INTERNATIONAL STANDARD

ISO 16100-6

First edition
2011-06-15

Industrial automation systems and integration — Manufacturing software capability profiling for interoperability —

Part 6: Interface services and protocols for matching profiles based on multiple capability class structures

Systèmes d'automatisation industrielle et intégration — Profil d'aptitude du logiciel de fabrication pour interopérabilité —

Partie 6: Services d'interface et protocoles pour la correspondance des profils fondés sur les structures de classe d'aptitude multiple

Reference number
ISO 16100-6:2011(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 16100-6 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Interoperability, integration, and architectures for enterprise systems and automation applications*.

ISO 16100 consists of the following parts, under the general title *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability*:

— *Part 1: Framework*

— *Part 2: Profiling methodology*

— *Part 3: Interface services, protocols and capability templates*

— *Part 4: Conformance test methods, criteria and reports*

— *Part 5: Methodology for profile matching using multiple capability class structures*

— *Part 6: Interface services and protocols for matching profiles based on multiple capability class structures*

Introduction

The motivation for ISO 16100 stems from the industrial and economic environment, in particular:

a) a growing base of vendor-specific solutions;

b) user difficulties in applying standards;

c) the need to move to modular sets of system integration tools;

d) recognition that application software and the expertise to apply that software are assets of the enterprise.

ISO 16100 is an International Standard for the computer-interpretable and human-readable representation of a capability profile. Its goal is to provide a method to represent the capability of manufacturing application software relative to its role throughout the life cycle of a manufacturing application, independent of a particular system architecture or implementation platform. This can lead to reduced production and information management costs to users and vendors/suppliers of manufacturing applications.

Certain diagrams in this part of ISO 16100 are constructed following UML conventions. Because not all concepts embodied in these diagrams are explained in the text, some familiarity with UML on the part of the reader is assumed.

Industrial automation systems and integration — Manufacturing software capability profiling for interoperability —

Part 6: Interface services and protocols for matching profiles based on multiple capability class structures

WARNING — This part of ISO 16100 provides a specification intended to be implemented in software. Incompatibilities may result in machine-to-machine communication in the case of software developed on the basis of translations of this part of ISO 16100 into languages other than the official ISO languages. Accordingly, it is strongly recommended that any implementations be developed only on the basis of the texts in the official ISO languages.

1 Scope

This part of ISO 16100 defines the detailed interface services and protocols used in a matching method based on multiple capability class structures. This part of ISO 16100 also defines a Capability Profile Template Interface (CPTI) Service Group, an Extended Capability Profile Interface (CPI) Service Group and an Extended Matcher Interface Service Group, which are extensions of the Type 1, Type 2 and Type 3 services, respectively, as specified in ISO 16100-3:2005, 5.4.

This part of ISO 16100 also defines the Capability Class Structure Interface (CCSI) Service Group, an additional service group used to create, register, access and modify a capability class structure for the reference manufacturing domain models, as specified in ISO 16100-5:2009, Clause 6.

This part of ISO 16100 also specifies detailed contents of the specific part of a capability profile template as defined in ISO 16100-5:2009, Clause 7.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 16100-1, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 1: Framework*

ISO 16100-2, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 2: Profiling methodology*

ISO 16100-3, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 3: Interface services, protocols and capability templates*

ISO 16100-4:2006, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 4: Conformance test methods, criteria and reports*

ISO 16100-5:2009, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 5: Methodology for profiling matching using multiple capability class structures*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 16100-1, ISO 16100-2, ISO 16100-3, ISO 16100-4 and ISO 16100-5 and the following apply.

3.1
capability class
〈software unit capability class〉 element within the capability profiling method that represents software unit functionality and behaviour with regard to the software unit's role in a manufacturing activity

NOTE 1 The role of a Manufacturing Software Unit (MSU) changes when used in different manufacturing activities; however, the MSU's corresponding capability class is positioned uniquely in an inheritance structure, but it can assume different positions in an aggregation structure.

NOTE 2 In this part of ISO 16100, a capability class template is identical to a capability profile template (see ISO 16100-2:2003, 6.3, for requirements for capability templates).

NOTE 3 Adaped from ISO 16100-2:2003, definition 3.3.

NOTE 4 In general, a capability class maps to an activity. The capability class is distinct within a capability inheritance structure and can form a capability aggregation structure with other capability classes.

3.2
capability class structure
hierarchy of capability classes

NOTE This structure is intended for modeling capability aggregation hierarchies in the target domains of ISO 16100-1:2009, Figure 2.

3.3
capability class structure template
extensible markup language (XML) schema representing a capability class structure

NOTE Adapted from ISO 16100-5:2009, definition 3.2.

3.4
capability profile template
schema for a manufacturing software capability profile

3.5
extended service interface
set of service access points defined in this part of ISO 16100 that handle manufacturing domain data, manufacturing domain models, capability class structures, capability profiles and capability profile templates

NOTE “Extended” refers to both the services specified in this part of ISO 16100 and the “basic” services specified in ISO 16100-3.

3.6
manufacturing domain data
information, represented by a unified modeling language (UML) class, about manufacturing resources, manufacturing activities, or items exchanged among manufacturing resources within a particular manufacturing domain

NOTE Adapted from ISO 16100-5:2009, definition 3.3.

3.7
manufacturing domain data template
extensible markup language (XML) schema representing a manufacturing domain data

[ISO 16100-5:2009, definition 3.4]

3.8
manufacturing domain model
particular view of a manufacturing domain, consisting of manufacturing domain data and relationships among them, corresponding to the domain's applications

[ISO 16100-5:2009, definition 3.5]

3.9
parts library
⟨manufacturing⟩ collection of part descriptions or catalogue

NOTE The term “parts library” also refers to a dictionary such as a PLIB dictionary in ISO 13584 or OTD in ISO 22745.

4 Symbols and abbreviated terms

BSU Basic Semantic Unit

CCS Capability Class Structure

CCSI Capability Class Structure Interface

CPI Capability Profile Interface

CPTI Capability Profile Template Interface

CSI Conformance Statement for the Implementation

ESI Extended Service Interface

ESP Extended Service Provider

ICD International Code Designator

MDD Manufacturing Domain Data

MDM Manufacturing Domain Model

MSU Manufacturing Software Unit

OTD Open Technical Dictionary

PLIB Parts Library (as specified in ISO 13584)

UML Unified Modeling Language

URL Uniform Resource Locator

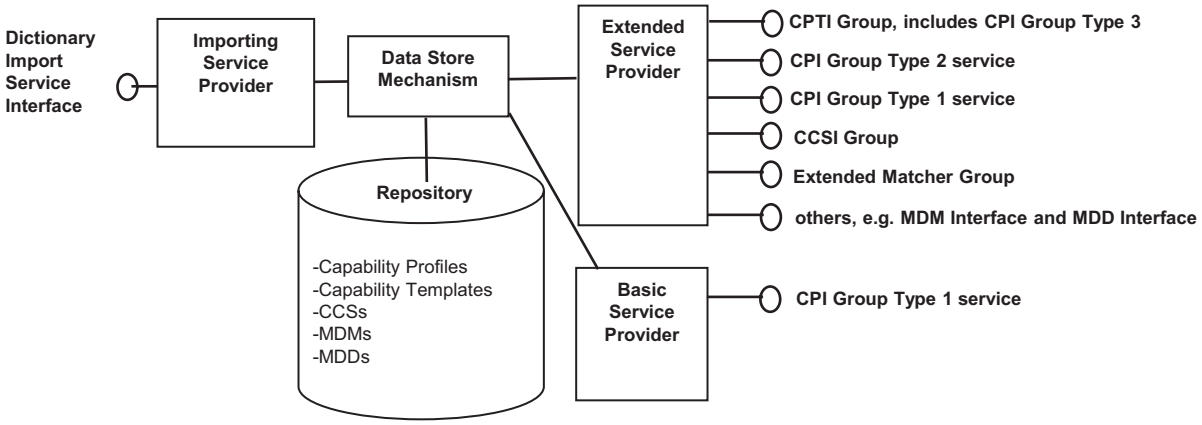
URN Uniform Resource Name

XML eXtensible Markup Language

5 Service provider interface services

5.1 Service sets

Figure 1 shows all the services, and their relations to Extended Service Providers and Basic Service Providers, for handling capability profiles, capability profile templates, CCSs, MDMs, MDDs and MDD objects. Basic Service Providers handle CPI group Type 1 services. Extended Service Providers handle CPTI, CPI and CCSI services. In addition, Extended Service Providers support extended matcher services and other service interfaces for handling MDMs and MDDs.



NOTE 1 This figure is not in accordance with UML conventions. The line between the Data Store Mechanism and the Repository represents the rules for adding, removing and changing contents of the Repository. The line between the Data Store Mechanism and the Extended Service Provider represents a mapping of the extended services to the Data Store services. The mapping is typically implementation-specific and therefore not part of the scope of this part of ISO 16100.

NOTE 2 The boldfaced elements in this figure are specifically addressed in this part of ISO 16100.

NOTE 3 The contents in the Repository are stored as XML files.

NOTE 4 The ESI access point is represented elsewhere in this part of ISO 16100 by the object *ServiceAccessPoint*.

NOTE 5 The Type 1 CPI service group, which is briefly described in ISO 16100-3:2005, 5.4, includes Type 1 matcher service.

NOTE 6 The Type 2 CPI service group, which is briefly described in ISO 16100-3:2005, 5.4, does not include Type 2 matcher services, which are part of the Extended Matcher Group.

NOTE 7 The Type 3 CPI service group is briefly described in ISO 16100-3:2005, 5.4.

Figure 1 — Extended Service Provider service sets

All services have the following characteristics:

a) when a service is conducted, there is one service provider and one service user, and no other third party is involved;

b) the service user initiates all service invocations, which are distinct from the lower communications layer service invocations;

c) a service user invocation is always accompanied by a response from the service provider;

d) any service user invocation and the corresponding response(s) are conducted in a bounded time frame, as determined by the service user or the service provider or both;

e) a service invocation at a service access point is processed when a response to a prior service invocation is completed;

f) an invocation is made for a single service; there is no invocation for a service group;

g) a service to register additions and updates to the repository uses the data store mechanism in Figure 1;

h) the state of an object in the repository is one of the following:

   1) stored: an object is stored in the repository after a creation request;

   2) registered: an object is registered into the repository after it is conformance tested;

   3) deleted: an object is deleted from the repository after a deletion request.

5.2 ESI service set

The generic ESI services provided by an ESP can be organized in specific combinations into a CPI Group Type 1 (see ISO 16100-3) and the four service groups listed below, which are described in more detail in Clause 6. Other service groups can exist, e.g. MDM, MDDs and MDD objects, but are not defined in this part of ISO 16100.

a) The CPTI Group, which includes CPI Group Type 3, allows the following services:

   1) create a new capability profile template;

   2) access a capability profile template;

   3) modify a capability profile template;

   4) conformance test a capability profile template;

   5) register an MSU capability profile;

   6) delete an MSU capability profile.

b) The CPI Group Type 2 (see ISO 16100-3) allows the following services:

   1) create a new MSU capability profile or a new requirement capability profile;

   2) access an MSU capability profile or a requirement capability profile;

   3) modify an MSU capability profile or a requirement capability profile;

   4) conformance test a capability profile;

   5) register a requirement capability profile or a requirement capability profile;

   6) delete a requirement capability profile or a requirement capability profile.

c) The CCSI Group allows the following services:

   1) create a new capability class structure;

   2) access a capability class structure;

   3) modify a capability class structure;

4) conformance test a capability class structure;

5) register a capability class structure;

6) delete a capability class structure.

d) The Extended Matcher Group allows the following services:

1) access an MSU capability profile or a requirement capability profile;

2) match two capability profiles, each referencing a different capability class structure.

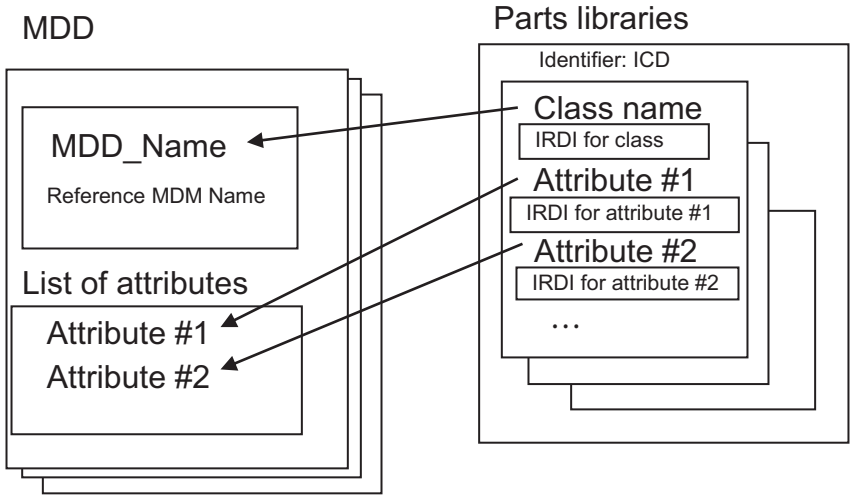
5.3 Dictionary Import Service Interface

5.3.1 Parts libraries imported to a repository

The Dictionary Import Service Interface enables the importing of a library (e.g. PLIB dictionary or OTD) to a repository.

5.3.2 Relationship between parts libraries and MDDs

A parts library imported to a repository can be used as part of MDDs in capability profiling. MDDs in an MDM include the definition of manufacturing activities. Since a PLIB dictionary and an OTD includes definitions of technical terms, it can be used in capability profiling in the same manner as MDDs are used. All classes and attributes in a PLIB dictionary or OTD can be identified by an unambiguous identifier in accordance with ISO 29002-5 and ISO 22745-13. The unambiguous identifier is a form of international registration data identifier (IRDI) as specified in ISO/IEC 11179-5. Classes and attributes within a PLIB dictionary can also be specified by a combination of a dictionary and a BSU, a code for each PLIB class and an attribute internal to the dictionary. A BSU for an attribute in a PLIB dictionary is the combination of the attribute code and the BSU of the parent class. For an OTD, it is not necessary to reference the BSU of the parent class, since ISO 22745 is classification-neutral and properties are defined independently of classes in an OTD. The relationship between a PLIB dictionary or OTD and an MDD is determined by mapping the unambiguous identifier for each MDD and attribute of an MDD, as shown in Figure 2.



NOTE 1 This figure is not in accordance with UML conventions.

NOTE 2 In ISO 13584 the term “property” is used instead of “attribute”.

Figure 2 — Relationship between parts libraries and MDDs

5.3.3 Mapping from a PLIB element to an MDD

The element “MDD_Name” in Figure 2 shall have the following extended attributes:

— “dictionary id”,

— “parent”,

— “BSU”,

— “version”, and

— “revision”.

The attribute set of “dictionary id”, “parent” and “BSU” can be used as an identifier of the MDD.

According to Figure 2, attributes in an MDD refer to attributes of items in a PLIB dictionary. An MDD corresponds to an element in a PLIB dictionary. Attributes belonging to the same element would be associated with one MDD. See Annex E.

6 Extended Service Interface

6.1 CPTI Group services

6.1.1 Scenarios handled by the CPTI Group

The CPTI Group handles the following capability profile template scenarios:

a) create a capability profile template for a specific class structure, either by partially filling the generic formal structure of the capability profile template or by modifying an existing capability profile template with a new capability profile template ID using the MDDs in the MDM, and then receive a capability profile template from a service provider;

b) request a capability profile template from a repository based on a capability profile template ID and then receive a capability profile template from a service provider;

c) modify a capability profile template, according to either user requirements or the results from a conformance test, and receive a modified capability profile template from a service provider;

d) test a capability profile template against the capability profile template conformance criteria found in ISO 16100-5:2009, Clause 8, and receive either a positive, negative, or matching level response from a service provider;

e) register a tested capability profile template into a repository and receive a “registered” status from the service provider;

f) delete a capability profile template based on a capability profile template ID and receive a “deleted” status from a service provider.

6.1.2 Capability profile template creation

6.1.2.1 Creation process

The process for creating a capability profile template for a capability class of a particular capability class structure consists of configuring a generic formal structure of a capability profile template by either

a) filling in specific values for certain attributes of certain elements in the generic formal structure of the capability profile template as defined in ISO 16100-5:2009, 6.3, or

b) modifying previously filled-in values in an existing capability profile template.

6.1.2.2 Capability profile template configuration

A generic formal structure of the capability profile template may be either partially or completely filled according to the application requirement.

For any capability profile template, the following attributes and elements shall be filled in or modified:

a) attributes “id” and “name” in the element “Capability Profile Template”;

b) attribute “domain_Name” in the element “Reference_MDM_Name”;

c) attribute “format_name” in the element “MDD_Description_Format” with one of the following four values:

   1) “Set_Of_MDD_Objects”;

   2) “List_Of_MDD_Objects”;

   3) “Time_Ordered_MDD_Objects”;

   4) “Event_Ordered_MDD_Objects”;

d) attributes “name” and “action” in the element “MDD_Name”.

Each value filled in or modified for the attributes “id” in 6.1.2.2 a), “domain_Name” in 6.1.2.2 b) and “name” in 6.1.2.2 d) shall be unique.

A capability profile template shall be used to create a capability profile associated with a capability class.

6.1.2.3 *createTemplate* service

6.1.2.3.1 Template based on a generic formal capability structure

The *createTemplate* service shall allow a template user to create a template based on a generic formal capability structure. When creating a template based on a formal capability structure, the *createTemplate* service uses at a minimum the *requestBlankTemplate*, *returnBlankTemplate*, *processFilledTemplate* and *returnProcessingResult* services. The *createTemplate* service shall consist of the following steps:

a) the template user invokes the *requestBlankTemplate* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the *requestBlankTemplate*;

b) the service provider invokes the *returnBlankTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnBlankTemplate* service are blank template and creation error;

c) the template user fills in the blank template using MDDs of the MDM and then invokes the *processFilledTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *processFilledTemplate* service is template ID;

d) the service provider checks the uniqueness of the template ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 3, above the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a template from a formal template structure.

