined as W3C XML datatypes, to support interoperability with other records systems (see W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes).

In addition to the metadata elements defined by MoReq2010, administrators may create new element definitions and add them to the element inventory. These are custom or user defined metadata. Administrators can construct templates, consisting of sets of custom element definitions, and apply them to entities. For example, if a record template is associated with an aggregation and a user creates or moves a record to that aggregation then all of the elements defined in the template will be included in the metadata of the record. If any of these elements are mandatory then values for them will need to be supplied before the record can be created or moved.

MoReq2010 allows templates for classes, aggregations and records, as well as two special template types for components and users.

Component templates are unusual because they, and the element definitions they contain, are provided by the supplier of the MCRS rather than being defined in the MCRS by an administrator. Component templates capture metadata from standard industry document formats such as HTML, PDF, PNG, JPEG and OOXML datafiles. When a user adds a new component whose data format is recognised, then the MCRS will automatically select and apply the appropriate component template to the new component entity and extract the embedded metadata from the document into the corresponding elements of the component entity. Because this capability must be built into the technology of the MCRS then only the supplier can determine which component templates a particular product will support.

User templates are defined from metadata elements that match the entries in the directory service used by the MCRS. By defining a user template an administrator is determining which metadata about users will be mapped from the user account in the directory service to the corresponding user entity in the MCRS. Although the administrator has control over which elements to include in the user template, the range of element definitions available will be restricted to those that the supplier supports for any given type and version of directory service application.

1.4.11 Disposing of records

The creation of disposal schedules and the disposal process is discussed in section 1.12 of the MoReq2010 specification. In MoReq2010 aggregations are not given disposal schedules. Only records are given disposal schedules which they may derive from their primary classification under the principle, mentioned previously, that fate follows primary classification. Of course the record may well inherit its primary classification from its parent aggregation so indirectly aggregations are also involved.

Administrators may create new disposal schedules and add them to a disposal schedule inventory. MoReq2010 departs from the previous version of the specification by associating each record with one and only one disposal schedule. This greatly simplifies the MoReq2010 disposal process which is shown as a state machine in the diagram below. In this diagram a record is shown as always being in one of six possible states.

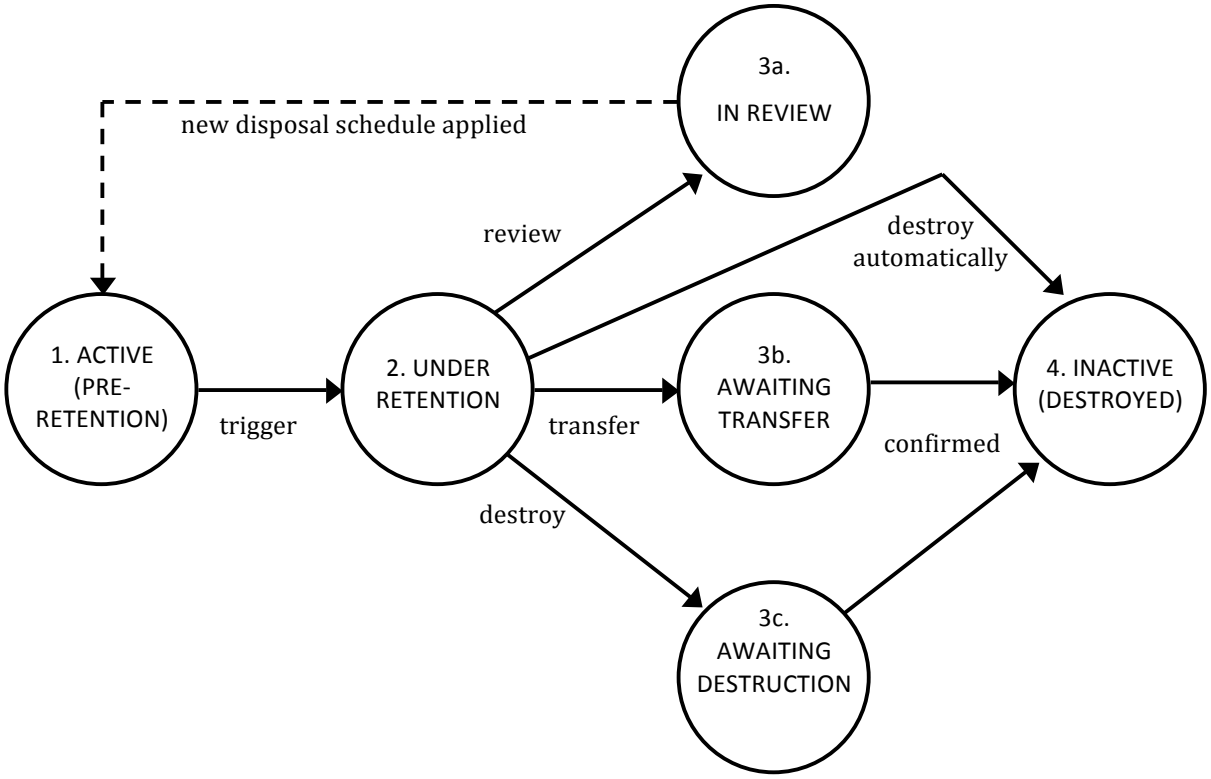
A record's disposal schedule is derived in the first instance from its primary classification. MoReq2010 requires that all aggregations and records must have a primary classification and that to be used as a primary classification a class must have an associated disposal schedule. Simply by creating a record in an aggregation it will automatically inherit the aggregation's primary classification and from it the disposal schedule associated with the primary classification.

If an administrator wishes to change the disposal schedule for a record then the administrator may give the record a new primary classification or override the default disposal schedule derived from its primary classification by applying a different

disposal schedule directly to the record. (Note that the ability to give a record a primary classification that is different to the primary classification of its parent aggregation replaces the concept of record types in previous versions of the specification.)

An administrator can override the default disposal schedule of a record that is derived from its primary classification at any time, but this is most likely to happen following a review in the disposal process as outlined in the diagram below (see step 3a.). For example, if a record is reviewed and a decision is made to destroy it in another two years then this decision will be enacted by applying a new disposal schedule ("destroy two years from now") to the record. It is unlikely that the reviewer will elect to change the record's primary classification simply to enact the review decision.

All records in the MCRS must at all times be in one of the six states described by the following state machine.



Even though they do not have disposal schedules applied to them, aggregations in MoReq2010 are always destroyed when the last record they contain is destroyed. Higher level aggregations are destroyed when the last aggregation they contain is destroyed.

When aggregations and records are destroyed they become inactive and their components are deleted. All entities in MoReq2010 have an active and an inactive state. However, in the case of aggregations and records, once they are made inactive they cannot be returned to an active state. Element definitions and function definitions

managed by the MCRS are configurable such that when an aggregation or record is made inactive some of its metadata elements are deleted and some of the events in its event history are deleted specifically to remove personal data and to ensure that a record cannot be reconstructed from its remaining metadata.

1.4.12 Disposal holds

MoReq2010 supports the concept of a disposal hold and requirements for disposal holds are contained in section 1.13. Disposal holds are becoming increasingly used in all jurisdictions to prevent records from being destroyed while legal action is pending.

MoReq2010 allows disposal holds to be created and aggregations and records to be added to them at any subsequent time. A disposal hold is therefore an entity in its own right that might represent, for example, a court order. The MCRS must maintain an inventory of all disposal holds.

While the disposal hold is active the MCRS must not destroy any aggregations or their contents down to any level of aggregation, or any individual records listed in the disposal hold. When the disposal hold is made inactive then these entities are allowed to proceed to destruction.

1.4.13 Searching and reporting

Section 1.14 contains requirements for searching and reporting. MoReq2010 does not distinguish particularly between searching and reporting. Any search criteria must be able to be saved and run as a report. Likewise any report should have an equivalent search that could be performed instead. MoReq2010 requires that an MCRS supports searches that give summary reports, with results shown as the numbers of entities matching each search category rather than listing them in detail.

1.4.14 Exporting

Importing and exporting have a particular meaning in MoReq2010 that is discussed in 1.4.2 Interoperability and XML schema, above. The requirements for exporting in section 1.15 therefore only allow entities to be exported as complete entities with their metadata and event histories intact. MoReq2010 does not allow for partial export, for example, only the components of a record without its metadata.

One of the issues with export is that because of the interrelatedness between entities in the MCRS it would be possible to choose to export a single entity and in the end export every entity in the MCRS because they are all related to one another in some way. For this reason, MoReq2010 specifies that while entities selected for export will be exported in full, any entities that they refer to will only be included in the form of an export header. The metadata that is included in an export header is configurable by an administrator but by default indicates only the system identifier of the entity and the system identifier of the entity type.

Optionally, when exporting the administrator may choose to include a certain number of generations of entity in full. For example, specifying one generation will export all the selected entities and the entities they refer to in full, specifying two generations will

include all the entities those entities refer to, and so on. Any entities not exported in full will be exported as export headers. Export headers may be matched up against full entities on import, if they have been imported previously or are imported later.

It should be noted that while export is a necessary part of the core requirements of MoReq2010, import is not part of the core requirements. This reflects the complexity of the import process. If every MCRS were required to support import from every other MCRS then the cost of a system to the end consumer would rise and the number of compliant systems would fall.

For this reason MoReq2010 requires that every MCRS must support export so that it does not become a blind alley where data once having been put into the system cannot be retrieved from it. However, not every MCRS must support import; only those that are used as middleware to sit behind front line solutions and second generation MCRS products that replace existing implementations in the future require this facility. Import is provided as an extension module to MoReq2010 and organisations can specify this functionality if they require it.

1.4.15 Non-functional requirements and other resources

In addition to functional requirements, MoReq2010 also contains non-functional requirements. Non-functional requirements are not tested as part of MoReq2010 compliance and certification. However the certification process may require the supplier to provide information about the MCRS application that is then reproduced in the testing report.

In MoReq2010, non-functional requirements differ from functional requirements by taking the form of questions. These are questions that should be asked by the organisation during procurement and during the installation of a MoReq compliant records system. In part these questions may be able to be answered by the supplier of the MCRS. However, it is more likely that answering the non-functional requirements of MoReq2010 will also require alignment between the MCRS as installed and the organisation's technical and security infrastructure under the guidance of the organisation's administrative policies and operating procedures. It is hoped that the ISO30300 series of standards, currently under development, will assist in highlighting these wider considerations and their ramifications.

Non-functional requirements for business continuity are covered in section 1.16. Business continuity includes back up and other forms of off site redundant duplication and disaster recovery following the damage, loss or failure of the MCRS. Business continuity also involves service level agreements (SLA) where service up time over a longer period of time, such as a year, is measured against agreed percentages, and planned outages for maintenance and upgrade are minimised or restricted to certain times of the week.

Section 1.17 contains non-functional requirements for performance and scalability. Performance is a measure of the responsiveness and the throughput of the MCRS while scalability is concerned with how much data it can store. The two are often linked as,

depending on the technology in use, the larger a system gets the slower it may perform necessary functions such as searching. Performance and scalability are also often affected by geographic distances between the user and the service, for example, the number of network hops and the network latency. There are no industry standards for performance and scalability of records systems and too many variables involved to establish them. An MCRS may run quite differently on a private network as compared to its performance on the public internet infrastructure. This lack of comparable performance and scalability indicators should not prevent the consumer from obtaining assurance from a supplier for a given site under measurable traffic loads and known conditions.

The final section of non-functional requirements in section 1.18, covers operational aspects of the MoReq2010 specification that the MCRS must comply with.

Other resources contained in this specification include a glossary of terms, a bibliography and references to other international records management standards and tables of MoReq2010 provided type definitions and codes.

1.5 Functional requirements for records systems

R1.5.1

The MCRS must automatically assign a system identifier to itself when it is first installed.

See requirements for system identifiers in 1.5.19 and formatting requirements in 1.11.1 and 1.11.2. Each separate instance of an MCRS must be uniquely identified. Even if the MCRS is installed as a duplicate of another system, a new system identifier should be generated on installation. This is necessary, among other reasons, to ensure the integrity of data exported from the MCRS.

In MoReq2010 the entity identified by the system identifier of the MCRS is referred to as the "system agent". It is used, for example, in event histories to identify which user performed a particular function when it was actually performed automatically by the system itself.

R1.5.2

The MCRS must automatically assign to itself a default language when it is first installed.

Every MCRS must be installed with a default language. A supplier may choose any default language for its MCRS software or may make provision for the default language to be chosen during installation. The core requirements of MoReq2010 do not make provision for the default language to be changed after installation. See the formatting requirements for language identifiers under R1.11.3.

Each textual metadata element in the MCRS is accompanied by a specified language indicator under R1.11.5. The core requirements of MoReq2010 cover monolingual MCRS solutions and the language indicator for all textual metadata will therefore be derived from the default language of the MCRS.

R1.5.3

The MCRS must automatically assign to itself a mandatory title and an optional description when it is first installed.

The supplier may provide default text for the title and optionally for the description when the MCRS is first installed. This may be changed later by an administrator under R1.5.5.

R1.5.4

The MCRS must include in its metadata the supplier's name and address; the supplier's website; the product name; product version and product support URI.

This metadata may only be changed by a product upgrade, which is outside the scope of the MoReq2010 specification.

R1.5.5
The MCRS must allow an administrator to change its title and description without deleting its title.

The title is mandatory.

R1.5.6
The MCRS must allow an administrator to set, change and delete an external identifier for the MCRS.

Different systems may interpret the external identifier in different ways. For web based systems it may be a URI, for other systems it may contain server, network or port information, for others it may be a purely textual description. Irrespective of its use, the external identifier may provide valuable contextual information about the MCRS, for example, when associated with records exported from the system.

R1.5.7
The MCRS must allow an administrator to set, change and delete an organisation name and organisation address for the MCRS.

The organisation name and organisation address are optional metadata but may provide valuable contextual information about the MCRS, for example, when associated with records exported from the system.

R1.5.8
The MCRS must allow an administrator to update the metadata of the MCRS entity by adding, changing and deleting organisation contacts consisting of pairs of contact name and contact detail.

Organisation contacts are optional metadata but may provide valuable contextual information about the MCRS, for example, when associated with records exported from the system.

R1.5.9
The MCRS must allow an administrator to change the weekday that starts each week and the month that starts the first quarter of the year.

These values are used in searching and reporting under R1.14.11. The supplier must provide default values for these elements when the MCRS is first installed.

R1.5.10
The MCRS must allow a user to inspect the identifying metadata for the MCRS.

Throughout MoReq2010 the term "inspect" is used to mean that the user is able to access the metadata of an entity without the ability to modify them. It does not imply that an MCRS must implement a particular type of visual interface, see R1.5.13.

The identifying metadata for the MCRS is listed under the rationale to R1.5.12.

R1.5.11

The MCRS must be able to report to a user, listing its level of MoReq2010 compliance, as installed. For the core requirements and each module of MoReq2010 that the MCRS is both certified against, and is installed in compliance with, the MCRS must list the identifier, title, description and version for that module.

This requirement is intended to provide users with a degree of assurance that an installed software application is MoReq2010 compliant when implemented at a particular site for a particular organisation, subject to the terms and conditions of the contract between the consumer, the supplier and the system integrator. Installed systems must not report MoReq2010 compliance against the core requirements or modules of MoReq2010 unless their suppliers have received official certification by the DLM Forum for the particular product and product version that is installed, except when they are undergoing testing for the purposes of certification by a DLM Forum accredited test centre. Some software products have many configuration settings and some depend on third-party applications for MoReq2010 compliance. Systems must not report MoReq2010 compliance against the core requirements or modules of MoReq2010 if they or any integrated third-party applications are not configured so as to exclude settings that contravene either the functional or non-functional requirements of MoReq2010.

The core requirements and all modules of MoReq2010 are issued with a unique identifier (UUID), a title, a description, and version information, see section 1.1. Each revision of either the core requirements or a module will be issued with a new unique identifier. The DLM Forum will keep a full list of all MoReq2010 identifiers, and the modules they refer to, on the MoReq2010 website. A full list of all MoReq2010 certified MCRS products and product versions, including the specific revisions of MoReq2010 modules that they have been certified against will be published on the DLM Forum website.

R1.5.12

When reporting compliance under R1.5.11, the report header must either include all of the MCRS identifying metadata, or allow the user to select which metadata to include in the report.

The MCRS identifying metadata includes:

- *The system identifier for the MCRS, see R1.5.1;*
- *The default language for the MCRS, see R1.5.2;*
- *The title and description of the MCRS, see R1.5.3 and R1.5.5;*
- *The supplier name, supplier address, supplier website URI, product name, product version and product support URI, see R1.5.4;*
- *The external identifier, if any, see R1.5.6;*
- *The organisation name and address, if any, see R1.5.7; and*
- *The organisation contacts for the MCRS, if any, see R1.5.8.*

See also the requirements for reporting in 1.14.42 and exporting under R1.15.6.

R1.5.13
The MCRS must have an interface that implements one of the MoReq2010 100 series interface modules.

An MCRS may implement more than one of the 100 series interface modules, but it must be fully compliant with and tested against at least one interface module in the 100 series.

R1.5.14
The MCRS must maintain an inventory of MoReq2010 entity type definitions and allow users to browse it.

MoReq2010 uses the term "inventory" to refer collectively to all the entities of a particular type within an MCRS. An inventory in MoReq2010 is a singleton concept, meaning that each MCRS must maintain only one inventory of entity type definitions and all entity type definitions must be included in this inventory. The term is used in MoReq2010 as a handy conceptualisation. For example, here it is used to define who may browse the entity type definitions in the MCRS, in this case, users, meaning both users and administrators.

Throughout MoReq2010 the term "browse" is used to mean "discoverable without using search". In this context, browsing means that a user should be able to access the entity type definition inventory and access all entity type definitions through a series of simple operations such as next, previous, first and last. MoReq2010 does not specify what these operations are or how they must be implemented.

The term "browse" does not imply that an MCRS must implement a particular type of visual interface, see R1.5.13.

R1.5.15
The MCRS must allow a user to inspect the metadata of an entity type definition in the entity type inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.5.16
The MCRS must allow an administrator to change the title and description of an entity type definition.

The table of entity types that must be supported by the MCRS can be found in section 1.23, including the MoReq2010 provided system identifier for each entity type definition and with suggested titles and descriptions in English.

Suppliers must implement the entity types listed in 1.23, with their MoReq2010 supplied system identifiers (see 1.11.2), but may provide their own default titles and descriptions for each entity type in any language. Administrators may then modify the titles and descriptions further within their organisational MCRS implementation in accordance with this requirement.

R1.5.17
The MCRS must allow users to search for and report on entity type definitions.

See the requirements for searching and reporting in 1.14.

R1.5.18
The MCRS must ensure that each entity created by the MCRS is a valid entity type.

The entity type must appear in the inventory of entity types. An entity's type cannot be changed.

R1.5.19
The MCRS must automatically generate a unique system identifier for each new entity that is created in the MCRS.

See format requirements for system identifiers under R1.11.1 and R1.11.2. An entity's system identifier must never change.

R1.5.20
The MCRS must automatically generate a timestamp for each new entity with the date and time it is created in the MCRS.

See formatting requirements for timestamps under R1.11.8 and R1.11.9. Even when the entity derives from another source, such as a user synchronised with a directory service, the timestamp must reflect the date and time it was created as an entity in the MCRS.

R1.5.21
The MCRS must automatically include a reference to the user that created the entity in the metadata of the entity.

Some entities are created automatically by the MCRS and not by user action. In these cases the system agent should be used as the creator of the entity. The system agent is defined in R1.5.1.

R1.5.22
The MCRS must ensure that all new entities are active.

All entities in the MCRS have an active/inactive status but are first created in the active state.

R1.5.23
The MCRS must ensure that each new entity created in the MCRS has a title and optionally an accompanying description.

MoReq2010 requires that all entities have a title but does not require that all titles be unique across the MCRS. Where the MCRS itself creates an entity then it must support a mechanism for automatically constructing the title and description, possibly by combining other metadata elements.

R1.5.24
Where MoReq2010 allows the title of an entity to be modified, the MCRS must ensure that the title is never deleted.

The title is mandatory.

R1.5.25
The MCRS must maintain an inventory of MoReq2010 alert type definitions and allow administrators to browse it.

The terms "inventory" and "browse" are explained in the rationale to R1.5.14.

R1.5.26
The MCRS must allow an administrator to inspect the metadata of an alert type definition in the alert type inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.5.27
The MCRS must allow an administrator to change the title and description of an alert type definition.

The table of alert types that must be supported by the MCRS can be found in section 1.24, including the MoReq2010 provided system identifier for each alert type definition and with suggested titles and descriptions in English.

Suppliers must implement the alert types listed in 1.24, with their MoReq2010 supplied system identifiers (see 1.11.2), but may provide their own default titles and descriptions for each alert type in any language. Administrators may then modify the titles and descriptions further within their organisational MCRS implementation in accordance with this requirement.

R1.5.28
The MCRS must allow administrators to search for and report on alert type definitions.

See the requirements for searching and reporting in 1.14.

R1.5.29
The MCRS must be able to generate alerts and raise them remotely with administrators that are not directly accessing the system.

MoReq2010 does not specify what alert mechanism must be supported and the MCRS may implement several different technologies. These must be listed and tested in order to obtain MoReq2010 certification.

Acceptable mechanisms for raising alerts may include:

- *By email;*

- *By push notification, for example SMS messaging;*
- *Published as an RSS/Atom feed;*
- *Via a subscription service, for example "twitter.com";*

Mechanisms that are not acceptable include:

- *Writing alerts to a system log file (is not proactive and requires secure access to the system logs from a remote location);*
- *By instant messaging (does not leave a traceable conversation to show whether the message was ever received);*

The alert notification mechanism used by the MCRS must allow administrators to choose whether to subscribe to alert notifications or not, see R1.7.16 and R1.7.17.

R1.5.30
The MCRS must maintain an inventory containing all the alerts it raises and allow administrators to browse it by alert type, status and created timestamp.

The terms "inventory" and "browse" are defined in the rationale to R1.5.14. By implication, an inventory must be ordered so that it may be browsed in a logical way. Otherwise browsing is no different to random access. In this requirement an MCRS must be able to order each of the alerts the inventory by whether they are active or inactive, by their recency, and by their type.

R1.5.31
The MCRS must automatically assign the appropriate alert type to each new alert it generates and make it active.

The MCRS maintains an inventory of alert types under R1.5.25. One option for the MCRS to automatically construct a title and description for a new alert under R1.5.23, is to base it on the alert type title and description, see R1.5.27.

R1.5.32
When an alert is generated then the MCRS must include in its metadata all entities that are the subject of the alert.

Alerts are always generated in reference to one or more subject entities in the MCRS, depending on the alert type.

R1.5.33
Whenever an alert is raised then the MCRS must include the event in the event history of not just the alert itself but in the event histories of each of the entities that are subjects of the alert.

In other words, the alert and the entities that are the subjects of the alert are considered to be the participating entities in the event under R1.6.28 and R1.6.29.

R1.5.34
The MCRS must include in each alert notification it issues, under R1.5.29, traversable references to the alert entity and optionally to each of the subject entities.

A "traversable reference" is one that allows a user to directly access an entity in the MCRS. An example of a traversable reference in an alert notification is a hypertext link directly to the alert entity embedded in an email message. By evoking the link the user accesses the entity directly. MoReq2010 does not specify what particular traversable reference technologies MCRS software should implement, however the user must not be required to use search to find the alert in the MCRS.

R1.5.35
The MCRS must allow an administrator make an active alert inactive by actioning it.

Actioning is an acknowledgement by an administrator that an alert has been received and any required action by the administrator is completed.

R1.5.36
When an alert is actioned under 1.5.35, the MCRS must include the event in the event history of not just the alert itself but in the event history of each of the entities that are the subject of the alert.

See 1.5.32.

R1.5.37
The MCRS must automatically generate a timestamp for any alert actioned under R1.5.35, and add it to the metadata of the alert.

R1.5.38
The MCRS must include the administrator that actions an alert under R1.5.35 in the metadata of the alert.

R1.5.39
The MCRS must allow an administrator to make an inactive alert active by recalling the previous action under 1.5.35, to show that it still requires further actioning.

R1.5.40
When an administrator recalls an action on an alert under R1.5.39, the MCRS must automatically clear the actioned timestamp set under R1.5.37 and remove the administrator that previously actioned the alert under R1.5.38 from the metadata of the alert, although they must still remain part of its event history.

R1.5.41
When an action is recalled under 1.5.39, the MCRS must include the event in the event history of not just the alert itself but in the event history of each of the entities that are the subject of the alert.

See 1.5.32 and 1.5.34.

R1.5.42
The MCRS must include a textual comment in the metadata of an alert and must allow an administrator to modify this comment as part of actioning an alert under R1.5.35, recalling an alert under 1.5.39, and at any other time.

The Comment may also be used by the MCRS to provide additional textual information relating to the alert not contained in either the title or description. For example, see R1.6.9.

R1.5.43
The MCRS must allow an administrator to select a number of alerts and perform any of the following functions as a single bulk operation:

- Action all selected alerts simultaneously, see R1.5.35;
- Recall all selected alerts simultaneously, see R1.5.39; and
- Give all the selected alerts the same comment simultaneously, see R1.5.42.

The term "select" refers to a user choosing an ad hoc and temporary grouping of entities, for example from a set of search results, in order to nominate them for collective action. When making a selection, the MCRS should not restrict the user's ability to choose which entities to include and which to exclude. MoReq2010 does not specify how an MCRS should support the user in making a selection, and the term "select" does not imply that an MCRS must implement a particular type of visual interface, see R1.5.13.

A "bulk operation" is a function performed simultaneously on a selected number of entities of the same entity type in a single operation. Further requirements for bulk operations can be found starting at R1.6.10.

R1.5.44
When performing any of the bulk operations listed under R1.5.43 the MCRS must allow the administrator to apply the same comment simultaneously to all the selected alerts.

See R1.5.42.

R1.5.45
The MCRS must allow administrators to search for and report on alerts.

See the requirements for searching and reporting in 1.14.

1.6 Functional requirements for performing functions

R1.6.1

The MCRS must perform each function as a single self-contained process unless otherwise indicated by MoReq2010.

As a general rule, MoReq2010 does not permit functions to be partially successful as this would leave the MCRS in an undefined state. For example if the metadata for an entity has been changed but there is no corresponding event history associated with the entity to record when it changed, what it was previously and who changed it. Under these circumstances the MCRS must roll back any changes made by the function before they are committed. MoReq2010 does not specify how this is done.

R1.6.2

The MCRS must maintain a system log that can be monitored by a system support technician on behalf of the organisation.

The term "log" refers to a datafile or other external data store kept separately to the MCRS so that it is readable even when the MCRS has failed to the point of being inaccessible. A system support technician is not a role within the MCRS, but a suitably qualified person appointed or contracted by the organisation to manage its business systems. The system support technician does not necessarily have to be an MCRS user.

R1.6.3

The system log for the MCRS, defined in R1.6.2, must support at least two modes of operation:

- an informational mode where every function performed by the MCRS creates a log entry; and
- an error mode where only those functions that fail, under R1.6.5 or R1.6.8, are logged.

The system log may support other modes and the supplier may choose to include other additional information as log entries.

R1.6.4

When operating in either of the two modes defined in R1.6.3, the MCRS must ensure that all log entries refer to entities only by their system identifiers and do not contain any textual metadata about the entity.

For example, the log must not contain users' names, record or aggregation titles, etc. This is a security precaution as the log is necessarily stored externally to the MCRS and for diagnostic reasons may be examined by system support technicians who are not subject to any of the access controls implemented by the MCRS.

R1.6.5

If the MCRS fails to complete a function performed by a user then it must create an entry in the system log, defined in R1.6.2, and provide feedback to the user that the function was unsuccessful.

The way in which an MCRS communicates a failed function back to the user will be dependent on the type of interface the MCRS uses, see R1.5.13.

R1.6.6

The system log entry created in R1.6.5 and R1.6.8 must contain at least the following information, subject to R1.6.4:

- The date/time of the failure;
- The identity of the function that was attempted;
- The identity of the user who performed the function;
- The identity of any participating entities; and
- Extended error information.

See the rationale to R1.6.28 for a definition of participating entities. Extended error information is specific to a particular MCRS but will generally contain diagnostic information including extended error codes, detail about the system state at the time the error occurred and software exceptions that were encountered. MoReq2010 does not specify what extended error information should be provided by an MCRS.

R1.6.7

Following the failure of a function under R1.6.5, the MCRS must provide a means by which a user can interrogate the system and retrieve extended error information following the failure of a function, without accessing the system log.

Extended error information is explained in the rationale to R1.6.6. MoReq2010 does not specify how the MCRS should provide extended error information to the user under this requirement. For example, it may be provided as part of the initial error feedback in R1.6.5 or it may be made available on follow-up request; extended error information may only be accessible temporarily, until the next error occurs or it may be written to a client cache; and so on.

R1.6.8

If the MCRS fails to complete a function performed by itself then it must create an entry in the system log, defined in R1.6.2, and raise an alert for the participating entity or entities, see R1.5.29, R1.5.32 and R1.5.33.

The MCRS must do this in all cases where there is not an active user performing the function and therefore no one to immediately report the error to.

R1.6.9

When the MCRS raises an alert under R1.6.8, it must include extended error information in the comment of the alert.

See R1.5.42.

R1.6.10
Where a user performs a single function for a selected number of entities as a bulk operation, then the MCRS must attempt to repetitively perform the same function once for each entity as if it were a single function under R1.6.1.

The rationale to R1.5.43 contains an explanation of a bulk operation. Each separate function within the bulk operation must be performed as a self-contained function, as described in R1.6.1.

R1.6.11
The MCRS must have a mechanism for automatically cancelling a bulk operation if it encounters a significant number of failures when it attempts to perform the function.

MoReq2010 does not specify what this mechanism should be or whether it can be overridden by the user or configured for different organisational needs. Neither does MoReq2010 provide a precise definition for the word "significant" when used in this requirement.

R1.6.12
The MCRS must have a mechanism to provide a user who has initiated a bulk operation with feedback on the progress of the bulk operation.

MoReq2010 does not specify what this mechanism should be or how progress should be indicated to the user during a bulk operation. Examples include, a progress bar in a visual graphical interface; an indication of the time taken so far and the expected time to complete; and/or the total number of entities to still to process and the number processed so far broken into successful and unsuccessful completions; etc.

R1.6.13
The MCRS must not prevent a user who has initiated a bulk operation for a selected number of entities under R1.6.10 from accessing the MCRS, and performing functions on entities other than those participating in the bulk operation.

The MCRS must not freeze the user's access to the system while it carries out a bulk operation.

R1.6.14
The MCRS must allow the user who requested a bulk operation under R1.6.10 to cancel it at any time before completion.

A user may choose to cancel a bulk operation for a number of reasons. The user may change his or her mind, or the bulk operation may be taking too long, for example, in the user's opinion.

R1.6.15
When a bulk operation is cancelled, under either R1.6.11 or R1.6.14, the MCRS must end it as soon as possible while retaining the integrity of the system under R1.6.1.

The MCRS must not start any new functions under the bulk operation and must decide whether to complete any partially completed functions stop them from completing and roll them back.

R1.6.16
When a bulk operation is completed or cancelled, the MCRS must provide summary feedback to the user, similar to that received under R1.6.5, for the bulk operation as a whole.

The user must not receive separate feedback individually for each participating entity.