6.1.2.3.2 Template created by modifying an existing capability profile template

The *createTemplate* service shall allow a template user to create a template modified from an existing template. When creating a template modified from an existing template, the *createTemplate* service uses at a minimum the *requestExistingTemplate*, *returnExistingTemplate*, *processModifiedTemplate* and *returnProcessingResult* services. The *createTemplate* service shall consist of the following steps:

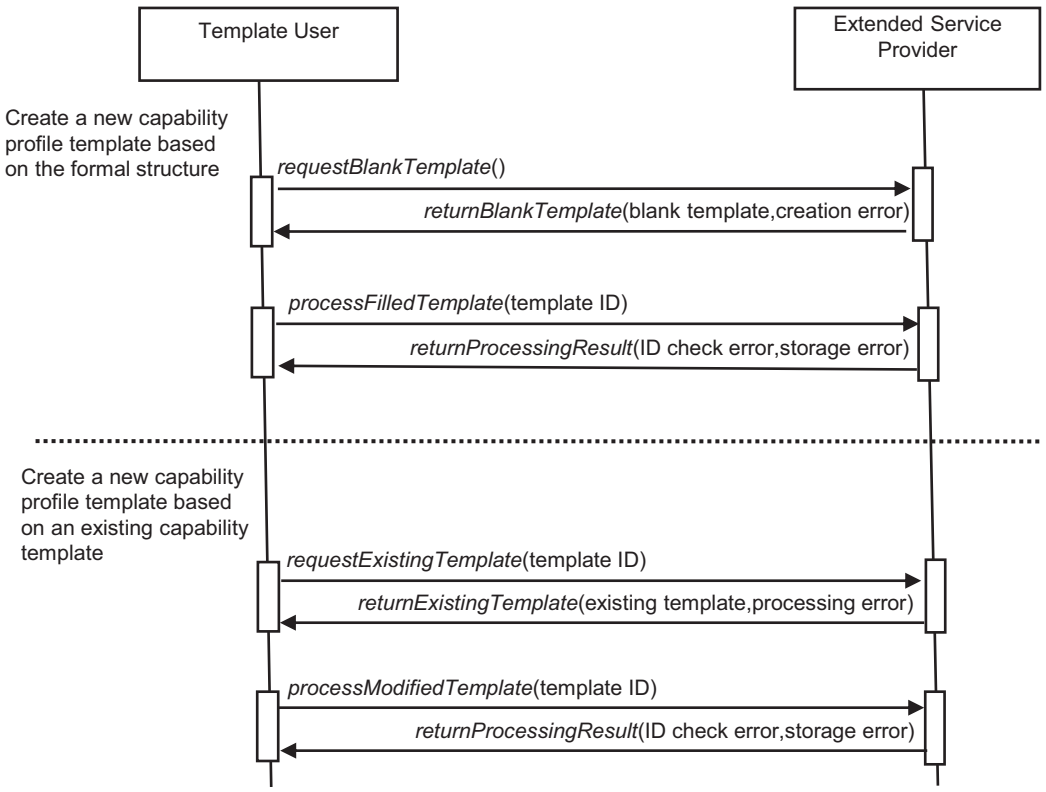
a) the template user invokes the *requestExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingTemplate* service is template ID;

b) the service provider invokes the *returnExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingTemplate* service are existing template and processing error;

c) the template user modifies the existing template and then invokes the *processModifiedTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *processModifiedTemplate* service is template ID;

d) the service provider checks the uniqueness of the template ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 3, below the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a template from an existing template.



NOTE 1 The dotted line separates the two creation cases.

NOTE 2 Template creation is always preceded by the preliminary step of registering and verifying unique template identifiers, which is outside the scope of ISO 16100.

Figure 3 — *createTemplate* service

6.1.3 *accessTemplate* service

The *accessTemplate* service, using the *requestExistingTemplate* and the *returnExistingTemplate* services, shall allow a template user to access an existing template.

The *accessTemplate* service shall consist of the following steps:

a) the template user invokes the *requestExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingTemplate* service is template ID;

b) the service provider invokes the *returnExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingTemplate* service are existing template and processing error.

Figure 4 provides a UML sequence diagram of the mandatory steps for accessing a template based on a template ID.

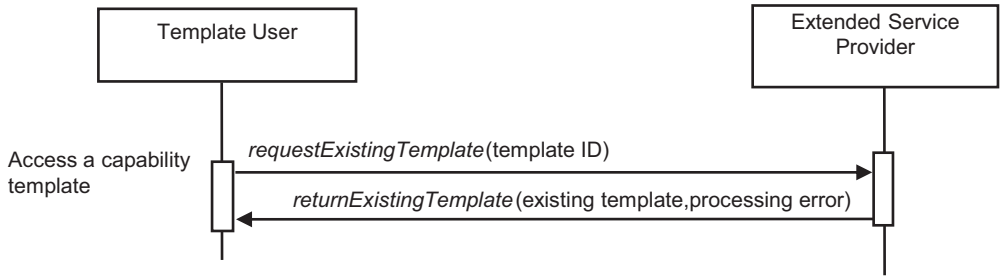


Figure 4 — *accessTemplate* service

6.1.4 *modifyTemplate* service

The *modifyTemplate* service, using the *requestExistingTemplate*, *returnExistingTemplate*, *processModifiedTemplate* and *returnProcessingResult* services, shall allow a template user to modify an existing template.

The *modifyTemplate* service shall consist of the following steps:

a) the template user invokes the *requestExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingTemplate* service is template ID;

b) the service provider invokes the *returnExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingTemplate* service are existing template and processing error;

c) the template user modifies the existing template and then invokes the *processModifiedTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *processModifiedTemplate* service is template ID;

d) the service provider checks the uniqueness of the template ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 5 provides a UML sequence diagram of the mandatory steps for the modification of an existing template based on a template ID.

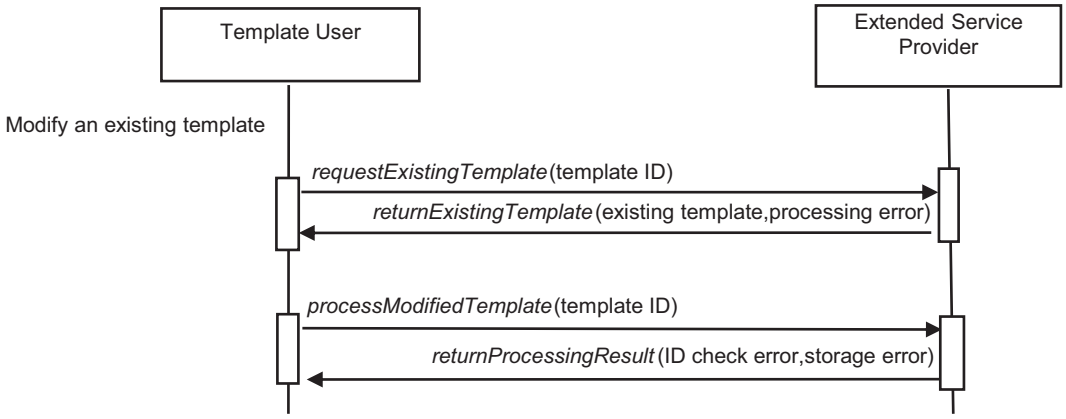


Figure 5 — *modifyTemplate* service

6.1.5 *validateTemplate* service

The *validateTemplate* service, using the *requestUnregisteredTemplate*, *returnUnregisteredTemplate*, *testTemplate*, *returnTestResult* services, shall allow a template user to conformance test an existing unregistered template.

The *validateTemplate* service shall consist of the following steps:

a) the template user invokes the *requestUnregisteredTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *requestUnregisteredTemplate* service is template ID;

b) the service provider invokes the *returnUnregisteredTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnUnregisteredTemplate* service are unregistered template and processing error;

c) the template user invokes the *testTemplate* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the *testTemplate* service;

d) the service provider invokes the *returnTestResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnTestResult* service are the test result and match status.

The value of the test result parameter of the *returnTestResult* service may be positive, negative, or the matching levels defined in ISO 16100-5:2009, 7.2. The value of the match status parameter of the *returnTestResult* service shall be equivalent to the output messages shown in ISO 16100-4:2006, Clause B.1.

The tested template shall be registered in the repository if the value of the test result parameter of the *returnTestResult* is positive or a Complete Match (see ISO 16100-5:2009, 7.2). Depending on user requirements, the tested template may be registered in the repository if the value of the test result parameter is either All Mandatory Match or Some Mandatory Match (see ISO 16100-5:2009, 7.2). The tested template may be modified again to meet the conformance criteria if the value of the test result parameter is negative or No Mandatory Match (see ISO 16100-5:2009, 7.2).

Figure 6 provides a UML sequence diagram of the mandatory steps for the conformance testing of an unregistered template based on a template ID.

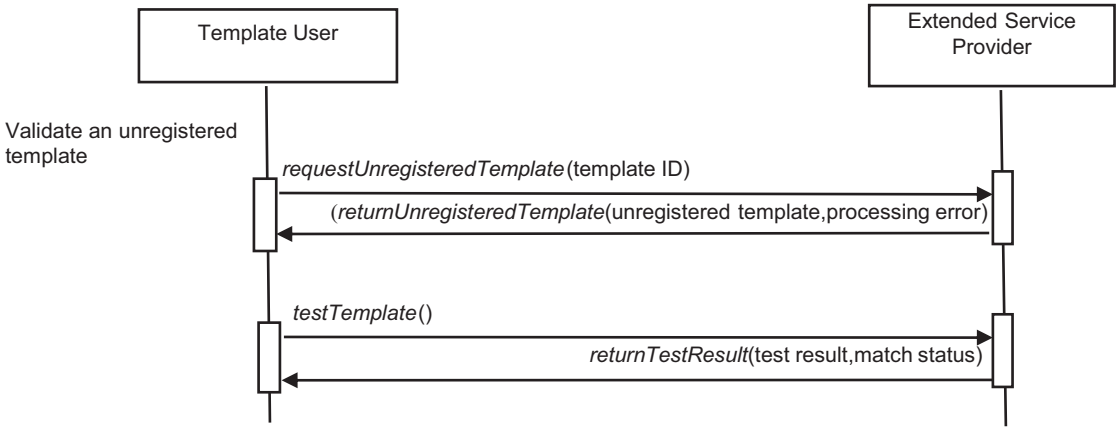


Figure 6 — *validateTemplate* service

6.1.6 *deleteTemplate* service

The *deleteTemplate* service, using the *requestExistingTemplate*, *returnExistingTemplate*, *removeTemplate* and *returnRemoveResult* services, shall allow a template user to delete an existing template.

The *deleteTemplate* service shall consist of the following steps:

a) the template user invokes the *requestExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingTemplate* service is template ID;

b) the service provider invokes the *returnExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingTemplate* service are existing template and processing error;

c) the template user invokes the *removeTemplate* service of the ServiceAccessPoint object, in which there are no parameters associated with the *removeTemplate* service;

d) the service provider invokes the *returnRemoveResult* service of the *ServiceAccessPoint* object, in which the parameter of the *returnRemoveResult* service is removal error.

Figure 7 provides a UML sequence diagram of the mandatory steps for the deletion of a template based on a template ID.

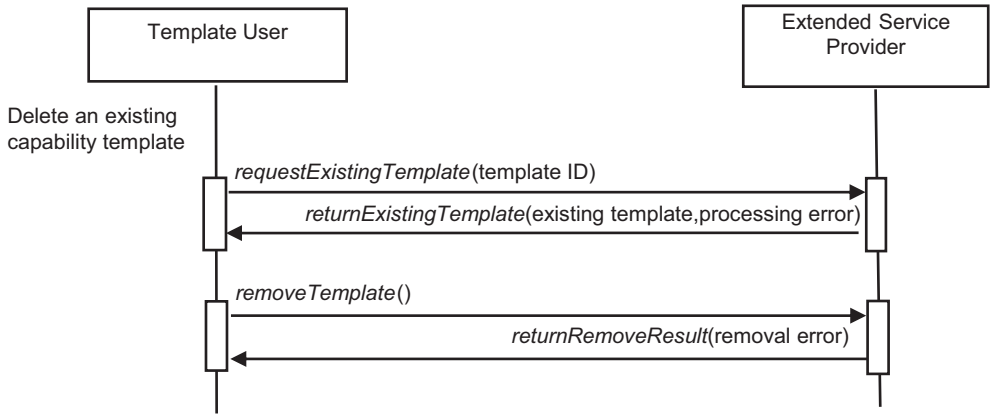


Figure 7 — *deleteTemplate* service

6.2 Extended CPI Group

6.2.1 Scenarios handled by the Extended CPI Group

The extended CPI service group shall handle the following capability profile scenarios:

a) create a capability profile, either by filling in at least the necessary attributes or elements of the template, or by modifying an existing capability profile with a new profile ID using the MDDs of the MDM, and then receiving the capability profile from a service provider;

b) access a profile, either from the repository via the ESI or from an MSU, based on a capability profile ID, and then receive a capability profile from either a service provider or an MSU;

c) modify a capability profile according to either user requirements or the results from a conformance test, and receive a modified capability profile from either a service provider or an MSU;

d) test a capability profile against the capability profile conformance criteria and receive either a positive, negative, or matching level response from a service provider;

e) register a tested capability profile into a Repository and receive a “registered” status from a service provider;

f) delete a capability profile based on a capability profile ID and receive a “deleted” status from a service provider.

6.2.2 Capability profile creation

6.2.2.1 Creation process

The process for creating a capability profile consists of either filling in specific values for certain attributes of certain elements in the capability profile template, or modifying previously filled-in values in an existing capability profile.

6.2.2.2 *createProfile* service

6.2.2.2.1 Profile based on the capability profile template

The *createProfile* service shall allow a profile user to request the creation of a profile based on the capability profile template. When creating a profile based on a capability profile template, the *createProfile* service uses at a minimum the *requestExistingTemplate*, *returnExistingTemplate*, *processFilledProfile* and *returnProcessingResult* services. The *createProfile* service shall consist of the following steps:

a) the profile user invokes *the requestExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingTemplate* service is template ID;

b) the service provider invokes the *returnExistingTemplate* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingTemplate* service are existing template and processing error;

c) the profile user fills in the existing template using MDD objects of the MDM and then invokes the *processFilledProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *processFilledProfile* service is profile ID;

d) the service provider checks the uniqueness of the profile ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 8, above the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a profile from an existing template.

6.2.2.2.2 Profile based on an existing capability profile

The *createProfile* service shall allow a profile user to request the creation of a profile modified from an existing profile. When creating a profile modified from an existing profile, the *createTemplate* service uses at a minimum the *requestExistingProfile*, *returnExistingProfile*, *processModifiedProfile* and *returnProcessingResult* services. The *createProfile* service shall consist of the following steps:

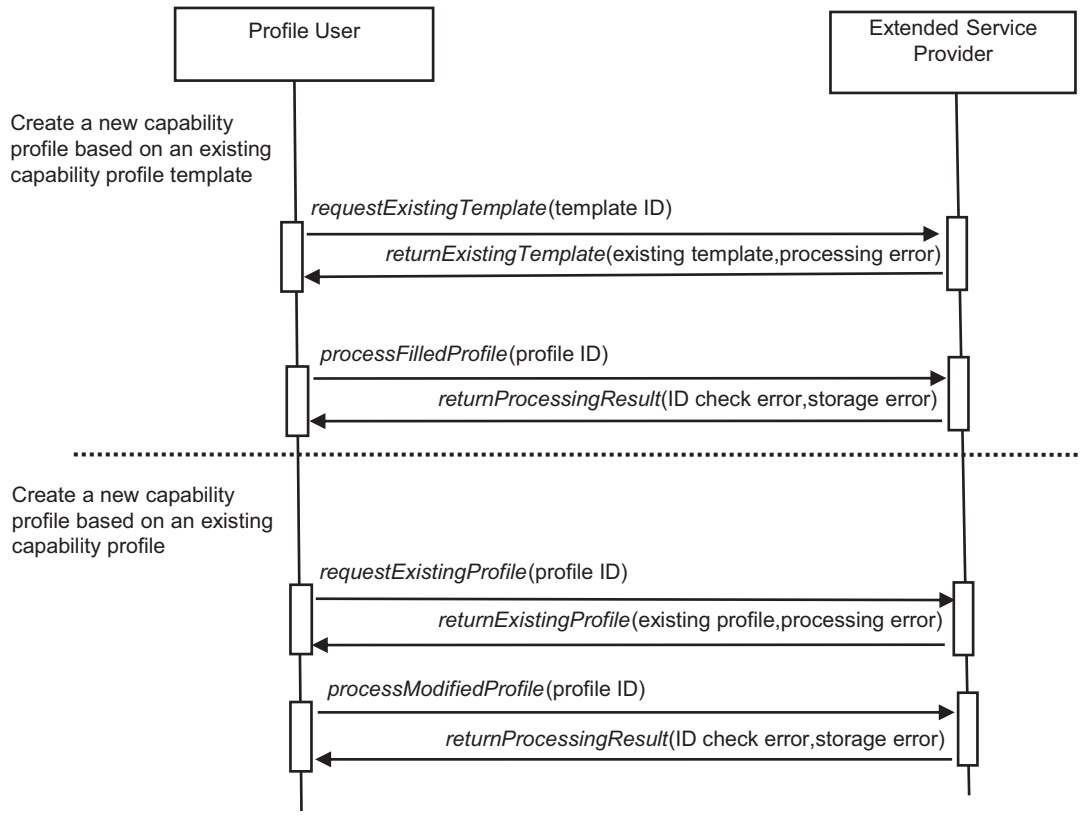
a) the profile user invokes the *requestExistingProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingProfile* service is profile ID;

b) the service provider invokes the *returnExistingProfile* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error;

c) the profile user modifies the existing profile using MDDs of the MDM and then invokes the *processModifiedProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *processModifiedProfile* service is profile ID;

d) the service provider checks the uniqueness of the profile ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 8, below the dotted line, provides a UML sequence diagram for the creation of a profile from an existing profile.



NOTE 1 The dotted line separates the two creation cases.

NOTE 2 Profile creation is always preceded by the preliminary step of registering and verifying unique template identifiers, which is outside the scope of ISO 16100.

Figure 8 — *createProfile* service

6.2.3 *accessProfile* service

6.2.3.1 Profile from an ESI distinct from an MSU

The *accessProfile* service shall allow a profile user to access an MSU capability profile from an ESI distinct from an MSU. The *accessProfile* service uses at a minimum the *requestExistingProfile* and the *returnExistingProfile* services. The *accessProfile* service shall consist of the following steps:

a) the profile user invokes the *requestExistingProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingProfile* service is profile ID;

b) the service provider invokes the *returnExistingProfile* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error.

Figure 9, above the dotted line, provides a UML sequence diagram of the mandatory steps for accessing a profile based on a profile ID via an ESI.

6.2.3.2 Profile from an MSU

The *accessProfile* service shall allow a profile user to access a required capability profile from an MSU. The *accessProfile* service uses at a minimum the *requestExistingProfile* and the *returnExistingProfile* services. The *accessProfile* service shall consist of the following steps:

a) the profile user invokes the *requestExisingProfile* service of the MSU object, in which there are no parameters with the *requestExistingProfile* service;

b) the service provider invokes the *returnExistingProfile* service of the MSU object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error.

Figure 9, below the dotted line, provides a UML sequence diagram of the mandatory steps for accessing a profile based on a profile ID via an MSU.

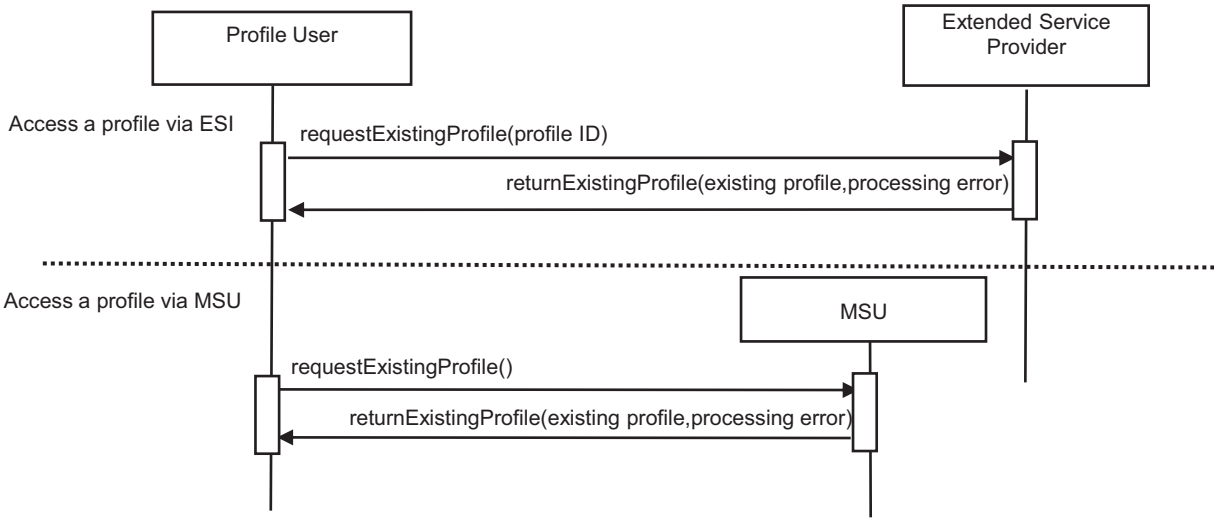


Figure 9 — *accessProfile* service

6.2.4 *modifyProfile* service

6.2.4.1 Profile accessed from an ESI

The *modifyProfile* service, using the *requestExistingProfile*, *returnExistingProfile*, *processModifiedProfile* and *returnProcessingResult* services, shall allow a profile user to modify an existing profile that was accessed from an ESI. The *modifyProfile* service shall consist of the following steps:

a) the profile user invokes the *requestExistingProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingProfile* service is profile ID;

b) the service provider invokes the *returnExistingProfile* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error;

c) the profile user modifies the existing profile using MDDs of the MDM and then invokes the *processModifiedProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *processModifiedProfile* service is profile ID;

d) the service provider checks the uniqueness of the profile ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 10, above the dotted line, provides a UML sequence diagram for the modification of a profile from an existing profile via an ESI.