R1.6.17
The feedback provided by the MCRS for a bulk operation under R1.6.16 must enable the user to clearly identify which of the selected entities were processed successfully, for which entities the function failed, and for which entities the function was not attempted.

See also R1.6.5. The user must have sufficient information to be able to restart the bulk operation at a later time by finding and selecting only those entities that were not successfully processed.

R1.6.18
Following the failure of one or more functions in a bulk operation, the MCRS must provide a means similar to R1.6.7 by which a user can interrogate the system and retrieve extended error information for the failed functions.

See R1.6.7.

R1.6.19
The MCRS must maintain an inventory of MoReq2010 function definitions and allow users to browse it.

The terms "inventory" and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.6.20
The MCRS must allow a user to inspect the metadata of a function definition in the function inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.6.21
The MCRS must allow an administrator to change the title and description of a function definition.

The table of function definitions that must be supported by the MCRS can be found in section 1.25, including the MoReq2010 provided system identifier for each function definition and with suggested titles and descriptions in English.

Suppliers must implement the function definitions listed in 1.25, with their MoReq2010 supplied system identifiers (see 1.11.2), but may provide their own default titles and descriptions for each function definition in any language. Administrators may then modify the titles and descriptions further within their organisational MCRS implementation in accordance with this requirement.

R1.6.22
For each function definition within the function inventory, except as specified in R1.6.24, the MCRS must allow an administrator to modify the function definition to prevent functions of that type from being included in event histories.

By default when the MCRS is first installed all functions must generate events. This requirement allows administrators to specify which functions generate events and which do not. The purpose of this requirement is to allow administrators to control the rate at which event histories grow by not including some types of events.

R1.6.23
The MCRS must allow an administrator to reset any function definition that has been modified under R1.6.22, to allow functions of that type to generate events that are included in event histories.

It should be noted that any events of this type that have occurred while the function definition was modified cannot be regenerated.

R1.6.24
The MCRS must ensure that events are always generated and added to the event histories of the function definitions for modifying a function definition under 1.6.22 and resetting a function definition under 1.6.23.

The purpose of this requirement is to ensure that any changes to the settings for whether some event types are included in event histories or not are always themselves captured as events.

R1.6.25
The MCRS must allow users to search for and report on function definitions.

See the requirements for searching and reporting in 1.14.

R1.6.26
Whenever any function described in MoReq2010 is performed for any entity in the MCRS, then the MCRS must automatically create a new event that describes the function that was performed, unless the function definition has been modified under R1.6.22.

The MCRS must maintain an event history as part of the metadata of every entity in the MCRS.

R1.6.27
The MCRS must automatically include the function definition for the function that was performed to create the event under R1.6.26 in the metadata of the event.

One option for the MCRS to automatically construct a title and description for a new event under R1.5.23 is to base it on the function definition title and description.

R1.6.28
When an event is generated then the MCRS must include in its metadata all entities that participated in the event.

For example, if a record is added to an aggregation then both the record and the aggregation have participated in the event. The function definitions in 1.25 list the participating entities.

R1.6.29
When an event is generated then the MCRS must add the event to the event history of every entity that participated in the event.

See R1.6.28 for a definition of participating entities. Note that the event should not be duplicated to be included in multiple event histories, but instead the same event, with the same system identifier, should appear in the event history of every participating entity.

R1.6.30
Whenever an event results in one or more changes to the metadata elements of a participating entity, then for each element that has changed, the MCRS must add change information to the event identifying the changed entity and the changed element and including the value of the element before the change and its new value after the change.

The function definitions in section 1.25 list the possible participating entities and, for each, which of their metadata elements may be changed by the function.

R1.6.31
Whenever a user inspects the metadata of an entity the MCRS must allow the user to browse the entity's event history.

The term "inspect" is defined in the rationale to R1.5.10. See also R1.6.36 and R1.6.37.

R1.6.32
For each function definition within the function inventory that applies to either aggregations or records, the MCRS must allow an administrator to change the function definition, so that that the MCRS deletes events based on that function definition for aggregations and records when they are destroyed.

The purpose of this requirement is to enable administrators to ensure that the detail of destroyed entities cannot be reconstructed from their event histories. Administrators may also specify that particular metadata elements are deleted from aggregations and records when they are destroyed, see R1.12.50 and R1.12.52.

Section 1.25 lists the function definitions for which this requirement applies. By default the MCRS should keep all events.

It should be noted that changing the function definition should not delete the events for aggregations and records that have already been destroyed.

R1.6.33
The MCRS must allow an administrator to change any function definition previously changed under R1.6.32, so that the MCRS keeps events based on that function definition for aggregations and records that are destroyed.

It should be noted that changing the function definition back cannot recover deleted events for aggregations and records that have already been destroyed.

R1.6.34
The MCRS must allow an administrator to select a number of function definitions and perform any of the following functions as a bulk operation:

- Prevent events being generated and added to event histories for all selected function definitions, see R1.6.22.
- Allow events to be generated and added to event histories for all selected function definitions, see R1.6.23.
- Ensure that events based on all selected function definitions are destroyed when aggregations and records are destroyed, see R1.6.32.
- Ensure that events based on all selected function definitions are retained when aggregations and records are destroyed, see R1.6.33.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.6.35
The MCRS must allow users to search for and report on any events belonging to entities whose metadata they are entitled to inspect.

See the requirements for searching and reporting in 1.14. If the user is not allowed access to the metadata then the event should not be discoverable by searching.

R1.6.36
The MCRS must not allow users to find, report on, browse to or inspect events under R1.6.31 or R1.6.35 unless the user is permitted to inspect the metadata of all participating entities.

The term "inspect" is defined in the rationale to R1.5.10 and the term "browse" is explained in the rationale to R1.5.14. The term "participating entities" is explained in the rationale to R1.6.29 and R1.6.30.

If a user browses the event history of an entity under R1.6.31 then the MCRS must ensure that events that the user is not permitted to inspect under this requirement do not appear as part of the event history.

If the user is not allowed to inspect the event then it should not be discoverable by searching in accordance with R1.14.20.

R1.6.37

The MCRS must allow the user to browse to the entity and inspect its metadata for any event discovered by searching under R1.6.35.

This requirement is the inverse of R1.6.31.

1.7 Functional requirements for agents and roles

R1.7.1

The MCRS must be integrated with one or more authentication services and must receive secure confirmation that a user is authorised and active before it allows that user to access the MCRS.

The authentication service may be external to the MCRS or built in. The MCRS may use more than one authentication service simultaneously. MoReq2010 does not specify whether a user must be authorised before each individual access to the MCRS or only once before a session consisting of multiple accesses within a specified period of time, both methods are acceptable and in common use. The term "secure confirmation" means that communication between the MCRS and the authentication service must be secured. MoReq2010 does not specify how this should be implemented.

R1.7.2

If a user attempts to access the MCRS and is not authorised by the authentication service, then the MCRS should raise an alert on the system agent.

The system agent is the singleton entity that represents the MCRS system, see R1.5.1. The alert must include the system agent as a participating entity under R1.5.32. Therefore, in accordance with R1.5.33, the alert events for all unauthorised attempts to access the MCRS must be added to the event history of the system agent.

R1.7.3

The MCRS must be integrated with one or more directory services that can securely provide information such as the name, group membership, and other details of each user on request.

The directory service may be external to the MCRS or built in. The MCRS may connect to more than one directory service simultaneously. Alternatively, the MCRS may use a single unified service as both an authentication service and a directory service. The term "securely provide information" means that communication between the MCRS and the directory service must be secured. MoReq2010 does not specify how this should be implemented.

R1.7.4

The MCRS must allow an administrator to associate a user template with each integrated directory service.

User templates are created under 1.11.22. The user template maps metadata maintained by the directory service in its users' accounts to the equivalent elements of user entities in the MCRS. The metadata elements that can be mapped in this way are dependent on how the organisation has configured its directory service. By allowing a different template for each directory service the MCRS can overcome differences in the metadata availability between different services.

R1.7.5
The MCRS must maintain a combined inventory of both user entities and group entities, collectively defined as agents, and allow users to browse it by both user and group.

The MCRS maintains the agent inventory to capture historical contextual information about users and groups related to records, aggregations and related entities. It must not be considered an alternative to having integrated authentication and directory services under 1.7.1 and 1.7.3. For this reason, the agent inventory contains only a subset of metadata about users that have accessed the MCRS at least once and their group memberships. For example, the organisation should not duplicate its entire corporate directory into the MCRS agent inventory.

The terms "inventory" and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.7.6
The MCRS must allow a user to inspect the metadata of a user or group entity in the agent inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.7.7
The MCRS must automatically create a new active user entity in the agent inventory for each authorised user that accesses the MCRS for the first time.

New user entities are created in the agent inventory whenever the MCRS is unable to match an existing user entity from the agent inventory with the external identifier for the user that is passed by the authentication service.

R1.7.8
The MCRS must automatically grant each new user it creates the default user role.

This requirement avoids the need for an administrator to manually assign a role to every new user before they can usefully access the functionality of the MCRS. If a user has no roles then the user cannot perform any functions in the MCRS, even though the user can access it. In the MoReq2010 core requirements the default user role is the only user role.

R1.7.9
The MCRS must include each user's external identifier in the metadata of the corresponding user entity.

The external identifier is the unique reference for the user's account used by the authentication and directory services. An example of an external identifier is a user's unique network sign on.

The MCRS will use the external identifier to match the user entity in the agent inventory to each user access, as described in the rationale to R1.7.7. Within the MCRS, internal

references to the user entity should always use the system identifier as this will never change and is universally unique rather than just locally unique.

R1.7.10
For each new user, the MCRS must automatically retrieve the user's name from the directory service and set it to the title of the new user entity.

MoReq2010 makes no provision for entering the user's details directly into the MCRS.

R1.7.11
For each new user entity, the MCRS must automatically retrieve any additional metadata elements from the directory service as specified by the user template associated with the directory service under 1.7.4, if any.

The value of each element in the template is read from the equivalent metadata in the directory service as specified by the template mapping.

R1.7.12
The MCRS must allow an administrator to change the status of any active user entity to inactive, to prevent the user from accessing the MCRS and to prevent the MCRS from synchronising the user entity with the directory service.

The MCRS should not attempt to synchronise inactive users with the directory service.

R1.7.13
If an authorised but inactive user attempts to access the MCRS then the MCRS must refuse the user access and raise an alert.

The alert must include the user as a participating entity under R1.5.32. Therefore, in accordance with R1.5.33, the alert event must be added to the event history of the inactive user entity.

R1.7.14
The MCRS must allow an administrator to change the status of any inactive user to active to allow the user entity access to the MCRS and to resume synchronisation with the directory service.

Note that the MCRS will automatically change the status of the user back to inactive if there is no entry for the user in the directory service under R1.7.23.

R1.7.15
Whenever the status of a user is changed from inactive to active, under R1.7.14, then the MCRS must first synchronise the user entity with the directory service before the user is able to access the MCRS.

If the user entity has been marked as inactive for some time then the metadata in the entity will not have been synchronised and may be out of date.

R1.7.16
The MCRS must allow an administrator to unsubscribe from alert notifications.

MoReq2010 includes a metadata element in the user entity that flags whether the administrator is subscribed to alerts or not. Some MCRS solutions may ignore this metadata element depending on the type of notification method adopted by the MCRS for raising alerts under R1.5.29. The MCRS solutions that ignore the MoReq2010 metadata element will be those that manage subscriptions outside the MCRS, for example using an RSS/Atom feeds which is already a subscription based service. Those MCRS solutions that do use the MoReq2010 flag should set it to true by default.

R1.7.17
The MCRS must allow an administrator to subscribe to alert notifications.

See the rationale to 1.7.16.

R1.7.18
The MCRS must raise an alert if, following any synchronisation, insufficient metadata is obtained from the directory service to enable administrators that have subscribed to alert notifications to receive them.

For some alert notification mechanisms the MCRS may need to obtain particular user details from the directory service. For example, if the MCRS sends alerts by email then email addresses will need to be copied from the directory service during synchronisation to ensure that alerts can be sent to administrators. This in turn means that the user template associated with the directory service must contain an appropriate mapping for synchronising email addresses into custom metadata elements, see R1.7.4, R1.7.11 and R1.7.19.

If the MCRS uses the MoReq2010 metadata element described in the rationale to R1.7.16, then this can be used to help decide for which users these additional metadata elements are mandatory.

R1.7.19
Whenever the MCRS synchronises a user entity with the directory service then it must automatically update the title of the user and any metadata elements specified by the user template associated with the directory service, if any.

See R1.7.10 and R1.7.11. Whenever synchronisation changes the values of metadata elements this should be captured in the synchronisation event in the user entity's event history under R1.6.29.

MoReq2010 does not specify how often the MCRS should synchronise user entities with the directory service. This may be once per user access, once per user session, or at regular timed intervals, such as daily. The MCRS may be configured to synchronise users only when they access it and perform functions. Alternatively, in a closely integrated environment, synchronisation may be externally triggered by the directory service in response to an

update of its own user accounts (in other words, "push" synchronisation rather than "pull" synchronisation). An MCRS with a built in directory service may not require synchronisation.

R1.7.20
Whenever the MCRS synchronises a user entity with the directory service it must automatically update the user's group memberships and then synchronise the active groups to which the user belongs.

R1.7.21
Whenever the MCRS synchronises a user entity with the directory service then it must automatically include the timestamp for the most recent synchronisation event in the metadata of the user entity.

If the MCRS uses a built in directory service then the last synchronised timestamp should be modified whenever the user entity is updated.

R1.7.22
The MCRS must allow an administrator to request the immediate synchronisation of an active user entity.

The user entity must be active and must be synchronised with the directory service before the user is next able to access the MCRS. If the MCRS uses a built in directory service then all user entities will always be up to date.

R1.7.23
If in attempting the synchronisation of a user entity the MCRS discovers that the corresponding user account has been deleted from the directory service then the MCRS must automatically change the status of the user entity to inactive.

See R1.7.12 and the rationale to R1.7.14.

R1.7.24
The MCRS must allow an administrator to modify the external identifier for a user entity.

This functionality is needed occasionally when a user has been accidentally deleted from the directory service and has been reinstated under a different user account. It does not apply to an MCRS with a built in directory service.

R1.7.25
If, during user synchronisation with the directory service, the MCRS discovers a group for which it does not already have a corresponding group entity, then it must create a new active group entity for that group in the agent inventory.

The MCRS must check the external identifier for the group to see whether a corresponding group entity has already been created in the agent inventory. If there is no corresponding group entity then the MCRS must automatically create one.

R1.7.26
The MCRS must automatically store the external identifier for each group as part of its group entity.

The external identifier is the unique reference for the group's account in the directory service.

R1.7.27
For each new group, the MCRS must automatically retrieve the group's name from the directory service and set it to the title of the group entity.

R1.7.28
The MCRS must allow an administrator to change the status of an active group to inactive to prevent the group entity from being automatically synchronised with the directory service by the MCRS.

In addition to not being synchronised with the directory service, any user that is a member of the inactive group must not obtain the benefit of any roles granted to the group under R1.7.37.

R1.7.29
The MCRS must allow an administrator to change the status of an inactive group to active to allow the group entity to resume synchronisation with the directory service.

R1.7.30
Whenever the status of a group is changed from inactive to active under R1.7.29 the MCRS must immediately synchronise the group entity with the directory service.

If the group entity has been marked as inactive for some time then the metadata in the entity may be out of date.

R1.7.31
Whenever the MCRS synchronises a group entity with the directory service then it must automatically update the title of the group from the name in the directory service.

MoReq2010 does not specify how often the MCRS should synchronise group entities with the directory service. See also the rationale to R1.7.19.

R1.7.32
Whenever the MCRS synchronises a group entity with the directory service then it must automatically include the timestamp for the most recent synchronisation event in the metadata of the group entity.

If the MCRS uses a built in directory service then the last synchronised timestamp should be modified whenever the group entity is updated.

R1.7.33
The MCRS must allow an administrator to request the immediate synchronisation of an active group entity.

The group entity must be active. If the MCRS uses a built in directory service then all group entities will always be up to date.

R1.7.34
If in attempting synchronisation the MCRS discovers that the corresponding group account has been deleted from the directory service then it must automatically set the status of the group entity to inactive.

See R1.7.28.

R1.7.35
The MCRS must allow an administrator to modify the external identifier for a group entity.

This functionality is needed occasionally when a group has been accidentally deleted from the directory service and has been reinstated under a different group account. It does not apply to an MCRS with a built in directory service.

R1.7.36
The MCRS must allow an administrator to grant roles to users and groups.

When a user is granted a role directly then the user will retain that role regardless of the user's group memberships.

R1.7.37
The MCRS must allow an active user to exercise any roles that have been granted either to that user directly or to any active groups to which the user belongs.

When a group entity is granted a role then all of the active users that are members of that group will inherit that role for so long as they remain group members and the status of the group is marked as active. Users do not inherit the roles granted to inactive groups.

R1.7.38
The MCRS must allow an administrator to rescind roles granted to user and group entities.

See R1.7.36 and R1.7.37. An administrator may rescind roles, including the administrator role from the administrator's own user entity. If a user has no roles then the user, though active and authorised, cannot perform any functions.

R1.7.39
The MCRS must provide a secure configuration option for granting the administrator role to a user or group if at any time the MCRS has no active administrator.

MoReq2010 does not specify how this is done. The configuration option exists only to allow the appointment of the first administrator for a newly installed MCRS, or in the event that an administrator accidentally, during normal operation, rescinds the administrator role from all active users. It must be highly secure and should not be used as a substitute for R1.7.36.

R1.7.40
Whenever the secure configuration option for granting the administrator role under R1.7.39 is used, the MCRS must make the system agent a participating entity under 1.6.28 and 1.6.29.

In other words, the event generated by R1.7.39 must be logged in the event history of the system agent, explained in R1.5.1, as well as in the event history of the user entity.

R1.7.41
The MCRS must allow an administrator to select a number of user entities from the agent inventory and perform any of the following functions as a bulk operation:

- Change the status of the selected user entities to inactive, see R1.7.12;
- Change the status of the selected user entities to active, see R1.7.14;
- Request the immediate synchronisation of all selected user entities, see R1.7.22;
- Grant a role to all selected user entities, see R1.7.36; and
- Rescind a role from all selected user entities, see R1.7.38.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.7.42
The MCRS must allow an administrator to select a number of group entities from the agent inventory and perform any of the following functions as a bulk operation:

- Change the status of the selected group entities to inactive, see R1.7.28;
- Change the status of the selected group entities to active, see R1.7.29;
- Request the immediate synchronisation of all selected group entities, see R1.7.33;
- Grant a role to all selected group entities, see R1.7.36;
- Rescind a role from all selected group entities, see R1.7.38.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.7.43
The MCRS must allow users to search for and report on users, groups and roles.

See the requirements for searching and reporting in 1.14.

1.8 Functional requirements for classification

R1.8.1

The MCRS must have a classification scheme capable of implementing at least one of the MoReq2010 200 series classification modules.

Since each MCRS must have a classification scheme it is possible for the classification scheme to be created automatically and given a default title and description when the MCRS is first installed. Otherwise an administrator must first create the classification scheme.

R1.8.2

Before any classes can be added to the classification scheme an administrator must select which method of classification it implements by choosing from one of the available MoReq2010 200 series classification modules.

If the MCRS software only supports one method of classification by being certified against only one of the 200 series classification modules then it may skip this step.

R1.8.3

The MCRS must create the classification scheme as an entity with the appropriate metadata.

The following requirements apply: R1.5.19, R1.5.20, R1.5.21 (the user may be the system agent under R1.8.1 or the administrator under R1.8.2), and R1.5.23 (the title and description may be a set to default values under the rationale to R1.8.1).

R1.8.4

The MCRS must allow an administrator to change the title and description of the classification scheme.

See R1.5.24.

R1.8.5

The MCRS must allow an administrator to associate a class template with the classification scheme so that the template is automatically applied to new classes created in the classification scheme.

See requirements for metadata and templates in section 1.11.

R1.8.6

The MCRS must allow an administrator to create new classes and add them to the classification scheme, using the class template associated with the classification scheme under R1.8.5, if any.

Since records must be classified on capture it is not possible to use an MCRS for managing records unless the classification scheme contains at least one class.

R1.8.7
The MCRS must allow an administrator to include textual scope notes in the metadata of a class.

Scope notes explain to users how a class should be used for classification.

R1.8.8
The MCRS must allow users to browse the classes in the classification scheme.

The term "browse" is defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.8.9
The MCRS must allow a user to inspect the metadata of a class in the classification scheme.

The term "inspect" is defined in the rationale to R1.5.10.

R1.8.10
The MCRS must allow users to search for and report on classes.

See the requirements for searching and reporting in 1.14.

R1.8.11
The MCRS must allow an administrator to change the values of any metadata elements associated with a class, including the title, description, scope notes, and any additional metadata in accordance with any element definitions applied to the class using a template.

Element definitions are included in the metadata of a class from a class template under R1.8.6. Scope notes are added under R1.8.7.

R1.8.12
The MCRS must allow an administrator to associate an aggregation template with a class so that it is automatically applied to aggregations classified under that class.

See requirements for metadata and templates in section 1.11.

R1.8.13
The MCRS must allow an administrator to remove an aggregation template from a class that was associated under R1.8.12.

R1.8.14
The MCRS must allow an administrator to associate a record template with a class so that it is automatically applied to records classified under that class.

See requirements for metadata and templates in section 1.11.

R1.8.15
The MCRS must allow an administrator to remove a record template from a class that was associated under R1.8.14.

R1.8.16
The MCRS must allow an administrator to associate an active disposal schedule with a class so that it becomes the default disposal schedule for records whose primary classification is that class.

See requirements for disposal schedules in section 1.12. To be used for primary classification a class must have an active disposal schedule associated with it.

R1.8.17
The MCRS must allow an administrator to replace the disposal schedule associated with a class under R1.8.16 for another active disposal schedule.

Any active record in the MCRS that currently uses the previous disposal schedule as its default disposal schedule must immediately adopt the new disposal schedule. This will apply to all active records for which this class is their primary classification. It does not apply to inactive records under R1.12.55.

R1.8.18
The MCRS must not prevent an administrator who has initiated an operation to replace one disposal schedule with another under R1.8.17, from accessing the MCRS, and performing functions on entities other than those participating in the same operation.

The MCRS must not freeze the administrator's access to the system while it carries out the replacement operation.

R1.8.19
The MCRS must allow an administrator to remove a disposal schedule from a class that was associated under R1.8.16, unless that class is being used as the primary classification for any aggregation or record in the MCRS.

A class cannot be used as a primary classification unless it has an associated disposal schedule. For this reason the disposal schedule cannot be removed unless the class is not being used for primary classification. Note that the disposal schedule can be replaced by another while the class is being used for primary classification under R1.8.17.

R1.8.20
The MCRS must allow an administrator to mark a class so that it can be directly applied to records.

By default a class can only be applied directly to classes and inherited by records. This requirement marks a class as being able to be applied directly to a record as well as a class. See also 1.8.22.

R1.8.21
The MCRS must allow an administrator to unmark a class marked under R1.8.20 so that it cannot be directly applied to records.

The class returns to the default state where it cannot be applied to records directly but only by class inheritance. Note that unmarking the class will not change any records that currently have the class applied directly.

R1.8.22
The MCRS must allow an administrator to change the status of any active class to inactive to prevent it being used for future classification, unless that class is being used as the primary classification for any active aggregation or active record in the MCRS.

If the class is being used as the primary classification of active aggregations and records then it cannot be made inactive. If it is only being used for secondary classification then it may be made inactive.

R1.8.23
The MCRS must allow an administrator to change the status of an inactive class to active to allow it to be used for future classification.

R1.8.24
The MCRS must allow an administrator to select a number of classes from the classification scheme and perform any of the following functions as a bulk operation:

- Change the value of a custom metadata element applied by template, see R1.8.11, to the same value for all selected classes;
- Associate the same aggregation template to all selected classes, see R1.8.12;
- Remove the aggregation template from all selected classes, see R1.8.13;
- Associate the same record template to all selected classes, see R1.8.14;
- Remove the record template from all selected classes, see R1.8.15;
- Associate the same disposal schedule to all selected classes, see R1.8.16;
- Remove the disposal schedule from all selected classes, see R1.8.19;
- Mark a class so that it may be applied to records directly, see R1.8.20;
- Unmark a class so that it may not be directly applied to records, see R1.8.21;
- Change the status of the selected classes to inactive, see R1.8.22;
- Change the status of the selected classes to active, see R1.8.23;

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.8.26
The MCRS must allow an administrator to replace any active class with another active class for all active aggregations and active records to which it is applied throughout the MCRS as a single operation.

The MCRS must find all aggregations and records to which the original class was been applied or inherited and change it for the replacement class. If the original class was being used as the primary classification then the alternative class must be used as the primary classification instead. This may then result in a change to the default disposal schedule for active records with this primary classification, similar though not the same as R1.8.17.

R1.8.27
The MCRS must not allow the administrator to make a class inactive under R1.8.23 if the original class has an associated disposal schedule under R1.8.16 and is being used for primary classification and the replacement class does not have an associated disposal schedule under R1.8.16.

R1.8.28
The MCRS must not prevent an administrator who has initiated an operation to replace one class with another under R1.8.26, from accessing the MCRS, and performing functions on entities other than those participating in the same operation.

The MCRS must not freeze the administrator's access to the system while it carries out the replacement operation.

R1.8.29
The MCRS must allow an administrator to delete any class that has never been used for classification.

A class cannot be deleted once it has been used as it has then become part of the history of aggregations and records in the MCRS. However, the class can be made inactive to prevent it being used for future classification, under R1.8.22.

R1.8.30
The MCRS must allow an administrator to add new classes to the classification scheme, in bulk, by reading from a datafile that describes their metadata.

The purpose of this requirement is to allow the classification scheme to be easily populated while the MCRS is being configured, and before it is used for the management of records. The datafile format to be used by the MCRS to populate the classification scheme is not specified.

This requirement is not the same as importing classes into the classification scheme (bulk import is not part of the MoReq2010 core requirements). In each case the classes described by the datafile are being newly created in the MCRS. It is therefore not appropriate to load event histories for these entities from the datafile.

R1.8.31
When new classes are added in bulk to a classification scheme, under R1.8.30, then the MCRS must not read the system identifier for each new class from the datafile but must automatically assign a new system identifier to each new class.

See R1.5.19. This requirement is intended to avoid the possibility that a given installation of an MCRS may be managing classes with the same system identifiers as similar classes in another MCRS. Such a situation would result in entities that were not universally unique.

R1.8.32

When new classes are added in bulk to a classification scheme, in accordance with R1.8.30, then the metadata read from the datafile must be valid and include any mandatory metadata elements required by MoReq2010, such as the title of the class, or required by the class template associated with the classification scheme, if any.

See R1.5.23, R1.8.5 and R1.8.6. The MCRS must not process the datafile if mandatory metadata elements are missing or invalid.

R1.8.33

When new classes are added in bulk to a classification scheme, in accordance with R1.8.30, then the MCRS must support the ability to add new classes that have an associated aggregation template and/or an associated record template.

See R1.8.12 and R1.8.14. The specified templates must have already been added to the MCRS or they must be described in the same datafile as the new class in accordance with R1.11.52.

R1.8.34

When new classes are added in bulk to a classification scheme, in accordance with R1.8.30, then the MCRS must support the ability to add new classes that have an associated disposal schedule.

See R1.8.16. The specified disposal schedule must have already been added to the MCRS or it must be described in the same datafile as the new class in accordance with R1.12.57.

1.9 Functional requirements for aggregation

R1.9.1

The MCRS must allow an administrator to create a new aggregation and add it to the MCRS without placing it into an existing aggregation.

This requirement describes a root aggregation that is not the child of another aggregation. Since records must be placed into an aggregation, it is not possible to capture records into an MCRS until it contains at least one aggregation.

R1.9.2

The MCRS must allow an administrator to create a new aggregation and add it to an open aggregation that does not contain any records.

A parent aggregation cannot have both child aggregations and child records. An aggregation must be open (and active) to have entities created in it, see R1.9.35.

R1.9.3

When a new aggregation is created under R1.9.1 or R1.9.2, then the MCRS must allow the administrator to classify it on creation by associating it with active classes from the classification scheme.

The primary classification must be selected as part of creating the aggregation.

R1.9.4

When a new aggregation is created and added to an open aggregation under R1.9.2, then the MCRS must ensure that the new aggregation automatically inherits all of it parent aggregation's classes and includes them with any new classes it is classified with under R1.9.3.

The aggregation must automatically inherit all the classes its parent aggregation is classified with. MoReq2010 does not provide a way to prevent a class from being inherited by a child entity.

R1.9.5

When creating a new aggregation under R1.9.1 or R1.9.2 the MCRS must allow an administrator to nominate which class associated with the new aggregation under R1.9.3 and R1.9.4 will be its primary classification, if the nominated class has a disposal schedule associated with it.

To be considered as a primary classification a class must have a disposal schedule associated with it. A child aggregation does not need to have the same primary classification as its parent although this is a sensible default unless the administrator chooses another under this requirement.

R1.9.6
When a new aggregation is created under R1.9.1 or R1.9.2 the MCRS must give it any additional metadata as specified by any aggregation template associated with any of the classes it is classified with under R1.9.3 and R1.9.4.

An aggregation template may be associated with a class under R1.8.12. See the requirements for metadata and templates in section 1.11.

R1.9.7
Where a template applied to an aggregation under R1.9.6 contains mandatory elements then the MCRS must ensure that valid values are provided by the administrator for these mandatory elements before the aggregation can be created.

See requirements for metadata and templates in section 1.11.

R1.9.8
When an aggregation is created under R1.9.1 or R1.9.2 the MCRS must create it as an open aggregation.

An open aggregation is one that child entities can be placed in or moved from.

R1.9.9
The MCRS must allow users to browse from an aggregation to its child entities and to its parent aggregation, if any.

The term and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.9.10
The MCRS must allow a user to inspect the metadata of an aggregation.

The term "inspect" is defined in the rationale to R1.5.10.

R1.9.11
The MCRS must allow an administrator to apply an aggregation template to an aggregation at any time after it is created and give it any additional metadata as specified by the template.

Templates are only applied automatically when an aggregation is created under R1.9.6. If the aggregation is later moved or its classification is changed then additional templates are not automatically applied. However, an administrator may decide to apply a template to an aggregation at any time.

R1.9.12
Where a template is applied to an aggregation under R1.9.11 then the MCRS must ensure that the administrator provides valid values for all mandatory elements or the template cannot be applied.

See requirements for metadata and templates in section 1.11.

R1.9.13
The MCRS must allow an administrator to change the values of any metadata elements associated with an aggregation, including the title, description, and any additional metadata in accordance with any element definitions applied to the aggregation using a template.

Element definitions are included in the metadata for an aggregation from an aggregation template under R1.9.6 and R1.9.11.

R1.9.14
The MCRS must allow an administrator to update any aggregation's classification by adding a class to it.

The class will become a secondary classification but may be promoted to primary classification under R1.9.19.

R1.9.15
When a class is added to an aggregation under R1.9.14 then the MCRS must ensure that it is automatically inherited by any child entities of that aggregation, and their children, and so on.

Once applied to a higher level aggregation, classes will follow a rule of cascading inheritance.

R1.9.16
The MCRS must allow an administrator to remove a class from an aggregation, except its primary classification or a class that it inherits from its parent aggregation, if any.