6.2.4.2 Profile accessed from an MSU

The *modifyProfile* service, using the *requestExistingProfile*, *returnExistingProfile*, *processModifiedProfile* and *returnProcessingResult* services, shall allow a profile user to modify an existing profile that was accessed from an MSU. The *modifyProfile* service shall consist of the following steps:

a) the profile user invokes the *requestExistingProfile* service of the MSU object, in which the parameter of *requestExistingProfile* service is profile ID;

b) the service provider invokes the *returnExistingProfile* service of the MSU object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error;

c) the profile user modifies the existing profile using MDDs of the MDM and then invokes the *processModifiedProfile* service of the MSU object, in which the parameter of *processModifiedProfile* service is profile ID;

d) the service provider checks the uniqueness of the profile ID and then invokes the *returnProcessingResult* service of the MSU object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 10, below the dotted line, provides a UML sequence diagram for the modification of a profile from an existing profile via an MSU.

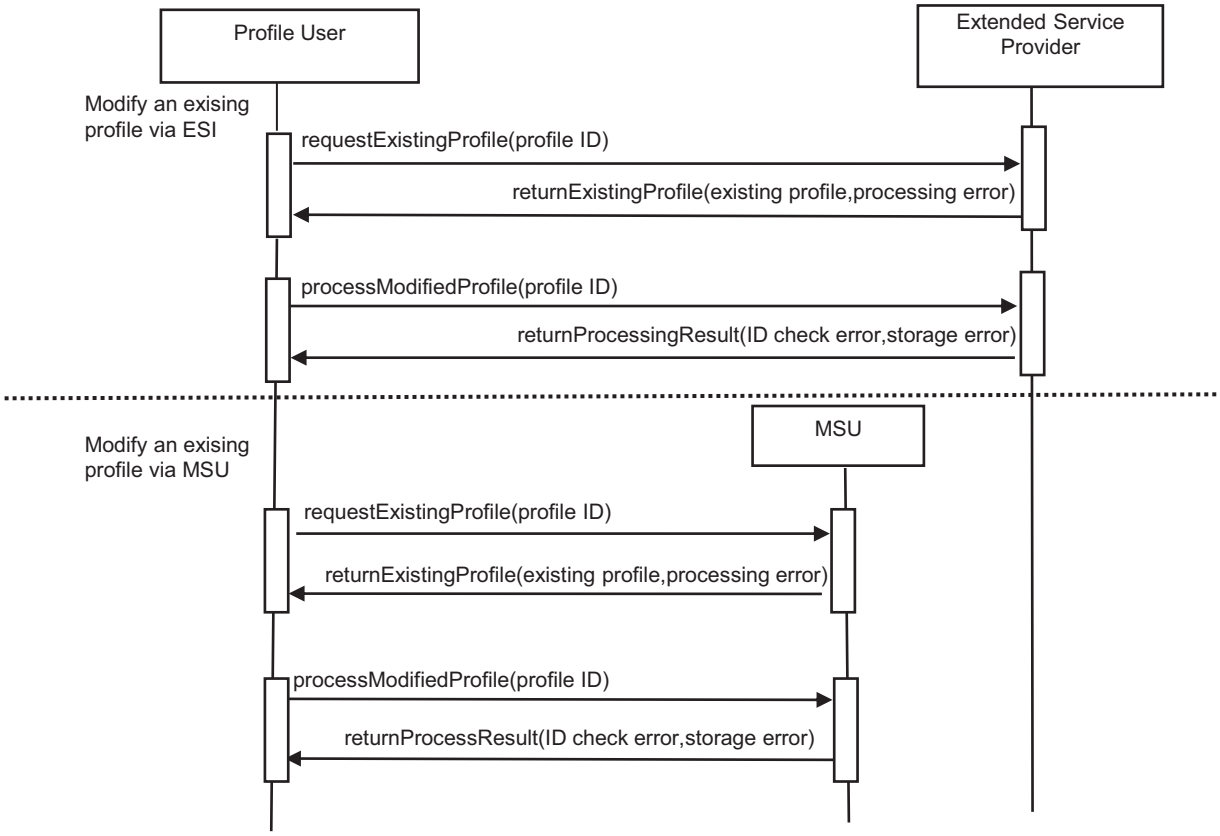


Figure 10 — *modifyProfile* service

6.2.5 *validateProfile* service

The *validateProfile* service, using the *requestExistingProfile*, *returnExistingProfile*, *testProfile*, *returnTestingResult* services, shall allow a profile user to conformance test an existing profile.

The *validateProfile* service shall consist of the following steps:

a) the profile user invokes the *requestExistingProfile* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingProfile* service is profile ID;

b) the service provider invokes the *returnExistingProfile* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error;

c) the profile user invokes the *testProfile* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the *testProfile* service;

d) the service provider invokes the *returnTestResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnTestResult* service are the test result and match status.

Figure 11 provides a UML sequence diagram of the mandatory steps for the conformance testing of an existing profile based on a profile ID.

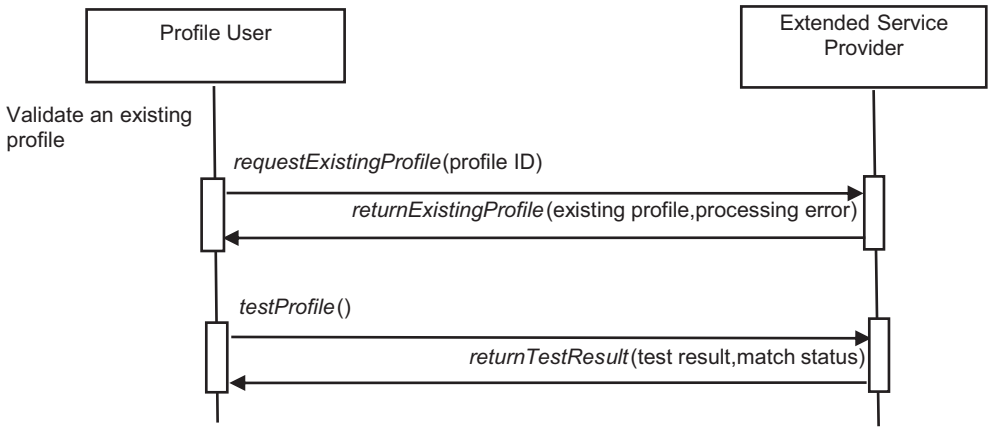


Figure 11 — *validateProfile* service

6.2.6 *deleteProfile* service

The *deleteProfile* service, using the *requestExistingProfile*, *returnExistingProfile*, *removeProfile* and *returnRemoveResult* services, shall allow a profile user to delete an existing profile from the Repository via an ESI.

The *deleteProfile* service shall consist of the following steps:

a) the profile user invokes the *requestExistingProfile* service of the ServiceAccessPoint object, in which the parameter of the *requestExistingProfile* service is profile ID;

b) the service provider invokes the *returnExistingProfile* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingProfile* service are existing profile and processing error;

c) the profile user invokes the *removeProfile* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the *removeProfile* service;

d) the service provider invokes the *returnRemoveResult* service of the *ServiceAccessPoint* object, in which the parameter of the *returnRemoveResult* service is removal error.

Figure 12 provides a UML sequence diagram of the mandatory steps for the deletion of a profile based on a profile ID.

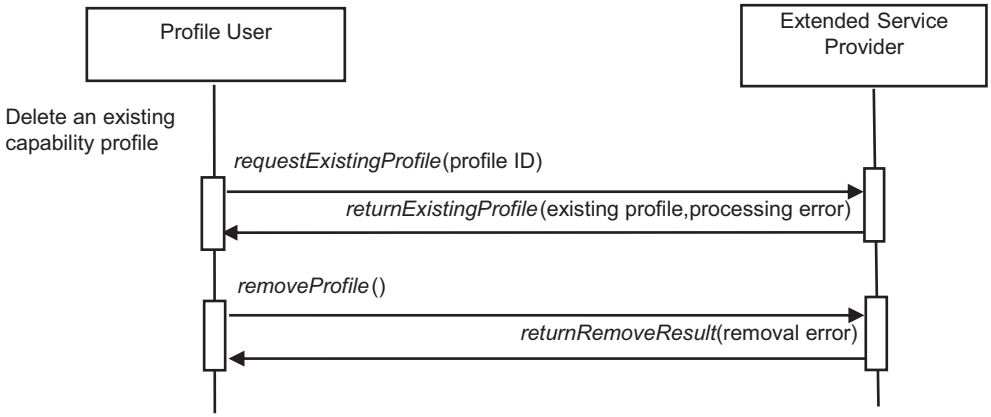


Figure 12 — *deleteProfile* service

6.3 CCSI Group

6.3.1 Scenarios handled by the CCSI Group

The CCSI Group shall handle the following capability profile template scenarios:

a) create a CCS, either by filling in a blank CCS template or by modifying an existing CCS, and then receive the created CCS from a service provider;

b) access a CCS from a repository by its CCS ID and then receive a CCS from a service provider;

c) modify a CCS according to either user requirements or the results from a conformance test, and receive a modified CCS from a service provider;

d) test a CCS against the CCS conformance criteria and receive either a positive, negative, or matching level response from a service provider;

e) register a CCS into a repository and receive a “registered” status from a service provider;

f) delete a CCS based on a CCS ID and receive a “deleted” status from a service provider.

6.3.2 Capability class structure creation

6.3.2.1 Creation process

The creation process for a capability class structure starts with an analysis of a particular manufacturing application. The manufacturing process, consisting of a set of activities, is the key consideration in this analysis. These activities are represented by nodes in a capability class structure. By creating new nodes, a user can create a new capability class structure within the same MDM.

The process for creating a capability class structure consists of either filling in a blank CCS template or modifying an existing CCS according to a particular application.

6.3.2.2 *createCCS* service

6.3.2.2.1 CCS from a blank CCS template

The *createCCS* service shall allow a CCS user to create a CCS by filling in a blank CCS template. When creating a CCS from a blank CCS template, the *createCCS* service uses at a minimum the *requestBlankCCS*, *returnBlankCCS*, *processFilledCCS* and *returnProcessingResult* services.

The *createCCS* service for creating a new CCS from a blank CCS template shall consist of the following steps:

a) the CCS user invokes the *requestBlankCCS* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the *requestBlankCCS* service;

b) the service provider invokes the *returnBlankCCS* service of the *ServiceAccessPoint* object, in which the parameters of the *returnBlankCCS* service are blank CCS and creation error;

c) the CCS user fills in the blank CCS template using MDDs of the MDM and then invokes the *processFilledCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *processFilledCCS* service is CCS ID;

d) the service provider checks the uniqueness of the CCS ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 13, above the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a CCS from a formal CCS structure.

6.3.2.2.2 CCS created by modifying an existing CCS

The *createCCS* service shall allow a CCS user to create a CCS by modifying an existing CCS. When creating a CCS modified from an existing CCS, the *createCCS* service uses at a minimum the *requestExistingCCS*, *returnExistingCCS*, *processModifiedCCS* and *returnProcessingResult* services. The *createCCS* service shall consist of the following steps:

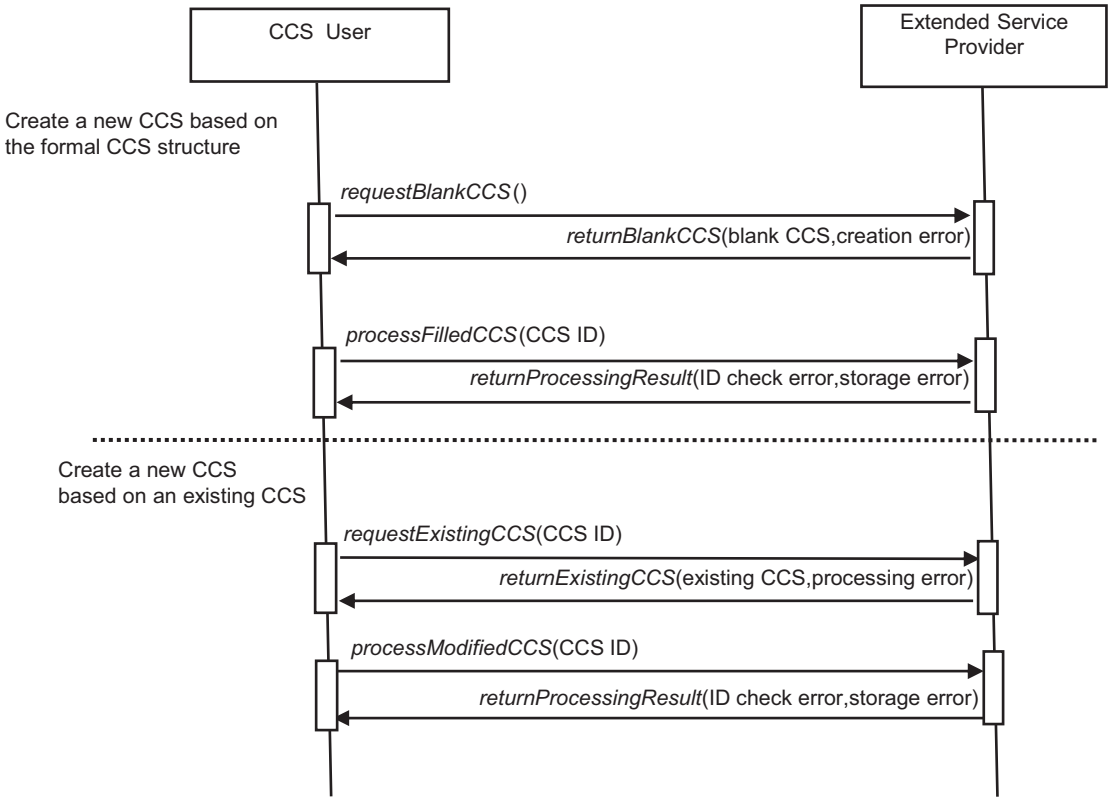
a) the CCS user invokes the *requestExistingCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingCCS* service is CCS ID;

b) the service provider invokes the *returnExistingCCS* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingCCS* service are existing CCS and processing error;

c) the CCS user modifies the existing CCS and then invokes the *processModifiedCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *processModifiedCCS* service is CCS ID;

d) the service provider checks the uniqueness of the CCS ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 13, below the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a CCS from an existing CCS.



NOTE 1 The dotted line separates the two creation cases.

NOTE 2 CCS creation is always preceded by the preliminary step of registering and verifying unique CCS identifiers, which is outside the scope of ISO 16100.

Figure 13 — *createCCS* service

6.3.3 *accessCCS* service

The *accessCCS* service, using the *requestExistingCCS* and the *returnExistingCCS* services, shall allow a CCS user to access an existing CCS.

The *accessCCS* service shall consist of the following steps:

a) the CCS user invokes the *requestExistingCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingCCS* service is CCS ID;

b) the service provider invokes the *returnExistingCCS* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingCCS* service are existing CCS and processing error.

Figure 14 provides a UML sequence diagram of the mandatory steps for accessing a CCS based on a CCS ID.

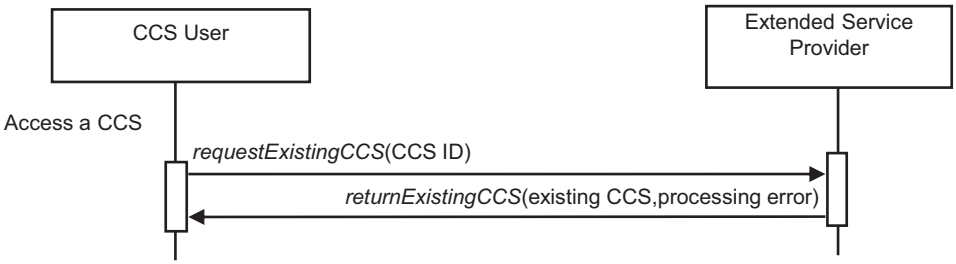


Figure 14 — *accessCCS* service

6.3.4 *modifyCCS* service

The *modifyCCS* service, using the *requestExistingCCS*, *returnExistingCCS*, *processModifiedCCS* and *returnProcessingResult* services, shall allow a CCS user to modify an existing CCS.

The *modifyCCS* service shall consist of the following steps:

a) the CCS user invokes the *requestExistingCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingCCS* service is CCS ID;

b) the service provider invokes the *returnExistingCCS* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingCCS* service are existing CCS and processing error;

c) the CCS user modifies the existing CCS and then invokes the *processModifiedCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *processModifiedCCS* service is CCS ID;

d) the service provider checks the uniqueness of the CCS ID and then invokes the *returnProcessingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnProcessingResult* service are ID check error and storage error.

Figure 15 provides a UML sequence diagram of the mandatory steps for the modification of an existing CCS based on a CCS ID.

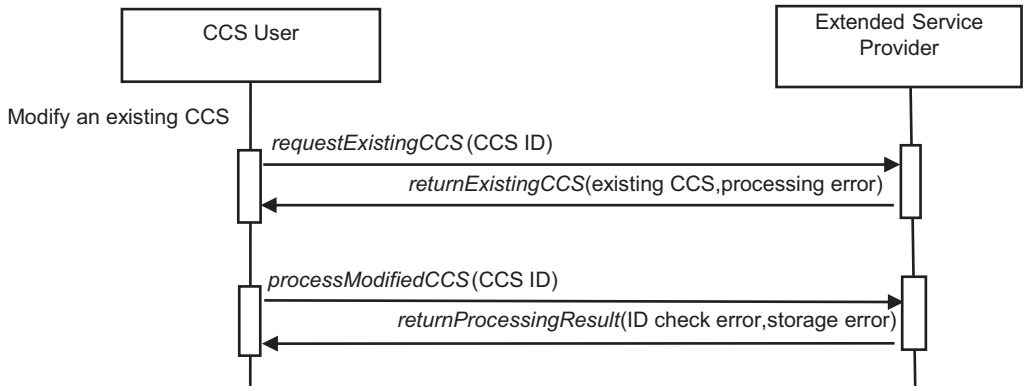


Figure 15 — *modifyCCS* service

6.3.5 *validateCCS* service

The *validateCCS* service, using the *requestExistingCCS*, *returnExistingCCS*, *testCCS* and *returnTestResult* services, shall allow a CCS user to conformance test an existing CCS.

The *validateCCS* service shall consist of the following steps:

a) the CCS user invokes the *requestExistingCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingCCS* service is CCS ID;

b) the service provider invokes the *returnExistingCCS* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingCCS* service are existing CCS and processing error;

c) the CCS user invokes the *testCCS* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the *testCCS* service;

d) the service provider invokes the *returnTestResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnTestResult* service are the test result and error status.