Inherited classes cannot be removed from the aggregation but they can be replaced as the primary classification and become a secondary classification, see R1.9.18. The only classes that can be removed from aggregations are those that have been applied under R1.9.14 or R1.9.3.

R1.9.17
When a class is removed from an aggregation under R1.9.16 then the MCRS must ensure that it is automatically removed from any child entities of that aggregation, and their children, and so on.

If the class is used by a child entity as its primary classification then see R1.9.18.

R1.9.18
Where a class is inherited by an active child aggregation or record and used as its primary classification, then if the class is removed under R1.9.16, or changed under R1.9.19 or R1.9.32, the MCRS must simultaneously change the primary classification of the child entity to the new primary classification inherited from its parent aggregation.

Every aggregation and record in the MCRS must have a primary classification at all times. This requirement will have a cascading effect similar to R1.9.15 and R1.9.17 as the primary classification at each level of aggregation is changed.

This requirement applies only to active aggregations and records which can have their primary classification changed. Inactive aggregations and records may not have their primary classification changed under R1.12.54.

R1.9.19
The MCRS must allow an administrator to nominate a different primary classification from the classes with which an aggregation is classified to replace its current primary classification, if the nominated class has a disposal schedule associated with it.

To be considered as a primary classification a class must have a disposal schedule associated with it. Note that a change to the primary classification of an aggregation will result in cascading changes to the primary classification of child entities, see R1.9.18.

R1.9.20
Whenever a child entity is added to an open aggregation the MCRS must include a timestamp in the metadata of the child entity to indicate the date and time it was included in the aggregation.

Child entities can be added to an aggregation by being created in them or moved to them under R1.9.2 and R.9.28. The child's timestamp is replaced if it is ever moved to a different aggregation but remains in its event history, if recorded.

R1.9.21
Whenever a child entity is added to an open aggregation the MCRS must include a timestamp in the metadata of the aggregation to indicate the date and time the most recent child entity was added to the aggregation.

Both the child's timestamp for its inclusion in an aggregation under R1.9.20, and the aggregation's timestamp for its most recent inclusion can be used as retention period start triggers, see R1.12.4.

R1.9.22
Whenever a child entity is added to an open aggregation the MCRS must include a sequence number in the metadata of the child entity to allow the contents of the aggregation to be ordered by when they were included.

The metadata model for MoReq2010 includes a next sequence number in the metadata for each aggregation to enable sequence numbers to be issued. The child's sequence number is replaced if it is ever moved to a different aggregation but remains in its event history, if recorded.

R1.9.23
If child entities are moved out of an aggregation for any reason, the MCRS must ensure that their original sequence numbers are not reissued to new arrivals.

The vacant sequence numbers are simply skipped.

R1.9.24

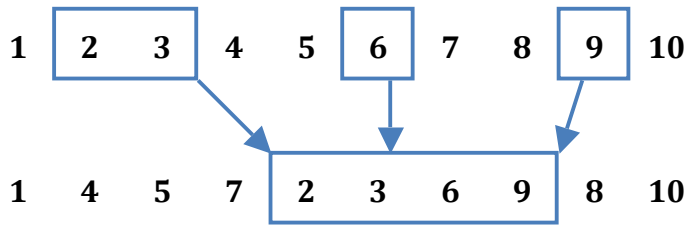
The MCRS must allow an administrator to reposition a child aggregation or record within its parent's aggregation sequence, if the parent aggregation is open.

The administrator moves the child aggregation or record to the new position currently occupied by a different child entity. The children of the aggregation all shuffle along swapping sequence numbers until they are effectively reordered. The child entity that was originally adjacent to the entity that was reordered takes the reordered entity's original sequence number.

Moving a child entity to a new position within its parent aggregation should not change the included timestamp added to the entity under R1.9.20.

This requirement can be used to reposition a set of selected child entities, not necessarily contiguous, in a single operation under R1.9.42. When this occurs the MCRS must first bring the child entities together as a block with their internal order preserved and then move the whole block together into a particular position within the sequence adjusting sequence numbers as necessary.

For example, if there were ten child entities in an aggregation then 2, 3, 6 and 9 could be selected under R1.9.42 and moved as a bulk operation to position 5 to result in a final order of 1, 4, 5, 7, 2, 3, 6, 9, 8 and 10, as shown in the diagram below:



Following this the sequence numbers would then be reallocated as described above.

R1.9.25

The MCRS must allow an administrator to split an open aggregation with two or more child entities at any point in its sequence to create two aggregations with at least one of the original child entities in each of the new aggregations.

The first part of the sequence is retained in the original aggregation while the second part of the sequence is moved to a new aggregation with the same metadata as the original aggregation, except the creation timestamp and the title, see R1.9.27.

Note that the original aggregation should not reissue the sequence numbers of the entities that have been moved and the new aggregation should not reissue the sequence numbers at the start of the original sequence under R1.9.23

R1.9.26
When an aggregation is split under R1.9.25 the MCRS must ensure that the new aggregation is created in the same parent aggregation as the original, if any, and that all of the metadata, including the classifications is copied to the new aggregation.

If the original aggregation has no parent aggregation then neither will the new aggregation. The new aggregation will have the same metadata including custom metadata as the original aggregation, except the creation timestamp, creator and the title, see R1.9.27, and of course it will have a different sequence number in its own parent aggregation to the original aggregation.

R1.9.27
When an aggregation is split under R1.9.25 the MCRS must allow an administrator to give the newly created aggregation a different title to the original aggregation.

The MCRS may propose the original title as the default for the administrator to modify.

R1.9.28
The MCRS must allow an administrator to move a child aggregation or record from one open aggregation to another open aggregation.

The child entity is removed from its current aggregation and placed into the new aggregation as the new last entity in the sequence.

R1.9.29
When a child entity is moved into an open aggregation under R1.9.28, or cloned under R1.10.33 its sequence number must be replaced with a new sequence number under R1.9.22, its aggregated timestamp must be replaced under R1.9.20, and the aggregation's timestamp for the most recent inclusion must be updated under R1.9.21.

R1.9.30
Where an administrator moves a selection of child entities from one open aggregation to another, under R1.9.28 and R1.9.42, then the MCRS must ensure that they are issued new sequence numbers under R1.9.29 in the same order as their previous sequence numbers in the aggregation they are moved to.

In other words, if a block of child entities is moved as a bulk operation then the MCRS should ensure that they stay in their same relative positions within the block.

R1.9.31
When an aggregation or record is moved to a new parent aggregation under R1.9.28, or cloned under R1.10.33, then any classes that it inherited from its previous parent aggregation must be removed and replaced with the classes it inherits from its new parent aggregation.

Note that if the entity that is moved is an aggregation (not applicable to cloning, see R1.10.33) then changing the classification of the entity in this way will have a cascading

effect similar to R1.9.15 and R1.9.17 where the inherited classes are also removed from its child entities and replaced by new inherited classes.

R1.9.32
When an active child aggregation or record is moved under R1.9.31, or cloned under R1.10.33, and the entity inherited the primary classification of its previous parent aggregation then the MCRS must simultaneously change the primary classification of the entity to match its new parent aggregation.

As with R1.9.31 if the entity that is moved is an aggregation (not applicable to cloning, see R1.10.33) then this will cause a cascading effect under R1.9.18. However, this does not apply to inactive aggregations and records that never change their primary classification under R1.12.54.

R1.9.33
The MCRS must allow an administrator to remove a child aggregation from its parent aggregation.

The administrator makes the former child aggregation a new root aggregation with no parent. This applies only to aggregations and not records.

R1.9.34
When a child aggregation is removed from its parent under R1.9.33 the aggregation's classifications must be applied directly to the aggregation, regardless of whether they were previously inherited or applied directly.

In effect there is no change to the aggregation's primary or secondary classifications and therefore no impact on any child entities in the aggregation.

R1.9.35
The MCRS must allow a user to close an open aggregation so that child entities cannot be added to it or moved out of it and its contents cannot be repositioned, so long as the aggregation does not have any open child aggregations.

Closing an aggregation prevents new child entities being added to it or removed from it under R1.9.2 and R1.9.28. It also prevents an administrator from changing the sequence of an aggregation under R1.9.24. Closing an aggregation is not the same as destroying it under a disposal schedule. An aggregation can only be made inactive by destroying it.

R1.9.36
The MCRS must allow an administrator to close an open aggregation and simultaneously close any open child aggregations and their open child aggregations, and so on.

Under 1.9.35 a user may only close one aggregation at a time. An administrator, however, may close an aggregation and cause a cascading series of closures.

R1.9.37
When an aggregation is closed under R1.9.35 or R1.9.36 the MCRS must include a timestamp in the metadata of the aggregation with the date and time the aggregation was closed.

The closed timestamp can be used as a retention period start trigger, see R1.12.4.

R1.9.38
If an aggregation is closed under R1.9.35 or R1.9.36 and it has no child entities or all of its child entities have been destroyed then the MCRS must destroy the aggregation under R1.12.52.

See also R1.12.51. Leaving an aggregation open will prevent it being destroyed when its children are destroyed. But as soon as it is closed it will trigger R1.12.52.

R1.9.39
The MCRS must allow a user to reopen a closed aggregation provided it has no parent aggregation or its parent aggregation is open.

It must never be possible for a closed aggregation to have a child aggregation that is open.

R1.9.40
When a closed aggregation is reopened then the MCRS must remove the closed timestamp applied under R1.9.37 from the metadata of the aggregation.

Note that the closing and subsequent reopening of the aggregation will remain as events in the metadata of the aggregation.

R1.9.41
The MCRS must allow an administrator to select a number of aggregations and perform any of the following functions as a bulk operation:

- Apply the same aggregation template to all the selected aggregations and specify the same values for any additional metadata elements added by the template to all the selected aggregations, see R1.9.11 and R1.9.12;
- Update any metadata element added by template to the same value for all the selected aggregations, see R1.9.13;
- Update the classification by adding a class to all selected aggregations, see R1.9.14;
- Remove a class from all selected aggregations, see R1.9.16;
- Nominate a new primary classification, the same class for all selected aggregations, see R1.9.19;
- Remove the selected aggregations from their parent aggregations, see R1.9.33;
- Close the selected aggregations, see R1.9.36;
- Reopen the selected aggregations, see R1.9.39;

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.9.42

The MCRS must allow an administrator to select a number of aggregations and records from the same parent aggregation and perform any of the following functions as a bulk operation:

- Reposition the selected child entities within their parent aggregation sequence, see R1.9.24 including the explanatory rationale;
- Move all selected child entitles to another aggregation, see R1.9.28 and particularly R1.9.30;

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

1.10 Functional requirements for records and components

R1.10.1

The MCRS must allow a user to create a new record and add it to an open aggregation that does not contain any child aggregations.

A parent aggregation cannot have both child aggregations and child records. An aggregation must be open (and active) to have entities created in it, see R1.9.35.

R1.10.2

When a new record is created under R1.10.1 the MCRS must ensure that it automatically inherits its parent aggregation's classifications including its parent's primary classification.

The record must automatically inherit all the classes its parent aggregation is classified with. MoReq2010 does not provide a way to prevent a class from being inherited by a record. A record must always have a primary classification.

R1.10.3

When a new record is created under R1.10.1 the MCRS must allow the user to classify it on creation by associating it with active classes from the classification scheme.

R1.10.4

When a record is classified by a user under R1.10.3 or R1.10.28 the MCRS must ensure that the user only adds classes that are marked as available for record level classification.

The user may only choose classes that have been so marked by an administrator under R1.8.20.

R1.10.5

When a new record is created under R1.10.1 the MCRS must allow the user to choose any class as the record's primary classification including classes added under R1.10.3, if the nominated class has a disposal schedule associated with it.

To be considered as a primary classification a class must have a disposal schedule associated with it. All records must always have a primary classification.

R1.10.6

When a new record is created under R1.10.1 the MCRS must give it any additional metadata as specified by any record template associated with any of the classes it is classified with under R1.10.2 and R1.10.3.

A record template may be associated with a class under R1.8.14.

R1.10.7
Where a template applied to a record under R1.10.6 contains mandatory elements then the MCRS must ensure that valid values are provided by the user for these mandatory elements before the record can be created.

See requirements for metadata and templates in section 1.11.

R1.10.8
When a new record is created under R1.10.1 the MCRS must ensure that the user captures or declares at least one component for the record.

A record may be made up of many components. MoReq2010 uses the term "capture" for components that are copied electronically by the MCRS into a managed repository and "declare" for physical and digital components that are managed by business systems externally to the MCRS. See R1.10.9.

R1.10.9
The MCRS must implement at least one of the MoReq2010 300 series component modules and for every component captured or declared under R1.10.8, the MCRS must include its component type in the metadata of the component.

The 300 series modules define different types of component that an MCRS may support. The MCRS may support one or more of the component types described by these records.

R1.10.10
Where the MCRS supports more than one of the MoReq2010 300 series component modules then the MCRS must allow a record to be made up of several components each of a different type.

Each component must comply with a supported module but one component may be of one type, defined by one module, while another component of the same record is of a different type, defined by a different module.

R1.10.11
When a user captures or declares a component under R1.10.8 the MCRS must automatically calculate and include a flag in the metadata of the component indicating whether or not the component can be searched using full text searching, under R1.14.6.

The value of this flag will depend on whether or not the component is textual and in a datafile format which the MCRS is able to index. It may also depend on which of the MoReq2010 series 300 component modules the MCRS implements. If the component cannot be searched using full text searching then a user will only be able to find it based on searching its metadata.

R1.10.12
Where the MCRS sets a flag in the metadata of a component indicating that it can be searched using full text searching under R1.10.11, the MCRS must store a valid language identifier with the component, indicating the language of the component text.

R1.11.3 defines a "valid" language identifier. The MoReq2010 core requirements describe a monolingual MCRS and the default language for the MCRS is set under R1.5.2.

The MCRS should attempt to identify the language of the component while indexing it for full text searching. If the MCRS cannot identify the language of the component it should assume that it is in the default language of the MCRS.

R1.10.13
When a user captures or declares a component under R1.10.8 the MCRS must automatically calculate and include a flag in the metadata of the component indicating whether or not the component is accessible by the MCRS as a datafile.

The value of this flag will depend on the type of the component and the MCRS implementation (see also the MoReq2010 Series 300 component modules). MoReq2010 does not specify any particular mechanism for setting the value of this metadata element. If the component cannot be accessed by the MCRS as a datafile then it can only be remotely managed by the MCRS. See R1.15.6.

R1.10.14
If the datafile accessibility flag is not set under R1.10.13 then the MCRS must ensure that it includes a location and external identifier in the metadata of the component.

A component's location element indicates where it can manually be found by a user. All record components that are not automatically accessible through the MCRS must have a location. The location may be in another business system. The external identifier is a reference to the component that identifies it at its location.

R1.10.15
When a user captures or declares a component under R1.10.8 the MCRS must automatically calculate and include a flag in the metadata of the component indicating whether or not the component can be automatically deleted by the MCRS.

The value of this flag will depend on the type of the component and the MCRS implementation (see also the MoReq2010 Series 300 component modules). MoReq2010 does not specify any particular mechanism for setting the value of this metadata element. If the component cannot be automatically deleted by the MCRS then it will require manual deletion by an administrator who must then confirm that it has been deleted, see R1.12.36.

R1.10.16
When a user captures or declares more than one component for a record under R1.10.8 and R1.10.9 the MCRS must not place any limit on the number of components that can be captured for the same record, provided R1.10.17.

R1.10.17
Once a record has been created under R1.10.1 the MCRS must ensure that components cannot be added, removed or replaced in the record.

The user must capture or declare all components of the record when the record is first created.

R1.10.18
When a record is created with more than one component under R1.10.16 the MCRS must include in the metadata of the record which component is the main component.

The main component is the component that should be retrieved first by a user accessing the record, for example, under R1.10.23. By way of explanation, a web page may be made up of an HTML document with associated CSS, XML and image datafiles. Of these the HTML document is the main component of the record because by invoking the HTML document in a web browser it will automatically retrieve, access and include the other components. Accessing one of the image datafile components will not have this effect.

R1.10.19
The MCRS must be able to identify components for which it provides component templates and accurately apply the relevant templates to those components when they are captured or declared under R1.10.8.

MoReq2010 does not specify how the MCRS identifies which components are compatible with which component templates or how those templates are applied. Component templates support the automatic extraction of embedded metadata from datafiles or the automatic extraction of metadata about a component managed by another business system. See R1.11.43 and R1.11.55. The supplier must provide information about the component templates that are supported by the MCRS as part of certification testing, which will then be included in the published test report for the MCRS.

R1.10.20
The MCRS must allow users to browse from a record to its parent aggregation.

The term and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.10.21
The MCRS must allow users to browse from a record to its components.

The term and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.10.22
The MCRS must allow a user to inspect the metadata of a record and its components.

The term "inspect" is defined in the rationale to R1.5.10.

R1.10.23
If a user is able to inspect the metadata of a record under R1.10.22 and the main component of the record has an accessible datafile under R1.10.13, then the MCRS must allow the user to retrieve the main component's datafile.

The main component of a record is defined under R1.10.18.

The term "retrieve" in this context means that the MCRS allows the user to download a copy of the component datafile. The original datafile remains in its permanent location. The user is then able to read and manipulate the copied datafile using another application. Once a copy of a datafile has been retrieved from the MCRS by a user the copy is no longer managed by the MCRS.

R1.10.24
If a user is able to inspect the metadata of a component under R1.10.22 and the component has an accessible datafile under R1.10.13, then the MCRS must allow the user to retrieve the component's datafile.

The term "retrieve" is defined in the rationale to R1.10.23.

R1.10.25
The MCRS must allow a user to apply a record template to a record at any time after it is created and give it any additional metadata as specified by the template.

Templates are only applied automatically when a record is first created under R1.10.6. If the record is later moved or its classification is changed then additional templates are not automatically applied. However, a user may decide to apply a template to a record at any time.

R1.10.26
Where a template is applied to a record under R1.10.25 then the MCRS must ensure that the user provides valid values for all mandatory elements or the template cannot be applied.

See requirements for metadata and templates in section 1.11.

R1.10.27
The MCRS must allow a user to change the values of any metadata elements associated with a record, including the title, description, and any additional metadata in accordance with any element definitions applied to the record using a template.

Element definitions are included in the metadata of a record from a record template under R1.10.6 and R1.10.25.

R1.10.28
The MCRS must allow a user to update any record's classification by directly applying a class to it, subject to R1.10.4.

The class will become a secondary classification but may be promoted to primary classification under R1.10.30.

R1.10.29
The MCRS must allow a user to remove a class from a record, except its primary classification or a class that it inherits from its parent aggregation, if any.

Inherited classes cannot be removed from the record but they can be replaced as the primary classification and become a secondary classification, see R1.10.30. The only classes that can be removed from records by users are those that have been directly applied under R1.10.3 or R1.10.28.

R1.10.30
The MCRS must allow a user to nominate a different primary classification from the classes with which an active record is classified to replace its current primary classification, if the nominated class has a disposal schedule associated with it.

To be considered as a primary classification a class must have a disposal schedule associated with it. The primary classification of a record may only be changed if the record is active, see R1.12.54.

R1.10.31
Whenever the primary classification of an active record is changed and the record uses the disposal schedule associated with its primary classification, the MCRS must automatically set the disposal schedule of the record to the disposal schedule associated with its new primary classification.

The MCRS must not change the disposal schedule if the default disposal schedule has been overridden, see R1.10.32. The primary classification of a record can change for many different reasons, see R1.9.18, R1.9.19, R1.9.32, R1.10.2, R1.10.5 and R1.10.30. This requirement only applies to active records, see R1.12.54 and R1.12.55.

R1.10.32
The MCRS must allow an administrator to override the default disposal schedule for an active record and apply a different disposal schedule.

This administrative power is usually associated with a review, see R1.12.42. The disposal schedule for an inactive record cannot be changed, see R1.12.55. See also R1.12.25.

R1.10.33
The MCRS must allow a user to place the same record in more than one aggregation by cloning it into a different parent aggregation from the original.

A cloned record shares the same components as the original but keeps a different set of metadata. A clone must always be created in a different aggregation.

R1.10.34
When the MCRS creates a clone of a record under R1.10.33 it must copy all of the metadata of the record, including the event history, but excluding the system identifier, date created, and creator to the clone and then move it to the specified parent aggregation.

The clone has a different system identifier but both the clone and the original share the same events in their event history up to the cloning event. From the cloning event forward the two records will not share a common event history. Note that the event history must not be duplicated but instead the clone must be added to each existing event in the original event history as a participating entity under R1.6.29.

R1.10.35
When a cloned record is created under R1.10.33 and placed in a different parent aggregation then the MCRS must automatically issue it with a new sequence number and included timestamp, see R1.9.29, replace its inherited classes, see R1.9.31, and possibly update its primary classification, see R1.9.32.

The MCRS must automatically update the cloned record in the same way as it would if it were moving the original record to a new parent aggregation under R1.9.28. Note that an administrator must move a record under R1.9.28 but a user can clone a record under R1.10.33.

R1.10.36
The MCRS must automatically include references to all records cloned from the same record under R1.10.33 in the metadata of the record and every clone.

Under MoReq2010 there is no concept of an "original" as all cloned records are indistinguishable, sharing a common event history up to the point where they branch apart from each other. A cloned record may be cloned again but under this requirement references to all cloned records must be included with every other cloned record, regardless of when they were cloned.

R1.10.37
When a record that has been cloned under R1.10.33 is destroyed under R1.12.50 then its components must not be destroyed unless the cloned record is the last active record of the set of cloned records that all share the same components.

The components are only destroyed when the last cloned record is destroyed.

R1.10.38
Inactive records may not be cloned.

See R1.10.33.

R1.10.39
The MCRS must allow a user to select a number of aggregations and create a cloned record from the same record on each aggregation as a bulk operation.

See R1.10.33 a cloned record cannot be created on the same aggregation as the record being cloned.

R1.10.40

The MCRS must allow a user to select a number of records and perform any of the following functions as a bulk operation:

- Apply the same record template to all the selected records and specify the same values for any additional metadata elements added by the template to all the selected records, see R1.10.25 and R1.10.26;
- Update any metadata element added by template to the same value for all the selected records, see R1.10.27;
- Update the classification by applying a class to all selected records, see R1.10.28;
- Remove a class from all selected records, see R1.10.29;
- Nominate a new primary classification, the same class for all selected records, see R1.10.30;
- Clone the selected records into the same new parent aggregation, see R1.10.33;
- Clone the selected records into a selected set of new parent aggregations, see R1.10.39.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.10.41

The MCRS must allow an administrator to select a number of records and perform the following function as a bulk operation:

- Override the default disposal schedule for the selected records by applying a different disposal schedule, see R1.10.32;

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

1.11 Functional requirements for metadata and templates

R1.11.1

The MCRS must generate all system identifiers as universally unique identifiers (UUID).

MoReq2010 does not specify which algorithm the MCRS should use to generate system identifiers, but the approaches listed in RFC4122 are recommended and can support "high allocation rates of up to 10 million per second per machine" (RFC4122, page 2).

R1.11.2

The MCRS must use the system identifiers that accompany the MoReq2010 specification, where provided.

For the standard alert types, the standard function definitions, and other entities provided by the specification, MoReq2010 supplies ready generated UUIDs to identify each module and revision of the specification,. For interoperability purposes the MCRS must use these standard system identifiers where supplied.

R1.11.3

The MCRS must use valid language identifiers that are compliant with RFC5646 and the IANA Language Subtag Registry.

See R1.5.2 and R1.11.5.

R1.11.4

The MCRS must be able to capture, store and export Unicode text and must use it for all textual metadata elements.

The latest version of the Unicode Standard is 6.0.

R1.11.5

The MCRS must store a valid language identifier with each textual metadata element.

R1.11.3 defines a "valid" language identifier. The MoReq2010 core requirements describe a monolingual MCRS and the default language for the MCRS is set under R1.5.2. Textual metadata input directly into the MCRS may therefore be assumed to be in the default language. This is not necessarily true of imported metadata. A planned extension module to MoReq2010 covers multilingual MCRS solutions.

R1.11.6

For the generation of timestamps the MCRS must have access to a highly accurate source of the current date, time and local time zone and must be able to adjust accordingly for changes to time zone, for example caused by shifts into and out of daylight savings/summer time, and the addition and subtraction of leap seconds.

See R1.11.8 and R1.11.9.

R1.11.7
The MCRS must be able to support different users operating in different time zones.

Depending on the architecture of the MCRS, this may be achieved by setting all timestamps to central server time, or by setting timestamps to local times and time zones but incorporating the facility to compare timestamps and search across timezones under R1.14.12. The latter option is preferred as keeping the time zone at point origin within a timestamp provides more contextual information.

R1.11.8
The MCRS must keep timestamps that are compatible with W3C XML dateTimeStamp formats.

See W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. The dateTimeStamp format is a variant of the XML date/time format which is in turn a subset of ISO 8601 that does not allow truncated or reduced formats.

R1.11.9
Where the MCRS automatically populates timestamp information, for example when assigning a value to the created metadata element under 1.5.20, then this must include time zone information and a precision of at least one millisecond.

Time zone information is necessary to determine when events actually occurred relative to each other.

R1.11.10
The MCRS must maintain an inventory of element definitions, including the MoReq2010 supplied element definitions, and allow users to browse it.

The terms "inventory" and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

When first installed the MCRS must include all of the MoReq2010 element definitions found in section 1.26. Administrators may create new element definitions local to a particular installation under R1.11.13.

R1.11.11
Where the MCRS supports the automatic extraction of embedded metadata from components or other business systems under R1.10.19, the associated element definitions must be included in the element definition inventory when the MCRS is first installed.

*Component templates are used where th*e MCRS supports the automatic extraction of embedded metadata from components or other business systems. *The supplier must provide MoReq2010 element definitions for all such metadata, see also R1.11.43.*

R1.11.12
The MCRS must allow a user to inspect the metadata of an element definition in the element inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.11.13
The MCRS must allow an administrator to create new element definitions and add them to the element inventory.

These local metadata elements may then be added to templates under R1.11.45.

R1.11.14
The MCRS must ensure that an administrator creating a new element definition under R1.11.13 nominates the element definition as being for either an element that holds a reference to an entity or an element that holds a datatype value.

The entity creator element definition, see R1.5.21, is an example of a definition for a metadata element that holds a reference to an entity; in this case a user entity. By contrast, the system identifier element definition, see R1.5.19, is an example of a definition for a metadata element that holds a datatype value; in this case a valid UUID.

R1.11.15
Where an administrator has nominated that a new element definition will hold a reference to an entity under R1.11.14, the MCRS must ensure that the administrator further specifies by nominating from the entity type inventory, which entity type or types the element will reference.

The administrator may select more than one entity type from the entity type inventory. For example, an element definition may specify an element that holds a reference to either a user entity or a group entity. The administrator may select a combination of any entity types from the inventory for an element definition.

R1.11.16
Where an administrator has nominated that a new element definition will hold a datatype value under R1.11.14, the MCRS must ensure that the administrator assigns a valid W3C XML datatype to the element definition.

MoReq2010 uses datatypes to describe element values that conform with the W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. The basis of a metadata element datatype in MoReq2010 is therefore effectively an XML simple type, whether a standard XML datatype or a user defined XML datatype.

MoReq2010 does not require that the MCRS use XML datatypes or validation internally. The MCRS may keep metadata values in a common database or other format. However, each metadata element value held in the MCRS must be capable of being expressed as the

specified XML datatype or it cannot be successfully exchanged between systems via export and import.

Note that a possible extension module to MoReq2010 to include metadata element definitions based on XML complex types has been proposed.

R1.11.17
The MCRS must ensure that for each element definition the datatype specified under R1.11.16 is valid and does not reference or link to any schema or other document.

The datatype definition must be self-contained within the element definition held by the MCRS, or long term preservation of the metadata cannot be assured. MoReq2010 does not allow XML definitions for datatypes that contain a URI to an external XSD schema.

R1.11.18
The MCRS must allow the administrator to specify which element definitions that reference datatypes are textual and require a language identifier under R1.11.5.

Some datatypes are text based but not textual and do not require an indicator of which language they are written in. For example, the datatype definition for a UUID is text based but not written in any particular language. For this reason the administrator must interpret the intent of the element definition and specify whether or not it is a textual field that should have an associated language.

R1.11.19
The MCRS must ensure that for each new element definition created under R1.11.13, the administrator specifies a minimum number of occurrences for the element.

A minimum number of occurrences of zero indicates an optional element. A minimum number of one or more indicates a mandatory element.

R1.11.20
The MCRS must ensure that for each new element definition created under R1.11.13, the administrator specifies a maximum number of occurrences for the element.

The maximum number of occurrences must be at least one and at least as many as the minimum number of occurrences. Most metadata elements will occur once. More than one maximum occurrence indicates a list element.

R1.11.21
The MCRS must ensure that the maximum number of occurrences for an element definition specified under R1.11.20 can be set to unlimited.

This represents a list that can of be any length.

R1.11.22

The MCRS must support the creation under R1.11.13 of element definitions that automatically map to metadata elements in the directory service associated with users' accounts.

The MCRS must be able to automatically extract metadata from the directory service under R1.7.4, R1.7.11, R1.7.18, and R1.7.19. To do this the MCRS must support the creation special mapped element definitions that can be added to a user template. MoReq2010 does not specify which user account metadata must be supported by mapped element definitions as this is dependent on how the organisation has configured its directory service and the particular directory service technologies supported by the MCRS.

Note that it is more usual for MCRS software to map metadata in the directory service to metadata elements that hold datatype values, see R1.11.16, rather than as references to other entities, see R1.11.15, although both are possible.

R1.11.23

The MCRS must allow an administrator to change the title and description of any element definition, including the MoReq2010 supplied element definitions.

The table of MoReq2010 element definitions found in section 1.26, includes suggested titles and descriptions in English. Suppliers may replace these with their own default titles and descriptions for each element definition, including titles and descriptions in any supported language. Administrators may then modify these titles and descriptions further within their organisational MCRS implementation in accordance with their business needs.

R1.11.24

The MCRS must allow an administrator to modify either the type of entity an element holds or the datatype it holds, as well as the minimum and maximum occurrences of the new element definition created under R1.11.13 if it has never been used.

The term "used" means that the element definition has never been applied to an entity. Once an element definition has been applied to at least one entity in the MCRS then the specification of the element definition must be frozen or existing metadata held by the MCRS would be invalidated.

R1.11.15, R1.11.16, R1.11.19 and R1.11.20 specify the entity type(s), datatype, minimum occurrences and maximum occurrences, respectively. R1.11.18 specifies whether the element definition is written in a language and is also modifiable until the entity is used.

This requirement does not apply to the MoReq2010 supplied element definitions under R1.11.10, the supplier provided element definitions for components under R1.11.11, or the element definitions for user templates under R1.11.22.

R1.11.25

The MCRS must allow an administrator to set, modify or delete a default value for each element definition created under R1.11.13.

Only element definitions created under R1.11.13 may be given default values. MoReq2010 supplied element definitions may not be given default values. The default value must be either an existing entity in the MCRS of the correct entity type, see R1.11.15, or a data value that is valid according to the datatype of the element definition, see R1.11.16. The default value for an element definition is the initial value given to elements of that definition when they are first added to entities, see R1.11.63.