Figure 16 provides a UML sequence diagram of the mandatory steps for the conformance testing of an existing CCS based on a CCS ID.

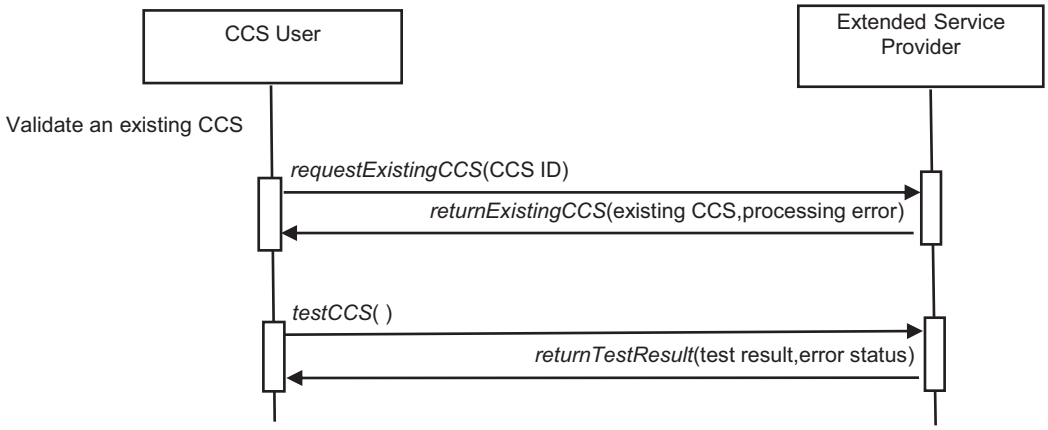


Figure 16 — *validateCCS* service

6.3.6 *deleteCCS* service

The *deleteCCS* service, using the *requestExistingCCS*, *returnExistingCCS*, *removeCCS* and *returnRemoveResult* services, shall allow a CCS user to delete an existing CCS.

The *deleteCCS* service shall consist of the following steps:

a) the CCS user invokes the *requestExistingCCS* service of the *ServiceAccessPoint* object, in which the parameter of the *requestExistingCCS* service is CCS ID;

b) the service provider invokes the *returnExistingCCS* service of the *ServiceAccessPoint* object, in which the parameters of the *returnExistingCCS* service are existing CCS and processing error;

c) the CCS user invokes the *removeCCS* service of the *ServiceAccessPoint* object, in which there are no parameters associated with the removeCCS service;

d) the service provider invokes the *returnRemoveResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnRemoveResult* service are removal error.

Figure 17 provides a UML sequence diagram of the mandatory steps for the deletion of a CCS based on a CCS ID.

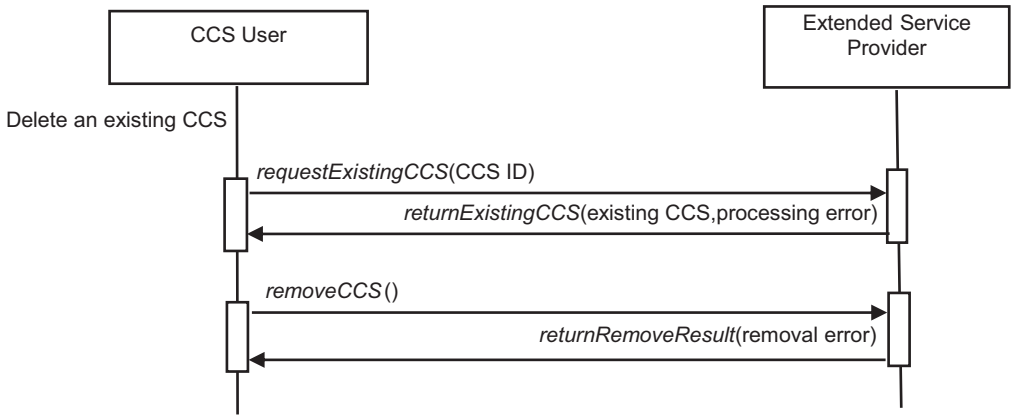


Figure 17 — *deleteCCS* service

6.4 The Extended Matcher Group

6.4.1 Scenarios handled by the Extended Matcher Group

The Extended Matcher Group shall handle the following scenarios for accessing and matching two capability profiles:

a) request an MSU capability profile, either from a repository to which the MSU capability profile has been registered or from an MSU to which the MSU capability profile is associated, and then receive the MSU capability profile;

b) match two capability profiles – an MSU capability profile and a required capability profile – and receive a matching result that consists of the matching level (see ISO 16100-5:2009, 7.2) and a report on matched and unmatched functions from a service provider.

6.4.2 *ExtendedMatcher* service

The *ExtendedMatcher* service, using the *requestProfile*, *returnProfile*, *requestMatching* and *returnMatchingResult* services, shall allow a user to request profiles either from a repository or from an MSU and to match an MSU capability profile with a required capability profile using a matcher.

The *ExtendedMatcher* service shall consist of the following steps:

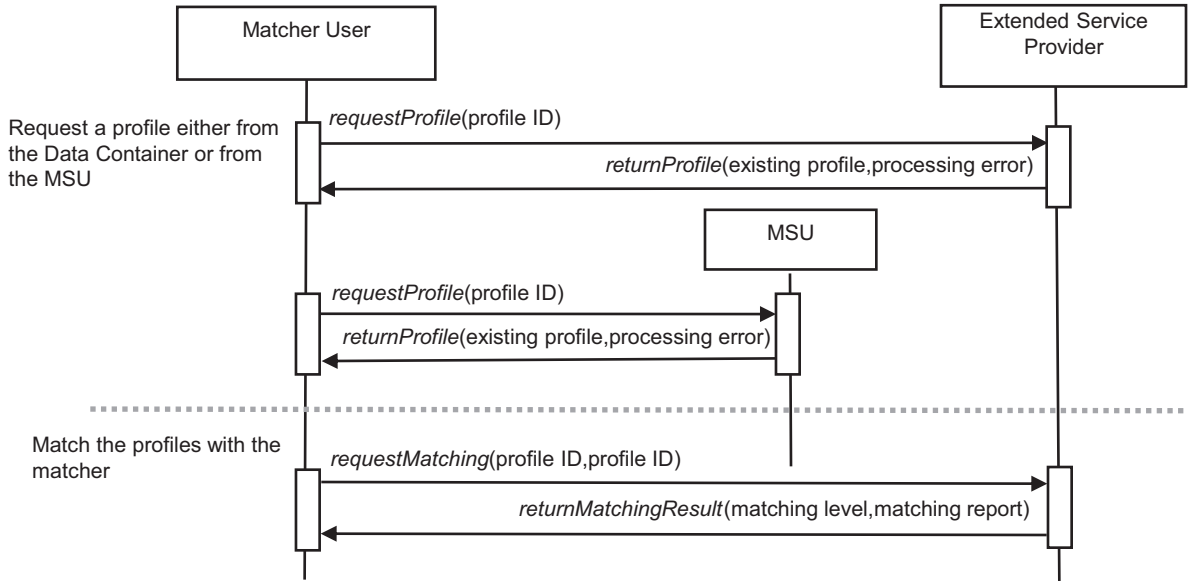
a) the matcher user invokes the *requestProfile* service of the *ServiceAccessPoint* object or the MSU object, in which the parameter of the *requestProfile* service is profile ID;

b) the service provider invokes the *returnProfile* service of the *ServiceAccessPoint* object or the MSU object, in which the parameters of the *returnProfile* service are existing profile and processing error;

c) the matcher user invokes the *requestMatching* service of the *ServiceAccessPoint* object, in which there are two profile ID parameters;

d) the service provider invokes the *returnMatchingResult* service of the *ServiceAccessPoint* object, in which the parameters of the *returnMatchingResult* service are matching level (see ISO 16100-5:2009, 7.2) and matching report (a report on matched and unmatched functions).

Figure 18 provides a UML sequence diagram of the mandatory steps for requesting a profile and matching two profiles.



NOTE The dotted line separates the profile request from the matching request.

Figure 18 — *ExtendedMatcher* services

6.4.3 Matcher conformance testing

The conformance methodology specified in ISO 16100-4 applies to this part of ISO 16100, which extends the CSIs for a capability profile matcher shown in ISO 16100-4:2006, Table 9. Tables 1 and 2 of this part of ISO 16100 shall be used for conformance testing. The conformance point types shown in Tables 1 and 2 are defined in ISO 16100-4:2006, Table 5.

Table 1 — CSIs for matcher performance

Conformance point and set number	Conformance point description	Specification reference	Conformance point type[a]	Abstract test criteria
Index_1	Receive required capability profile	ISO 16100-5:2009, Clause 7	A	Verify the required capability profile to be in conformance with ISO 16100-5:2009, Table 2
Index_2	Receive MSU's capability profile	ISO 16100-5:2009, Clause 7	A	Verify the MSU's capability profile to be in conformance with ISO 16100-5:2009, Table 2
Index_3	Compare the MDM IDs of the two profiles	ISO 16100-5:2009, Clause 7	A	—
Index_3.1	Extract the MDM IDs from the two profiles	ISO 16100-5:2009, Clause 7	A	Verify the dictionary IDs
Index_3.2	Compare the two MDM IDs	ISO 16100-5:2009, Clause 7	A	Verify and display the result of the comparison
Index_3.2.1	Report result 1 and go to Type 1 Matcher	ISO 16100-5:2009, Clause 7	A	Verify: 1) "IDs are the same" 2) "Go to Type 1 Matcher"
Index_3.2.2	Report result 2	ISO 16100-5:2009, Clause 7	A	Verify: "IDs are not the same"
Index_4	Compare the MDM names of the two profiles	ISO 16100-5:2009, Clause 7	A	—
Index_4.1	Extract the MDM names from the two profiles	ISO 16100-5:2009, Clause 7	A	Verify the MDM names
Index_4.2	Compare the two MDM names	ISO 16100-5:2009, Clause 7	A	Verify the result from the comparison
Index_4.2.1	Report result 1 and end the matching	ISO 16100-5:2009, Clause 7	A	Verify: 1) "MDM names are not the same" 2) "The two profiles come from different MDMs" 3) "There has been no match"
Index_4.2.2	Report result 2	ISO 16100-5:2009, Clause 7	A	Verify: "MDM names are the same"
Index_5	Compare the capability definition formats of the two profiles	ISO 16100-5:2009, Clause 7	A	—
Index_5.1	Extract the capability definition formats from the two profiles	ISO 16100-5:2009, Clause 7	A	Verify the capability definition formats
Index_5.2	Compare the capability definition formats	ISO 16100-5:2009, Clause 7	A	Verify the result from the comparison
Index_5.2.1	Report result 1	ISO 16100-5:2009, Clause 7	A	Show report 1: 1) "Capability definition formats are not the same" 2) "Convert MDDs and MDD objects in capability definitions to a single capability definition format"

Table 1 (*continued*)

Conformance point and set number	Conformance point description	Specification reference	Conformance point type[a]	Abstract test criteria
Index_5.2.2	Report result 2	ISO 16100-5:2009, Clause 7	A	Verify: "Capability definition formats are the same"
Index_6	Compare the two capability definitions	ISO 16100-5:2009, Clause 7	A	—
Index_6.1	Compare contents of element "Set_of_MDD_Objects" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	—
Index_6.1.1	Compare attributes "name" and "action" in the elements "MDD_Name" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result
Index_6.2	Compare contents of the two "List_of_MDD_Objects" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	—
Index_6.2.1	Compare the sequences of the elements "MDD_Name" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result
Index_6.2.2	Compare attributes "name" and "action" in the elements "MDD_Name" for the same position in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result
Index_6.3	Compare contents of the two "Time_Ordered_MDD_Objects" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	—
Index_6.3.1	Compare time orders of the elements "MDD_Name" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result
Index_6.3.2	Compare attributes "name" and "action" in the two elements "MDD_Name" for the same position in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result
Index_6.4	Compare contents in the "Event_Ordered_MDD_Objects"	ISO 16100-5:2009, Clause 7	A	—
Index_6.4.1	Compare the event orders of the elements "MDD_Name" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result
Index_6.4.2	Compare attributes "name" and "action" in the two elements "MDD_Name" for the same position in the two capability definitions	ISO 16100-5:2009, Clause 7	A	Verify the result

[a] See ISO 16100-4:2006, Table 5.

Table 2 — CSIs for matcher reporting

Conformance point and set number	Conformance point description	Specification reference	Conformance point type[a]	Abstract test criteria
Index_1	MatchingLevelReport	ISO 16100-5:2009, 7.2	A	Matching level is in one of the following: Complete Match All Mandatory Match Some Mandatory Match No Mandatory Match
Index_2	DetaitedListReport	ISO 16100-5:2009, 7.2	A	Verify matched functions and unmatched functions
Index_3	CompareMatchingLevel&Detailed Report MatchingLevel	—	—	—
Index_3.1	ForCompleteMatch	—	—	Verify that both sets of functions in the two profiles are fully equivalent
Index_3.2	ForAllMandatoryMatch	—	—	Verify that both sets of mandatory functions in the two profiles are fully equivalent
Index_3.3	ForSomeMandatoryMatch	—	—	Verify the list of equivalent mandatory functions in the two profiles and the list of non-equivalent mandatory functions
Index_3.4	ForNoMandatoryMatch	—	—	Present both sets of functions in the two profiles as fully non-equivalent

[a] See ISO 16100-4:2006, Table 5.

7 Formal ESI protocol description

7.1 Generic service syntax

The service URN syntax specified in ISO 16100-3 applies to this part of ISO 16100.

A generic service URN starts with the string “service:”. This service URN includes the service type followed by the relevant service access point up to, but not including the final “:” where the address specification starts. The service-specific attribute information follows the address specification encoded according to the URN grammar.

The complete service URN shall be of the following syntax:

“service:<service-type>:<service-access-point>://<address>;<attribute-list>”

The <service-type> item in the URN string shall represent a generic service identified in 5.1.

The <service-access-point> item in the URN string shall represent an access point of an ESI service group; these service groups are identified in 5.2.

The <address> item in the URN string shall represent the path to the ESI service provider.

The attribute list consists of a list of semicolon ";" separated attribute assignments. The attribute assignments shall have the form:

<attribute-id>=<attribute-value>

In addition, for keyword attributes, the form <attribute-id> shall be used.

The detailed service descriptions in all the UML diagrams in Clause 6 shall be expressed using the formal syntax as specified in the remaining subclauses in Clause 7.

7.2 CPTI Group service protocol

7.2.1 Capability profile template creation

7.2.1.1 Creation based on the formal structure

The *createTemplate* service generates a template based on the formal structure and shall consist of the following steps:

a) the *requestBlankTemplate* service requests a blank template and has the service type:

<service-type>="requestBlankTemplate"

b) the *returnBlankTemplate* service returns the blank template and has the service type:

<service-type>="returnBlankTemplate"

with the corresponding attributes:

— template_content="the_template_content"

— access_status="the_access_status"

c) the *processFilledTemplate* service requests acceptance of a filled template and has the service type:

<service-type>="processFilledTemplate"

with the corresponding attribute:

— template_ID="the_template_id"

d) the *returnProcessingResult* service returns the processing results and has the service type:

<service-type>="returnProcessingResult"

with the corresponding attributes:

— ID_check_error="ID_check_error"

— storage_error="storage_error"

7.2.1.2 Creation based on an existing template

The *createTemplate* service also generates a template based on an existing template and shall consist of the following steps:

a) the *requestExistingTemplate* service requests an existing template and has the service type:

<service-type>=“requestExistingTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”;

b) the *returnExistingTemplate* service returns the existing template and has the service type:

<service-type>=“returnExistingTemplate”

with the corresponding attributes:

— template_content=“the_template_content”

— process_status=“the_process_status”

c) the *processModifiedTemplate* service requests acceptance of the modified template and has the service type:

<service-type>=“processModifiedTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”

d) the *returnProcessingResult* service returns the processing results and has the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.2.2 Capability profile template access

The *accessTemplate* service accesses a template and shall consist of the following steps:

a) the *requestExistingTemplate* service accesses an existing template and has the service type:

<service-type>=“requestExistingTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”;

b) The *returnExistingTemplate* service returns the requested template and shall have the service type:

<service-type>=“returnExistingTemplate”

with the corresponding attributes:

— template_content=“the_template_content”

— process_status=“the_process_status”

7.2.3 Capability profile template modification

The *modifyTemplate* service modifies a template and shall consist of the following steps:

a) the *requestExistingTemplate* service accesses an existing template and has the service type:

<service-type>=“requestExistingTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”

b) the *returnExistingTemplate* service returns the requested template and has the service type:

<service-type>=“returnExistingTemplate”

with the corresponding attributes:

— template_content=“the_template_content”

— process_status=“the_process_status”

c) the *processModifiedTemplate* service requests acceptance of the modified template and has the service type:

<service-type>=“processModifiedTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”

d) The *returnProcessingResult* service returns the processing results and shall have the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.2.4 Capability profile template conformance test

The *validateTemplate* service tests a template and shall consist of the following steps:

a) the *requestUnregisteredTemplate* service accesses an unregistered template and has the service type:

<service-type>=“requestUnregisteredTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”

b) the *returnUnregisteredTemplate* service returns the requested template and shall have the service type:

<service-type>=“returnUnregisteredTemplate”

with the corresponding attributes:

— template_content=“the_template_content”

— process_status=“the_process_status”

c) the *testTemplate* service validates an unregistered template and shall have the service type:

<service-type>=“testTemplate”

d) the *returnTestResult* service returns the tested results and the status and shall have the service type:

<service-type>=“returnTestResult”

with the corresponding attributes:

— test_result =“the_test_result”

— test_status=“the_test_status”

7.2.5 Capability profile template deletion

The *deleteTemplate* service deletes an existing template and shall consist of the following steps:

a) the *requestExistingTemplate* service accesses an existing template and shall have the service type:

<service-type>=“requestExistingTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”

b) the *returnExistingTemplate* service returns the requested template and shall have the service type:

<service-type>=“returnExistingTemplate”

with the corresponding attributes:

— template_content= “the_template_content”

— process_status=“the_process_status”

c) the *removeTemplate* service removes a capability profile template from the Repository and shall have the service type:

<service-type>=“removeTemplate”

d) the *returnRemoveResult* service returns the removal status and shall have the service type:

<service-type>=“returnRemoveResult”

with the corresponding attribute:

— remove_status=“the_remove_status”

7.3 Extended CPI Group service protocols

7.3.1 Capability profile creation

7.3.1.1 Creation based on the capability profile template

The *createProfile* service generates a profile based on the capability profile template and shall consist of the following steps:

a) the *requestExistingTemplate* service requests an existing template and shall have the service type:

<service-type>=“requestExistingTemplate”

with the corresponding attribute:

— template_ID=“the_template_id”

b) the *returnExistingTemplate* service returns the requested template and shall have the service type:

<service-type>=“returnExistingTemplate”

with the corresponding attributes:

— template_content=“the_template_content”

— access_status=“the_access_status”

c) the *processFilledProfile* service requests acceptance of a filled profile and shall have the service type:

<service-type>=“processFilledProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

d) the *returnProcessingResult* service returns the processing results and shall have the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_ error=“storage_error”

7.3.1.2 Creation based on an existing profile

The *createProfile* service also generates a profile based on an existing profile and shall consist of the following steps:

a) the *requestExistingProfile* service requests an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

b) the *returnExistingProfile* service returns the existing profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content=“the_profile_content”

— process_status=“the_process_status”

c) the *processModifiedProfile* service requests acceptance of the modified profile and shall have the service type:

<service-type> = “processModifiedProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

d) the *returnProcessingResult* service returns the processing results and shall have the service type:

<service-type> = “returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.3.2 Capability profile access

7.3.2.1 Access via an ESI

The *accessProfile* service accesses a profile via an ESI and shall consist of the following steps:

a) the *requestExistingProfile* service accesses an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

b) the *returnExistingProfile* service returns the requested profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content=“the_profile_content”

— process_status=“the_process_status”

7.3.2.2 Access via an MSU

The *accessProfile* service also accesses a profile via an MSU and shall consist of the following steps:

a) the *requestExistingProfile* service accesses an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

b) the *returnExistingProfile* service returns the requested profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content= “the_profile_content”

— process_status=“the_process_status”

7.3.3 Capability profile modification

The *modifyProfile* service modifies a profile and shall consist of the following steps:

a) the *requestExistingProfile* service accesses an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

b) the *returnExistingProfile* service returns the requested profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content= “the_profile_content”

— process_status=“the_process_status”

c) the *processModifiedProfile* service requests acceptance of the modified profile and shall have the service type:

<service-type>=“processModifiedProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

d) the *returnProcessingResult* service returns the processing result and shall have the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.3.4 Capability profile conformance test

The *validateProfile* service tests an existing profile and shall consist of the following steps:

a) the *requestExistingProfile* service accesses an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

b) the *returnExistingProfile* service returns the requested profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content=“the_profile_content”

— process_status=“the_process_status”

c) the *testProfile* service validates an unregistered profile and shall have the service type:

<service-type>=“testProfile”

d) the *returnTestResult* service returns the tested results and the status and shall have the service type:

<service-type>=“returnTestResult”

with the corresponding attributes:

— test_result=“the_test_result”

— test_status=“the_test_status”

7.3.5 Capability profile deletion

The *deleteTemplate* service deletes an existing template and shall consist of the following steps:

a) the *requestExistingProfile* service accesses an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

b) the *returnExistingProfile* service returns the requested profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content= “the_profile_content”

— process_status=“the_process_status”

c) the *removeProfile* service removes a capability profile from the Repository and shall have the service type:

<service-type>=“removeProfile”

d) the *returnRemoveResult* service returns the removal status and shall have the service type:

<service-type>=“returnRemoveResult”

with the corresponding attribute:

— remove_status=“the_remove_status”

7.4 CCSI Group service protocols

7.4.1 Capability class structure creation

7.4.1.1 Creation based on the formal structure

The *createCCS* service generates a CCS based on the formal structure and shall consist of the following steps:

a) The *requestBlankCCS* service requests a blank CCS and shall have the service type:

<service-type>=“requestBlankCCS”

b) the *returnBlankCCS* service returns the blank CCS and shall have the service type:

<service-type>=“returnBlankCCS”

with the corresponding attributes:

— CCS_content=“the_CCS_content”

— access_status=“the_access_status”

c) the *processFilledCCS* service requests acceptance of a filled CCS and shall have the service type:

<service-type>=“processFilledCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

d) the *returnProcessingResult* service returns the processing results and shall have the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.4.1.2 Creation based on an existing CCS

The *createCCS* service also generates a template based on an existing CCS and shall consist of the following steps:

a) the *requestExistingCCS* service requests an existing CCS and shall have the service type:

<service-type>=“requestExistingCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

b) the *returnExistingCCS* service returns the existing CCS and shall have the service type:

<service-type>=“returnExistingCCS”

with the corresponding attributes:

— CCS_content=“the_CCS_content”

— process_status=“the_process_status”

c) the *processModifiedCCS* service requests acceptance of the modified CCS and shall have the service type:

<service-type>=“processModifiedCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

d) the *returnProcessingResult* service returns the processing results and shall have the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.4.2 Capability class structure access

The *accessCCS* service accesses a CCS and shall consist of the following steps:

a) the *requestExistingCCS* service requests an existing CCS and shall have the service type:

<service-type>=“requestExistingCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

b) the *returnExistingCCS* service returns the existing CCS and shall have the service type:

<service-type>=“returnExistingCCS”

with the corresponding attributes:

— CCS_content=“the_CCS_content”

— process_status=“the_process_status”

7.4.3 Capability class structure modification

The *modifyCCS* service modifies a CCS and shall consist of the following steps:

a) the *requestExistingCCS* service requests an existing CCS and shall have the service type:

<service-type>=“requestExistingCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

b) the *returnExistingCCS* service returns the existing CCS and shall have the service type:

<service-type>=“returnExistingCCS”

with the corresponding attributes:

— CCS_content=“the_CCS_content”

— process_status=“the_process_status”

c) the *processModifiedCCS* service requests acceptance of the modified CCS and shall have the service type:

<service-type>=“processModifiedCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

d) the *returnProcessingResult* service returns the processing results and shall have the service type:

<service-type>=“returnProcessingResult”

with the corresponding attributes:

— ID_check_error=“ID_check_error”

— storage_error=“storage_error”

7.4.4 Capability class structure conformance test

The *validateCCS* service tests a CCS and shall consist of the following steps:

a) the *requestExistingCCS* service requests an existing CCS and shall have the service type:

<service-type>=“requestExistingCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

b) the *returnExistingCCS* service returns the existing CCS and shall have the service type:

<service-type>=“returnExistingCCS”

with the corresponding attributes:

— CCS_content=“the_CCS_content”

— process_status=“the_process_status”

c) the *testCCS* service validates an unregistered CCS and shall have the service type:

<service-type>=“testCCS”

d) The *returnTestResult* service returns the tested results and the status and shall have the service type:

<service-type>=“returnTestResult”

with the corresponding attributes:

— test_result =“the_test_result”

— test_status=“the_test_status”

7.4.5 Capability class structure deletion

The *deleteCCS* service deletes an existing template and shall consist of the following steps:

a) the *requestExistingCCS* service requests an existing CCS and shall have the service type:

<service-type>=“requestExistingCCS”

with the corresponding attribute:

— CCS_ID=“the_CCS_id”

b) the *returnExistingCCS* service returns the existing CCS and shall have the service type:

<service-type>=“returnExistingCCS”

with the corresponding attributes:

— CCS_content=“the_CCS_content”

— process_status=“the_process_status”

c) the *removeCCS* service removes a CCS from the Repository and shall have the service type:

<service-type>=“removeCCS”

d) the *returnRemoveResult* service returns the removal status and shall have the service type:

<service-type>=“returnRemoveResult”

with the corresponding attribute:

— remove_status=“the_remove_status”

7.5 Extended Matcher Group service protocols

The *ExtendedMatcher* service matches an MSU capability profile with a required profile using a matcher and shall consist of the following steps:

a) the *requestExistingProfile* service accesses an existing profile and shall have the service type:

<service-type>=“requestExistingProfile”

with the corresponding attribute:

— profile_ID=“the_profile_id”

b) the *returnExistingProfile* service returns the requested profile and shall have the service type:

<service-type>=“returnExistingProfile”

with the corresponding attributes:

— profile_content=“the_profile_content”

— process_status=“the_process_status”

c) the *requestMatching* service matches two accessed profiles and shall have the service type:

<service-type>=“requestMatching”

with the corresponding attributes:

— profile_ID_1=“the_profile_id_1”

— profile_ID_2=“the_profile_id_2”

d) The *returnMatchingResult* service returns the matching results and shall have the service type:

<service-type>=“returnMatchingResult”

with the corresponding attributes:

— matching_level=“the_matching_level”

— matching_report=“the_matching_report”

8 Dictionary import service and protocol

8.1 The *DictionaryImporting* service

The *DictionaryImporting* service, using the *requestImportDictionary*, *returnImportDictionary, requestDictionary and returnDictionary* services, shall allow a user to import a parts library into a Repository and to review the contents in the Repository (see Figure 19).

The *DictionaryImporting* service shall consist of the following steps:

a) the dictionary user invokes the *requestImportDictionary* service of the *ImportServicePoint* object, in which the parameter associated with the *requestImportDictionary* service is dictionary ID;

b) the service provider invokes the *returnImportResult* service of the *ImportServicePoint* object, in which the parameters of the *returnImportResult* service are import result and processing error;

c) the dictionary user invokes the *requestDictionary* service of the *ImportServicePoint* object, in which the parameter of the *requestDictionary* service is dictionary ID;

d) the service provider invokes the *returnDictionary* service of the *ImportServicePoint* object, in which the parameters of the *returnDictionary* service are an existing dictionary and processing error.

8.2 The *DictionaryImporting* protocol

The *DictionaryImporting* service imports dictionaries into a Repository and shall consist of the following steps:

a) the *requestImportDictionary* service requests the importing of dictionaries and shall have the service type:

<service-type>=“requestImportingDictionary”

with the corresponding attribute:

— dictionary_ID=“the_dictionary_id”

b) the *returnImportDictionary* service returns the importing dictionaries and shall have the service type:

<service-type>=“returnImportingresult”

with the corresponding attributes:

— importing_result=“the_importing_result”

— process_status=“the_process_status”

c) the *requestDictionary* service requests a dictionary and shall have the service type:

<service-type>=“requestDictionary”

with the corresponding attribute:

— dictionary_ID=“the_dictionary_id”

d) the *returnDictionary* service returns the requested dictionary and shall have the service type:

<service-type>=“returnDictionary”

with the corresponding attributes:

— dictionary_content=“the_dictionary_content”

— process_status=“the_process_status”

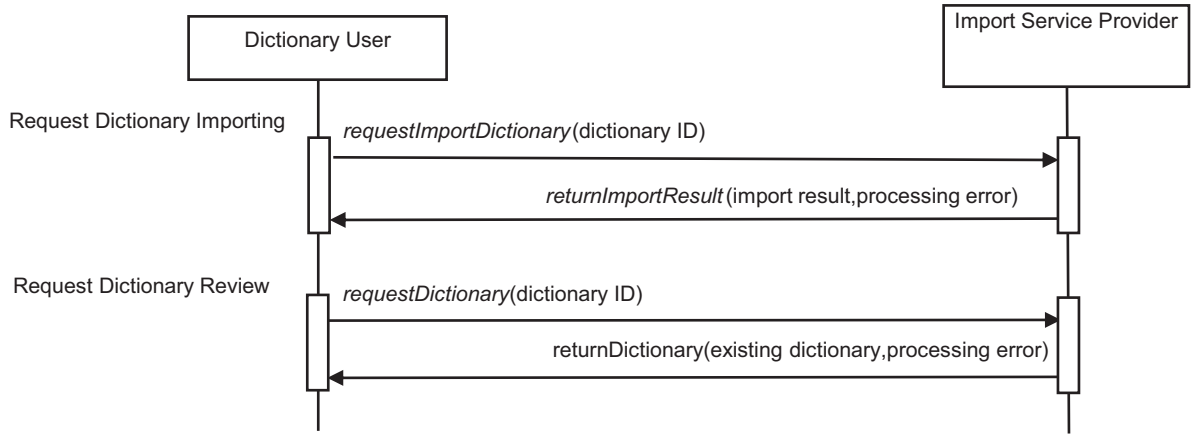


Figure 19 — The *DictionaryImporting* service

Annex A
(informative)

Capability model with MDDs

A.1 Capability model diagram

A manufacturing activity consists of one or more actions within a manufacturing process associated with a set of manufacturing functions described in ISO 16100-1:2009, 5.3. Each activity can be modeled with MDDs in accordance with ISO 15745-1.

An application activity tree has a one-to-one mapping to a capability class tree. The activity model can be mapped to the class capability model, as shown in Figure A.1.

Figure A.1 describes the following capabilities of a CCS in terms of MDDs:

a) action(s) in the activity;

b) constraints or information exchanged in the action(s);

c) resources used to support the action(s);

d) relationships between the predecessors and/or successors in the action(s).

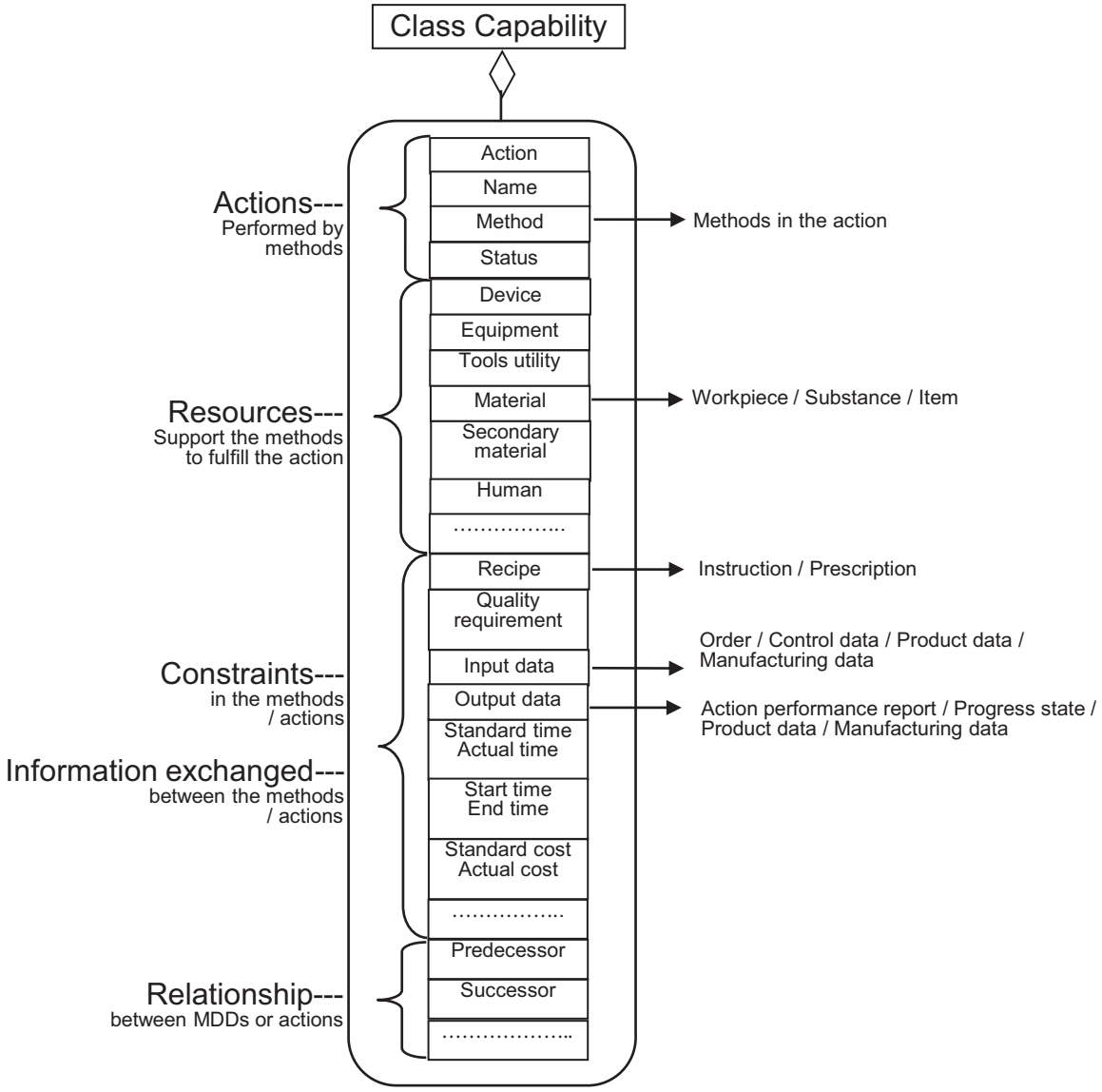


Figure A.1 — Mapping a capability model to an activity model using MDDs

A.2 XML syntax for the capability model

The following is the XML syntax for the capability model represented in the specific part of the template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CapabilityProfiling">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="type">
           <xs:complexType>
              <xs:attribute name="id" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="CapabilityProfile">
            <xs:complexType>
                <xs:sequence>
                  <xs:element name="pkgtype">
                    <xs:complexType>
                      <xs:attribute name="version" type="xs:string" form="unqualified"/>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="Common" type="CommonPartType"/>
                  <xs:element name="Specific" type="SpecificPartType"/>
                </xs:sequence>
                <xs:attribute name="date" type="xs:string" form="unqualified"/>
             </xs:complexType>
          </xs:element>
        </xs:sequence>
     </xs:complexType>
  </xs:element>
<xs:complexType name="CommonPartType">
</xs:complexType>
<xs:complexType name="SpecificPartType">
  <xs:sequence>
    <xs:element name="Reference_MDM_Name">
      <xs:complexType>
         <xs:attribute name="domain_Name" type="xs:string" form="unqualified"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="MDD_Description_Format" type="MDD_Description"/>
       <xs:complexType name="MDD_Description">
          <xs:sequence>
             <xs:choice>
                <xs:element name="Set_of_MDD_Objects">
                   <xs:complexType>
                      <xs:sequence minOccurs="0" maxOccurs="unbounded">
                         <xs:element name="MDD_Name_action">
                            <xs:attribute name="name" type="xs:string" use="required" fixed=""/>
                            <xs:attribute name="method" type="xs:string" use="required" fixed=""/>
                            <xs:attribute name="status" type="xs:string" default=""/>
                         </xs:element>
                         <xs:element name="MDD_Name_exchanged_information">
                            <xs:complexType>
                               <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                  <xs:choice>
                                     <xs:element name="information_out">
                                        <xs:complexType>
                                           <xs:attribute name="name" type="xs:string"
form="unqualified"/>
                                        </xs:complexType>
                                     </xs:element>
                                     <xs:element name="information_in">
                                        <xs:complexType>
                                           <xs:attribute name="name" type="xs:string"
form="unqualified"/>
                                        </xs:complexType>
                                     </xs:element>
                                  </xs:choice>
                               </xs:sequence>
                            </xs:complexType>
                         </xs:element>      // end of exchanged_information
                         <xs:element name="MDD_Name_constraints">
                            <xs:complexType>
                               <xs:sequence minOccurs="0" maxOccurs="unbounded">
```

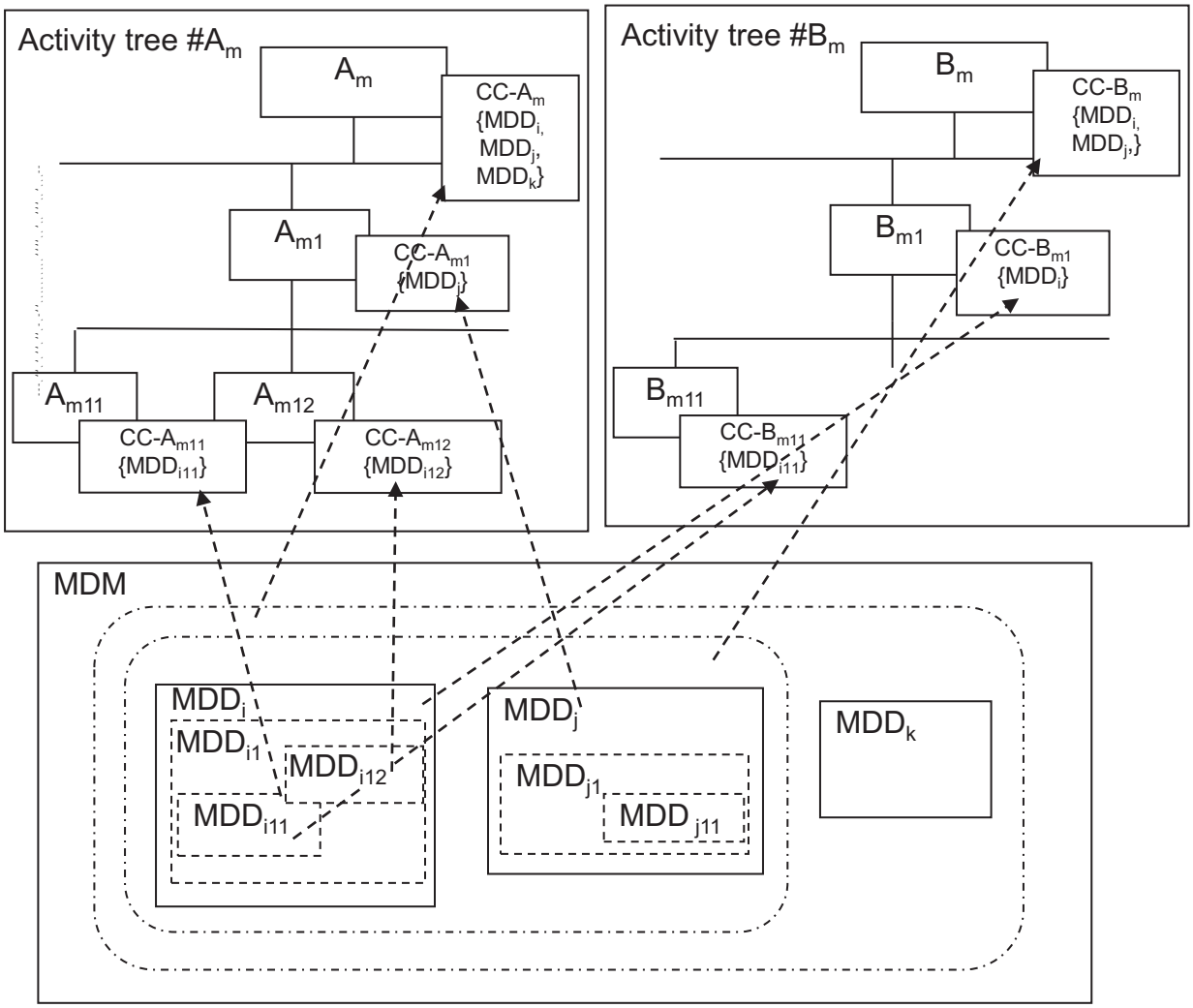
```
                                    <xs:element name="Constraint_name">
                                       <xs:complexType>
                                          <xs:attribute name="name" type="xs:string"
form="unqualified"/>
                                       </xs:complexType>
                                    </xs:element>
                                 </xs:sequence>
                              </xs:complexType>
                           </xs:element>       // end of constraints
                           <xs:element name="MDD_Name_resources">
                              <xs:complexType>
                                 <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                    <xs:element name="Resource_name">
                                       <xs:complexType>
                                          <xs:attribute name="name" type="xs:string"
form="unqualified"/>
                                       </xs:complexType>
                                       </xs:element>
                                    </xs:sequence>
                                 </xs:complexType>
                              </xs:element>     // end of resources
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>                // end of "Set_of_MDD_Objects"
                     <xs:element name="List_of_MDD_Objects">
                        ……..
                     </xs:element>
                     <xs:element name="Time_ordered_MDD_Objects ">
                        ……..
                     </xs:element>
                     <xs:element name="Event_ordered_MDD_Objects ">
                        ……..
                     </xs:element>
                  </xs:choice>
                  <xs:element name="List_of_lower_level">
                     <xs:complexType>
                        <xs:sequence minOccurs="0" maxOccurs="unbounded">
                           <xs:element name="Subactivity_name">
                              <xs:complexType>
                                 <xs:attribute name="name" type="xs:string" form="unqualified"/>
                              </xs:complexType>
                           </xs:element>
                           <xs:element name="subtemplate_name">
                               <xs:complexType>
                                  <xs:attribute name="name" type="xs:string" form="unqualified"/>
                              </xs:complexType>
                           </xs:element>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>  // end for MDD_Description
            </xs:complexType>
         </xs:sequence>
      </xs:complexType>    // end for SpecificPart
</xs:schema>
```

A.3 Relationships of MDDs and corresponding MDM

Figure A.2 shows two activity trees, A and B, with different CCSs. Each node in these trees has its own capability class and a corresponding capability profile template. Each template uses the MDDs to describe its capabilities. These MDDs are selected from the same set of MDDs in a particular MDM, i.e.

$$MDDi \in MDM \mid i \in [1..\ n]$$

Each MDD in a particular MDM is defined and distinct, with each MDD's unique identifier being the start point for semantic matching.



NOTE The labels CC-A_m, CC-A_{m1}, CC-A_{m11} and CC-A_{m12} signify the specific parts of the capability classes for activity node A_m, node A_{m1}, node A_{m11} and node A_{m12}, respectively. Similarly, CC-B_m, CC-B_{m1} and CC-B_{m11} signify the specific parts of the capability classes for activity node B_m, node B_{m1} and node B_{m11}, respectively. MDDs are elements in the specific part.

Figure A.2 — Relationships between an activity and its MDDs

Annex B
(informative)

Simplified matching of capability profile templates

B.1 Examples of capability profile templates

B.1.1 Example 1

The following is the XML syntax for the capability profile template of activity A21 (“getOperationMethod”) in ISO 16100-5:2009, Clause B.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="CapabilityProfiling">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Template">
                <xs:complexType>
                    <xs:attribute name="id" type="xs:string" use="required" fixed="A21"/>
                    <xs:attribute name="name" type="xs:string" use="required"
fixed="getOperationMethod" />
                </xs:complexType>
            </xs:element>
            <xs:element name="type">
                <xs:complexType>
                    <xs:attribute name="id" type="xs:string"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="CapabilityProfile">
                <xs:complexType>
                    <xs:sequence>
                       <xs:element name="pkgtype">
                          <xs:complexType>
                             <xs:attribute name="version" type="xs:string" form="unqualified"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="Common" type="CommonPartType"/>
                        <xs:element name="Specific" type="SpecificPartType"/>
                    </xs:sequence>
                    <xs:attribute name="date" type="xs:string" form="unqualified"/>
                </xs:complexType>
             </xs:element>
          </xs:sequence>
      </xs:complexType>
  </xs:element>
  <xs:complexType name="CommonPartType">
       ……..
       ……..
  </xs:complexType>
  <xs:complexType name="SpecificPartType">
     <xs:sequence>
        <xs:element name="Reference_MDM_Name">
            <xs:complexType>
                <xs:attribute name="domain_Name" type="xs:string" use="required"
fixed="MESApplicationDomain" form="unqualified"/>
            </xs:complexType>
         </xs:element>
         <xs:element name="format_name" type="MDD_Description"/>
      </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MDD_Description">
    <xs:sequence>
      <xs:element name="Set_Of_MDD_Objects">
        <xs:complexType>
          <xs:sequence>
```

```
            <xs:element name="action1">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
                          <xs:element name="information_out">
                               <xs:complexType>
                                   <xs:attribute name="name" use="required" type="xs:string"
form="unqualified" fixed="operation_type"/>
                               </xs:complexType>
                          </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <xs:element name="Resources">
                       ……
                       ……
                  </xs:element>
                  <xs:element name="Constraints">
                       ……
                       ……
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required" fixed="setOperationType"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Set"/>
                <xs:attribute name="status" type="xs:string" default="mandatory"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="action2">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                       <xs:sequence minOccurs="1" maxOccurs="1">
                          <xs:element name="information_in">
                              <xs:complexType>
                                  <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="operation_method(plan)"/>
                              </xs:complexType>
                          </xs:element>
                       </xs:sequence>
                     </xs:complexType>
                   </xs:element>
                   <xs:element name="Resources">
                     ……
                     ……
                  </xs:element>
                  <xs:element name="Constraints">
                      ……
                      ……
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required" fixed="getOperationMethod"
form="unqualified"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Get"
form="unqualified"/>
                <xs:attribute name="status" type="xs:string" default="optional"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="action3">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
                          <xs:element name="information_in">
                              <xs:complexType>
                                  <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="recipe(plan)"/>
                              </xs:complexType>
                          </xs:element>
                       </xs:sequence>
                    </xs:complexType>
                  </xs:element>
```

```
                    <xs:element name="Resources">
                       ……..
                       ……..
                    </xs:element>
                    <xs:element name="Constraints">
                        ……..
                        ……..
                    </xs:element>
                 </xs:sequence>
                 <xs:attribute name="name" type="xs:string" use="required" fixed="getRecipe"
form="unqualified"/>
                 <xs:attribute name="method" type="xs:string" use="required" fixed="Get"
form="unqualified"/>
                 <xs:attribute name="status" type="xs:string" default="optional"/>
              </xs:complexType>
           </xs:element>
        </xs:sequence>
     </xs:complexType>
  </xs:element>
  <xs:element name="List_of_lower_level">
       ……
  </xs:element>
 </xs:sequence>
</xs:complexType>
</xs:schema>
```

B.1.2 Example 2

The following is the XML syntax for the capability profile template of activity B11 (“receiveManufacturingInstruction”) in ISO 16100-5:2009, Clause B.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="CapabilityProfiling">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Template">
               <xs:complexType>
                  <xs:attribute name="id" type="xs:string" use="required" fixed="B11"/>
                  <xs:attribute name="name" type="xs:string" use="required"
fixed="receiveManufacturingInstruction" />
               </xs:complexType>
            </xs:element>
            <xs:element name="type">
                <xs:complexType>
                   <xs:attribute name="id" type="xs:string"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="CapabilityProfile">
                <xs:complexType>
                   <xs:sequence>
                      <xs:element name="pkgtype">
                         <xs:complexType>
                            <xs:attribute name="version" type="xs:string" form="unqualified"/>
                         </xs:complexType>
                      </xs:element>
                      <xs:element name="Common" type="CommonPartType"/>
                      <xs:element name="Specific" type="SpecificPartType"/>
                   </xs:sequence>
                   <xs:attribute name="date" type="xs:string" form="unqualified"/>
                </xs:complexType>
             </xs:element>
         </xs:sequence>
      </xs:complexType>
  </xs:element>
  <xs:complexType name="CommonPartType">
     …….
     …….
  </xs:complexType>
  <xs:complexType name="SpecificPartType">
     <xs:sequence>
        <xs:element name="Reference_MDM_Name">
```

```
            <xs:complexType>
                <xs:attribute name="domain_Name" type="xs:string" use="required"
fixed="MESApplicationDomain" form="unqualified"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="format_name" type="MDD_Description"/>

    </xs:sequence>
 </xs:complexType>
 <xs:complexType name="MDD_Description">
    <xs:sequence>
      <xs:element name="Set_Of_MDD_Objects">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="action1">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
                           <xs:element name="information_in">
                               <xs:complexType>
                                  <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="product_order(plan)"/>
                               </xs:complexType>
                           </xs:element>
</xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <xs:element name="Resources">
…
…
                  </xs:element>
                  <xs:element name="Constraints">
                  …
…
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required" fixed="getProductOrder"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Get"/>
                <xs:attribute name="status" type="xs:string" default="mandatory"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="action2">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="information_in">
                              <xs:complexType>
                                  <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="operating_instruction(plan)"/>
                              </xs:complexType>
                          </xs:element>
                       </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <xs:element name="Resources">
                     ……………………………………….
                     …………………………………..…
                  </xs:element>
                  <xs:element name="Constraints">
                     ……………………………………..…
                     ……………………………………….…
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required"
fixed="getOperatingInstruction" form="unqualified"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Get"
form="unqualified"/>
                <xs:attribute name="status" type="xs:string" default="optional"/>
              </xs:complexType>
            </xs:element>
```

```
            <xs:element name="action3">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
                          <xs:element name="information_in">
                                <xs:complexType>
                                    <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="item"/>
                                </xs:complexType>
                           </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <xs:element name="Resources">
                      ……
                      ……
                  </xs:element>
                  <xs:element name="Constraints">
                       ……
                       ……
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required" fixed="getItem"
form="unqualified"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Get"
form="unqualified"/>
                <xs:attribute name="status" type="xs:string" default="mandatory"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="List_of_lower_level">
          ……
          ……
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

B.1.3 Example 3

The following is the XML syntax for the capability profile template of activity A33 (“monitorOperationCondition”) in ISO 16100-5:2009, Clause B.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="CapabilityProfiling">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Template">
                    <xs:complexType>
                        <xs:attribute name="id" type="xs:string" use="required" fixed="A33"/>
                        <xs:attribute name="name" type="xs:string" use="required"
fixed="MonitorOperationCondition" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="type">
                    <xs:complexType>
                        <xs:attribute name="id" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="CapabilityProfile">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="pkgtype">
                                <xs:complexType>
                                    <xs:attribute name="version" type="xs:string"
form="unqualified"/>
                                </xs:complexType>
```

```
                            </xs:element>
                            <xs:element name="Common" type="CommonPartType"/>
                            <xs:element name="Specific" type="SpecificPartType"/>
                        </xs:sequence>
                        <xs:attribute name="date" type="xs:string" form="unqualified"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="CommonPartType">
    </xs:complexType>
    <xs:complexType name="SpecificPartType">
        <xs:sequence>
            <xs:element name="Reference_MDM_Name">
                <xs:complexType>
                    <xs:attribute name="domain_Name" type="xs:string" use="required"
fixed="MESApplicationDomain" form="unqualified"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="format_name" type="MDD_Description"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MDD_Description">
    <xs:sequence>
      <xs:element name="Set_Of_MDD_Objects">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="action1">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
                          <xs:element name="information_out">
                               <xs:complexType>
                                    <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="actual_equipment"/>
                               </xs:complexType>
                           </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <xs:element name="Resources">
                     ……..…
                     ……..…
                  </xs:element>
                  <xs:element name="Constraints">
                      ……..…
                       ……..…
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required"
fixed="setActualEquipment"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Set"/>
                <xs:attribute name="status" type="xs:string" default="mandatory"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="action2">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="exchanged_information">
                    <xs:complexType>
                      <xs:sequence minOccurs="1" maxOccurs="1">
                           <xs:element name="information_in">
                                <xs:complexType>
                                    <xs:attribute name="name" type="xs:string" use="required"
form="unqualified" fixed="state(result)"/>
                                </xs:complexType>
                           </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>

                  <xs:element name="Resources">
```

```
                    ……
                    ……
                  </xs:element>
                  <xs:element name="Constraints">
                      ……
…
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="required" fixed="getState"
form="unqualified"/>
                <xs:attribute name="method" type="xs:string" use="required" fixed="Get"
form="unqualified"/>
                <xs:attribute name="status" type="xs:string" default="optional"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="List_of_lower_level">
        <xs:complexType>
          <xs:sequence minOccurs="1" maxOccurs="4">
            <xs:element name="subactivity_name1">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A331"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subtemplate_name1">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A331"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subactivity_name2">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A332"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subtemplate_name2">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A332"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subactivity_name3">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A333"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subtemplate_name3">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A333"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subactivity_name4">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A334"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="subtemplate_name4">
              <xs:complexType>
                <xs:attribute name="id" type="xs:string" use="required" fixed="A334"
form="unqualified"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

B.2 Procedure for generating a capability profile template

The procedure for generating a capability profile template according to the formal structure of a capability profile template is shown in Figure B.1.

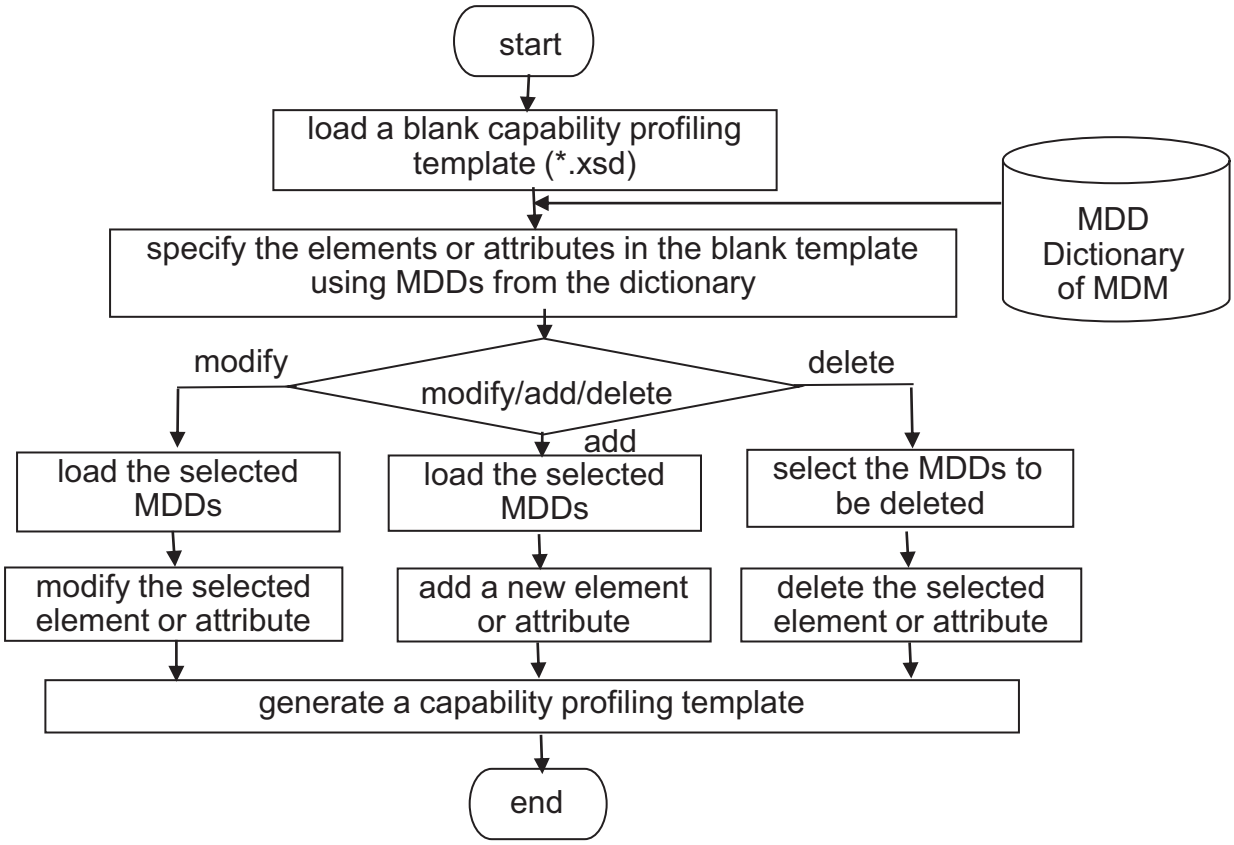
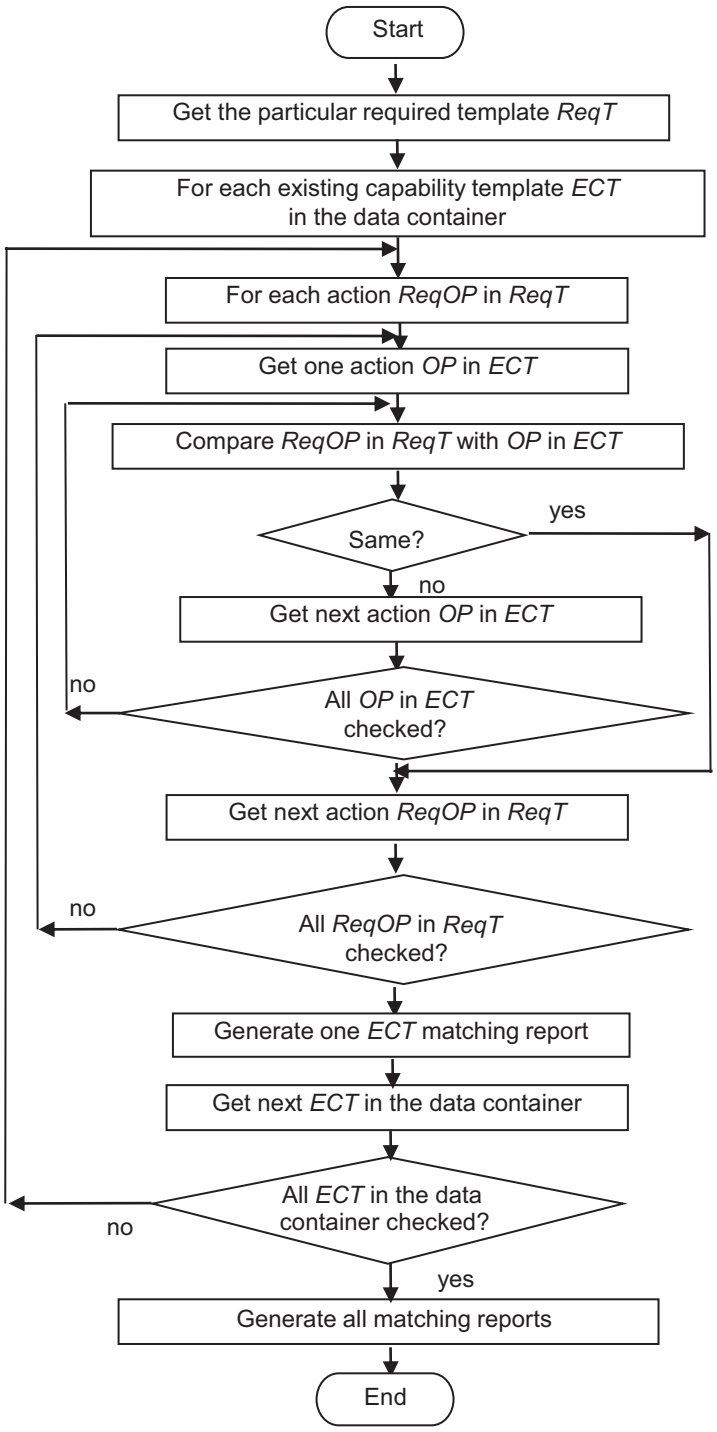


Figure B.1 — Procedure for generating a capability profile template

B.3 Process for selecting a proper capability profile template

The process for selecting a proper capability profile template in the repository in accordance with the required template is shown in Figure B.2.