R1.11.26

Further to R1.11.25, the MCRS must allow the administrator to specify more than one default value and up to the maximum number of occurrences, if the element definition is for a list of values.

Element definitions can describe lists of values under R1.11.20, while R1.11.21 limits the maximum number of occurrences in a list to a particular value.

R1.11.27

Where the administrator specifies a default value for an element definition under R1.11.25 and the element definition requires a language identifier under R1.11.18, then the MCRS must set the language for the default value(s) to the default language for the MCRS.

The default language for the MCRS is specified under R1.5.2. Note that for elements with a list of values there is only one language identifier for the list as a whole. See also R1.11.59.

R1.11.28

For each element definition within the element inventory, the MCRS must allow an administrator to set the element definition, so that that the MCRS deletes elements belonging to that element definition from aggregations and records when they are destroyed.

The purpose of this requirement is to enable administrators to ensure that the detail of destroyed entities cannot be reconstructed from their metadata elements. Administrators may also specify that particular events are deleted from the event histories of aggregations and records when they are destroyed, see R1.6.32. By default the MCRS should keep all metadata elements.

It should be noted that changing the element definition should not delete the corresponding elements from aggregations and records that have already been destroyed.

R1.11.29

The MCRS must never allow an administrator under R1.11.28 to set the system identifier element definition for deletion from aggregations or records when they are destroyed.

All entities, whether active or inactive, must always retain their original system identifier, see R1.5.19.

R1.11.30
The MCRS must allow an administrator to reset any element definition previously modified under R1.11.28, so that the MCRS retains metadata elements belonging to the element definition when aggregations and records are destroyed.

It should be noted that resetting the element definition back cannot recover any deleted elements belonging to the element definition for aggregations and records that have already been destroyed.

R1.11.31
The MCRS must allow an administrator to delete an element definition created under R1.11.13 if it has never been used.

The term "used" is defined in the rationale to R1.11.24. Once an element definition has been applied to an entity it cannot be deleted but it can be made inactive under R1.11.33. The MoReq2010 supplied metadata elements must never be deleted.

R1.11.32
When an element definition is deleted under R1.11.31, the MCRS must ensure that it is removed from all templates to which it has been added, see R1.11.47.

R1.11.33
The MCRS must allow an administrator to change the status of an element definition created under R1.11.13 or R1.11.22 from active to inactive, to prevent it being added to templates or applied to entities in the future.

Only active element definitions are added to entities when a template is applied to them, see R1.11.57. The MoReq2010 supplied metadata elements must never be made inactive.

R1.11.34
When an element definition is made inactive under R1.11.33, the MCRS must allow the administrator to optionally specify that it be simultaneously automatically removed by the MCRS from all templates to which it has been added.

If the administrator elects not to automatically remove the element definition from templates it still cannot be applied to entities, see R1.11.33 and R1.11.57.

R1.11.35
The MCRS must ensure that users cannot modify the values of elements belonging to element definitions that have been made inactive under R1.11.33.

When an element definition is made inactive then the corresponding elements of that type remain as valid metadata for entities in the MCRS, however these elements are effectively switched to "read only" status and cannot be further updated.

R1.11.36
The MCRS must allow an administrator to change the status of an inactive element definition to active, to allow it to be included in templates and applied to entities.

If the element definition has been automatically removed from templates under R1.11.34 then the administrator will also need to add it to at least one template under R1.11.45 before it can be applied to entities.

R1.11.37
The MCRS must allow an administrator to select a number of element definitions from the element inventory and perform any of the following functions as a bulk operation:

- Give the same default value or no default value to all selected element definitions, see R1.11.25;
- Set all selected element definitions, so that that the MCRS deletes elements belonging to those definitions from aggregations and records when they are destroyed, see R1.11.28;
- Reset all selected element definitions, so that the MCRS retains elements belonging to those definitions when aggregations and records are destroyed, see R1.11.30;
- Change the status of the selected element definitions to inactive, see R1.11.33; and
- Change the status of the selected element definitions to active, see R1.11.36.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.11.38
The MCRS must maintain an inventory of templates, and allow users to browse it.

The terms "inventory" and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.11.39
The MCRS must allow a user to inspect the metadata of a template in the template inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.11.40
The MCRS must allow an administrator to create a new template and add it to the element inventory.

MoReq2010 does not include any predefined templates, although they may be specified by some jurisdictions. Predefined templates may also be loaded from a datafile, see R1.11.52.

R1.11.41
The MCRS must ensure that each new template created under R1.11.40 is associated with an entity type by the administrator, so that it can be applied to entities of that type.

It is required that each template be created for a particular entity type. MoReq2010 does not include provision for templates that apply to more than one entity types. There is also no provision to change the entity type to which a template applies.

R1.11.42
The MCRS must allow an administrator to associate templates with the following entity types under R1.11.41:

- Classes;
- Aggregations;
- Records; and
- Users.

These are the only types of user defined templates specified within the core requirements of MoReq2010. The MCRS may also come with one or more built in component templates provided by the supplier under R1.11.43.

R1.11.43
If the MCRS supports one or more component templates then these must be added to the template inventory at installation along with the associated element definitions under R1.11.11.

The automatic extraction of embedded metadata from components and about component from other business systems by the MCRS is specified in R1.10.19. Component templates for this must be provided by the supplier as they are dependent on the specific technology built into individual MCRS solutions.

R1.11.44
The MCRS must allow an administrator to modify the title and description of a template.

R1.11.45
The MCRS must allow an administrator to add any active element definition created under R1.11.13 to a class, aggregation or record template so that it is automatically applied with the template.

Only active element definitions may be added to templates, see R1.11.33. The MoReq2010 supplied element definitions cannot be added to templates as they are automatically applied to entities of the appropriate type as specified by these functional requirements.

R1.11.46
The MCRS must allow an administrator to add any active element definition created under R1.11.22 to a user template so that it is automatically applied with the template.

User templates define which user account metadata can be synchronised from the directory service. Only metadata elements that map to the directory service may be included in user templates.

R1.11.47
The MCRS must allow an administrator to remove element definitions from templates if they have been added under R1.11.45 or R1.11.46.

Both active and inactive element definitions may be removed from a template. Note that adding or removing element definitions from a template will not affect the elements belonging to existing entities to which the template has been previously applied.

R1.11.48
The MCRS must allow an administrator to delete a template if it has never been applied to an entity.

Once the template has been used it cannot be deleted, but it may be made inactive under R1.11.49.

R1.11.49
The MCRS must allow an administrator to change the status of a template from active to inactive, to prevent it being applied to entities in the future.

It must be possible to make any template inactive, including user and supplier provided component templates if the organisation does not wish to use it.

R1.11.50
The MCRS must allow an administrator to change the status of an inactive template to active, to allow it to be applied to entities.

R1.11.51
The MCRS must allow an administrator to select a number of templates from the template inventory and perform any of the following functions as a bulk operation:

- Add element definitions to the selected templates, see R1.11.45 and R1.11.46;
- Remove element definitions from the selected templates, see R1.11.47;
- Change the status of the selected templates to inactive, see R1.11.49; and
- Change the status of the selected templates to active, see R1.11.50.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.11.52
The MCRS must allow an administrator to add new element definitions to the element inventory and new templates to the template inventory, in bulk, by reading from a datafile that describes their metadata.

The purpose of this requirement is to allow element definitions and templates to be easily populated while the MCRS is being configured, and before it is used for the management of records. The datafile format to be used by the MCRS to populate the element inventory and the template inventory is not specified. Element definitions and templates may be loaded from the same datafile that is used to load classes into a classification scheme

under R1.8.30 and particularly R1.8.33, or disposal schedules into the schedule inventory under R1.12.57.

This requirement is not the same as importing element definitions and templates (bulk import is not part of the MoReq2010 core requirements). In each case the entities described by the datafile are being newly created in the MCRS. It is therefore not appropriate to load event histories for these entities from the datafile.

R1.11.53
When new element definitions and templates are created in bulk from a datafile, under R1.11.52, then the MCRS must not read the system identifier for each entity from the datafile but must automatically assign a new system identifier to each new entity.

See R1.5.19. This requirement is intended to avoid the possibility that a given installation of an MCRS may be managing element definitions and/or templates with the same system identifiers as similar entities in another MCRS. Such a situation would result in entities that were not universally unique.

R1.11.54
When new element definitions and templates are added in bulk, in accordance with R1.11.55, then the MCRS must ensure that the metadata read from the datafile for each entity is valid and includes all mandatory metadata elements required by MoReq2010.

The MCRS must not create any element definition or template with missing or invalid metadata.

R1.11.55
Whenever an entity is created the MCRS must apply a template to the new entity if there is one available and the template is active and for the same entity type as the new entity.

Class templates associated with the classification scheme will be applied to new classes under R1.8.6 and R1.8.32. Aggregation templates and record templates associated with an aggregation or record's primary classification will be applied under R1.9.6 and R1.10.6. User templates associated with the directory service will be applied under R1.7.11. Component templates are applied automatically by the MCRS to new components under R1.10.19. Each template must be associated with an entity type under R1.11.41. An inactive template cannot be applied to an entity.

R1.11.56
The MCRS must allow an administrator to apply active templates to aggregations and records if they are for that entity type at any time.

Each template must be associated with an entity type under R1.11.41. Templates may be applied to aggregations under R1.9.11 and records under R1.10.25. An inactive template must not be applied to an entity.

R1.11.57

When a template is applied to an entity under R1.11.55 or R1.11.56 then the MCRS must create elements based on the active element definitions in the template and add them to the entity.

Elements are only added to entities for active element definitions in the template. An entity may have any number of elements associated with it, subject to R1.11.58.

Elements, like event histories, are not entities, they are part of the metadata of an entity. An element definition however is considered to be a separate entity in its own right as is each separate event.

R1.11.58

When elements based on the element definitions in a template are added to an entity under R1.11.57, the MCRS must not create duplicates of any elements that already exist for that entity and belong to the same element definitions.

Each element added to an entity's metadata must belong to a different element definition. Note that element definitions can describe lists of values, see R1.11.19 and R1.11.20, but the list as a whole should be considered as a single element.

If a template is applied to an entity and the entity already has an element corresponding to an element definition in the template then the element definition in the template is ignored.

R1.11.59

When elements based on the element definitions in a template are added to an entity for the first time under R1.11.57, the MCRS must ensure that the language identifier for the element is set to the default language for the MCRS.

R1.11.18 specifies that textual element definitions must be identified. Elements based on these element definitions must have an accompanying language under R1.11.5. The default language for the MCRS is specified under R1.5.2. Note that for elements with a list of values there is only one language for the list as a whole.

R1.11.60

When elements based on the element definitions in a template are added to an entity for the first time under R1.11.57, the MCRS must copy into the element the default value or values specified in the element definition along with the accompanying language identifiers, if any.

R1.11.25 describes how element definitions may be given default values, while R1.11.26 describes how list definitions may be given multiple default values. R1.11.27 requires default values to be accompanied by a language identifier.

R1.11.61

The MCRS must ensure that once metadata elements are added to an entity by applying a template they cannot be removed except as part of the destruction of an aggregation or record.

It is important to note that under MoReq2010, once a template has been applied to an entity and the elements defined by the template have been added to the entity, the MCRS should not continue associating the entity with the template. If the template definition is subsequently changed under R1.11.45, R1.11.46 or R1.11.47 then this has no effect on the entity or its elements. Other templates can also be applied to the entity which under R1.11.57 will result in more elements belonging to different element definitions being added to the entity's metadata.

Once metadata elements have been created and added to an entity, MoReq2010 does not allow them to be removed from the entity, except under controlled circumstances when aggregations and records are destroyed, under R1.12.50 and R1.12.52. The element definition must indicate that the element is to be deleted from aggregation and record entities at destruction, under R1.11.28. However, metadata elements may be made "read only" by making their element definitions inactive under R1.11.33.

1.12 Functional requirements for disposing of records

R1.12.1

The MCRS must maintain an inventory of disposal schedules, and allow users to browse it.

The terms "inventory" and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.12.2

The MCRS must allow a user to inspect the metadata of any disposal schedule in the disposal schedule inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.12.3

The MCRS must allow an administrator to create a new disposal schedule and add it to the disposal schedule inventory.

The schedule inventory should support any number of schedules.

R1.12.4

For each new disposal schedule created under R1.12.3 the MCRS must ensure that the administrator includes a retention period start trigger in the metadata of the disposal schedule with one of the following values:

- NOW
- NEVER
- DATE OF LAST REVIEW
- DATE RECORD CREATED
- DATE RECORD INCLUDED IN AGGREGATION
- DATE LAST RECORD INCLUDED IN AGGREGATION
- DATE AGGREGATION CREATED
- DATE AGGREGATION CLOSED
- METADATA ELEMENT DATE

Each MCRS may describe these trigger codes in a different way and in a different language for display purposes, but they must all be supported. MoReq2010 specifies identifiers for each of these codes in the tables in section 1.28 which must be used when exporting disposal schedules in section 1.15.

A retention trigger of NOW is used as a means of initiating the retention period immediately without waiting for the next cyclical update under R1.12.32.

R1.12.5

Where an administrator selects a retention trigger of NEVER under R1.12.4 the MCRS must not permit the simultaneous inclusion of any of the following elements in the metadata of the disposal schedule:

- A retention period trigger element under R1.12.6;
- A retention period interval under R1.12.8;
- A retention period interval duration under R1.12.9;
- A retention period offset under R1.12.11;
- A retention period offset month under R1.12.12;
- A disposal action under R1.12.13.
- A disposal period interval under R1.12.14; and
- A disposal period interval duration under R1.12.15.

A trigger event of NEVER is used to mark a record for permanent retention. The metadata elements listed are not applicable if the record is to be retained permanently.

R1.12.6

Where an administrator selects a retention trigger of METADATA ELEMENT DATE under R1.12.4 the MCRS must ensure that the administrator identifies a retention period trigger element definition and includes it in the metadata of the disposal schedule.

A retention trigger of METADATA ELEMENT DATE means that the trigger date will be obtained by the MCRS from a metadata element that is applied to a record by a template or inherited by the record from its parent aggregation.

R1.12.7

Where an administrator selects a retention period trigger element definition under R1.12.6 the MCRS must ensure that the value of the element definition selected by the administrator is based on a datatype under R1.11.16 and that the datatype describes a date, date/time, or timestamp.

The retention period trigger element must correspond to an element definition that specifies a date based datatype.

R1.12.8

For each new disposal schedule created under R1.12.3 the MCRS must ensure that the administrator includes a retention period interval in the metadata of the disposal schedule consisting one of the following values:

- NO RETENTION PERIOD
- DAYS
- WEEKS
- MONTHS
- YEARS

Each MCRS may describe these period codes differently and in a different language for display purposes. MoReq2010 specifies identifiers for each of these codes in the tables in section 1.28 which must be used when exporting disposal schedules.

A value of NO RETENTION PERIOD *means that there will be no period of retention for the record. The retention date will be set under R1.12.33 to the same value as the disposal schedule trigger date. NO RETENTION PERIOD must be used in combination with a retention trigger of NOW (see R1.12.4) to construct a disposal schedule for the immediate disposal of a record.*

R1.12.9

Where an administrator selects a retention period interval of DAYS, WEEKS, MONTHS or YEARS under R1.12.8 the MCRS must also include a retention period interval duration in the metadata of the disposal schedule consisting of a number greater than zero.

The retention period interval duration must be a whole number greater than zero. Therefore the shortest retention period is one day.

R1.12.10

When specifying a retention period interval duration under R1.12.9 the MCRS must allow the administrator to set a retention period of up to 100 years, or the equivalent in days, weeks or months.

The equivalent is a maximum of 36,524 days, 5,217 weeks or 1,200 months. The MCRS should not allow more than these values as a substitute for constructing a disposal schedule that properly specifies permanent retention using the NEVER value under R1.12.4.

R1.12.11

For each new disposal schedule created under R1.12.3 the MCRS must ensure that the administrator includes a retention period offset in the metadata of the disposal schedule consisting one of the following values:

- NO OFFSET
- START OF NEXT MONTH
- START OF NEXT QUARTER
- START OF SPECIFIED MONTH

Each MCRS may describe these offset codes differently and in a different language for display purposes. MoReq2010 specifies identifiers for each of these codes in the tables in section 1.28 which must be used when exporting disposal schedules.

An offset period of NO OFFSET means that the retention date will fall exactly on the calculated date. An offset period of START OF NEXT MONTH will extend the retention period to the end of the month in which it falls.

R1.12.12

Where an administrator selects a retention period offset of START OF NEXT QUARTER or START OF SPECIFIED MONTH under R1.12.11 then the MCRS must also include a retention period offset month in the metadata of the disposal schedule.

An offset period of START OF NEXT QUARTER will extend the retention period to the end of the quarter of the year in which it falls. For this option the retention period offset month is interpreted as being the first month of the first quarter of the year. For example, if the month were set to APRIL then the quarters would run April to June, July to September, October to December and January to March.

An offset period of START OF SPECIFIED MONTH will extend the retention period to the end of the month before the start of the retention period offset month. For example, to extend the retention period to the end of the calendar year in which it falls, the administrator would set the retention period offset month to JANUARY.

R1.12.13

For each new disposal schedule created under R1.12.3 the MCRS must ensure that the administrator includes a disposal action in the metadata of the disposal schedule consisting one of the following values:

- REVIEW
- TRANSFER
- DESTROY AFTER CONFIRMATION
- DESTROY WITHOUT CONFIRMATION

Each MCRS may describe these period codes differently and in a different language for display purposes. MoReq2010 specifies identifiers for each of these codes in the tables in section 1.28 which must be used when exporting disposal schedules.

Note that destruction is the only real end point in the disposal cycle. A record may be given a disposal schedule with a disposal action of REVIEW. However an administrator must replace this with a different disposal schedule to complete the review process under R1.12.42.

Similarly a record may be given a disposal schedule of TRANSFER. However transfers must always be confirmed before they are destroyed under R1.12.47.

Some components that are not under the direct control of the MCRS cannot be destroyed without confirmation. Records of this type given a DESTROY WITHOUT CONFIRMATION disposal schedule will interpret it as DESTROY AFTER CONFIRMATION under R1.12.36.

R1.12.14

For each new disposal schedule created under R1.12.3 the MCRS must ensure that the administrator includes a disposal period interval in the metadata of the disposal schedule consisting one of the following values:

- DAYS
- WEEKS

Each MCRS may describe these period codes differently and in a different language for display purposes. MoReq2010 specifies identifiers for each of these codes in the tables in section 1.28 which must be used when exporting disposal schedules.

The disposal period refers to maximum time it should take an administrator to complete the disposal action nominated under R1.12.14, before the MCRS raises a reminder with an alert, under R1.12.38.

R1.12.15
When specifying a disposal period interval under R1.12.14 the MCRS must ensure that the administrator also includes a disposal period interval duration in the metadata of the disposal schedule consisting of a number greater than zero.

The disposal period interval duration must be a whole number greater than zero. Therefore the shortest disposal period is one day.

R1.12.16
When specifying a disposal period interval duration under R1.12.15 the MCRS must not allow the administrator to set a disposal period greater than 90 days or 13 weeks.

R1.12.17
For each new disposal schedule created under R1.12.3 the MCRS must allow an administrator to include a textual mandate in the metadata of a disposal schedule.

The mandate describes the authority and jurisdiction under which the disposal schedule operates.

R1.12.18
For each new disposal schedule created under R1.12.3 the MCRS must allow an administrator to include a mandate URI in the metadata of a disposal schedule.

The mandate URI can contain a hyperlink reference to the published mandate, if it is URI addressable.

R1.12.19
For each new disposal schedule created under R1.12.3 the MCRS must allow an administrator to include textual scope notes in the metadata of a disposal schedule.

Scope notes explain to administrators under what circumstances a disposal schedule should be applied.

R1.12.20
The MCRS must allow an administrator to modify the following metadata for any disposal schedule that has never been applied to any class, aggregation or record:

- The retention period start trigger, see R1.12.4;
- The retention period trigger element, if applicable, see R1.12.6;
- The retention period interval, see R1.12.8;
- The retention period interval duration, if applicable, see R1.12.9;
- The retention period offset, see R1.12.11;
- The retention period offset month, if applicable, see R1.12.12;
- The disposal action, see R1.12.13.
- The disposal period interval, see R1.12.14; and
- The disposal period interval duration, if applicable, see R1.12.15;

Once a disposal schedule has been used and applied to entities, these aspects of the disposal schedule cannot be altered. Note that these metadata are not applicable to disposal schedules where a retention period start trigger of NEVER has been specified, see R1.12.5.

R1.12.21
The MCRS must allow an administrator to modify the following metadata for any disposal schedule at any time:

- The title and description, see R1.5.23;
- The textual mandate, see R1.12.17;
- The mandate URI, see R1.12.18; and
- The scope notes, see R1.12.19.

R1.12.22
The MCRS must allow an administrator to change the status of any active disposal schedule to inactive, so long as it is not applied to any active or inactive class or active record.

Only disposal schedules that are not in use can be made inactive. The disposal schedule must first be removed from any classes, aggregations or records to which it is applied, possibly by applying R1.12.25.

Inactive classes can be made active again under R1.8.23. The disposal schedule for an inactive aggregation or record cannot be changed under R1.12.55.

An inactive disposal schedule cannot be applied to any entity until it has been made active again.

R1.12.23
The MCRS must allow an administrator to change the status of an inactive disposal schedule to active, to allow it to be applied to active classes, aggregations and records.

R1.12.24
The MCRS must allow an administrator to select a number of disposal schedules from the disposal schedule inventory and perform any of the following functions as a bulk operation:

- Change the status of the selected disposal schedules to inactive, see R1.12.22; and
- Change the status of the selected disposal schedules to active, see R1.12.23.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.12.25
The MCRS must allow an administrator to replace any active disposal schedule with another active disposal schedule for all active classes, inactive classes, and active records to which it is applied throughout the MCRS as a single operation.

The MCRS must find all affected entities to which the original disposal schedule has been applied and apply the replacement disposal schedule to them instead.

R1.12.26
The MCRS must not prevent an administrator who has initiated an operation to replace one disposal schedule with another under R1.12.25, from accessing the MCRS, and performing functions on entities other than those participating in the same operation.

The MCRS must not freeze the administrator's access to the system while it carries out the replacement operation.

R1.12.27
When the MCRS replaces one disposal schedule with another for one or a number of entities in a single operation either under R1.8.17, R1.10.31 or R1.10.32 it must not check and update the disposal progress of any affected entity under R1.12.32 until the replacement operation is complete.

It is possible by changing the primary classification of aggregations, for example, under R1.9.18 to trigger a cascading change of primary classification which may eventually lead to a change of disposal schedule for a number of records simultaneously under R1.10.31. The MCRS must avoid taking disposal actions on any entities in accordance with their original disposal schedules while they are simultaneously queued to have their disposal schedules replaced.

R1.12.28
Whenever a record has its disposal schedule replaced it must simultaneously clear the following elements from its metadata:

- The retention date, see R1.12.33
- Disposal action, see R1.12.35;
- Disposal action due date, see R1.12.35;
- Disposal action reminder sent timestamp, see R1.12.38;

But not the last reviewed timestamp under R1.12.44 or the last review comment under R1.12.45.

The disposal schedule of an active record can be replaced under R1.8.17, R1.10.31, R1.10.32 or R1.12.42*.*

R1.12.29
Whenever the disposal schedule for a record is first applied or subsequently replaced the MCRS must automatically generate a timestamp and update it in the metadata of the record denoting when disposal was last scheduled for that record.

See formatting requirements for timestamps under R1.11.8. The disposal schedule of an active record can be replaced under R1.8.17, R1.10.31, R1.10.32 or R1.12.42.

R1.12.30
The MCRS must raise an alert if a record is given a disposal schedule with a retention period start trigger of DATE OF LAST REVIEW and that record does not have a reviewed timestamp has been included in its metadata under R1.12.44.

Disposal schedules based on the reviewed date should only be used for disposal schedules applied to records during or after a review.

R1.12.31
The MCRS must raise an alert if a record is given a disposal schedule with a retention period start trigger of METADATA ELEMENT DATE and that record does not have the element definition specified by the retention period trigger element included in its metadata or the metadata of its parent aggregation.

The retention period trigger element definition is specified in R1.12.6. The same element definition must be included in the metadata of the record or its parent aggregation but it does not need to contain a value yet at the time the disposal schedule is applied to the record. Elements are applied to records and aggregations by using templates, see R1.11.58, R1.11.59 and R1.11.60*.*

R1.12.32
Either whenever their metadata changes or periodically and at least daily the MCRS must check and update the retention and disposal progress of all active records, except those with disposal schedules that specify a retention period start trigger of NEVER under R1.12.4.

See also non-functional requirement N1.18.1. The MCRS must always have checked and updated the progress of a record within 24 hours of any change to its circumstances. Inactive records have already been destroyed under R1.12.50 and need not be checked. Likewise once the check has ascertained that a record has a disposal schedule specifying permanent retention, no further action is required (the check is still necessary in case this disposal schedule has been changed).

R1.12.33

Whenever an MCRS checks a record under R1.12.32 then it must compare the retention period start trigger in the record's disposal schedule to the relevant element in the record's or its parent aggregation's metadata and, if possible, calculate and set a new retention date.

The following retention period start triggers apply:

- *NOW relates to the disposal scheduled timestamp, see R1.12.29;*
- *DATE OF LAST REVIEW relates to the reviewed timestamp, see R1.12.44;*
- *DATE RECORD CREATED relates to the created timestamp, see R1.5.20;*
- *DATE RECORD INCLUDED IN AGGREGATION relates to aggregated timestamp, see R1.9.20;*
- *DATE LAST RECORD INCLUDED IN AGGREGATION relates to the last aggregated timestamp for the record's parent aggregation, see R1.9.21;*
- *DATE AGGREGATION CREATED relates to the record's parent aggregation's created timestamp, see R1.5.20;*
- *DATE AGGREGATION CLOSED relates to the record's parent aggregation's closed timestamp, see R1.9.37; and*
- *METADATA ELEMENT DATE relates to the element definition for the record identified by the retention trigger element, see R1.12.31.*

The MCRS will not be able to set a retention date if it has a retention period start trigger of DATE AGGREGATION CLOSED and the record's parent aggregation has not yet been closed, or it has a retention period start trigger of METADATA ELEMENT DATE and the element referred to by the retention period trigger element has not been given a value yet.

Otherwise the MCRS must calculate a retention date based on the retention period start trigger and element under R1.12.4 and R1.12.6, plus the retention period interval and duration under R1.12.8 and R1.12.9, plus the retention period offset and month under R1.12.11 and R1.12.13.

R1.12.34

If following R1.12.33 the MCRS determines that the retention date has not yet elapsed, then the MCRS must ensure that the record has none of the following metadata elements, even if these have been previously set:

- Disposal action, see R1.12.35;
- Disposal action due date, see R1.12.35; and
- Disposal action reminder sent timestamp, see R1.12.38.

These elements may have been set under R1.12.35 and R1.12.38 if the retention date was previously calculated for an earlier day which elapsed previously.

R1.12.35
If following R1.12.33 the MCRS determines that the retention date has elapsed, then the MCRS must include in the metadata of the record a disposal action and a disposal action due date.

The disposal action is copied from the disposal action of the record's disposal schedule, see R1.12.13. The disposal action due date is calculated from the retention date plus the disposal interval period and duration, see R1.12.14 and R1.12.15.

R1.12.36
Whenever disposal schedule for a record specifies a disposal action of DESTROY WITHOUT CONFIRMATION under R1.12.13, but the record has one or more components which are flagged as requiring confirmation before deletion under R1.10.15, then the MCRS must set the disposal action under R1.12.35 to DESTROY AFTER CONFIRMATION instead.

Some components of some records must be separately deleted by an administrator or external system, especially if they are not under the direct control of the MCRS. If the record contains one of these components then the MCRS must automatically adjust the disposal action when it is set.

R1.12.37
Whenever the disposal action for an active record is set to DESTROY WITHOUT CONFIRMATION under R1.12.35, and remains so after checking R1.12.36, the MCRS must immediately destroy the record under R1.12.50 unless it is the subject of a disposal hold under R1.13.8.

This is the only time when records are deleted without any administrator interaction. The action cannot be performed if the records are the subject of a disposal hold in which case the MCRS must keep trying periodically under R1.12.32 until the disposal hold is made inactive under R1.13.9 and the records can be destroyed.

R1.12.38
Whenever an MCRS checks a record under R1.12.32 for which the disposal action due date has elapsed, and which is not subject to a disposal hold under R1.13.8, and which does not already have a disposal action reminder sent timestamp, then it must raise an alert and include a disposal action reminder sent timestamp in the metadata of the record.

At this point the record is overdue for disposal so an alert is raised.

R1.12.39
When raising an alert under R1.12.38 the MCRS must consolidate the alerts for all overdue records belonging to the same parent aggregation so that only one alert is raised for each aggregation.

MoReq2010 does not specify how this is done and the technology used may vary depending on how the MCRS implements R1.12.32.

R1.12.40
The MCRS must allow an administrator to search for all active records due for disposal and browse them by disposal action, aggregation, classification and disposal action due date.

Records due for disposal are those with a disposal action set and a disposal action due date set under R1.12.35.

R1.12.41
When an administrator conducts a search under R1.12.40 the MCRS must not include in the results any active records that are subject to a disposal hold under R1.13.8.

If the records are subject to a disposal hold then their disposal action cannot be carried out.

R1.12.42
The MCRS must allow an administrator to review an active record with a disposal action set to REVIEW by directly applying a disposal schedule to the record to replace its current disposal schedule, unless the record is under a disposal hold, see R1.13.8.

When records have a disposal schedule that specifies a disposal action of REVIEW, the disposal schedule must be replaced once the retention period has expired with a disposal schedule that specifies a different retention period or a different disposal action. If the record is the subject of a disposal hold then it cannot even be reviewed.

R1.12.43
When an administrator is applying a disposal schedule to a record during a review under R1.12.42, the MCRS must allow the administrator to retain the record's existing disposal schedule if its retention period start trigger is DATE OF LAST REVIEW under R1.12.4.

A disposal schedule of "retain for x from date of last review" will result in a different retention date if replied to a record following a review.

R1.12.44
When an administrator completes a review by applying a disposal schedule to a record under R1.12.42 the MCRS must generate a timestamp indicating the last reviewed date and time and include it in the metadata for the record.

The last reviewed timestamp will replace the previous last reviewed timestamp, if any.

R1.12.45
When an administrator completes a review by applying a disposal schedule to a record under R1.12.42 the MCRS must allow the administrator to add a textual last review comment to the metadata of the record to describe the review findings.

If the administrator does not add a last review comment then any previous last review comment must be removed from the metadata of the record or it will be misleading. The previous last review comment will be retained in the entity's event history.

R1.12.46
When an administrator completes a review under R1.12.42 and applies a disposal schedule to a record then the MCRS must implement R1.12.28 even if the same disposal schedule has been applied again under R1.12.43.

The record is then ready for the next scheduled disposal.

R1.12.47
The MCRS must allow an administrator to confirm for any active record with a disposal action of TRANSFER or DESTROY WITH CONFIRMATION that the specified disposal action has been carried out, unless the record is under a disposal hold, see R1.13.8.

Records being transferred require confirmation from an administrator to ensure that the transfer has taken place. MoReq2010 does not specify how the transfer is made; it may partially involve exporting the records under 1.15.2.

Similarly, unless they are subject to R1.12.36, records require confirmation from an administrator before they are destroyed. This confirmation can mean one of two different things. For records with components under the management of the MCRS it can be an instruction to go ahead and destroy the records. For records with components not under the management of the MCRS it can be confirmation that the components have been "manually" destroyed so that the MCRS can now destroy the records in parallel.

If the records are subject to a disposal hold then the MCRS must ensure that an administrator cannot confirm the transfer or destruction of the records until the disposal hold is made inactive under R1.13.9.

R1.12.48
Whenever an administrator confirms a transfer or destroy action for an active record under R1.12.47 then the MCRS must allow the administrator to add a textual disposal comment to the metadata of the record to give more information about the confirmation decision and the details of any transfer, etc.

R1.12.49
Whenever an administrator confirms a transfer or destroy action for an active record under R1.12.47 then the MCRS must immediately destroy the record under R1.12.50 unless it is the subject of a disposal hold under R1.13.8.

R1.12.50
Whenever a record is destroyed its status must be set to inactive and the MCRS must perform the following operations:

- Delete from the metadata of the record all elements whose element definitions indicate that they should be deleted on destruction, see R1.11.28;
- Delete from the event history of the record all entries whose function definitions indicate that they should be deleted on destruction, see R1.6.32.

In addition, if the record does not share its components with active clones under R1.10.37 then the MCRS must:

- Set the status of each component of the record to inactive;
- Delete the datafile belonging to each component of the record if they are under the control of the MCRS, see R1.10.13;
- Delete from the metadata of each component of the record all elements whose element definitions indicate that they should be deleted on destruction, see R1.11.28; and
- Delete from the event history of each component of the record all entries whose function definitions indicate that they should be deleted on destruction, see R1.6.32.

When a record or a component enter the inactive state they cannot be reactivated. The MCRS must ensure that deletions of component datafiles, metadata elements and event history events are permanent and irrecoverable.

R1.12.51

Whenever a record is destroyed under R1.12.50 and there are no remaining active records in the record's parent aggregation, which is closed under R1.9.35 or R1.9.36, then the MCRS must destroy the aggregation under R1.12.52.

See also R1.12.53 and R1.12.38. An open aggregation may be closed later but closing it will trigger destruction under R1.12.52.

R1.12.52

Whenever an aggregation is destroyed its status must be set to inactive and the MCRS must perform the following operations:

- Delete from the metadata of the aggregation all elements whose element definitions indicate that they should be deleted on destruction, see R1.11.28; and
- Delete from the event history of the aggregation all entries whose function definitions indicate that they should be deleted on destruction, see R1.6.32.

When an aggregation enters the inactive state it cannot be reactivated. The MCRS must ensure that deletions of metadata elements and event history events are permanent and irrecoverable.

R1.12.53

Whenever an aggregation belonging to a parent aggregation is destroyed under R1.12.50 and there are no remaining active aggregations in the record's parent

aggregation, which is closed under R1.9.35 or R1.9.36, then the MCRS must destroy the aggregation under R1.12.52.

See also R1.12.51 any number of levels of aggregation can be destroyed by upwards cascade. An open aggregation may be closed later but closing it will trigger destruction under R1.12.52.

R1.12.54
Whenever a record or an aggregation has been destroyed under R1.12.50 and R1.12.52 respectively the MCRS must ensure that the entity's primary classification is fixed and is not subsequently changed.

The entity retains the primary classification under which it was destroyed in perpetuity for historical and contextual reasons. Other classifications relating to the now inactive entity may be changed for future management of the remaining stub. Note that the primary classification may also be made inactive under R1.8.22 but cannot be deleted.

R1.12.55
Whenever a record has been destroyed under R1.12.50 the MCRS must ensure that the record's disposal schedule is fixed and is not subsequently changed.

The record retains the disposal schedule under which it was destroyed. To change it would be misleading. Note that the disposal schedule may also be made inactive under R1.12.22 but cannot be deleted.

R1.12.56
The MCRS must allow an administrator to select a number of records from the search in R1.12.40 and perform any of the following functions as a bulk operation:

- Review the records by applying the same disposal schedule to all selected records, see R1.12.42;
- Apply the same last review comment to all selected records while completing the review, see R1.12.45;
- Confirm either a transfer action or a disposal action on all selected records, see R1.12.47; and
- Add the same disposal comment to all selected records while confirming their disposal, see R1.12.48.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R1.12.57
The MCRS must allow an administrator to add new disposal schedules to the disposal schedule inventory, in bulk, by reading from a datafile that describes their metadata.

The purpose of this requirement is to allow the disposal schedule inventory to be easily populated while the MCRS is being configured, and before it is used for the management of

records. The datafile format to be used by the MCRS to populate the disposal schedule inventory is not specified. Disposal schedules may be loaded from the same datafile that is used to load classes into a classification scheme under R1.8.30 and particularly R1.8.34, or element definitions and templates into the schedule inventory under R1.11.52.

This requirement is not the same as importing disposal schedules (bulk import is not part of the MoReq2010 core requirements). In each case the entities described by the datafile are being newly created in the MCRS. It is therefore not appropriate to load event histories for these entities from the datafile.

R1.12.58

When new disposal schedules are created in bulk from a datafile, under R1.12.57, then the MCRS must not read the system identifier for each entity from the datafile but must automatically assign a new system identifier to each new entity.

See R1.5.20. This requirement is intended to avoid the possibility that a given installation of an MCRS may be managing disposal schedules with the same system identifiers as similar entities in another MCRS. Such a situation would result in entities that were not universally unique.

R1.12.59

When new disposal schedules are added in bulk, in accordance with R1.12.57, then the MCRS must ensure that the metadata read from the datafile for each entity is valid and includes all mandatory metadata elements required by MoReq2010.

The MCRS must not create any disposal schedules with missing or invalid metadata.

1.13 Functional requirements for disposal holds

R1.13.1
The MCRS must maintain an inventory of disposal holds, and allow users to browse it.

The terms "inventory" and "browse" are defined and further explained in the rationales to R1.5.14 and R1.5.30.

R1.13.2
The MCRS must allow a user to inspect the metadata of any disposal hold in the disposal hold inventory.

The term "inspect" is defined in the rationale to R1.5.10.

R1.13.3
The MCRS must allow an administrator to create a new disposal hold and add it to the disposal hold inventory.

The disposal hold inventory should support any number of disposal holds.

R1.13.4
The MCRS must allow an administrator to change the title and description of a disposal hold.

R1.13.5
The MCRS must allow an administrator to add aggregations and records to a disposal hold.

There should be no limit to the number of aggregations and records that can be added to a disposal hold.

R1.13.6
The MCRS must allow an administrator to remove an aggregation or a record from a disposal hold.

When an aggregation or a record is removed from a disposal hold then any pending destruction actions against them or any of the contents of the aggregation must immediately be carried out by the MCRS, see R1.12.37, R1.12.42 and R1.12.50.

R1.13.7
The MCRS must include a textual comment in the metadata of a disposal hold and must allow an administrator to modify this comment as part of creating the disposal hold under R1.13.3, lifting the disposal hold under R1.13.9, reasserting the disposal hold under R1.13.11 and at any other time.

The comment allows a user to specify additional information about the disposal hold that is not included in the title or description.

R1.13.8
The MCRS must prevent any aggregation, or any of the aggregations and records it contains, or any single record that has been added to an active disposal hold from being reviewed, transferred or destroyed for as long as that disposal hold remains active.

When an aggregation is added to a disposal hold then this will freeze any disposal action on its children and their children, and so on.

R1.13.9
The MCRS must allow an administrator to change the status of a disposal hold from active to inactive, to allow the destruction of the aggregations and records it specifies.

When a disposal hold is made inactive then any pending destruction actions against any of the aggregations or records subject to the disposal hold including any of the contents of any of the aggregations will be carried out by the MCRS, see R1.12.37, R1.12.42 and R1.12.50.

R1.13.10
When the status of a disposal hold is changed from active to inactive under R1.13.9 then the MCRS must include the administrator that lifted the disposal hold and a timestamp as part of the metadata of the disposal hold.

The link type definition for each link must be specified when the link is first created and cannot be changed after that.

R1.13.11
The MCRS must allow an administrator to change the status of an inactive disposal hold to active, to allow it to prevent the destruction of aggregations and records.

Note that changing a disposal hold from active to inactive will not recover any aggregations or records that were destroyed when the disposal hold was made inactive under R1.13.9.

R1.13.12
When an administrator makes a disposal hold active under R1.13.11 then the administrator that lifted the disposal hold and the timestamp added to the metadata under R1.13.10 should be cleared.

This information will still be retained in the event history for the disposal hold.

1.14 Functional requirements for searching and reporting

R1.14.1
A user must be able to search for entities in the MCRS.

All search results will contain a list of entities.

R1.14.2
When searching for entities under R1.14.1, or reporting under R1.14.39 and R1.14.44, the user must be able to specify at least 20 search criteria for each search.

Search criteria restrict the set of possible search results. In the search query "find entities where created timestamp is today", the phrase "created timestamp is today" is an example of a search criterion.

R1.14.3
When specifying criteria under R1.14.2 each search criterion must refer to either a metadata element or the content of a component.

In the example given in the rationale to R1.14.2 the term "created timestamp" refers to a metadata element.

R1.14.4
When specifying criteria under R1.14.2 for use in searching the user must be able to combine the search criteria together logically using any combination of AND, OR or NOT.

These operators come from Boolean algebra. AND refers to the intersection of two sets, OR to the union of two sets and NOT to the complement of a set. Note that the MCRS does not need to explicitly use the terms "AND", "OR" and "NOT" but may use any logically equivalent means of expressing the same operators.

R1.14.5
When combining search criteria under R1.14.4 the user must be able to use parentheses or their equivalent to change the order of operation.

Under the conventions of Boolean logic, NOT has the highest precedence, followed by AND, with OR having the lowest precedence. Parentheses can be used to change the precedence of operators (operations inside parentheses are always performed first). Parentheses should be able to be wrapped around other parenthesised operations to any depth.

R1.14.6
The MCRS must be able to perform full text searching of all textual metadata elements and any components identified under R1.10.11.

Full text searching involves searching for whole words within the search text. When indexing text for full text searching white space and punctuation characters are generally ignored. See also non-functional requirements N1.17.9, N1.17.10 and N1.17.11.

R1.14.7
When performing full text searching under R1.14.6 the MCRS must be able to calculate a relevancy score for each entity found.

The relevancy score should help to identify those entities which better fit the search text and are therefore more likely to be the entity that the user was searching for. MoReq2010 does not specify the algorithm to be used for calculating the relevancy score.

R1.14.8
When constructing search criteria under R1.14.3, the user must be able to find components whose content contains some target search text.

MoReq2010 uses the term "contains" in this context to mean full text searching under R1.14.6. When using full text searching the user must also provide some target text to search for. In the example, "find components whose content contains 'hospital'" the target search text is the word "hospital".

R1.14.9
When constructing search criteria for metadata elements under R1.14.3, the user must be able to define search criteria with the following conditions:

- Whether the metadata element exists (has been added to the entity), see R1.11.60; and
- Whether the metadata element has a value.

These search criteria apply to all metadata elements regardless of the element type. Users must be able to search for entities which either have or do not have a particular metadata element added to them, regardless of the value of the element. Users must also be able to search for whether the element associated with an entity has been given a value or remains empty.

R1.14.10
When constructing search criteria for textual metadata elements under R1.14.2, the user must be able to define search criteria for at least the following conditions:

- Equivalence – where the value of the metadata element exactly matches the search text; and
- Contains – where the content of the textual metadata element contains some target search text.

The term "contains" is explained in the rationale of R1.14.8 and refers to full text searching. Equivalence refers to an exact match on the whole value, rather than a part of the text.

R1.14.11
When constructing search criteria for date, date/time and timestamp metadata elements under R1.14.2, the user must be able to define search criteria for at least the following conditions:

- Values occurring before a particular date, date/time or timestamp;
- Values occurring after a particular date, date/time or timestamp;
- Values occurring on a particular date;
- Values occurring today;
- Values occurring yesterday;
- Values occurring tomorrow;
- Values occurring this week;
- Values occurring last week;
- Values occurring next week;
- Values occurring this month;
- Values occurring last month;
- Values occurring next month;
- Values occurring this quarter;
- Values occurring last quarter;
- Values occurring next quarter;
- Values occurring this year;
- Values occurring last year; and
- Values occurring next year.

Being able to define a time range in a relative sense, such as "this month" is particularly important for saved searches and reporting, see R1.14.34 and R1.14.49.

R1.14.12
When searching using timestamp based criteria under R1.14.11 the MCRS must be able to factor in the user's local time zone to accurately find data that falls into the specified period.

See R1.11.7.

R1.14.13
When constructing search criteria for numeric metadata elements under R1.14.2, the user must be able to define search criteria for at least the following conditions:

- Equals;
- Greater than; and
- Less than.

R1.14.14
When constructing search criteria for Boolean metadata elements under R1.14.2, the user must be able to define search criteria for at least the following conditions:

- True – the flag has been set; and
- False – the flag has been cleared.

Boolean metadata elements are sometimes called flags under MoReq2010.

R1.14.15
When constructing search criteria for metadata elements that hold references to entities, under R1.11.15 and R1.11.16, the user must be able to define search criteria for at least the following conditions:

- Is – where the metadata element value matches a particular entity nominated by the user; and
- Is in – where the metadata element value matches a particular entity that is included in the search results of another search.

Inclusion is the equivalent of a "join" in search terminology where a search specifies in one of its search criteria an element that's value matches an entity returned by another search. For example, the query "Find aggregations where the creator user is in 'find users whose status is inactive'" would find all aggregations in the MCRS that have been created by users who are now inactive.

Joining or chaining searches in this way, using inclusion, can be resource intensive, see non-functional requirement N1.17.16.

R1.14.16
The MCRS must allow a user performing a search under R1.14.1, or generating a report under R1.14.39 or R1.14.44, to search for entities of a particular entity type or a combination of entity types.

For example the user must be able to search for "entities", or just "classes", or for "users and groups", etc.

R1.14.17
The MCRS must allow a user performing a search under R1.14.1, or generating a report under R1.14.39 or R1.14.44, to search for aggregations, records and components that are descendants of any aggregation.

Descendants do not have to be the immediate children of an aggregation.

R1.14.18
The MCRS must be able to express in natural language the detail of any search criteria under R1.14.2, in the default language of the MCRS.

The default language of the MCRS is specified in R1.5.2. The detail is constructed from the search criteria and simply refers to a natural language description for inclusion in the event history, see R1.14.32, R1.14.33 and R1.14.48, and report headings, see R1.14.42.

Examples of the level of detail required have been given in the rationale to R1.14.2, R1.14.8 and R1.14.15.

R1.14.19
By default, the MCRS must return only active entities in the results of searches conducted under R1.14.1, and reports generated under R1.14.39 and R1.14.44, unless the user performing the search has requested the inclusion in the search results of inactive entities as well.

MoReq2010 does not specify how the user requests the inclusion of inactive entities nor whether the system wide default can be changed for individual users or roles.

R1.14.20
The MCRS must return only entities in the results of searches conducted under R1.14.1, and reports generated under R1.14.39 and R1.14.44, whose metadata the user performing the search is able to inspect.

The term "inspect" is defined in the rationale to R1.5.10. Users must not be able to search and find entities which they cannot inspect in this way.

R1.14.21
The MCRS must not prevent a user who has initiated a search under R1.14.1 from accessing the MCRS, and performing functions while the search is in progress.

The MCRS must not freeze the user's access to the system while it carries out a search.

R1.14.22
The MCRS must allow the user who initiates a search under R1.14.21 to cancel it at any time before completion.

A user may choose to cancel a search for a number of reasons. For example, the user may change his or her mind, or the search may be taking too long.

R1.14.23
The MCRS must allow a user who is conducting a search under R1.14.1, or a detail report under R1.14.39, to specify which metadata element values to include for each entity returned in the search results.

The search results may include, for example, the title and description of each entity and its created timestamp.

R1.14.24
The MCRS must allow the user specifying which metadata elements to include for each entity returned in the search results under R1.14.23, to also specify which metadata elements to include for any related entities.

The search results may include, for example, the title and date closed of the parent aggregation for each entity.

R1.14.25
The MCRS must allow a user who is conducting a search under R1.14.1, or a detail report under R1.14.39, that includes a full text search to include the relevancy score calculated under R1.14.7, for each entity returned in the search results.

R1.14.26
The MCRS must order the search results returned from a search initiated under R1.14.1, or a detail report under R1.14.39, by their relevancy, see R1.14.25, or any combination of returned metadata element values, see R1.14.23 and R1.14.24, as requested by the user.

The user must be able to specify the order by which results are returned. This can include a lexicographical ordering based on the default language of the MCRS, under R1.5.2.

R1.14.27
When the user requests that search results be ordered by metadata element values under R1.14.26, the MCRS must allow the user to order the results by more than one element and to specify either ascending or descending values.

For example, the user may specify that search results are to be ordered first by parent aggregation in ascending order of title and within parent aggregation by sequence number (see R1.9.22) in descending order.

Note that this requirement only applies to ordering by metadata elements, the order of relevancy scores, when requested under R1.14.25, should always be from most relevant to least relevant.

R1.14.28
When ordering search results by timestamp criteria under R1.14.26 the MCRS must be able to present results in accurate order despite different time zone considerations.

See R1.11.7 and R1.14.12.

R1.14.29
The MCRS must paginate search results so that the user receives a sub-set of the first search results in the order previously specified under R1.14.26, and may then request additional sub-sets, in order, until the user has received all the matching search results.

MoReq2010 does not specify the mechanism to be used for pagination. Page sizes of between 10 to 50 results per page are typical for many common search applications, and the MCRS may even make the number of results per page configurable by the user (this is not required). Pagination of search results implies that the interface to the MCRS provides methods for requesting at least the next page of search results and possibly for requesting other pages out of sequence, such as the previous page, first page, last page and a page at random. MoReq2010 does not specify the type or number of pagination controls required by the interface.

R1.14.30
The MCRS must include in the results of searches performed under R1.14.1 a reference to every entity returned that allows the user to inspect its metadata or perform any function on it.

The term "inspect" is defined in the rationale to R1.5.10. See also R1.14.20.

R1.14.31
The MCRS must allow a user to select a set of entities from the results of a search so as to perform, for example, a bulk operation.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43.

R1.14.32
Whenever a user performs a search under R1.14.1, the MCRS must generate an event and include it in the event history of the user's user entity, unless the search is cancelled under R1.14.22 before any results have been returned to the user.

Note also that the MCRS will not generate the event if the search function definition has been modified under R1.6.22.

R1.14.33
When generating an event under R1.14.32, the MCRS must include search detail, defined under R1.14.18, as metadata in the event.

In other words the event will contain a description of the search performed by the user in natural language.

R1.14.34
The MCRS must allow users to save searches so that they may be performed again.

MoReq2010 does not specify how searches are to be saved and saved searches are not themselves considered entities that can be exported to another MCRS. The search engine technology used may therefore be proprietary and may differ between MCRS products and implementations.

R1.14.35
The MCRS must allow users to share saved searches with other users.

Saved searches may be useful only to a particular user or more generally within an organisation.

R1.14.36
The MCRS must allow users to access a saved search and perform it, with or without first modifying it.

A user may access a saved search modify it and then save it as another saved search under R1.14.34.

R1.14.37
The MCRS must allow users to delete saved searches.

As noted in the rationale to R1.14.34, MoReq2010 does not consider saved searches to be entities that provide greater context to a record over time. For this reason they cannot be manipulated or exported to other systems and MoReq2010 does not specify the metadata of a saved search.

R1.14.38
The MCRS must allow an administrator to add saved searches to the MCRS, in bulk, by reading them from a datafile.

The purpose of this requirement is to allow the MCRS to be populated with commonly used saved searches when it is first configured. The datafile format can be proprietary.

R1.14.39
The MCRS must allow a user to generate a detail report for any set of search criteria specified under R1.14.2 or from a saved search, see R1.14.34.

Generating a detail report is the same as specifying a set of search results and all requirements R1.14.2 through R1.14.20 apply equally to the generation of a detail report. A user may use a search that has been performed or a saved search as the basis for specifying a detail report.

R1.14.40
The MCRS must allow a user generating a detail report under R1.14.39, or a summary report under R1.14.44, to specify a format for the report.

Suppliers must provide a list of the reporting formats they support when their product is certified. Common reporting formats include:

- *Comma or tab separated values;*
- *Spreadsheet formats, such as OOXML and ODF;*
- *XML and HTML based formats; and*
- *PDF or other document formats.*

R1.14.41
The MCRS must allow a user generating a detail report under R1.14.39, or a summary report under R1.14.44, to give the report a heading to appear on the first and every subsequent page of the report.

In addition to a report heading on each page the MCRS may add additional information such as a cover page, footnotes, etc.

R1.14.42
The MCRS must allow a user to include any of the following in the report heading of a detail or summary report under R1.14.41:

- System information, such as the title and description of the MCRS, the default language and the organisation name and address, see R1.5.12.
- Date or date/time the report is generated;
- Page numbering;
- Metadata elements from the user entity of the user generating the report, such as the user's title; and
- Search detail, defined under R1.14.18, for the search against which the report has been generated.

In the case of a summary report, defined under R1.14.44, based on more than one search, there will be several search detail descriptions each of which relates to one of the totals in the summary report.

R1.14.43
The MCRS must generate column labels for the body of each page of a detail report generated under R.11.39 to reflect the metadata elements included in the report under R1.14.23, as appropriate.

Requirements R1.14.23 through R1.14.28 for searches are equally applicable to which metadata elements may be included in the body of a detail report and how the results are ordered.

R1.14.44
The MCRS must allow a user to generate a summary report containing a number of sets of search criteria under R1.14.2 but including the number of entities in the MCRS that meet the search criteria rather than listing them.

A summary report gives the totals for the number of entities that meet each listed set of search criteria. Unlike a detail report that lists all the entities that meet a particular set of search criteria under R1.14.39, a summary report is based on more than one search but contains only the total number of entities in each search category.

R1.14.45
The MCRS must not prevent a user who has generated a report under R1.14.39 or R1.14.44 from accessing the MCRS, and performing functions while the report is being generated.

The MCRS must not freeze the user's access to the system while it generates a report.

R1.14.46
The MCRS must allow the user who initiates a report under R1.14.39 or R1.14.44 to cancel it at any time before it is generated.

Cancelling a report while it is being generated is not the same as cancelling it at the printer while it is being printed.

R1.14.47
Whenever a user generates a report under R1.14.39 or R1.14.44, the MCRS must create an event and include it in the event history of the user's user entity, unless the report is cancelled under R1.14.46 before it has been generated.

Note also that the MCRS will not generate the event if the detail report function definition or the summary report function definition has been modified under R1.6.22.

R1.14.48
When generating an event under R1.14.47, the MCRS must include search detail, defined under R1.14.18, as metadata in the event.

In the case of a summary report the event will have more than one search detail element included in its metadata.

R1.14.49
The MCRS must allow users to save report definitions so that they may be performed again.

MoReq2010 does not specify how report definitions are to be saved and they are not considered entities that can be exported to another MCRS.

R1.14.50
The MCRS must allow users to share report definitions with other users.

Report definitions may be useful only to a particular user or more generally within an organisation.

R1.14.51
The MCRS must allow users to access a report definition and generate a report, with or without first modifying it.

A user may access a report definition modify it and then save it as another saved report definition under R1.14.49.

R1.14.52
The MCRS must allow users to delete saved report definitions.

MoReq2010 does not specify the metadata of a saved report definition or require that it must be preserved over time to provide context to a record.

R1.14.53
The MCRS must allow an administrator to add report definitions to the MCRS, in bulk, by reading them from a datafile.

The purpose of this requirement is to allow the MCRS to be populated with commonly used saved searches when it is first configured. The datafile format can be proprietary.

Report definitions can be loaded in bulk from the same datafile as saved searches under R1.14.38.

1.15 Functional requirements for exporting

R1.15.1

The MCRS must support the export of entities, their metadata and their components, from the MCRS by an administrator to an XML datafile.

THIS REQUIREMENT DEFINITION IS A PLACEHOLDER

MoReq2010 will eventually replace this requirement with one that specifies the export of entities using the W3C EXI (Efficient XML Interchange) standard rather than to an XML datafile. EXI is not currently a recommendation of the W3C but has been a candidate recommendation since December 2009.

Future inclusion of EXI in MoReq2010 will ensure that all export from the MCRS is sent to an EXI data stream rather than being written to datafile storage. This will resolve current technical issues with export such as how to provide a standardised mechanism for breaking large exports across multiple datafiles and what compression technologies an MCRS must support.

It may also provide the future basis for live stream based transfer of records between different MCRS solutions.

R1.15.2

The MCRS must allow an administrator to select any of the following for export from the MCRS under R1.15.1:

- Any entity;
- Any selection of entities;
- One or more active records with a disposal action of TRANSFER selected from the records due for transfer under R1.12.47;
- The classification scheme and all its classes;
- All the entities in any inventory; or
- All the entities in the MCRS.

The term "select" is explained in the rationale to R1.5.43.

R1.15.3

When preparing to export under R1.15.1, the MCRS must prepare a list of entities to export in full and a list of entities to export as export headers according to the following rules which must be applied in order:

- Rule 1: any entity included in the selection under R1.15.2 must be added to the list of entities to export in full;
- Rule 2: any aggregation or record that is included in a disposal hold listed under Rule 1 must be added to the list of entities to export in full;

- Rule 3: any aggregation or record that is a descendent of an aggregation listed under Rule 1 or Rule 2 must be added to the list of entities to export in full;
- Rule 4: any component of any record that is listed under Rules 1, 2 and 3 must be added to the list of entities to export in full;
- Rule 5: any user entities that are members of groups listed under Rule 1 must be added to the list of entities to export in full;
- Rule 6: the user entity of the administrator who is performing the export must be added to the list of entities to export in full;
- Rule 7: any events that are part of the event histories of entities listed under Rules 1 through 6 must be added to the list of entities to export in full;
- Rule 8: any entities that are included in the metadata of any entities, including events, that are listed under Rules 1 through 7, must be added to the list of entities to export in full;
- Rule 9: any events in the event histories of entities that are included in the metadata of any entities that are listed under Rule 8 must be added to the list of entities to export in full;
- Rule 10: any entities included in the metadata of any entities, including events, that are listed under Rules 1 through 9, that are not themselves listed as entities to export in full must be added to the list of entities to export as export headers;
- Rule 11: the entity type definition of every entity listed in either list under Rules 1 through 10 must be added to the list of entities to export in full;
- Rule 12: the function definition of every event listed under Rule 7 and Rule 9 must be added to the list of entities to export in full; and
- Rule 13: the element definition of every element belonging to every entity listed in either list under Rules 1 through 12 must be added to the list of entities to export in full.
- Rule 14: any entity in the list of entities to export in full that the administrator performing the export is not permitted to inspect must be transferred to the list of entities to export as export headers.

Entities that are included in the metadata of another entity under Rule 7 and Rule 9 include, but are not limited to:

- *The user that creates an entity, see R1.5.21;*
- *The parent aggregation of an aggregation or record, see R1.9.4 and R1.10.1;*
- *The primary and other classifications of an aggregation or record, see R1.9.5 and R1.10.5;*
- *The disposal holds applied to an aggregation or record, see R1.13.5;*
- *The disposal schedule of a record, see R1.10.31 and R1.10.32;*
- *The classification scheme of a class, see R1.8.6;*
- *The templates associated with a class, see R1.8.12 and R1.8.14;*
- *The groups of which a user entity is a member, see R1.7.20; and*
- *The entities referred to by custom metadata elements, see R1.11.15.*

The term "inspect" is defined in the rationale to R1.5.10. Users must not be able to export entities in full which they cannot normally inspect.

R1.15.4

When an administrator performs an export under R1.15.1 the MCRS must generate a system identifier for the export.

See format requirements for system identifiers in 1.11.1 and 1.11.2. Each separate export from an MCRS represents a snapshot in time of a part of the MCRS. Clearly and uniquely identifying each export as it is made allows another MCRS to import a series of exports over time and more easily and precisely piece the information back together.

R1.15.5

When an administrator performs an export under R1.15.1 the MCRS must allow the administrator to add a textual export comment to be included in the export under R1.15.6, and the export event under R1.15.9.

The export comment is an explanation of why the export was performed and what entities are included in the export.

R1.15.6

When the MCRS exports entities under R1.15.1 then the MCRS must export only the entities in the lists prepared under R1.15.3, in the form specified under R1.15.3 (either in full or as an export header), using the MoReq2010 XML schema.

The XML schema specifies the full export format for entities under MoReq2010. The following is a summary of the export process using this schema:

- *The MCRS starts transmission and opens the outer XML tag;*
- *The MCRS exports a starting timestamp;*
- *The MCRS exports the system identifier for the export from R1.15.4;*
- *The MCRS exports the export comment from R1.15.5;*
- *The MCRS exports the system identifier of the administrator making the export;*
- *The MCRS exports system information made up of identical metadata values to those specified under R1.5.12;*
- *For each entity listed under R1.15.3 the MCRS exports the entity as part of a block of entities of the same entity type together (in no particular order);*
- *If the entity is listed for export in full then the MCRS includes its metadata and event history with the entity;*
- *If the entity is listed for export as an export header then the MCRS exports only its system identifier and its entity type system identifier;*
- *For every component listed for export in full the MCRS exports the component's datafile as Base64 encoded XML, if it has access to the datafile under R1.10.13;*
- *The MCRS exports an "export successful" tag as specified in the XML schema;*
- *The MCRS exports a finishing timestamp; and*
- *The MCRS closes the outer XML tag and ends transmission.*

R1.15.7

The MCRS must allow the administrator who requested an export under R1.15.1 to cancel it at any time before completion.

See the rationale to R1.15.6. When an export is cancelled the MCRS must take the following steps:

- *Complete the export of the entity that is currently being exported;*
- *Close the current block of entities by entity type even if not all entities of that type have been exported;*
- *Export an "export cancelled" tag as specified in the XML schema;*
- *Export a finishing timestamp; and*
- *Close the outer XML tag and end transmission.*

R1.15.8

Whenever entities are exported by an administrator under R1.15.1, the MCRS must generate an event and include it in the event history of the administrator's user entity and in the event history of every entity that was exported in full under R1.15.3.

In other words, the administrator's user entity, and every entity exported in full, is considered to be a participating entity in the event under R1.6.28 and R1.6.29.

Note that the event must be applied to the event history of only those entities that were exported in full. If the export is cancelled under R1.15.7 then not all entities that were due to be exported in full will have been exported in full. The entities that were not exported should not receive the export event in their event histories.

If no entities were exported in full before the export was cancelled or failed then the administrator will be the only participating entity.

R1.15.9

When generating an export event under R1.15.8 the MCRS must include the following in the metadata of the event:

- The system identifier of the export from R1.15.4;
- The export comment from R1.15.5
- An export success flag based on whether the export was successfully completed under R1.15.6, or was cancelled under R1.15.7, or failed due to other technical transmission reasons.

The MCRS must generate an export event regardless of the outcome of the export.

1.16 Non-functional requirements for business continuity

N1.16.1
Does the MCRS support atomic transactions?

Atomic transactions are required to ensure the integrity of the data if the MCRS fails during an operation.

N1.16.2
How often does the organisation require the MCRS to be backed up?