Key

ECT	Existing Capability Profile Template
Req T	Required Template
ReqOP	Required Operation
OP	Operation

Figure B.2 — Procedure for selecting a capability profile template in the repository

Annex C
(informative)

Profiles based on capability profile templates

C.1 Profile of “getOperationMethod”

The following is the XML syntax for the profile of activity A21 (“getOperationMethod”) in ISO 16100-5:2009, Clause B.2.

```
<?xml version="1.0" encoding="utf-8" ?>
<CapabilityProfiling xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation="D:\newDB\MonitorOperationCondition.xsd">
  <Template id="A21" name="getOperationMethod" />
  <type id="MSU profile" />
  <CapabilityProfile>
    <pkgtype version="1.1.1" />
    <Common>
      <MSU_Capability>
        <ID>getOperationMethod</ID>
      </MSU_Capability>
      <ReferenceCapabilityClassStructure id="MESTreeA" />
      <Capability_Class_Name name="getOperationMethod" />
      <Reference_Capability_Class_Structure_Name name="MESTreeA" />
    </Common>
    <Specific>
      <Reference_MDM_Name domain_Name="MESApplicationDomain" />
      <format_name>
        <Set_Of_MDD_Objects>
          <action1 name="setOperationType" method="Set" status="mandatory">
            <exchanged_information>
              <information_out_in>
                <information_out name="operation_type"/>
              </information_out_in>
            </exchanged_information>
            <Resources/>
            <Constraints/>
          </action1>
          <action2 name="getOperationMethod" method="Get" status="optional">
            <exchanged_information>
              <information_out_in>
                <information_in name="operation_method(plan)" />
              </information_out_in>
            </exchanged_information>
            <Resources />
            <Constraints />
          </action2>
          <action3 name="getRecipe" method="Get" status="optional">
            <exchanged_information>
              <information_out_in>
                <information_in name="recipe(plan)" />
              </information_out_in>
            </exchanged_information>
            <Resources />
            <Constraints />
          </action3>
        </Set_Of_MDD_Objects>
        <List_of_lower_level />
      </format_name>
    </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

C.2 Profile of “receiveManufacturingInstruction”

The following is the XML syntax for the profile of activity B11 (“receiveManufacturingInstruction”) in ISO 16100-5:2009, Clause B.3.

```
<?xml version="1.0" encoding="utf-8" ?>
<CapabilityProfiling xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation="D:\newDB\MonitorOperationCondition.xsd">
  <Template id="B11" name="receiveManufacturingInstruction" />
  <type id="MSU profile" />
  <CapabilityProfile>
    <pkgtype version="1.1.1" />
    <Common>
      <MSU_Capability>
        <ID>receiveManufacturingInstruction</ID>
      </MSU_Capability>
      <ReferenceCapabilityClassStructure id="MESTreeB" />
      <Capability_Class_Name name="receiveManufacturingInstruction" />
      <Reference_Capability_Class_Structure_Name name="MESTreeB" />
    </Common>
    <Specific>
    <Reference_MDM_Name domain_Name="MESApplicationDomain" />
      <format_name>
        <Set_Of_MDD_Objects>
          <action1 name="getProductOrder" method="Get" status="mandatory">
            <exchanged_information>
              <information_out_in>
                <information_in name="product_order(plan)" />
              </information_out_in>
            </exchanged_information>
            <Resources />
            <Constraints />
          </action1>
          <action2 name="getOperatingInstruction" method="Get" status="optional">
            <exchanged_information>
              <information_out_in>
                <information_in name="operating_instruction(plan)" />
              </information_out_in>
            </exchanged_information>
            <Resources />
            <Constraints />
          </action2>
          <action3 name="getItem" method="Get" status="optional">
            <exchanged_information>
              <information_out_in>
                 <information_in name="item" />
               </information_out_in>
            </exchanged_information>
            <Resources />
            <Constraints />
          </action3>
        </Set_Of_MDD_Objects>
        <List_of_lower_level />
      </format_name>
    </Specific>
  </CapabilityProfile>
</CapabilityProfiling>
```

C.3 Profile of “monitorOperationCondition”

The following is the XML syntax for the profile of activity A33 (“monitorOperationCondition”) in ISO 16100-5:2009, Clause B.2.

```
<?xml version="1.0" encoding="utf-8" ?>
<CapabilityProfiling xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation="D:\newDB\MonitorOperationCondition.xsd">
  <Template id="A33" name="MonitorOperationCondition" />
  <type id="MSU profile" />
  <CapabilityProfile>
    <pkgtype version="1.1.1" />
    <Common>
    <MSU_Capability>
      <ID>MSUMonitorOperationCondition</ID>
    </MSU_Capability>
    <ReferenceCapabilityClassStructure id="MESTreeA" />
    <Capability_Class_Name name="MonitorOperationCondition" />
    <Reference_Capability_Class_Structure_Name name="MESTreeA" />
  </Common>
  <Specific>
    <Reference_MDM_Name domain_Name="MESApplicationDomain" />
    <format_name>
      <Set_Of_MDD_Objects>
        <action1 name="setMonitorCondition" method="Set" status="mandatory">
          <exchanged_information>
            <information_out_in>
              <information_out name="actual equipment1" />
            </information_out_in>
          </exchanged_information>
          <Resources />
          <Constraints />
        </action1>
        <action2 name="getMonitorCondition" method="Get" status="optional">
          <exchanged_information>
            <information_out_in>
              <information_in name="state1" />
            </information_out_in>
          </exchanged_information>
           <Resources />
           <Constraints />
         </action2>
       </Set_Of_MDD_Objects>
       <List_of_lower_level>
         <subactivity_name1 id="A331" />
         <subtemplate_name1 id="A331" />
         <subactivity_name2 id="A332" />
         <subtemplate_name2 id="A332" />
         <subactivity_name3 id="A333" />
         <subtemplate_name3 id="A333" />
         <subactivity_name4 id="A334" />
         <subtemplate_name4 id="A334" />
       </List_of_lower_level>
     </format_name>
   </Specific>
 </CapabilityProfile>
</CapabilityProfiling>
```

Annex D
(informative)

Procedure for generating a capability class structure

The procedure for generating a CCS is shown in Figure D.1.

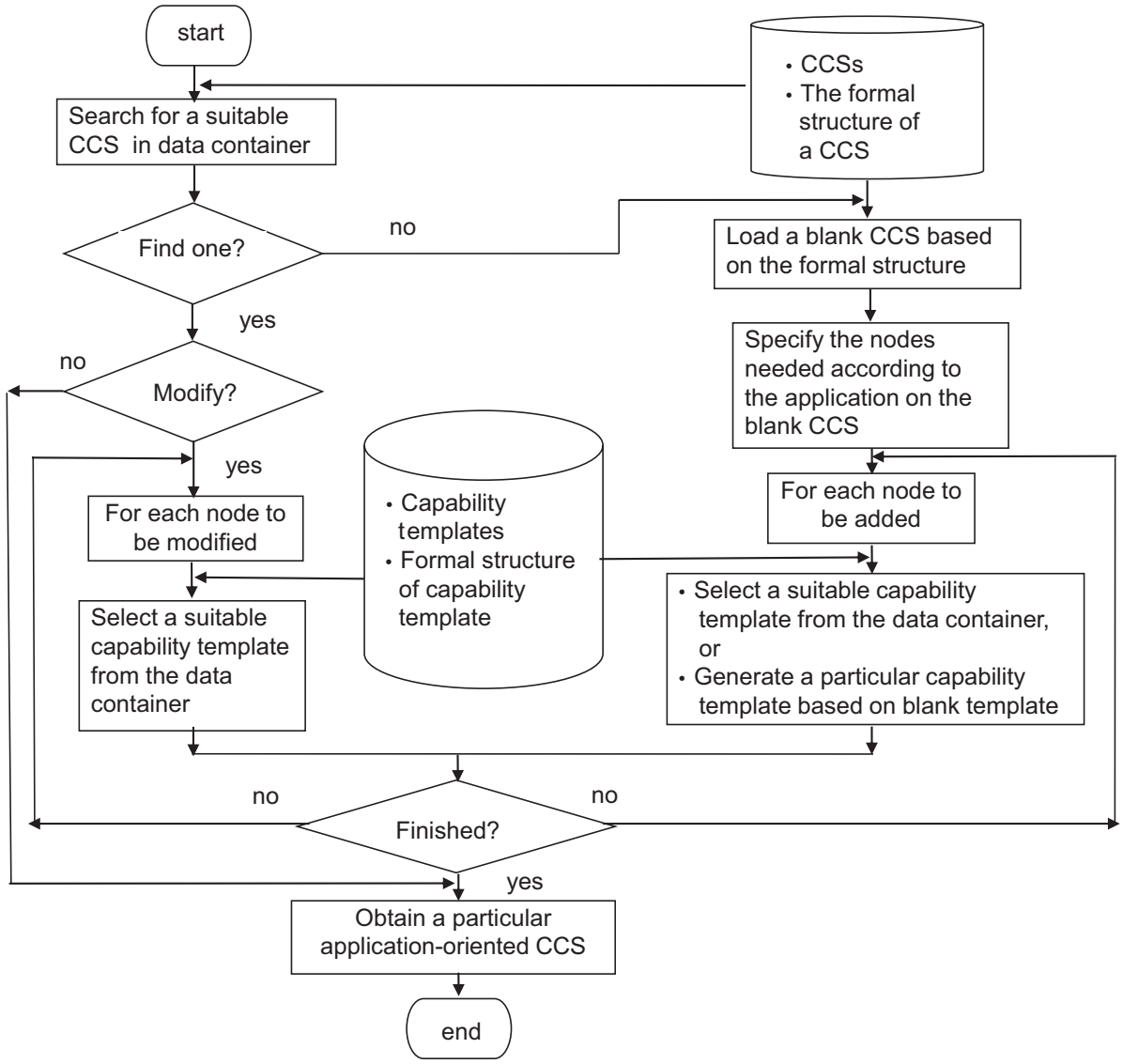


Figure D.1 — Procedure for generating a particular CCS

Annex E
(informative)

Mapping Parts Library (PLIB) to MDDs

E.1 Benefits of mapping

A PLIB dictionary, which can be machine-readable, includes definitions of part family descriptions (PLIB class) and attribute (known as “property” in ISO 13584). This information is contained in a code known as the Basic Semantic Unit (BSU).

Multi-language labels can be added to one BSU entry to support a multi-language definition. By use of this mechanism, the capability profiles can use different attribute names (PLIB property names) as the same definition. The matching mechanism will then simply be a comparison of machine-readable code since the BSU is also machine readable. The benefit is that such a comparison is easier than a comparison between a BSU and the structure of an XML object.

Figure E.1 shows a relationship between PLIB and MDDs.

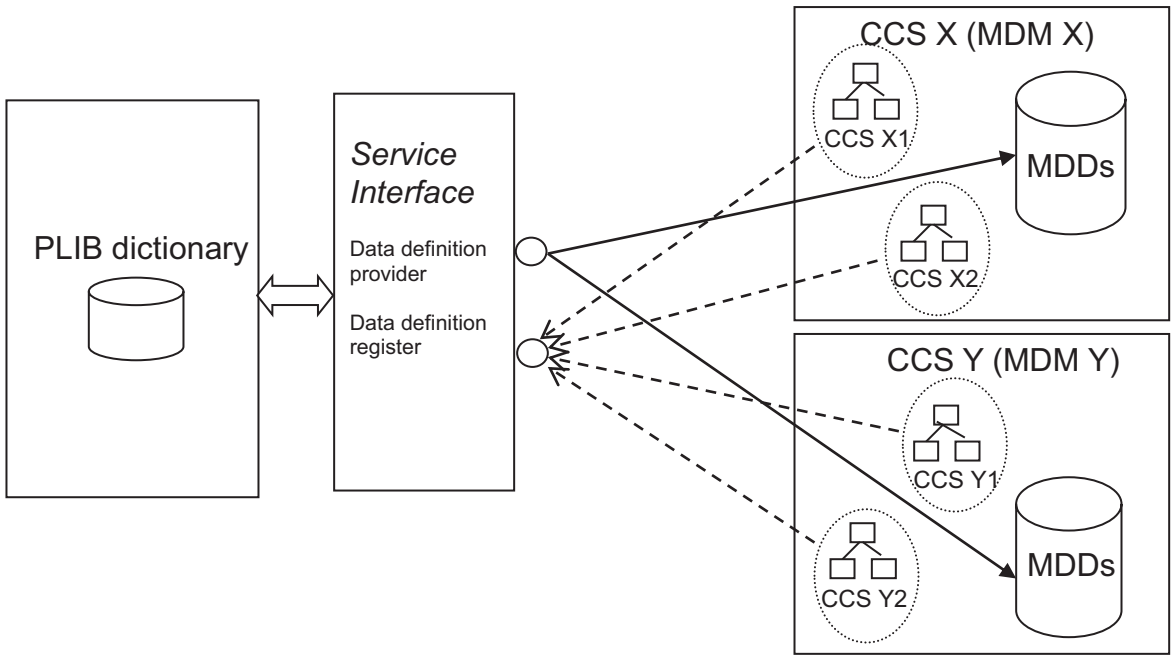


Figure E.1 — Relationship between PLIB and MDDs through a service interface

Figure E.2 shows two MDDs from one PLIB dictionary. The MDDs have the same BSU code, but different names. The compare mechanism recognizes that the MDDs are the same because of the same BSU code.

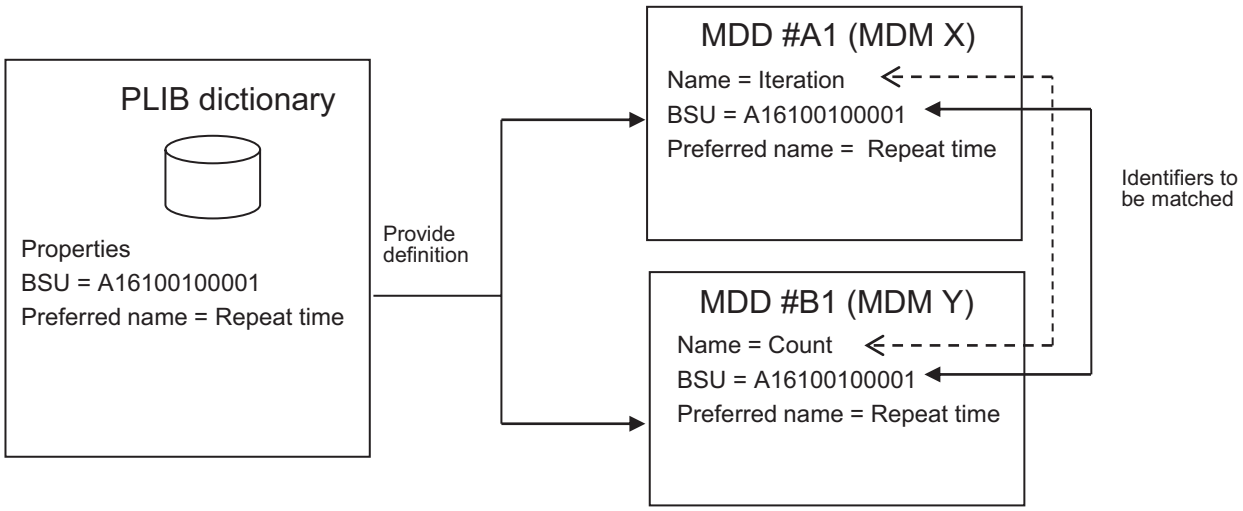


Figure E.2 — Example of comparing MDDs using PLIB-linked concept identifiers

E.2 Mapping example

The process for creating a particular capability profile template begins with the formal MDD template structure described in ISO 16100-5:2009, 6.5.2. The particular MDD template is created after a subset of PLIB-based MDDs are mapped to the formal MDD structure.

The following is the XML schema for an example of a particular MDD template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="MDD">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="MDD_Name">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" form="process order"/>
                        <xs:attribute name="DICTIONARY_ID" type="xs:string" form="9999/ISO16100-
XXX"/>
                        <xs:attribute name="PARENT" type="xs:string" form="F16100002E"/>
                        <xs:attribute name="BSU" type="xs:string" form="F16100001C"/>
                        <xs:attribute name="PREFERRED_NAME" language="en" type="xs:string"
form="process order"/>
                        <xs:attribute name="VERSION" type="xs:string" form="001" />
                        <xs:attribute name="REVISION" type="xs:string" form="001" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="Reference_MDM_Name">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" form="unqualified"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="List_Of_Attributes">
                    <xs:complexType>
                        <xs:sequence minOccurs="0" maxOccurs="unbounded">
                            <xs:element name="Attribute">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Attribute_Name">
                                            <xs:complexType>
                                                <xs:attribute name="name" type="xs:string"
form="Source"/>
                                                <xs:attribute name="DICTIONARY_ID" type="xs:string"
form="9999/ISO16100-XXX"/>
                                                <xs:attribute name="PARENT" type="xs:string"
form="F16100001"/>
```

```
                                                    <xs:attribute name="BSU" type="xs:string"
form="A161000001"/>
                                                    <xs:attribute name="PREFERRED_NAME" language="en"
type="xs:string" form="Source"/>
                                                    <xs:attribute name="PREFERRED_NAME" language="ja"
type="xs:string" form="Japanese-Source-Name"/>
                                                    <xs:attribute name="VERSION" type="xs:string"
form="001" />
                                                    <xs:attribute name="REVISION" type="xs:string"
form="001" />
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="Attribute_Type">
                                                <xs:complexType>
                                                    <xs:attribute name="type" type="xs:string"
form="unqualified"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                        <xs:attribute name="id" type="xs:string" form="unqualified"/>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="Attribute">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="Attribute_Name">
                                                <xs:complexType>
                                                    <xs:attribute name="name" type="xs:string"
form="Destination"/>
                                                    <xs:attribute name="DICTIONARY_ID" type="xs:string"
form="9999/ISO16100-XXX"/>
                                                    <xs:attribute name="PARENT" type="xs:string"
form="F16100001"/>
                                                    <xs:attribute name="BSU" type="xs:string"
form="A161000002"/>
                                                    <xs:attribute name="PREFERRED_NAME" language="en"
type="xs:string" form="Destination"/>
                                                    <xs:attribute name="PREFERRED_NAME" language="ja"
type="xs:string" form="Japanese-Destination-Name"/>
                                                    <xs:attribute name="VERSION" type="xs:string"
form="001" />
                                                    <xs:attribute name="REVISION" type="xs:string"
form="001" />
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="Attribute_Type">
                                                <xs:complexType>
                                                    <xs:attribute name="type" type="xs:string"
form="unqualified"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                        <xs:attribute name="id" type="xs:string" form="unqualified"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
</xs:schema>
```

Annex F
(informative)

Mapping OTD to MDDs

F.1 Benefits of mapping

An open technical dictionary (OTD) is a machine-readable collection of entries for various types of concepts, including classes, properties, controlled property values, units of measure, qualifiers of measure and currencies. Each entry contains, at a minimum, an unambiguous identifier, a term and a definition. An identification guide can be used to specify, for a particular domain of interest, the minimum set of properties required to describe a member of a class as well as any constraints or relationships between the properties.