For most businesses a daily backup is sufficient. In mission critical organisations backups may be required every hour or even more often. Such operations should consider incremental back up strategies and off site mirroring, see N1.16.3 and N1.16.4.

N1.16.3
Does the MCRS have the facility for back up or is data redundancy ensured by another means?

Such as mirroring all information to a stand by system or site.

N1.16.4
Does the MCRS support incremental back up and during data recovery is it able to roll forward to the transaction immediately before the failure?

Many high use and mission critical businesses will require that the minimum number of transactions are lost through failure of the MCRS.

N1.16.5
How long does it take to back up the MCRS?

This will depend on the size of the organisation's data holdings and the back up method and media. It is important for an organisation to understand before it purchases an MCRS how much data it will store and how long it will take to back up.

N1.16.6
Does backing up or other maintenance require the MCRS to be taken off line?

If not taken off line what is the impact on performance during the period it is being backed up?

N1.16.7
How long does it take to restore the data of the MCRS?

See the rationale to N1.16.5.

N1.16.8
How long will it take to rebuild the MCRS software and then restore the data?

See the rationale to N1.16.5 for estimate of data holdings, etc.

N1.16.9
Can the MCRS be run redundantly on a hot standby, cold standby and/or warm standby?

Many organisations prefer standby servers and sites instead of or in addition to traditional tape back ups.

N1.16.10
What system components can fail leaving the MCRS operational?

For example, is the MCRS built using service orientated architecture and are some parts of the MCRS redundant allowing some services to fail and be restarted while the MCRS remains operational.

N1.16.11
What types of back up and storage media does the MCRS support?

For example, magnetic discs, magnetic tapes, optical media, cloud storage, etc.

N1.16.12
How does the MCRS monitor its storage requirements and does it warn a system support technician when available storage space falls below a preconfigured minimum amount.

The term "system support technician" is defined in R1.6.2.

N1.16.13
Does the MCRS allow hardware to be maintained and replaced while it is operational?

For example, hot swapping drives.

N1.16.14
Does the MCRS facility have an uninterruptable power supply?

A UPS is usually necessary for N1.16.15.

N1.16.15
Can the MCRS shut itself down gracefully in the event of a power outage or in response to another external threat?

Gracefully means that all processes are shut down in the right order to preserve the integrity of the system and all it to restart and continue operation.

1.17 Non-functional requirements for performance and scalability

N1.17.1

Are there any technical limitations on the size to which the MCRS can grow?

Some systems are restricted by database size, file system segments, or other consideration.

N1.17.2

Are there any technical limitations on the capacity of the MCRS in terms of the number of entities it can hold or entities of a particular type?

There is no theoretical limit to how many entities, how much data and how much content an MCRS can hold under the MoReq2010 specification, are there any practical considerations however?

N1.17.3

What depth of aggregation structure can the MCRS support?

How many levels of aggregation?

N1.17.4

Are there any limits to any of the following:

- The number of entities that can be children of an aggregation?
- The number of entities that can be held by an inventory?
- The number of classes that can be included in a classification scheme?
- The number of components in a record?
- The number of metadata elements in a template?
- The number of metadata elements applied to an entity?

N1.17.5

Is there a limits to the number of entities that can be selected by a user simultaneously?

The term "select" is defined in the rationale to R1.5.43.

N1.17.6

What types of record does the organisation use?

How much metadata do they have, what size content?

N1.17.7

How many records does the organisation capture or declare per hour on average and in periods of peak load?

Can the MCRS meet this capacity?

N1.17.8
How many records are accessed, viewed or downloaded per hour on average and in periods of peak load?

Access may be through browsing or searching.

N1.17.9
How does the MCRS perform full text searching?

Does it use an OEM search engine and if so is this separately configurable?

N1.17.10
Can the MCRS perform any of the following language pattern based search techniques as part of its full text search capability:

- Word stemming?
- Homonym matching?
- Synonym finding?

These search techniques all require that the search engine have a knowledge of which language the searcher is using to search.

N1.17.11
Can the MCRS perform any of the following search techniques as part of its full text search capability:

- Concept searching?
- Phrase searching?
- Proximity searching?

N1.17.12
Can the MCRS perform searching using wild cards and pattern matching?

N1.17.13
When searching how long does it take the first page of search results to appear for:

- Simple searches with one criterion?
- Complex searches with several criteria?
- Searches with/without full text searching?

This will depend in part on the organisation's network and server facilities and contention from other systems.

N1.17.14
When searching how long does it take the second and subsequent pages of search results to appear after they have been requested?

N1.17.15
Is there a limit to the maximum number of search results that the MCRS will find and return?

For large numbers of search results the MCRS may approximate the total number of entities rather than calculate it.

N1.17.16
Is there a limit to the number of chains or joins the MCRS can include in a search under R1.14.15?

What is the impact on resource load and throughput when users employ this search strategy? Does this have an impact on N1.17.17?

N1.17.17
Does the MCRS have a search timeout interval and does the MCRS itself ever cancel searches because they are taking too long?

See also R1.14.22. What is the MCRS search timeout interval and can it be configured? If the MCRS search times out what notification is returned to the user?

N1.17.18
If the MCRS is able to retrieve datafiles under R1.10.23, how long does it take to retrieve a typical datafile?

See non-functional requirement N1.17.6, for the size of a typical datafile.

1.18 Non-functional requirements for operational compliance

N1.18.1

How often does the MCRS update the disposal progress of records under R1.12.32?

What is the maximum period before disposal progress is updated?

N1.18.2

Which authentication services does the MCRS support under R1.7.1?

The MCRS may support one or a number of commercial or proprietary authentication services, or the authentication service may be built in.

N1.18.3

Which directory services does the MCRS support under R1.7.3?

The MCRS may support one or a number of commercial or proprietary directory services, or the directory service may be built in.

N1.18.4

Which elements of the directory services supported under N1.18.3 can the MCRS include in a user template and synchronise with under R1.11.22?

The MCRS may support one or a number of commercial or proprietary directory services, or the directory service may be built in.

N1.18.5

Which component templates and elements does the MCRS support under R1.10.19, R1.11.11 and R1.11.46?

See the rationale to these requirements.

N1.18.6

How does the supplier denote the major and minor versions of the MCRS?

What may or may not be included in a major or minor version is important for upgrade and compatibility of the organisation's MCRS.

N1.18.7

What was the last major and minor version of the MCRS software and when were they released?

It is also useful to know the timeframe for release of the next minor and major versions.

N1.18.8

How often does the supplier develop and release new major versions?

Typically most suppliers will do this every two to four years.

N1.18.9
What is involved and what is the cost of upgrading the MCRS to a new version?

This may involve additional consultancy support.

N1.18.10
How long will the MCRS be off line for during an upgrade?

Typically this should not take longer than one day even for a planned outage to upgrade to a new major version.

N1.18.11
What is the supplier's policy for product support and issuing hot fixes or patches for critical errors?

This can affect the maintenance schedule with unplanned outages.

N1.18.12
Does the supplier have a 24 x 7 service desk for critical support issues?

If not what are the supplier's support hours?

N1.18.13
Does the supplier have an on line service desk application where the organisation can monitor the progress of all the issues it raises?

If not what provision does the supplier make for issue tracking?

N1.18.14
What are the supplier's service level agreements for critical and other priority issue turn around and bug fixing?

What statistics can the supplier provide to show that it is currently meeting its service and support targets?

N1.18.15
Is the MCRS an open source software application and if not has a copy of the source code of the MCRS been lodged in escrow with a neutral third-party?

It must be possible for the organisation to obtain access to the MCRS should the supplier ever go out of business.

1.19 Glossary of terms

Term	Explanation and relationship to general concepts
Administrator	(role) a role that allows access to all entities in the MCRS. *The administrator can perform any function.*
Aggregation	(verb) the act of grouping together records or aggregations of records. (entity) a grouping of records or of lower layer aggregations. *The lowest layer of aggregation is an aggregation of records. Higher layers of aggregation group together lower layer aggregations. Aggregation is hierarchical so that each aggregated entity has exactly one parent and relationships between different aggregations cannot be circular.* ISO 23081-2 More formally defined equivalent to the use of the term "aggregation" in ISO 23081-2.
Alert	(noun and verb) a mechanism by which the **MCRS** can collectively issue a warning to administrators. *Alerts can be implemented in different ways but do require a pro-active or "push" notification system rather than a passive or "pull" notification mechanism.*
Browse	(verb) a process whereby a user changes focus from one related entity to another without having to use the search facility to find the next entity. See R1.5.14.
Bulk Operation	(noun) a technique defined by MoReq2010 for performing the same function to more than one entity simultaneously with the same starting information. See R1.5.43.
Class	(entity) an entity in a classification scheme that can be associated with a record or an aggregation. *Classes are vital in providing the business context for records.* ISO 23081-2 Not related to the term "class" in ISO 23081-2, see Entity Type.

Term	Explanation and relationship to general concepts
Classification	(verb) the act of associating a class with a record or an aggregation.
Classification Scheme	(noun) a specific system of classification.
Clone	(noun and verb) a copy of a record in a different aggregation that shares its components with all the other clones of the same record. The act of creating the cloned record. *See R1.10.33.*
Component	(noun) a necessary and integral part of a record that has its own discrete storage requirements. *An example of a component is a digital datafile. Every record has at least one component. A record's component(s) may be electronic or physical.*
Datafile	(noun) A machine readable computer file containing data in any digital format. The term "datafile" has been coined specifically for use in MoReq2010. *The word "file" used in this context does not refer to an* ***aggregation****.*
Datatype	(noun) A data type definition that is included as part of a metadata element definition. *MoReq2010 requires that datatypes be defined in accordance with the "W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes" specification.*
Disposal action	(noun) the action that must be taken according to a **disposal schedule** when a record reaches the end of its retention period. *Either review and apply a new disposal schedule; transfer out of the MCRS; destroy without confirmation; or destroy with confirmation.*
Disposal hold	(entity) an entity designed to block the disposal process for records due to be disposed of by a **disposal schedule**. *Disposal holds are necessary in many countries for legal and other reasons.*

Term	Explanation and relationship to general concepts
Disposal schedule	(entity) a lifecycle plan for a record that specifies how long it will be retained and how it will be disposed of at the end of its retention period.
Element	(noun) an item of metadata belonging to an entity.
Element Definition	(entity) an entity that defines a metadata element. Element definitions are used to construct templates.
Entity	(noun) all data in the **MCRS** is split between entities.
Entity Type	(entity) a type definition for an entity.
Event	(entity) an occurrence in the lifecycle of any entity in the **MCRS** that has been remembered and recorded as an event entity. *Event histories are made up of events.*
Event History	(noun) a history made up of **events** of everything that has happened to an entity in the **MCRS**.
Function	(noun) a definition of an action that can be carried out either by the MCRS or at user instigation.
Group	(entity) a group of **users** managed by a directory service.
Inheritance	(concept) the concept of applying the same classification to a child entity as has already been applied to its parent aggregation.
Inspect	(verb) a technique defined by MoReq2010 to allow a user access to the metadata elements of an entity. See R1.5.10.
Inventory	(noun) a collection of all the entities of a particular entity type. See R1.5.14. *The inventory allows the entities to be **browsed**.*
Main Component	(element) The component of a record with more than one component which should be retrieved first by a user accessing the record. *See R1.10.18.*
MCRS	(abbreviation) see "MoReq Compliant Records System"

Term	Explanation and relationship to general concepts
MoReq Compliant Records System	(noun) a records system which is fully compliant with the MoReq2010 specification. *Within the functional requirements of any given module of MoReq2010, a "MoReq Compliant Records System" refers specifically to a records system compliant with the requirements of that module, any of its prerequisite modules, and the core requirements.*
Record	(noun) the smallest discrete unit managed as a single entity. *Records may be made up of several components but the whole record is managed atomically as a single entity.* ISO 23081-2 A record is the equivalent of an "item" in ISO 23081-2.
Records System	(noun) any system which manages records. *A records system may be separate from other systems, and operate independently, or it may be built into a larger business or information system and operate in an integrated way. Similarly, a records system may be general purpose, managing many different kinds of record or it may be a dedicated service for capturing and managing only one particular business output. MoReq2010 does not distinguish between records systems based on their level of integration into other business systems or the specific kinds of records they target.* ISO 15489-1 A records system is defined in ISO 15489-1 "Information systems, business applications and communication systems, and the business processes which they support, should be designed, modified or redesigned so that adequate records can be created and captured as a routine part of undertaking business activities." P.8
Retrieve	(verb) the method by which a user obtains and downloads a component of a record from the MCRS. See R1.10.23.
Role	(entity) a profile assigned to a user, or inherited by a user from one of the user's **groups**, that allows the user to perform a set of **functions** in the **MCRS**.

Term	Explanation and relationship to general concepts
Select	(verb) a technique defined in MoReq2010 that allows a user to isolate a group of entities so that they can have the same **bulk operation** performed on them, and for other reasons. See R1.5.43.
System Identifier	(element) a universally unique system identifier for every entity in the MCRS, the MCRS itself and every major version of every module of MoReq2010.
System Log	(noun) an error report generated by the MCRS but deliberately not kept in the MCRS. See R1.6.2.
System Support Technician	(noun) a person who manages the organisations technical infrastructure. The system support technician may not be a user of the MCRS. See R1.6.2.
Template	(entity) a mechanism for applying sets of user-defined or custom metadata **elements** to **entities**. *Applying a template to an entity will result in a pre-defined list of entities being created for that entity.*
User	(noun) a known user of the system who is authenticated and able to access the MCRS. (role) a default role given to all new users of the MCRS.

1.20 Relationship to other specifications

Not part of draft consultation.

1.21 Acknowledgements

Not part of draft consultation.

1.22 Bibliography

Not part of draft consultation.

1.23 Table of entity types

Reference	System identifier	Title
E1.23.1	0996870f-cf03-4463-a0fd-622e8a294132	Aggregation
E1.23.2	520b2c5c-99f3-4b72-9b07-ce50992969ed	Alert
E1.23.3	6e2a1c1e-657e-425d-921e-ac638e9b8193	Alert type
E1.23.4	74e510a9-f271-4ebf-8c67-402d1c9bb89e	Class
E1.23.5	42b92376-0465-4fef-9309-f0f9e97c2111	Classification scheme
E1.23.6	dac12fce-875d-4cb0-b70c-da1dff3bf01e	Component
E1.23.7	cbcc1422-b2dd-4727-b4ab-3d74f5313483	Disposal hold
E1.23.8	cbaa9fd4-621a-4090-b106-0cda270f107a	Disposal schedule
E1.23.9	23249c89-9dbc-4434-9d61-3c7c995b0d6d	Element definition
E1.23.10	6dbd44e0-e73e-4e8e-a8a0-23081b71e110	Entity type
E1.23.11	f6623edb-36e7-4b3b-8a3e-517e5c0dc582	Event
E1.23.12	c6d23e76-6e3e-45dc-9f20-161c3aa01964	Function definition
E1.23.13	b462312a-b51e-4941-aa2d-01a5312499df	Group

Reference	System identifier	Title
E1.23.14	4d963718-d05b-4b7e-b568-9da3350f2d23	Record
E1.23.15	fbd9aacd-28ef-4a4c-8b3c-4a1dac92e669	Role
E1.23.16	06c14ecb-07c6-43bf-baba-7c0876460411	System
E1.23.17	77ee8257-c901-478a-bc52-bad53a568189	Template
E1.23.18	85811e20-6e86-41a1-aebe-1495697484a0	User

1.24 Table of alert types

Reference	System identifier	Title
A1.24.1	0996870f-cf03-4463-a0fd-622e8a294132	Function failed
A1.24.2	bf169e97-441b-4578-8be2-c5a061f204b5	Unauthorised user
A1.24.3	92842c76-2b89-4d9c-9f1c-105cd5cf5a9f	Cannot receive alert
A1.24.4	4e670125-cbac-4448-9e02-7785e35051cc	Dispose after review but no review
A1.24.5	50833961-76a7-48bc-83fb-1f84b461d1ea	No trigger element for retention period
A1.24.6	c8b6aba0-7f43-4df6-a73d-cf3c8693859d	Disposal action due date elapsed

1.25 Table of function definitions

Reference	System identifier	Title	Suitable for Bulk Operation?	Applies to Entity Types
F1.25.1	cbed1d56-638d-4743-99a9-82af9286732d	Action alert	Yes	Alert
F1.25.2	a141713c-362e-4aac-953a-1ffd07fbc179	Add element	Yes	Template
F1.25.3	9f9bd3bf-17a6-4b48-a261-de8d7ff0a647	Add template	Yes	Classification scheme, Class
F1.25.4	c925202a-c09c-4fc6-992d-f4b3bf5619b1	Activate	Yes	Most entity types
F1.25.5	21d2695d-0c09-4d53-8922-6054716622b1	Aggregate	Yes	Aggregation, Record
F1.25.6	f18b21d9-5d3c-48ab-83e6-1cfc9aca0a17	Apply disposal schedule	Yes	Class, Record
F1.25.7	a3281652-1b37-4008-adbf-4612e1b9ab09	Apply hold	Yes	Disposal hold
F1.25.8	171c8477-38b2-44ac-ac1f-c3780de7c69d	Apply template	Yes	User, Class, Aggregation, Record, Component
F1.25.9	0e51cd19-abbf-40d9-9858-44cfceef6a9e	Change title	No	Most entity types
F1.25.10	13457b90-3fdf-40ef-a028-10236b1aa249	Change metadata	When specified	All entity types
F1.25.11	209737b7-dbde-4032-9060-f0fed5348638	Classify	Yes	Aggregation, Record
F1.25.12	47dc7c9c-0ad6-49c4-b245-5360bdf2af7f	Clone	Yes	Record

Reference	System identifier	Title	Suitable for Bulk Operation?	Applies to Entity Types
F1.25.13	db3e7456-d4bb-4a3a-b489-076b9a03e6b5	Close	Yes	Aggregation
F1.25.14	6a4f03c5-1d76-438c-a711-94c798b15419	Confirm disposal	Yes	Record
F1.25.15	8240cefa-45ec-4e13-9c9d-f7fc8ef9d6c9	Create	No	All entity types
F1.25.16	c285216d-56a6-4148-9cb8-e80561e1e573	Deactivate	Yes	Most entity types
F1.25.17	2ec7a47b-fd52-4bbb-89d2-611ea0adb0fe	Declassify	Yes	Aggregation, Record
F1.25.18	f986eb7e-bd0a-440d-8a05-e3479265369f	Delete	Yes	Many entity types
F1.25.18	42792e3d-8fb0-4fc8-9c1c-03894b56aa64	Destroy	No	Record
F1.25.20	34775542-ed2b-4b0b-85e7-558a7755c17f	Export	Yes	All entity types
F1.25.21	0bb667e9-a816-44c5-b734-1fdc047a483d	Grant role	Yes	Group or User
F1.25.22	3fd234f2-3f3c-4488-90c5-94c755b0acda	Inspect metadata	No	All entity types
F1.25.23	d28d6180-7a1e-4733-a593-3b122582e034	Lift hold	Yes	Disposal hold
F1.25.24	1a63c088-6cf9-4279-8b1c-f221b2888940	Move	Yes	Aggregation, Record
F1.25.25	f292e918-fc75-4629-a7cd-1dc7ba6a0159	Progress disposal	No	Record
F1.25.26	7c5abf9a-c7c3-411d-b558-b7c23f5d1439	Recall alert	Yes	Alert

Reference	System identifier	Title	Suitable for Bulk Operation?	Applies to Entity Types
F1.25.27	3f888227-27e1-484f-99a5-e6249fc9e6ed	Reclassify	Yes	Aggregation, Record
F1.25.28	597c27ba-c839-437e-8575-d4957348f647	Remove disposal schedule	Yes	Class, Record
F1.25.29	13a08f3d-2929-4c80-9b81-379dbb45eef0	Remove element	Yes	Template
F1.25.30	0417df39-a88b-4bfe-bf57-6fdbdecdbd8e	Remove template	Yes	Classification scheme, Class
F1.25.31	0b3683a6-57ea-4dcb-a1cd-1159fe47429f	Reopen	Yes	Aggregation
F1.25.32	525eeadb-b4d1-47c1-8739-69243be53ff0	Replace disposal schedule	Yes	Class, Record
F1.25.33	ed713064-fdc2-4f14-8602-d06970f0f8d0	Replace template	Yes	Classification scheme, Class
F1.25.34	0f7f3e64-aa52-4119-b510-dc8d96bc6b6d	Report	Yes	User and all entity types
F1.25.35	12f97cc0-2249-45a6-83d5-a8655ecf11b8	Report compliance	No	System
F1.25.36	4bd9c1d0-3db1-478a-aa06-b2699ecb8085	Reposition	Yes	Aggregation, Record
F1.25.37	0bc55335-d79e-491b-9023-30b86df0b626	Rescind role	Yes	Group or User
F1.25.38	11656a88-74ce-41d1-bc4e-dca4fbb87235	Retrieve datafile	Yes	Component
F1.25.39	d65950b1-e5fe-4577-af60-bbebf37eea9a	Review	Yes	Record

Reference	System identifier	Title	Suitable for Bulk Operation?	Applies to Entity Types
F1.25.40	a3038e89-1fbe-4a12-ab2a-283e68d9ad99	Save Report	Yes	User
F1.25.41	7c21a48a-5cde-406f-ac0a-c20c0729d47f	Save Search	Yes	User
F1.25.42	89efcd09-0d40-45bd-aa32-c91266b10f93	Search	Yes	User
F1.25.43	b9225d11-0c27-438e-b850-c697bc9bac9c	Split aggregation	Yes	Aggregation
F1.25.44	cb81c05e-1e56-4e1f-bd62-1e3eedd81deb	Subscribe alerts	No	User
F1.25.45	b89d527a-20ac-4e80-8e3c-5f1bca12372f	Synchronise	Yes	Group or User
F1.25.46	49d35844-72c0-4f37-92f8-0828be3a077b	Unsubscribe alerts	No	User

1.26 Table of element definitions

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.1	008d05bb-9ae3-424f-bc1b-34a5bce17648	Actioned	0..1	Timestamp	Alert
M1.26.2	7d0ba589-24bd-4bbc-8558-bb1155377637	Add to event history	1..1	Boolean	Function definition
M1.26.3	8908d599-2c2f-49da-8bb1-9dcdd07ca3ba	Aggregated	0..1	Timestamp	Aggregation
M1.26.4	0e3042a7-6c72-4ab6-b03e-f8cc895b0406	Aggregated	1..1	Timestamp	Record
M1.26.5	4554c528-d985-4c6f-9ea1-8a8623a7c191	Aggregation template	0..1	Template reference	Class
M1.26.6	ab6ed17b-1bb2-4978-b6eb-be1a5246f0c5	Alert comment	0..1	Text	Alert
M1.26.7	095c37d7-071d-47d9-bcf3-6302c6ff47dd	Alert subject	0..Many	Entity reference	Alert
M1.26.8	88aece74-da01-4f20-b850-8179f88b9007	Alert subscription	0..1	Text	User
M1.26.9	2a020b91-586a-4a84-af42-649195042635	Alert type	1..1	Alert type reference	Alert
M1.26.10	a5e6f694-a48f-4a9b-8a71-58a9615f89d8	Can auto delete	0..1	Boolean	Component
M1.26.11	f2468942-6c45-4e92-b2a0-5d8bf1fcc9e3	Can classify records	1..1	Boolean	Class
M1.26.12	56adbe51-0172-4263-b3c9-a128c3e80757	Change	0..Many	Complex	Event

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.13	d55a2dc0-5bc7-47e3-b5e2-26edc86c24f8	Class template	0..1	Template reference	Classification scheme
M1.26.14	77aa4235-9968-4efb-a460-85cb46298b15	Classification	0..Many	Class reference	Aggregation, Record
M1.26.15	f627be8d-38bf-4a0a-8579-17e9f0808d16	Classification method	1..1	Code	Classification scheme
M1.26.16	4910715a-54af-4c40-8fc7-ea0a2bf9497f	Clone	0..Many	Record reference	Record
M1.26.17	16faf0ce-af1f-41e8-a9db-2e645c0bbfc1	Closed	0..1	Timestamp	Aggregation
M1.26.18	a28d5cda-4b19-4f97-81ae-e4a4b8264dde	Component type	1..1	Code	Component
M1.26.19	758f707b-07a7-4e01-8812-a5eea15b6df2	Created	1..1	Timestamp	All entity types
M1.26.20	8341ad11-4efa-477d-a395-091960c3b6dd	Creator	1..1	User entity reference	All entity types
M1.26.21	5cc16a50-6336-4468-8e53-8c86a78d9899	Datafile accessible	0..1	Boolean	Component
M1.26.22	902d1bcd-3493-47a1-a5b5-d45a1d5fe9e4	Datatype	0..1	XML datatype	Element definition
M1.26.23	26db6414-1f0e-4a6f-92ed-8cdec49069ef	Default language	1..1	Language Identifier	System
M1.26.24	949ef9df-8c24-4ec0-bcdc-ae721e37406c	Default value	0..Many	Union	Element definition

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.25	dc913a3e-44dd-4dbc-9ac7-ffcfe40cc841	Description	0..1	Text	All entity types
M1.26.26	4e450713-0609-4921-abc4-bb25e987119d	Destroy event	0..1	Boolean	Function definition
M1.26.27	16373560-53ad-497a-a0ff-3b6ec22c0f0a	Disposal action	0..1	Code	Disposal schedule, Record
M1.26.28	24882ea8-cd28-4830-9f5a-0b84f174847e	Disposal action due	0..1	Timestamp	Record
M1.26.29	bd8c66d5-5c28-4ecc-a619-1d72ea94d2d1	Disposal action reminder sent	0..1	Timestamp	Record
M1.26.30	4465f886-b7b8-4806-a240-b84aab4fb78a	Disposal interval	0..1	Code	Disposal schedule
M1.26.31	1eb07d5c-7de8-4f39-9f34-f48cf2934103	Disposal interval duration	0..1	Numeric	Disposal schedule
M1.26.32	a2b8b14d-c48a-4b41-917d-0ced8f692938	Disposal schedule	0..1	Disposal schedule reference	Class
M1.26.33	93c4acf7-1405-4f7e-91cb-419a58e0847a	Disposal schedule	1..1	Disposal schedule reference	Record
M1.26.34	4408653c-5411-4e55-97eb-f750aa4907af	Disposal scheduled	1..1	Timestamp	Record
M1.26.35	6f92663b-f940-4bed-a056-d4bd5ab25198	Disposed	0..1	Timestamp	Record
M1.26.36	0e99a86c-5fe1-42b9-82c5-a010c520207d	Disposed comment	0..1	Timestamp	Record

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.37	e9ca35f2-092f-4aff-8de4-9a8f1b3937fb	Entity type	0..Many	Entity Type reference	Element definition
M1.26.38	3373f254-fbd1-4125-903c-d03b308f8b3d	Entity type	1..Many	Entity Type reference	Template
M1.26.39	ca7ff8f2-ef63-4eb8-8e65-bea9a31f02dc	Event history	1..Many	Event reference	All entity types except events
M1.26.40	b7fd889c-52fd-4e69-9e0b-7a700704e478	Export comment	0..1	Text	Event
M1.26.41	4e7d8a87-690e-4009-97e7-20f20d6964f3	Export success	0..1	Code	Event
M1.26.42	6058fb39-1c8d-44a1-86cc-c2cc4caa0b7e	External identifier	0..1	Text/URI	System
M1.26.43	6f7bb024-afc7-45ea-8f03-f19101fe003d	External identifier	1..1	Text/URI	Group, User
M1.26.44	2652791f-6e24-4193-a721-55e6bf1f6bad	Function performed	1..1	Function definition reference	Event
M1.26.45	7cc91a82-df3e-45d1-80d7-04d2e0c8a7e0	Group membership	0..Many	Group reference	User
M1.26.46	07725d79-69d2-4d2f-adb8-a1eeceb59822	Held entity	0..Many	Union of Aggregation and Record reference	Disposal hold

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.47	ca2a2fe9-31b7-4557-a0f5-a87fe41343e0	Hold comment	0..1	Text	Disposal hold
M1.26.48	f36c960c-4067-4485-be80-2e6cf1fc0fb1	Keep on destroy	0..1	Boolean	Function definition, Element definition
M1.26.49	128a7839-c590-4f5e-9b06-effa0b19b8ae	Last added to	0..1	Timestamp	Aggregation
M1.26.50	23cf1fde-4398-4265-a3d2-b2f770a2f95c	Last review comment	0..1	Text	Record
M1.26.51	e925017d-6c69-4cdc-8013-de160b556629	Last reviewed	0..1	Timestamp	Record
M1.26.52	9e2c48f9-2a1d-4698-b8ae-6dea88fb2605	Last synchronised	1..1	Timestamp	Group, User
M1.26.53	6f658b9b-25a1-430f-b817-66a6c1e7a952	Main component	0..1	Component reference	Record
M1.26.54	21d8001b-7d48-4715-a6d1-ae38f75a5420	Mandate	0..1	Text	Disposal schedule
M1.26.55	45ceada5-50c3-41ce-8b0f-4df2858a2c91	Mandate URI	0..1	URI	Disposal schedule
M1.26.56	9aa6e8b7-4d22-4bcf-9423-a20e072aa203	Max occurs	1..1	Numeric	Element definition
M1.26.57	6d134141-1060-4c55-b6a9-dffc5772f351	Min occurs	1..1	Numeric	Element definition
M1.26.58	62a720a4-39dd-47ba-8e05-73adaa5673c7	Element	0..Many	Element Definition reference	Template
M1.26.59	d1e14863-d005-4e2f-a436-51fcf0bcd65e	Organisation address	0..1	Text	System

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.60	13f22d69-4adb-4aef-b88a-1522fbff6a83	Organisation contact	0..Many	Complex	System
M1.26.61	4d4eb481-2b7b-4931-bb96-605e9de84f00	Organisation name	0..1	Text	System
M1.26.62	98cd8cd8-5eea-458c-98bd-2c288227b976	Parent aggregation	0..1	Aggregation reference	Aggregation
M1.26.63	b801458d-2a88-429f-9047-95d3ec594fd0	Parent aggregation	1..1	Aggregation reference	Record
M1.26.64	b3ac8410-3446-44b1-9dfd-697336140dd1	Participating entity	1..Many	Entity reference	Event
M1.26.65	ed7f78c5-94cb-4580-85e5-5e7d9a334d22	Primary classification	1..1	Class reference	Aggregation, Record
M1.26.66	04414b30-907c-407d-8f28-456998a92afd	Product name	1..1	Text	System
M1.26.67	a9d16335-38cc-4f1b-ab2c-67914f79328d	Product support URI	1..1	URI	System
M1.26.68	6605523e-3485-419d-9b98-68a82e145132	Product version	1..1	Text	System
M1.26.69	25107bf2-0ef1-48d9-92f1-5930f1f6b1f1	Receive alerts	0..1	Boolean	User
M1.26.70	fd19c5a0-791b-4b7a-aee0-94b74a34c08a	Record template	0..1	Template reference	Class
M1.26.71	2c684ecf-33ac-47c1-9694-48385a9ef214	Retain until	0..1	Timestamp	Record

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.72	6f9ab2ec-845b-405d-a03a-67b03f457616	Retention interval	1..1	Code	Disposal schedule
M1.26.73	67c9842b-0801-4181-90d4-987ce25fb8c5	Retention interval duration	0..1	Numeric	Disposal schedule
M1.26.74	482f6d88-3569-4caa-8a56-ea1a499b25c7	Retention offset	0..1	Code	Disposal schedule
M1.26.75	906cc323-4d35-4d0e-953b-0a064e55e6d8	Retention offset month	0..1	1..12	Disposal schedule
M1.26.76	043029ba-01a6-4e50-91b4-43c624378cbd	Retention trigger	1..1	Code	Disposal schedule
M1.26.77	0f18e54a-b82a-44ee-b4c7-00d5f0e2fc2b	Retention trigger element	0..1	Element Definition reference	Disposal schedule
M1.26.78	473c88ea-2ba0-4f5b-9c50-6588ee52616a	Scope notes	0..1	Text	Class, Disposal schedule
M1.26.79	60dd871b-77f1-4a12-9890-3b2e52c78705	Sequence	1..1	Numeric	Record
M1.26.80	59d0ecdb-f390-4ff5-ac5d-4e3803c51f8a	Sequence	0..1	Numeric	Aggregation
M1.26.81	9b6d92ee-2d1d-4230-ad87-34f578d0d74c	Status	1..1	Code	All entity types
M1.26.82	02a8e957-c97a-4203-b0b5-19d3c2e132b8	Supplier’s address	1..1	Text	System
M1.26.83	b8febdd7-de86-47f1-8f9b-c25a61ede61e	Supplier’s name	1..1	Text	System
M1.26.84	a84dfa90-ef81-4450-9add-1b1f3e41e595	Supplier’s website	1..1	URI	System

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M1.26.85	0996870f-cf03-4463-a0fd-622e8a294132	System identifier	1..1	UUID	All entity types
M1.26.86	2c277d09-4024-4d62-9946-bea6b36d86c9	Text language	0..1	Language Identifier	Component
M1.26.87	86244bf5-ab39-46b7-ab65-fc9ab6fe78f3	Text searchable	0..1	Boolean	Component
M1.26.88	5a0bd632-64b8-4485-afb7-9b94020b1e98	Title	1..1	Text	All entity types

1.27 Table of general codes

Reference	System identifier	Value	Applies to Element
C1.27.1	55d42941-0b68-42aa-8b47-5c64770b6298	ACTIVE	Status
C1.27.2	2dac1f7c-785a-4f57-beaa-0d4616d8f90a	INACTIVE	Status
C1.27.3	204162c1-6496-4619-8853-02ac5529c866	EXPORT SUCCESSFUL	Export success
C1.27.4	84ced01c-8e7f-4e62-b10c-b34e4dc0b082	EXPORT CANCELLED BY USER	Export success
C1.27.5	de5ae200-bf15-47a5-b471-df97c376435c	EXPORT FAILED	Export success

1.28 Table of disposal codes

Reference	System identifier	Value	Applies to Element
D1.28.1	61d61843-e4df-4155-afde-46e94d5c8a80	NOW	Retention trigger
D1.28.2	9f950f0b-c931-4842-9278-600f24540d8e	NEVER	Retention trigger
D1.28.3	5981e324-3ea9-42a1-a656-afc2c1a72e96	DATE OF LAST REVIEW	Retention trigger
D1.28.4	46d23f24-798f-4e66-84a3-3bb3d4353cc7	DATE RECORD CREATED	Retention trigger
D1.28.5	cb564717-580b-4ca4-b941-1a18d9f715db	DATE RECORD INCLUDED IN AGGREGATION	Retention trigger
D1.28.6	5705140f-117d-418b-9515-5eab8f1e0c7a	DATE LAST RECORD INCLUDED IN AGGREGATION	Retention trigger
D1.28.7	bce92663-4126-4289-8067-ada2a7e54447	DATE AGGREGATION CREATED	Retention trigger
D1.28.8	9344ab3d-d283-463b-968c-657f740d1844	DATE AGGREGATION CLOSED	Retention trigger
D1.28.9	03ac2a75-6005-41c7-8f67-641bc8f84588	METADATA ELEMENT DATE	Retention trigger
D1.28.10	be2ca101-1e81-4c1f-ad2b-cff2fd515afb	NO RETENTION PERIOD	Retention interval
D1.28.11	9fd351a9-d83c-44aa-b76f-b0954f968b89	DAYS	Retention interval
D1.28.12	6ef7af61-b5a1-4b8e-9435-1c7b201961b4	WEEKS	Retention interval

Reference	System identifier	Value	Applies to Element
D1.28.13	9d89cedd-a191-4049-aef6-3c8a0416cd50	MONTHS	Retention interval
D1.28.14	8d013a79-1b9f-4fe3-9a5c-56c969f5c5a2	YEARS	Retention interval
D1.28.15	217352ec-ef58-4d95-9451-fee216d3114f	NO OFFSET	Retention offset
D1.28.16	cdadd0b8-1a60-46e8-99a3-e6bc47edb106	START OF NEXT MONTH	Retention offset
D1.28.17	7c1bd9d7-e6d0-4bd6-949c-c2ace490abc0	START OF NEXT QUARTER	Retention offset
D1.28.18	fe45b94e-7ffd-4f1a-b974-512bf51fe480	START OF SPECIFIED MONTH	Retention offset
D1.28.19	8c17f566-07fb-4489-9962-b64fb9a75ba0	REVIEW	Disposal action
D1.28.20	4dd8e319-468b-47ad-b1bd-e653b21a3680	TRANSFER	Disposal action
D1.28.21	53c78ce7-db75-4de4-a7ca-6ff5b9796bd3	DESTROY AFTER CONFIRMATION	Disposal action
D1.28.22	a9d12e67-4c5c-457d-bb08-d217fc66071f	DESTROY WITHOUT CONFIRMATION	Disposal action
D1.28.23	f2f112fc-fa16-459e-b505-0700a3119679	DAYS	Disposal interval
D1.28.24	08ed2e55-7c36-4f5d-b2a5-d7a1544f81ce	WEEKS	Disposal interval

100 Series – Interface types

101 Graphical User Interface (GUI)

101.1 Important information

101.1.1 Identifier

37ebe5d0-fe2c-4142-ae67-17d66929bdb5

101.1.2 Title

MoReq2010 – Graphical User Interface (GUI) Requirements

101.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the requirements for all compliant records systems that implement a graphical user interface or GUI.

101.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

101.1.5 Prerequisites and co-requisites

The MCRS must be compliant with the following modules of MoReq2010:

- 1. Core requirements

The MCRS may be compliant with the following modules of MoReq2010:

- none

101.1.6 Intellectual Property Rights

101.1.7 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

101.1.8 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

101.2 Key concepts

A graphical user interface or GUI is the most common way for users to interact with modern computer technologies. Information is typically transmitted to a GUI through input devices such as keyboards, mice, tablets, pens/styli, touch screens and microphones that transmit voice commands. Feedback from a GUI is primarily visual through a screen or display, however some audio and other supplementary feedback is sometimes provided. Key features of a GUI are:

- It allows users to see more than one item or piece of information at a time and to see some information as images or graphics;
- It allows users to manipulate and perform actions on items by clicking with a mouse, tapping with a finger or typing on a keyboard;
- It allows users to navigate visually between entities by providing multiple item displays such as list views and tree views;
- It can allow users to select multiple items at once by lassoing them, or similar technique;
- It is able to display information sorted in different ways and to indicate the status of items with various visual effects;
- It is able to overlay different information from parts of a system in a single coherent display;
- It can provide pop-up notifications and dialogue boxes for the user to interact with;
- It can simulate a form for the entry of metadata about an item;
- It can guide the user through complex processes through techniques such as a wizard;
- It can limit the options that a user can choose through pick lists and menus;
- It can disable functions that are not appropriate for a particular item; and
- It can display search results and the content of items directly to the screen or display.

This module contains functional and non-functional requirements for an MCRS which implements a GUI.

101.3 Functional requirements for graphical user interfaces

R101.3.1

The MCRS must implement a graphical user interface (GPI) that allows users access to its functions.

The term "GUI" describes any primarily visual interface that is controlled by a human operator. GUIs may be contrasted with earlier command line interfaces where systems are controlled by typing instructions or commands into a terminal.

R101.3.2

The MCRS must provide the full functionality specified by MoReq2010 through its GUI under R101.3.1.

To be certified the GUI must not implement only a part of the functionality specified by MoReq2010.

R101.3.3

When browsing, the MCRS must be able to display the contents of the classification scheme, inventory or aggregation being browsed in a logically ordered manner that allows the user to change focus from one entity to the next.

The term "browse" is defined in the rationale to R1.5.14.

R101.3.4

While browsing under R101.3.3, the MCRS must be able to allow the user to inspect an entity without losing the current browsing order.

The GUI must allow the user to move between browsing and inspecting and back again without having to find the entity again.

R101.3.5

Wherever an entity appears in the interface the MCRS must allow the user to perform functions on it without changing to a different mode of operation.

For example, the entity might appear in search results, while browsing, while being inspected, in the metadata of another entity, etc.

R101.3.6

The search function must be available to a user of the MCRS at all times.

The user should not have to navigate to a special screen in order to conduct a search.

R101.3.7

The MCRS must ensure that users cannot enter metadata into element entry boxes contrary to the datatype of the element.

The MCRS must enforce and support the entry of correct datatypes including any constraints in the datatype.

R101.3.8
The MCRS must provide feedback to users who have initiated long operations and provide a control to allow users to cancel them before they have completed.

An example of a suitable feedback mechanism is a progress bar. Long operations include bulk operations under R1.6.14, searching under R1.14.22, reporting under R1.14.46, and exporting under R1.15.7.

R101.3.9
The MCRS must provide on line help for every part of its interface and make it available from every part of its interface.

It must be possible to access the help system from any screen and the help must be relevant to the process the user is performing.

R101.3.10
Where the GUI is hosted within a web browser interface the MCRS must meet the W3C Web Content Accessibility Guidelines (WCAG) 2.0 and the W3C Accessible Rich Internet Applications (WAI-ARIA) 1.0 to at least conformance level "A".

Conformance levels of "AA" and "AAA" are preferred. Although the W3C guidelines have been developed specifically for web applications they provide a useful guide to accessibility principles in any GUI based application and are therefore highly recommended.

101.4 Non-functional requirements for graphical user interfaces

N101.4.1

How user friendly is the GUI?

Ergonomics is an important albeit subjective judgment of all GUIs. For example, how many mouse clicks and movements does it take to perform commonly used functions? The best GUIs are often described as intuitive to use.

N101.4.2

How much use does the GUI make of graphical elements in its design?

For example, does the GUI use icons to represent different entity types and states? Is it easy to see the difference between active and inactive entities?

N101.4.3

Does the GUI use common gestures in appropriate ways to help make the MCRS more usable and useful?

For example, does the GUI allow records to be captured by dragging and dropping datafiles from the user's desktop, does the GUI allow scrolling and zooming, etc.?

N101.4.4

Does the GUI artificially restrict the depth of aggregation structure that can be used or limit the number of entities in an inventory, etc.?

For example, see N1.17.3. The MCRS may have no theoretical limit to the number of levels of aggregation it can support but there may only be a limited number that fit into size or width of the GUI for display purposes?

N101.4.5

Can the GUI be personalised to suit individual users?

To what extent can a user set up the GUI in a particular way? Is it customisable for screen colour, layout, fonts and font sizes? Can the user save their settings and return to them later?

N101.4.6

Are audio alerts and notifications configurable?

Does the GUI make use of audio alerts and notifications? Can volume settings be individually adjusted under N101.4.5?

N101.4.7

Does the MCRS provide easy access to recently accessed entities through the GUI so that the user can access them later without searching?

MoReq2010 does not define "recent" in this context. The intent of this non-functional requirement is that the MCRS should provide a means for users to return to entities that they created, inspected or modified recently without having to search or browse to find them.

N101.4.8

Does the MCRS allow users to keep a list of favourite entities and access them easily?

The intent of this non-functional requirement is that the MCRS should allow users to access their favourite entities without having to browse or search to find them. This is particularly important because users may often only need to have access to a small number of aggregations to do most of their daily work capturing, declaring and referring to records. Many users will wish to keep these aggregations close at hand, rather than constantly search for them. Another good use of favourites is for particular saved searches and reports that are often performed by the user, see R1.14.35 and R1.14.50.

N101.4.9

How does the GUI handle error conditions and failed functions?

Are notifications to users intelligible, informative and friendly? Do they suggest a useful course of action?

N101.4.10

How useful is the on line help provided with the GUI?

Is it context sensitive? Does it contain embedded videos, on line tutorials, and/or e-learning techniques?

N101.4.11

How accessible is the GUI to users with accessibility challenges?

Does the organisation have groups of users with particular accessibility requirements? Does the MCRS GUI cater for the needs of these users individually?

N101.4.12

Does the GUI respond to voice commands?

Is it possible to navigate the GUI and perform functions using voice commands only?

101.5 Glossary of terms

Term	Explanation and relationship to general concepts
Graphical User Interface	(noun) An **interface** to the **MCRS** that allows human users to directly access the **MCRS** and manipulate its **entities** through a primarily visual interface.
GUI	See **graphical user interface**.

102 Application Programming Interface (API)

102.1 Important information

102.1.1 Identifier

1d10e5f4-09a3-4e04-a788-202198764168

102.1.2 Title

MoReq2010 – Application Programming Interface (API) Requirements

102.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the requirements for all compliant records systems that implement an application programming interface or API.

102.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

102.1.5 Prerequisites and co-requisites

The MCRS must be compliant with the following modules of MoReq2010:

- 1. Core requirements

The MCRS may be compliant with the following modules of MoReq2010:

- none

102.1.6 Intellectual Property Rights

102.1.7 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

102.1.8 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

102.2 Key concepts

An application programming interface or API is a published and standardised interface to a system such as an MCRS that allows other software to interact with it. MoReq2010 defines an API as consisting of a number of methods which when called by the external system will implement the functionality described in this specification.

When an MCRS implements an API it must be fully self-sufficient and implement all of the functionality described by MoReq2010 without relying on any other interface to provide any of the functionality mandated by the specification.

Each MoReq2010 function must correspond to a single method call and it should not be necessary through the API to call more than one method to perform a single function defined by the specification. The supplier must provide a manual to the API describing the methods and their pre- and post-conditions.

This module contains functional and non-functional requirements for an MCRS which implements an API. Non-functional requirements cover how the API is maintained, supported, versioned, and various methods introduced and deprecated over time by the supplier.

102.3 Functional requirements for application programming interfaces

R102.3.1

The MCRS must implement an application programming interface (API) that allows users access to its functions.

The term "API" describes any interface that is accessed by a client application or another business system, rather than by a human operator. A human operator may interface with the business system and indirectly manipulate entities in the MCRS, but there is no direct interaction with people.

R102.3.2

The MCRS must provide the full functionality specified by MoReq2010 through its API under R102.3.1.

To be certified the API must not implement only a part of the functionality specified by MoReq2010.

R102.3.3

The MCRS must include within its API under R102.3.1 a method that performs the same operation for each function specified by MoReq2010.

Every single function specified by MoReq2010 should be performed by an equivalent single method call. It should not take multiple calls to the API to perform a single function.

R102.3.4

For every function that can be performed as a bulk operation as specified by MoReq2010 the MCRS must allow the function to be performed for all selected entities within a single method call under R102.3.3.

For example, by passing to the method call a list of selected entities. The MCRS must perform the bulk operation not the calling process or system.

R102.3.5

For every method call under R102.3.3 and R102.3.4 the MCRS must return an error code that indicates the success or failure of the call.

The calling process, application or system should not have to make a second method call to discover the outcome of the previous method call.

R102.3.6

The MCRS must provide a facility to use the error code returned under R102.3.5 to obtain extended error information for why a particular method failed.

Extended error information is explained in the rationale to R1.6.6 and R1.6.7.

R102.3.7
The extended error information returned by R102.3.6 must be available in every language supported by the MCRS.

The MCRS should return the extended error information in the default language of the MCRS. The default language of the MCRS is explained in R1.5.2.

R102.3.8
The MCRS must provide a facility as part of its API under R102.3.1 for allowing users to initiate long operations and later cancel them before they have completed.

For example, see bulk operations under R1.6.14, searching under R1.14.22, reporting under R1.14.46, and exporting under R1.15.7.

R102.3.9
A process, application or other system accessing the API of the MCRS under R102.3.1 must be authorised as a user of the MCRS under R1.7.1 with the same access and abilities of a user.

Every access through the API must come from a user as specified in the MoReq2010 requirements.

R102.3.10
The supplier must publish a complete manual to the API that lists its methods in full, their pre- and post-conditions, and describes their purpose and how a process application or other system may access the MCRS and call them.

The manual may be produced in paper or electronic form as a traditional or non-traditional document, for example, as a Wiki. The manual should give examples of working code and development hints, tips and tutorials.

102.4 Non-functional requirements for application programming interfaces

N102.4.1
What type(s) of API does the MCRS support?

For example, web services, REST, programming languages (Java, C++, Python, etc.), Microsoft .NET framework, SOA, etc.

N102.4.2
What platforms does the API run on?

For example, Google App Engine, Amazon EC2, Microsoft Windows Azure, Microsoft Windows, Linux, Mac OS X, Google Android, Apple iOS, etc.

N102.4.3
What protocols does the API use?

For example, HTML, SOAP, RPC, TCP/IP, XML, etc.

N102.4.4
In addition to the manual in R102.3.10, what other training or educational programmes does the supplier and/or third-party organisations provide?

How easy is the API to learn? Is the customer organisation freely able to build applications that interface to the API or must this be done by the supplier's developers? Are there third-party organisations providing API training, education and support?

N102.4.5
How often is the API updated and how is it versioned?

How easy is the API to learn? Is the customer organisation freely able to build applications that interface to the API or must this be done by the supplier's developers?

N102.4.6
How are new method calls introduced and old ones deprecated?

See also questions about the suppliers general support process and policy on new software versions for the MCRS under 1.18.

N102.4.7
In relation to N102.4.5 and N102.4.6, can code written to interface with one version of the API run against previous versions?

Is there a major/minor numbering system?

102.5 Glossary of terms

Term	Explanation and relationship to general concepts
API	See **application programming interface**.
Application Programming Interface	(noun) An **interface** to the **MCRS** that allows processes, applications and other systems authorised as users to access the **MCRS** and manipulate its **entities**.
Error code	(noun) A value returned by calling a **method** in an **API** that describes whether the underlying function was successful or not.
Method	(noun) An operation that can be carried out by a process, application or other system accessing an **API** that provides functionality exactly corresponding to a function in the **MoReq2010** specification.

200 Series – Classification types

201 Hierarchical Classification

201.1 Important information

201.1.1 Identifier

fb9a7912-dd77-4c00-afc7-b7ad19284cb3

201.1.2 Title

MoReq2010 – Requirements for hierarchical classification

201.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the requirements for all compliant records systems that implement a hierarchical classification scheme.

201.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

201.1.5 Prerequisites and co-requisites

The MCRS must be compliant with the following modules of MoReq2010:

- 1. Core requirements

The MCRS may be compliant with the following modules of MoReq2010:

- none

201.1.6 Intellectual Property Rights

201.1.7 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

201.1.8 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

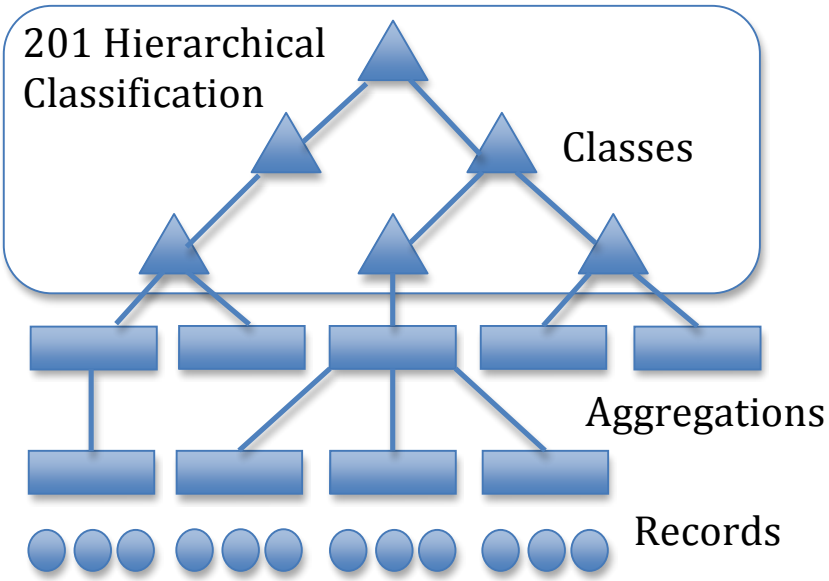
Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

201.2 Key concepts

This module of MoReq2010 describes requirements for hierarchical classification. Hierarchical classification provides backward compatibility with the classification scheme used in previous versions of the specification where it was the only method of classification supported.

Hierarchical classification schemes organise classes into a multi-tiered tree structure. A typical three tiered classification scheme might be organised by business function, and then within each function by business activity, and finally within each activity by transaction.

One advantage of hierarchical classification is that its structure is easy to represent and to navigate by browsing. Another advantage is that an organisation may join a hierarchical classification scheme to its aggregation model to produce a single navigable business structure containing several tiers of classes followed by one or more levels of aggregation, as shown in the diagram below.



In order to assist this process MoReq2010 allows only the lowest tier of hierarchical classification to be used in classification. Another way of expressing this is that parent classes cannot be used for classification of aggregations or records.

While relatively simple to implement, the hierarchical approach assumes that both the classification model and the aggregation model that the organisation uses are compatible. This may not be the case in some environments, such as case work.

Hierarchical classification is the most common form of classification in general use by different organisations. However, it is sometimes criticised as being overly restrictive in imposing a single top down view of a business. For example, two records that are

closely related according to a secondary criterion may be placed into widely separated classes (as measured by the shortest path between the classes when traversing the hierarchical classification structure).

Some organisations may prefer to implement an alternative method of classification that suits a particular business need. For this reason, MoReq2010 is able to accommodate different classification modules.

201.3 Functional requirements for hierarchical classification

R201.3.1

The MCRS must ensure that all classes within a hierarchical classification scheme are organised in a tree structure such that each class has at most one parent class and may have many child classes.

Hierarchical classification schemes should support any number of tiers of hierarchy and any number of child classes in a single parent class.

R201.3.2

The MCRS must include a mandatory textual title separator element and a title direction indicator element in the metadata of a hierarchical classification scheme.

The title separator and title direction indicator elements are mandatory and must be included in the metadata of the classification scheme when it is created under R1.8.3. The purpose of the title separator and title direction indicator is to build the prefix for the title of a class as explained under R201.3.6.

Valid values for the title direction indicator are:

- *LEFT-TO-RIGHT*
- *RIGHT-TO-LEFT*

R201.3.3

The MCRS must include a mandatory textual sub-title element in the metadata of each class in a hierarchical classification scheme.

The purpose of the sub-title is to build the suffix for the title of a class as explained under R201.3.6.

R201.3.4

The MCRS must not allow an administrator to directly change the title of a class contrary to R1.8.11, and must instead allow the administrator only to modify the textual sub-title of the class included under R201.3.3.

The sub-title is mandatory.

R201.3.5

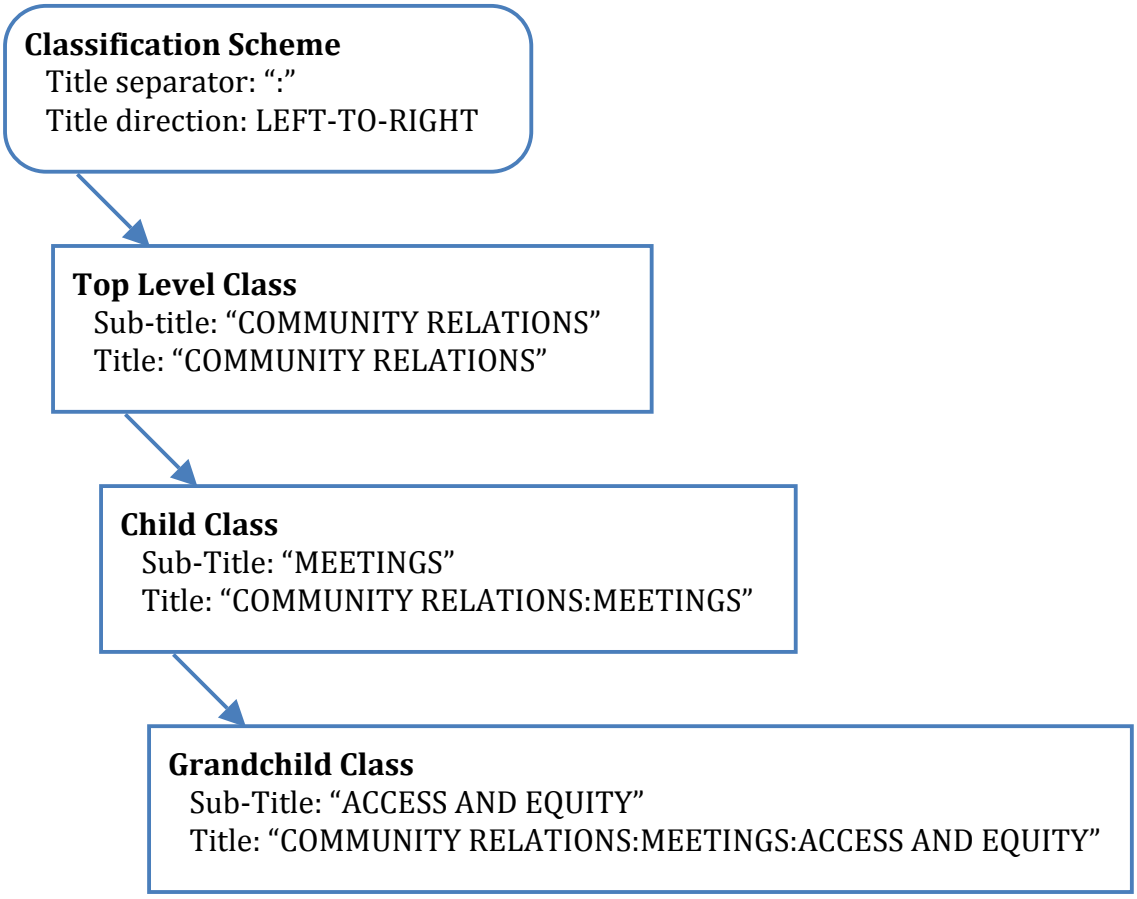
Whenever the sub-title of a class changes the MCRS must automatically build the title of the class and change it and the title of any child classes, and their child classes, and so on.

The titles of classes in hierarchical classification schemes are updated in accordance with R201.3.6.

R201.3.6

When the MCRS automatically builds the title of a class under R201.3.5, it must do so by taking the sub-title of each parent class and place them in order using the title separator and the title direction indicator from R201.3.2.

The diagram below is an example of how the title of a class in a hierarchical classification scheme is built. In each case the title of the class is generated automatically by the MCRS and the administrator is able to modify only the sub-title.



R201.3.7

When an administrator creates a new class under R1.8.6 the MCRS must allow the administrator to optionally specify a parent class for the new class.

Classes without a parent class are the top level classes within a hierarchical classification scheme.

R201.3.8

The MCRS must allow an administrator to move a class with no parent class into another class.

Any classes within the class move with it.

R201.3.9
The MCRS must allow an administrator to move a class with a parent class out of the parent class so that it becomes a new top level class.

All of the classes within the class move with it.

R201.3.10
The MCRS must allow an administrator to move a class from one parent class to another.

All of the classes within the class move with it.

R201.3.11
The MCRS must not allow a class to be created under R201.3.7, or moved under R201.3.8 or R201.3.9, if the proposed parent class has been used to classify an active aggregation or record.

A class in a hierarchical classification scheme can be used to classify or used as a parent for other classes, but not both.

R201.3.12
The MCRS must not allow parent classes to be used to classify aggregations or records.

R201.3.13
The MCRS must ensure that when a parent class is associated with an aggregation template under R1.8.12, that it is automatically inherited by any child classes that do not have a different aggregation template directly associated with them.

Changing or removing the aggregation template affects every child class and their children, and so on.

R201.3.14
The MCRS must ensure that when a parent class is associated with a record template under R1.8.14, that it is automatically inherited by any child classes that do not have a different record template directly associated with them.

Changing or removing the record template affects every child class and their children, and so on.

R201.3.15
The MCRS must ensure that when a parent class is associated with a disposal schedule under R1.8.16, that it is automatically inherited by any child classes that do not have a different disposal schedule associated with them.

Changing or removing the disposal schedule affects every child class and their children, and so on. This may result in cascading changes to aggregations, as described under R1.8.17 and may not always be possible as explained under R1.8.19.

R201.3.16
The MCRS must ensure that when a parent class is marked by an administrator so that they it be applied directly to records under R1.8.20, that the marking is automatically inherited by any child classes that have not been specifically unmarked under R1.8.21.

Unmarking the class under R1.8.21 will have the opposite effect.

R201.3.17
The MCRS must ensure that an administrator cannot make a parent class inactive under R1.8.22 unless all of its children are already inactive.

An inactive parent class in a hierarchical classification scheme cannot contain an active child class.

R201.3.18
The MCRS must ensure that an administrator cannot make a child class active under R1.8.23 unless its parent is already active.

See the rationale to R201.3.17.

R201.3.19
The MCRS must not allow an administrator to delete a parent class under R1.8.29.

The administrator must first move or delete the child classes it contains.

R201.3.20
When new classes are added in bulk to a hierarchical classification scheme, in accordance with R1.8.30, then the datafile must specify the hierarchical relationships between the new classes to be added to the MCRS, and the hierarchical relationships between the new classes and any existing classes already in the hierarchical classification scheme.

This requirement could be achieved if, for example, the datafile specified the parent class for each new class to be added to the hierarchical classification scheme either by its system identifier in the MCRS, its index in the datafile, its title, or some other individual metadata element. MoReq2010 does not specify the format of the datafile or how the MCRS should interpret it.

201.4 Non-functional requirements for hierarchical classification

There are no non-functional requirements for hierarchical classification.

201.5 Glossary of terms

Term	Explanation and relationship to general concepts
Hierarchical Classification Scheme	(entity type) A **classification scheme** where the **classes** are organised as a **tree structure**.
Tree Structure	(noun) An acyclic connected graph of nodes, where each node has zero or more child nodes and at most one parent node.

201.6 Table of element definitions

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M201.6.1	c7084729-74dd-4a52-a903-081c8dbf5f2c	Parent class	0..1	Class reference	Class
M201.6.2	77df308e-0f7d-4bf6-8162-57b339b3f96d	Sub-title	1..1	Text	Class
M201.6.3	f8624063-991a-403a-a629-b9644dbe98d3	Title direction	1..1	Code	Classification scheme
M201.6.4	49a9595f-e6cf-4633-92cf-f9efe077a7ce	Title separator	1..1	Text	Classification scheme

201.7 Table of general codes

Reference	System identifier	Value	Applies to Element
C201.7.1	4a7647fd-e9da-494f-83d1-1e95bcf45122	HIERARCHICAL	Classification method
C201.7.2	3377baa9-21e7-4eeb-b45e-3c7f0c187b64	LEFT-TO-RIGHT	Title direction
C201.7.3	e6014728-4c40-4ea8-96c4-9ad016bab24d	RIGHT-TO-LEFT	Title direction

202 Monolingual Thesaurus Classification

202.1 Important information

202.1.1 Identifier

2890bc8b-4e31-4266-9be7-1f12b5043a45

202.1.2 Title

MoReq2010 – Requirements for classification using a monolingual thesaurus

202.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the requirements for all compliant records systems that implement a monolingual thesaurus based classification scheme.

202.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

202.1.5 Prerequisites and co-requisites

The MCRS must be compliant with the following modules of MoReq2010:

- 1. Core requirements

The MCRS may be compliant with the following modules of MoReq2010:

- none

202.1.6 Intellectual Property Rights

202.1.7 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

202.1.8 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

202.2 Key concepts

Thesaurus classification schemes organise classes as terms in a thesaurus. They are typically used for subject based classification. The particular thesaurus approach adopted by this module is compliant with an ISO 2788 standard monolingual thesaurus.

Thesauri classification schemes allow for more complex relationships between classes than pure hierarchical classification schemes. For example, terms in a thesaurus can have polyhierarchical relationships where a single class can have two or more parent classes (or broader terms).

When classes are organised into a thesaurus based classification scheme they are usually called "terms" and may be preferred or non-preferred.

A preferred tem will have:

- none or several broader terms (the equivalent of parent classes);
- none or several narrower terms (the equivalent of child classes); and
- none or several related terms.

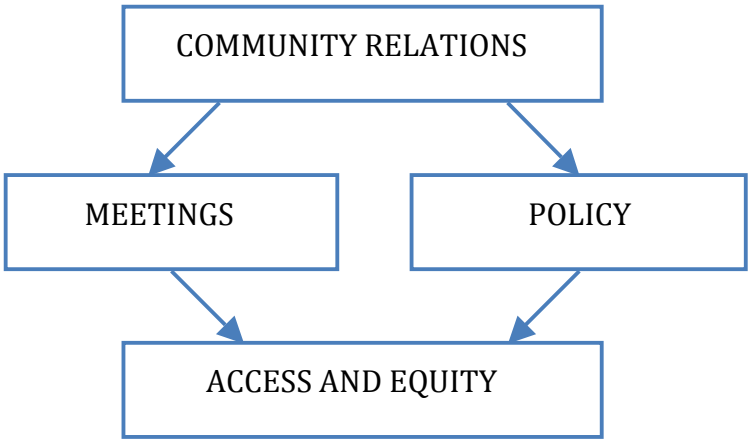
A non-preferred term will not have any broader, narrower, or related terms but will have instead:

- one preferred term (only).

Classes with no parents in a thesaurus based classification scheme are called "top terms". Users may browse a thesaurus based classification scheme in more ways than other types of classification schemes as they may traverse between related terms as well as moving between broader and narrower terms.

A thesaurus based classification scheme also allows a user to select a non-preferred term when classifying an aggregation or record or when searching by class. The class representing the preferred term is then automatically substituted for the non-preferred term by the MCRS without the user having to search and find it.

Polyhierarchical relationships between classes in a thesaurus based classification scheme allow users to classify aggregations and records differently to purely hierarchical models, as shown in the following diagram:



For a full explanation of polyhierarchical relationships, see section 8.3.7 of ISO 2788.

202.3 Functional requirements for monolingual thesauri

R202.3.1

When an administrator creates a new class under R1.8.6 the MCRS must allow the administrator to optionally specify one or more active broader term classes for the new class.

Classes without a broader term class are the top term classes within a thesaurus based classification scheme.

R202.3.2

When an administrator creates a new class under R1.8.6 the MCRS must allow the administrator to optionally specify one or more active related term classes for the new class.

A term may have any number of related terms.

R202.3.3

When an administrator creates a new class under R1.8.6 the MCRS must allow the administrator to optionally specify one active preferred term class for the new class, but not if it has a broader term class under R202.3.1 or a related term class under R202.3.2.

A non-preferred term cannot have broader, narrower or related terms.

R202.3.4

The MCRS must allow an administrator to modify the metadata of an active class to add broader terms and related terms subject to R202.3.5 and R202.3.6 respectively.

The administrator cannot add broader and related terms to a non-preferred term.

R202.3.5

When an administrator nominates a class as the broader term of another class under R202.3.1 or R202.3.4 the MCRS must ensure that the administrator can only choose classes that are active, which are not themselves descendants of the class whose metadata is being changed, are not related terms, and where neither is a non-preferred term.

The chain of broader classes cannot be circular. In other words, a class cannot be made the broader or narrower ancestor of itself. An inactive class cannot be made a broader term of an active class.

R202.3.6

When an administrator nominates a class as the related term of another class under R202.3.2 or R202.3.4 the MCRS must ensure that the administrator can only choose classes that are active, where one term is not descended from another, and where neither is a non-preferred term.

A term cannot be a related term to its ancestor or descendent terms. However, two terms that are both descendants of a common ancestor may be made related terms. An inactive term cannot be made a related term.

R202.3.7
When two classes become related terms under R202.3.2 or R202.3.4 then the MCRS must modify the metadata of both classes and add the related term element to both classes.

The two classes will cross-reference each other.

R202.3.8
The MCRS must allow an administrator to modify the metadata of an active class to add an active preferred term, as long as the class has no broader terms, no related terms and no preferred term.

A class may only have one preferred term. A class with a preferred term is also called a non-preferred term.

R202.3.9
The MCRS must ensure that a non-preferred term always has the same aggregation template, record template and disposal schedule allocated to it as its preferred term under R1.8.12, R1.8.14 and R1.8.16 respectively and must remove these from the class when they are removed from the preferred term under R1.8.12, R1.8.14 and R1.8.16.

A non-preferred term is not used directly for classification but must have the same classification status as its preferred term, see R202.3.13. The MCRS must ensure that these entities remain synchronised.

R202.3.10
The MCRS must ensure that a non-preferred term always has the same marking under R1.8.20 and R1.8.21 as its preferred term indicating that it may/may not be applied directly to records.

A non-preferred term is not used directly for classification but must have the same marking as its preferred term, see R202.3.13.

R202.3.11
The MCRS must allow an administrator to modify the metadata of an active class to remove broader terms, related terms and preferred terms, subject to R202.3.12.

R202.3.12
When a related term is removed from a class under R202.3.9 the MCRS must also remove it simultaneously from the class it was previously related to regardless of whether that class is active or inactive.

This requirement is the reverse of R202.3.7.

R202.3.13
The MCRS must allow a user to choose a non-preferred term to classify an aggregation or record but must immediately substitute the preferred term for the non-preferred term in the metadata of the entity.

A user may refer to a class by its non-preferred term but the class must not be used for classification. Only the preferred class may be used for classification.

R202.3.14
The MCRS must allow a user to choose a non-preferred term when searching or reporting by class but must instead search on the preferred term for that class.

A user may refer to a class by its non-preferred term.

R202.3.15
When a user searches or reports by class and selects a term that is the broader term for other terms then the MCRS must return search results based on all entities that are either classified by that term or classified by any narrower term.

A narrower term in this requirement means any descendant of the class down to any generation. In other words the search must return a match on a narrower term as if it were a match directly on the broader term that was used in the search criterion.

R202.3.16
The MCRS must ensure that a user can browse the classification scheme under R1.8.8 by moving between broader and narrower terms, related terms and preferred and non-preferred terms.

The term "browse" is defined in the rationale to R1.5.14 and further explained in the rationale to R1.5.30.

R202.3.17
The MCRS must allow an administrator to swap a non-preferred term with its preferred term if both are active.

The previously preferred term is made the non-preferred term and the non-preferred term is made the preferred term. To do this the MCRS must observe the following procedure:

- *The broader terms of the preferred term are transferred to the non-preferred term;*
- *The related terms of the preferred term are transferred to the preferred term; and*
- *The preferred term is removed from the non-preferred term and a preferred term is added to the previously preferred class.*

Note that the non-preferred term already has the same templates, disposal schedule and marking as the preferred term under R202.3.9 and R202.3.10.

R202.3.18
Whenever an administrator swaps a non-preferred term with its preferred term under R202.3.17 it must also substitute the classes with any active aggregations or active records that are classified with the previously preferred class.

This will result in a cascading change to the metadata of aggregations and records similar to R1.9.15, R1.9.17 and R1.9.18. The classification swap is independent of whether the class is being used as a primary classification or not.

Note that the classification swap does not apply to aggregations and records that are already inactive (because they have been destroyed).

R202.3.19
The MCRS must not prevent an administrator who has initiated an operation to swap a preferred term with a non-preferred term under R202.3.17 from accessing the MCRS, and performing functions on entities other than those participating in the same operation.

The MCRS must not freeze the administrator's access to the system while it carries out the swap.

R202.3.20
The MCRS must ensure that an administrator cannot make a broader term inactive under R1.8.22 unless all of its descendants are already inactive.

An inactive broader term cannot have active narrower terms.

R202.3.21
The MCRS must ensure that an administrator cannot make a class active under R1.8.23 unless its broader term is already active, if any.

See R202.3.20.

R202.3.22
The MCRS must ensure that an administrator cannot make a preferred term inactive under R1.8.22 unless all of its non-preferred terms are already inactive.

An active non-preferred term cannot have an inactive preferred term.

R202.3.23
The MCRS must ensure that an administrator cannot make a non-preferred term active under R1.8.23 unless its preferred term is already active.

See R202.3.22.

R202.3.24
The MCRS must not allow an administrator to delete a broader term under R1.8.29.

The administrator must first move or delete the narrower classes it contains.

R202.3.25

When new classes are added in bulk to a monolingual thesaurus classification scheme, in accordance with R1.8.30, then the datafile must specify the broader/narrower, related and preferred/non-preferred relationships between the new classes to be added to the MCRS, and the relationships between the new classes and any existing classes already in the monolingual thesaurus classification scheme.

MoReq2010 does not specify the format of the datafile.

202.4 Non-functional requirements for monolingual thesauri

N202.4.1

How does the MCRS allow for term substitution under R202.3.13 and R202.3.14 when classifying, searching and reporting?

It is important that the solution to this be elegant in the implementation by the MCRS and that it is apparent to the user what has happened and why.

202.5 Glossary of terms

Term	Explanation and relationship to general concepts
Broader Term	(noun) A **term** in a monolingual **thesaurus** based **classification scheme** which includes the **narrower term**. A term may have more than one broader term which is described by ISO2788 as "polyhierarchical". (element) An **element** holding a reference to a class's parent class.
Narrower Term	(noun) A **term** in a monolingual **thesaurus** based **classification scheme** which is included by the **broader term**.
Non-Preferred Term	(noun) A **term** in a monolingual **thesaurus** based **classification scheme** which represents a synonym or quasi-synonym of another term. *Non-preferred terms cannot be used for classification; the equivalent preferred term is used instead.*
Preferred Term	(element) A **term** in a monolingual **thesaurus** based **classification scheme** which is used to classify an entity and search for entities in place of the **non-preferred term**. (element) An **element** holding a reference to a class's parent class.
Related Term	(element) A **term** in a monolingual **thesaurus** based **classification scheme** which is related to the term. Relationships between related terms are used to give context and to allow the user to traverse the classification scheme other than from top to bottom. There are not used for classification. (element) An **element** holding a reference to a class's parent class.
Term	(noun) An alternative name for a **class** in a monolingual **thesaurus** based **classification scheme**.
Top Term	(noun) A **term** in a monolingual **thesaurus** based **classification scheme** with no **broader term**.

202.6 Table of function definitions

Reference	System identifier	Title	Suitable for Bulk Operation?	Applies to Entity Types
F202.6.1	4e3227d3-a663-4738-b651-f2b6f2a6244b	Add broader term	No	Classes
F202.6.2	6d27c8f2-94cb-4ac4-aca6-ff9ccb93f6e8	Add preferred term	No	Classes
F202.6.3	f64cdebc-cc04-4ab2-b947-74f6b938a2d7	Add related term	No	Classes
F202.6.4	f5cc118e-b551-40bd-9c60-b686a28e7d66	Remove broader term	No	Classes
F202.6.5	5e11612c-ef59-4c6c-8a36-e042901cac55	Remove preferred term	No	Classes
F202.6.6	b17dc7a5-0af3-4901-9b4f-7529b38e11e1	Remove related term	No	Classes
F202.6.7	93b416eb-477a-419c-8fa9-51609428c9f5	Swap terms	No	Classes, Aggregations and Records

202.7 Table of element definitions

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M202.7.1	9107eeaa-c6b8-4c0f-adb7-6f24a3170772	Broader term	0..Many	Class reference	Class
M202.7.2	fddef3fb-ffc4-44d0-9d65-e5dddbac98d6	Preferred term	1..1	Class reference	Class
M202.7.3	49508b67-47e2-4fb6-9a2b-64b2c27a499d	Related term	0..Many	Class reference	Class

202.8 Table of general codes

Reference	System identifier	Value	Applies to Element
C202.8.1	47ed014b-58db-4626-963a-4cfa9d5951a9	MONOLINGUAL THESAURUS	Classification method

300 Series – Component types

301 Managed Components

301.1 Important information

301.1.1 Identifier

94324fe3-cafa-4ed1-a621-60e947ab59ef

301.1.2 Title

MoReq2010 – Requirements for managed components

301.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the requirements for all compliant records systems that implement managed components.

301.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

301.1.5 Prerequisites and co-requisites

The MCRS must be compliant with the following modules of MoReq2010:

- 1. Core requirements

The MCRS may be compliant with the following modules of MoReq2010:

- none

301.1.6 Intellectual Property Rights

301.1.7 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

301.1.8 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

301.2 Key concepts

Traditional electronic records systems preserve digital datafiles by making copies of them and storing the copies inside a repository controlled by the records themselves. Where this occurs, under MoReq2010 these datafiles are termed managed components. Records with managed components are said to be "captured" into the records system.

Managed components have the following characteristics:

- They are wholly self-contained and do not have dependencies on any resources stored outside themselves or the other components in the same record;
- To access the component the user must obtain a copy of it from the records system repository; and
- The component can be automatically deleted by the records system when it is destroyed.

The following assumptions must also be made to use managed components:

- The records can be successfully captured as one or several datafile components;
- The original datafile for the component and any copies made by users when accessing the managed component are managed by systems or policies external to the records system;
- In copying the component from another business system enough of its original context will be captured with it to identify it later;
- It is of a type that does not require external resource dependencies; and
- It is in a format that can meaningfully be used or understood in the future when accessed in isolation from the records system.

This module contains functional requirements and non-functional requirements for the use of managed components.

301.3 Functional requirements for managed components

R301.3.1

When a managed component is captured under R1.10.8 the MCRS must copy the original datafile into its own managed repository.

The MCRS must maintain a managed repository of component datafiles and ensure that they are not modified or modifiable except by the MCRS itself.

R301.3.2

When a managed component is captured under R1.10.8 the MCRS must set the component type under R1.10.9 to MANAGED COMPONENT.

See the code in the table at 301.7.

R301.3.3

When a managed component is captured under R1.10.8 the MCRS must include its original datafile name without extension and the datafile extension in the metadata of the component.

The datafile extension is a suffix to the datafile name that is separated by a full stop from the rest of the datafile name but does not itself contain a full stop (Unicode character U+002E). For example, in the datafile, "MoReq2010 v1.1.pdf" the datafile name is "MoReq2010 v1.1" and the datafile extension is "pdf".

R301.3.4

When a managed component is captured under R1.10.8 the MCRS must include its datafile size in bytes in the metadata of the component.

The datafile size should reflect the size of the component when captured.

R301.3.5

When a managed component is captured under R1.10.8 the MCRS must extract from the datafile, the timestamp it was created on and the timestamp it was last modified on and store these in the metadata of the component.

This will depend on the nature of the operating system and how the datafile is transmitted to the MCRS.

R301.3.6

When a managed component is captured under R1.10.8 the MCRS must always set the metadata of the component under R1.10.13 to indicate that the component is accessible by the MCRS as a datafile.

The datafile is accessible from the repository of the MCRS.

R301.3.7

When a managed component is captured under R1.10.8 the MCRS must always set the metadata of the component under R1.10.15 to indicate that the component can be automatically deleted.

R301.3.8

When a user retrieves a managed component under R1.10.23 or R1.10.24 then the MCRS must enable the user to obtain an exact copy of the datafile held in its repository.

The term "retrieve" is defined in the rationale to R1.10.23.

R301.3.9

When a managed component is destroyed under R1.12.50 the MCRS must ensure that all copies of its datafile, including back up copies, are deleted and overwritten.

See non-functional requirements N301.4.5 and N301.4.6.

301.4 Non-functional requirements for managed components

N301.4.1

What business systems and applications does the MCRS capture managed components from?

The MCRS may be acquired as a dedicated system closely integrated or built into a business system or application, such as corporate email, or it may be a general purpose system with a wide number of integrations.

N301.4.2

What datafile formats does the MCRS recognise and what component templates does it support under R1.10.19 for extracting embedded metadata from datafiles?

See also the rationale to R1.10.19

N301.4.3

How does the organisation manage the original datafiles of managed components after they have been copied into the repository of the MCRS?

For example, are old emails purged from the email system or old documents purged from shared and personal drives, and so on?

N301.4.4

How does the organisation manage copies of datafiles of managed components after they have been retrieved by users from the repository of the MCRS, under R301.3.8?

Despite the tight control of datafiles by the MCRS within its own repository the organisation may still need to deal with proliferation of components outside its sphere of influence.

N301.4.5

What mechanism does the MCRS use for overwriting datafiles when they have been destroyed under R301.3.9?

Many operating systems do not overwrite the binary data of a datafile when it is deleted from storage media. The footprint previously occupied by the datafile in the repository of the MCRS must be overwritten to prevent the datafile being wholly or substantially recovered, for example, from close analysis of magnetic disc surfaces.

N301.4.6

What time period is acceptable to the organisation for removing all copies of a destroyed component's datafile from back up media?

If the MCRS uses traditional child–parent-grandparent back up strategies for its repository then datafiles belonging to destroyed components may be retained for months or even years as part of a "grandparent" back up. Alternatively if the MCRS provides

mirrored storage or similar architectures for data redundancy then all copies of a datafile can be deleted near simultaneously.

See also business continuity issues surrounding restored data sets in the non-functional requirements in section 1.16.

301.5 Glossary of terms

Term	Explanation and relationship to general concepts
Managed Component	(entity) A **component** whose corresponding **datafile** is stored by the **MCRS** in its **repository**. *See also **component**.*
Repository	(noun) A secure data store containing **datafiles** that match **components** which is managed and wholly controlled by the **MCRS**. *Traditional electronic records systems copy all digital records to a repository under their control.*

301.6 Table of element definitions

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M301.6.1	6fbabc80-f27a-4b25-b429-d6ed72a09987	Datafile created	1..1	Timestamp	Component
M301.6.2	7cc9d619-cf17-4b19-a0da-e570909457e4	Datafile extension	1..1	Text	Component
M301.6.3	395f06b8-6228-4ffe-bd4a-4cf029f6739d	Datafile name	1..1	Text	Component
M301.6.4	b00617f9-cf90-4785-b56e-da26484ffa2e	Datafile size	1..1	Numeric	Component
M301.6.5	f32168eb-dfb0-466d-b92c-51ed0c4a4b8a	Datafile updated	1..1	Timestamp	Component

301.7 Table of general codes

Reference	System identifier	Value	Applies to Element
C301.7.1	b2848652-88cd-45e8-9077-d6dcb062c212	MANAGED COMPONENT	Component type

302 Unmanaged Components

302.1 Important information

302.1.1 Identifier

57b79695-cb4a-484c-867e-c1bdfd0c09d1

302.1.2 Title

MoReq2010 – Requirements for unmanaged components

302.1.3 Description

MoReq2010 - Model Requirements for records systems. Contains the requirements for all compliant records systems that implement unmanaged components.

302.1.4 Version

0.92 DRAFT FOR PUBLIC CONSULTATION

302.1.5 Prerequisites and co-requisites

The MCRS must be compliant with the following modules of MoReq2010:

- 1. Core requirements

The MCRS may be compliant with the following modules of MoReq2010:

- none

302.1.6 Intellectual Property Rights

302.1.7 Requirements

Each requirement in MoReq2010 may be followed by an explanatory rationale in italics. The rationale is intended to provide clarity and amplification for the requirement itself.

All requirements are mandatory. Functional requirements are prefixed by the letter "R", non-functional requirements are prefixed by the letter "N".

302.1.8 Prefixes

The following prefixes are used for reference purposes throughout MoReq2010:

- **A** – Alert type definition;
- **C** – Codes other than disposal schedule codes;
- **D** – Disposal schedule code;
- **E** – Entity type definition;
- **F** – Function definition;
- **M** – Metadata element definition;
- **N** – Non-functional requirement;
- **R** – Requirement (functional); and
- **T** – Test case.

Note that these reference numbers are for document look up purposes only and are relative to a specific minor version of MoReq2010 and may change in any future major or minor version. Records systems and other applications should refer to the universally unique identifiers provided for each type definition and not the reference numbers above.

302.2 Key concepts

Unmanaged components are all those record components that are not captured and stored as datafiles by an MCRS within its own repository. They can include:

- Datafiles stored in other business systems, applications and on shared storage (this is sometimes described as to manage records in place);
- Transactional information, sometimes called "structured" records, stored in database tables in financial and other business systems;
- External digital resources that can be individually identified or located, for example with a URI;
- Physical documents, including paper records; and
- Information stored on physical media, including CDs, DVDs, tapes and microform.

It is important to note that an MCRS may be built for a specific purpose to do with only one of these kinds of unmanaged component. MoReq2010 does not require that an MCRS be able to manage all possible types of unmanaged component in order to comply with this module.

Dealing with unmanaged components often requires that the MCRS have some kind of integration with another business system or application. Where this occurs, the MCRS may be able to do one or some of the following:

- Index the unmanaged component or have access to another system's full text index of the component;
- Redirect users to access the unmanaged component in another system by viewing or downloading it;
- Be able to retrieve the component's datafile from the other system and forward it on to the user on request; or
- Be able to initiate the deletion of the managed component automatically through an integration with the other system.

At the very least the MCRS must be able to describe the physical or virtual location of the unmanaged component so that a user can access it independently and so that it can be destroyed by an administrator.

302.3 Functional requirements for unmanaged components

R302.3.1

When an unmanaged component is declared under R1.10.8 the MCRS must set the component type under R1.10.9 to UNMANAGED COMPONENT.

See the code in the table at 302.8.

R302.3.2

When an unmanaged component is declared under R1.10.8 the MCRS must include a textual component sub-type description in the metadata of the component.

There are many different types of unmanaged component, including paper documents, DVDs, etc. There is no complete listing of all possible unmanaged components. The component sub-type is intended to provide explanatory text describing the nature and form of the component.

R302.3.3

When an unmanaged component is declared under R1.10.8 the MCRS must attempt to collect the following information and include it in the metadata of the component:

- System of origin – this could be the business system or application containing the component or organisation or business process that has produced it;
- Date of origin – this could be the date of the component in the system of origin;
- Originator – this could be the author or authors of the component; and
- Originator's reference – this could be the original reference to the component in the system of origin.

This metadata is purely descriptive and not necessary for processing the component. The meaning of this metadata will depend on the specific MCRS implementation and its integration with external business systems. As a general rule the MCRS should capture as much contextual information about the component as possible.

R302.3.4

When an unmanaged component is declared under R1.10.8 the MCRS must determine if it is able search it through full text searching and set the component flag under R1.10.11 accordingly.

The MCRS may be able to search the content of an unmanaged component because:

- *The business system is able to provide the component as a datafile, see R302.3.5;*
- *The business system is able to provide a consumable index of the text in the component; or*
- *The MCRS and the business system both utilise the same enterprise search engine.*

R302.3.5
When an unmanaged component is declared under R1.10.8 the MCRS must determine if it is able to retrieve the component as a datafile from the system of origin on request, and set the datafile accessibility flag under R1.10.13 appropriately.

This means that the datafile which is not managed by the MCRS can be retrieved by integration with the system of origin and served to users.

R302.3.6
When an unmanaged component is declared under R1.10.8 the MCRS must always ensure that the component has a location and external identifier set under R1.10.14 regardless of whether the datafile accessibility file is set under R1.10.13.

Even when the MCRS can retrieve the unmanaged component it should also store the component's location and external identifier.

R302.3.7
When the datafile accessibility flag of an unmanaged component is set under R1.10.13 and a user requests the datafile under R1.10.23, then the MCRS must allow the user to retrieve it from the system of origin as if it came from the MCRS.

The term "retrieve" is defined in the rationale to R1.10.23.

R302.3.8
If an MCRS attempts to retrieve a datafile from its system of origin under R1.10.23 because the datafile accessibility flag has been set under R302.3.7 and the integration fails for any reason, then the MCRS must raise an alert.

This can happen if the system of origin is offline or fails to locate and respond with the content of the unmanaged component.

R302.3.9
When an unmanaged component is declared under R1.10.8 the MCRS must determine if it is integrated with the system of origin and able to delete the component from the system of origin on request, and set the automatic deletion flag under R1.10.15 appropriately.

This requires a system to system integration between the MCRS and the system of origin of the unmanaged component.

R302.3.10
If an MCRS attempts to destroy a datafile under R1.12.50 because the automatic deletion flag has been set under R302.3.9 and the integration fails for any reason, then the MCRS must raise an alert.

This can happen if the system of origin is offline or fails to locate or delete the content of the unmanaged component.

R302.3.11
When the MCRS raises an alert under R302.3.8 or R302.3.10 it must include extended error information in the Comment of the alert.

See R1.5.42.

R302.3.12
The MCRS must allow an administrator to change the status of an active unmanaged component to inactive to indicate that it has been lost or deleted from the system of origin.

Because unmanaged components are not controlled by the MCRS it is possible that they will be deleted accidentally or as a result of system error. Physical components may also be lost or misplaced.

R302.3.13
When an unmanaged component belonging to an active record is made inactive under R302.3.12 the MCRS must not delete any of the metadata or event history of the unmanaged component, until the record to which the component belongs is later destroyed under R1.12.50.

Making an unmanaged component inactive when its associated record remains active is not the same as destroying the component under R1.12.50. It is intended only to signify that the content linked to by unmanaged component is no longer available in the system of origin.

R302.3.14
When an unmanaged component belonging to an active record is made inactive under R302.3.12 the MCRS must allow the administrator to include a textual inactive comment in the metadata of the component to explain the reason why it has been made inactive.

It must be clear to a user of the MCRS why a component of an active record has been made inactive.

R302.3.15
The MCRS must allow an administrator to change the status of an inactive unmanaged component to active to indicate that it has been found or restored, if the record to which the component belongs is still active.

It is only possible to make an inactive unmanaged component active if the record it belongs to is active and has not already been destroyed. Making the component active reestablishes the connection between the MCRS and the system of origin in managing the record.

R302.3.16
When an administrator makes an inactive component active under R302.3.15 the MCRS must automatically remove any inactive comment given to the component under R302.3.14.

Leaving the inactive comment in place would be misleading. It will still be available through the event history of the record.

R302.3.17

The MCRS must allow an administrator to select a number of unmanaged components and perform any of the following functions as a bulk operation:

- Make all selected unmanaged components inactive simultaneously, see R302.3.12.
- Make all selected unmanaged components active simultaneously, see R302.3.15.

The terms "select" and "bulk operation" are explained in the rationale to R1.5.43. Requirements for bulk operations can be found starting at R1.6.10.

R302.3.18

When an administrator makes selected unmanaged components inactive simultaneously as a bulk operation under R302.3.17 the MCRS must allow the administrator to apply the same inactive comment simultaneously to all the selected unmanaged components.

See R302.3.14.

302.4 Non-functional requirements for unmanaged components

N302.4.1

What types of unmanaged component does the MCRS support?

The MCRS may be intended for a vertical market and only support a very specific type of unmanaged component. Alternatively the MCRS may provide wide ranging support for a number of different unmanaged component types.

N302.4.2

What business systems and applications does the MCRS interface to and provide integrated support for unmanaged components?

The supplier must provide information about the business systems and applications and the types of unmanaged components that are supported by the MCRS as part of certification testing, which will then be included in the published test report for the MCRS.

N302.4.3

What component templates does the MCRS support for unmanaged components under R1.10.19 that allow the extraction of metadata about the unmanaged component from its system of origin?

See also the rationale to R1.10.19

N302.4.4

What safeguards has the supplier or organisation put in place to ensure that unmanaged components cannot be deleted or modified once they have been declared as records in the MCRS?

Records and their components must be unalterable. If unmanaged components are located outside the control of the MCRS then the organisation must employ safeguards to ensure the integrity of the component is preserved in its environment. Safeguards may take the form of security measures, organisational policies and operational procedures.

N302.4.5

How does the organisation manage copies of datafiles of managed components after they have been retrieved by the MCRS from the system of origin, under R302.3.7?

The organisation should discourage the proliferation of datafiles belonging to corporate records that might outlive the records themselves.

N302.4.6

How does the organisation ensure that records are deleted from their system of origin if the MCRS is unable to instigate such deletion under R302.3.9?

Unmanaged components must be deleted from the system of origin and all back up copies destroyed as well. This must be confirmed by an administrator if the unmanaged components cannot be deleted automatically.

N302.4.7

What policies and procedures does the organisation provide for administrators that are advised by alert that components are unable to be found or failed to be deleted from their systems of origin under R302.3.8 or R302.3.10?

These issues may have to be addressed manually by an administrator in the system of origin.

N302.4.8

How will the organisation ensure that the MCRS and the system of origin are properly synchronised if one or the other has to be restored from back up following a disaster recovery?

If the MCRS and the system of origin are out of synch then the MCRS may have unmanaged component entities and records declared as coming from the system of origin which no longer have any equivalent in the system of origin. Alternatively the system of origin may have data for records which have previously been declared in the MCRS but will not be able to be found there because the MCRS has been rolled back.

302.5 Glossary of terms

Term	Explanation and relationship to general concepts
Unmanaged Component	(entity) A **component** whose artefact is physical, including paper records, or is digital but stored on physical media, such as CDs, DVDs, tape and microform, or is stored by another business system or application not directly under the management of the **MCRS**. *See also* ***component***.

302.6 Table of alert types

Reference	System identifier	Title
A302.6.1	d409e2c2-5551-43bd-868c-4589897566ad	Failed to delete unmanaged component
A302.6.2	8fcd53ee-3c24-4a63-b96e-ed0800d7f5a5	Failed to retrieve unmanaged datafile

302.7 Table of element definitions

Reference	System identifier	Title	Cardinality	Data Type	Applies to Entity Types
M302.7.1	ea094107-d166-443e-b95c-755dd6ad097c	Component sub-type	1..1	Text	Component
M302.7.2	ae85d800-055b-43ef-8c4b-58dab22b30d8	Date of origin	0..1	Date	Component
M302.7.3	1f8d8248-b4c3-4ab7-ba0d-d0c7ac2d3008	Inactive comment	0..1	Text	Component
M302.7.4	cb9218a4-f0cb-4c0f-a002-4eb1f4ed3903	Originator	0..1	Text	Component
M302.7.5	11284819-952d-4113-bf43-bc5c55ae4642	Originator's reference	0..1	Text	Component
M302.7.6	df252418-ea43-415d-b985-764ececbad8b	System of origin	0..1	Text	Component

302.8 Table of general codes

Reference	System identifier	Value	Applies to Element
C302.8.1	d239c185-dace-425b-b158-401ebf8c6269	UNMANAGED COMPONENT	Component type