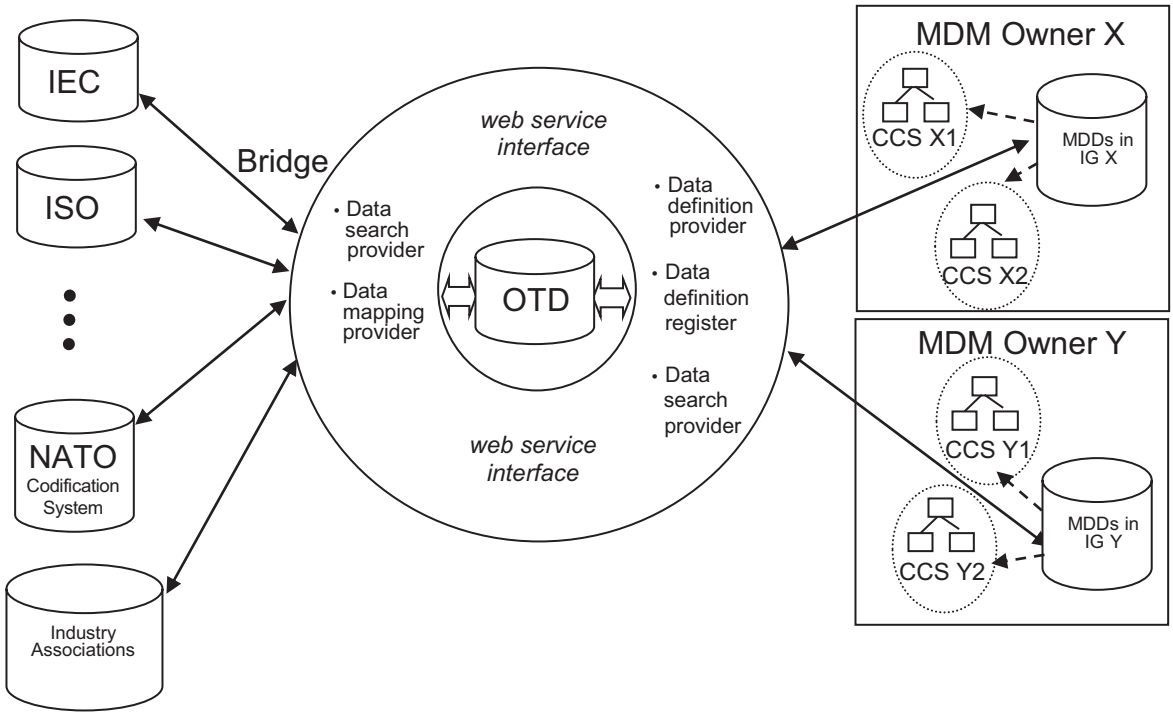
In accordance with ISO 22745, an item is described by the class to which it belongs and a set of property-value pairs. In an ISO 22745-compliant item description, each concept is represented by its unambiguous identifier from the OTD. ISO 22745 is classification-neutral.

A description of an ISO 16100 MDD that represents information about an activity, resource, software capability or relationship between MDDs can be encoded as an ISO 22745-compliant item description, using concepts defined in an OTD. An MSU capability or an application-required capability can be expressed in terms of OTD-based MDDs. An ISO 22745 item description can be linked to equivalent item descriptions, in both ISO 22745 and other standards (such as ISO 13584).

To match two OTD-based MDDs, a comparison of unambiguous concept identifiers is performed. If two OTD concept identifiers represent the same concept, this fact should be recorded in the OTD using a concept equivalence relationship. The identifiers are either explicitly given in the MDDs or are extracted via links.

Figure F.1 shows a relationship between MDDs and an OTD.

Figure F.2 shows two MDDs (#A1 refers to an MDD of capability class A1; #B1 refers to an MDD of capability class B1) that reference the same OTD item global identifier and the same preferred name. The two MDDs with different names in their respective dictionaries are represented by their corresponding OTD terms in the shared OTD resource (e.g. server). The shared global identifier is the common bridge between the individual dictionaries for matching MDDs.



Key

IG Identification Guide (see ISO 22745-2)

◄- - - a subset of MDDs can be used in the CCS

◄——► bi-directional information exchange

Figure F.1 — Relationship between OTD and MDDs through a service interface

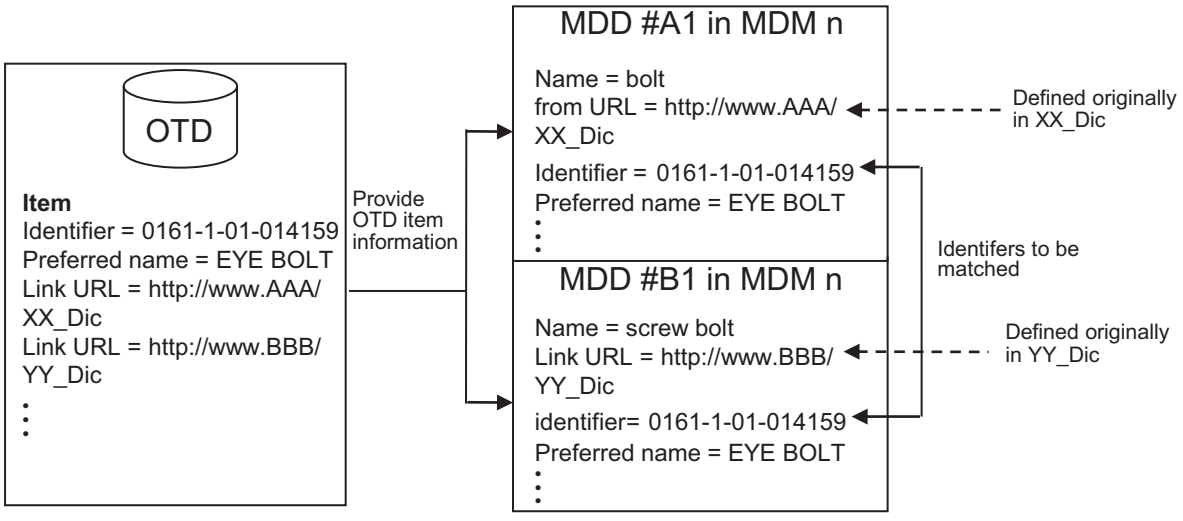


Figure F.2 — Example of comparing MDDs using OTD-linked concept identifiers

F.2 Mapping example

As in Clause E.2, following the formal MDD template structure described in ISO 16100-5:2009, 6.5.2, an example of a capability profile template begins with creating an MDD template using a subset of OTD-based MDDs that are expressed in terms of elements defined in an OTD that pertain to a specific MDM.

The following is the XML schema for an example of a particular MDD template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="MDD">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="MDD_Name">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" form="qualified" fixed="process
                    operation"/>
                        <xs:attribute name="GlobalItemIdentifier" type="xs:string" form="qualified"
                    fixed="process_MDM_IGxxx"/>
                        <xs:attribute name="GlobalItemCategory" type="xs:string" form="qualified"
                    fixed=" process_MDM_IGxxx "/>
                        <xs:attribute name="Title" type="xs:string" form="qualified" fixed="
                    process_MDM_IGxxx "/>
                        <xs:attribute name="Definition" type="xs:string" form="qualified" fixed="
                    process_MDM_IGxxx "/>
                        <xs:attribute name="DateAdded" type="xs:string" form="qualified" fixed="
                    process_MDM_IGxxx "/>
                        <xs:attribute name="SecretariatReference" type="xs:string" form="qualified"
                    fixed=" process_MDM_IGxxx "/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Reference_MDM_Name">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" form="unqualified"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="List_Of_Attributes">
                    <xs:complexType>
                        <xs:sequence minOccurs="0" maxOccurs="unbounded">
                            <xs:element name="Attribute">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Attribute_Name">
                                            <xs:complexType>
                                                <xs:attribute name="AttributeTitle"
                                            type="xs:string" form="qualified" fixed="
                                            process_MDM_IGxxx "/>
                                                <xs:attribute name="AttributeDefinition"
                                            type="xs:string" form="qualified" fixed="
                                            process_MDM_IGxxx "/>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="Attribute_Type">
                                            <xs:complexType>
                                            <xs:attribute name="GlobalAttributeCategory"
                                        type="xs:string" form="qualified"
                                        fixed="process_MDM_IGxxx"/>
                                            <xs:attribute name="AttributeDateAdded"
                                        type="xs:string" form="qualified" fixed=" process_MDM_IGxxx
                                        "/>
                                            <xs:attribute name="AttributeSecretariatReference"
                                        type="xs:string" form="qualified" fixed=" process_MDM_IGxxx
                                        "/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="GlobalAttributeIdentifier" type="xs:string"
                                form="qualified" fixed="process_MDM_IGxxx"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:schema>
```

Annex G
(informative)

Procedure for matching two profiles

The Extended Matcher Group service set can be used to match capability profiles based on multiple CCSs. The purpose of matching two profiles is to select a suitable MSU from the Repository according to a required capability profile for a given activity in the same manufacturing application.

The procedure of “Capability profile matching procedure” is shown in ISO 16100-5:2009, Figure 12. The five steps in that procedure are as follows:

— step 1: compare two dictionary IDs of two profiles;

— step 2: compare two MDM names;

— step 3: compare two definition formats;

— step 4: convert MDD objects into the same definition;

— step 5: compare one by one the capability definition in the required capability profile with those in the MSU capability profiles.

In each step, an MDD object's unique identifier (unique within one MDM) is the starting point for the semantic comparison among MDD objects. The “compare the capability definition” (step 5) focuses on the comparison of “MDD_Description”. The procedure (e.g. “List_of_MDD_Objects”) of the comparison is shown in Figure G.1.

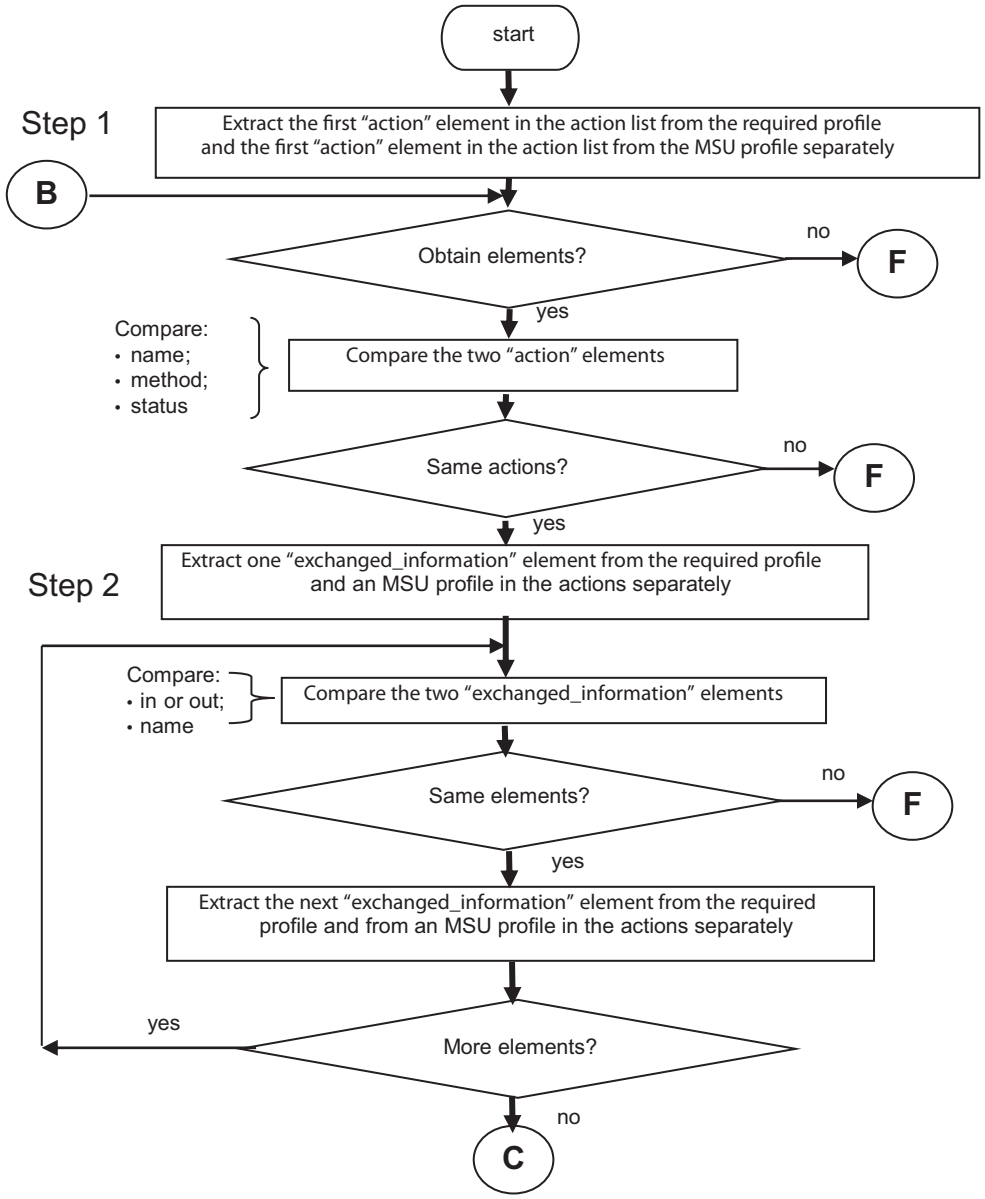
In Figure G.1, there are two kinds of loops: the outer loop and the inner loop. In the outer loop, loop B is used to compare the actions according to the order in the required profile. The same rules are used in “Time_Order_Of_MDD_Objects” and “Event_Order_Of_MDD_Objects”. The “action” elements are ordered according to the time sequence in “Time_Order_Of_MDD_Objects” of “MDD_Description”. Likewise, the “action” elements are ordered according to the event sequence in “Event_Order_Of_MDD_Objects” of “MDD_Description”. The comparisons of the “action” elements are strictly done according to the requirement order. The exception to this is for “Set_Of_MDD_Objects”, where each action element in the required profile is sequentially compared to all the action elements in the MSU profile until there is a matching success or failure.

The inner loop actually consists of four loop Bs. The first inner loop is used to compare the two MDD_Name_action elements from the two individual profiles. According to the formal structure shown in Figure 2, the comparisons of two action elements are followed by action names, methods and statuses.

The second inner loop is used to compare the MDD_Name_exchanged_information elements in the two actions. According to the formal structure shown in Figure 2, each exchanged_information element in the required profile is sequentially compared to all exchanged_information elements in the MSU profile until there is a matching success or failure. Because there might be several exchanged_information elements in one action, each exchanged_information comparison is followed by information_in or information_out and the element name.

The third inner loop is used to compare the MDD_Name_constraint elements in the two actions. According to the formal structure shown in Figure 2, each constraint element in the required profile is sequentially compared to all constraint elements in the MSU profile until there is a matching success or failure. Because there might be several constrains in one action, each constraint comparison is followed by the constraint name.

The fourth inner loop is used to compare the MDD_Name_resource elements in the two actions. According to the formal structure shown in Figure 2, each resource element in the required profile is sequentially compared to all resource elements in the MSU profile until there is a matching success or failure. Because there might be several resources in one action, each resource comparison is followed by the resource name.



NOTE A, B, C and F are procedure connection points, with F signifying a matching failure.

Figure G.1 (*continued*)

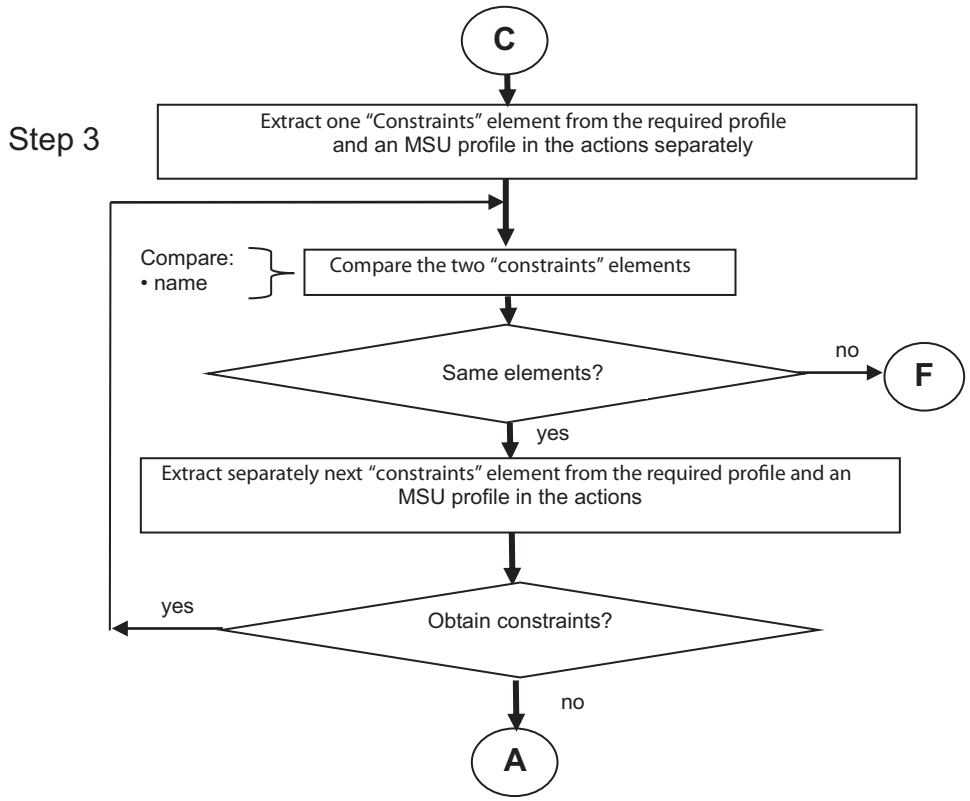


Figure G.1 (*continued*)

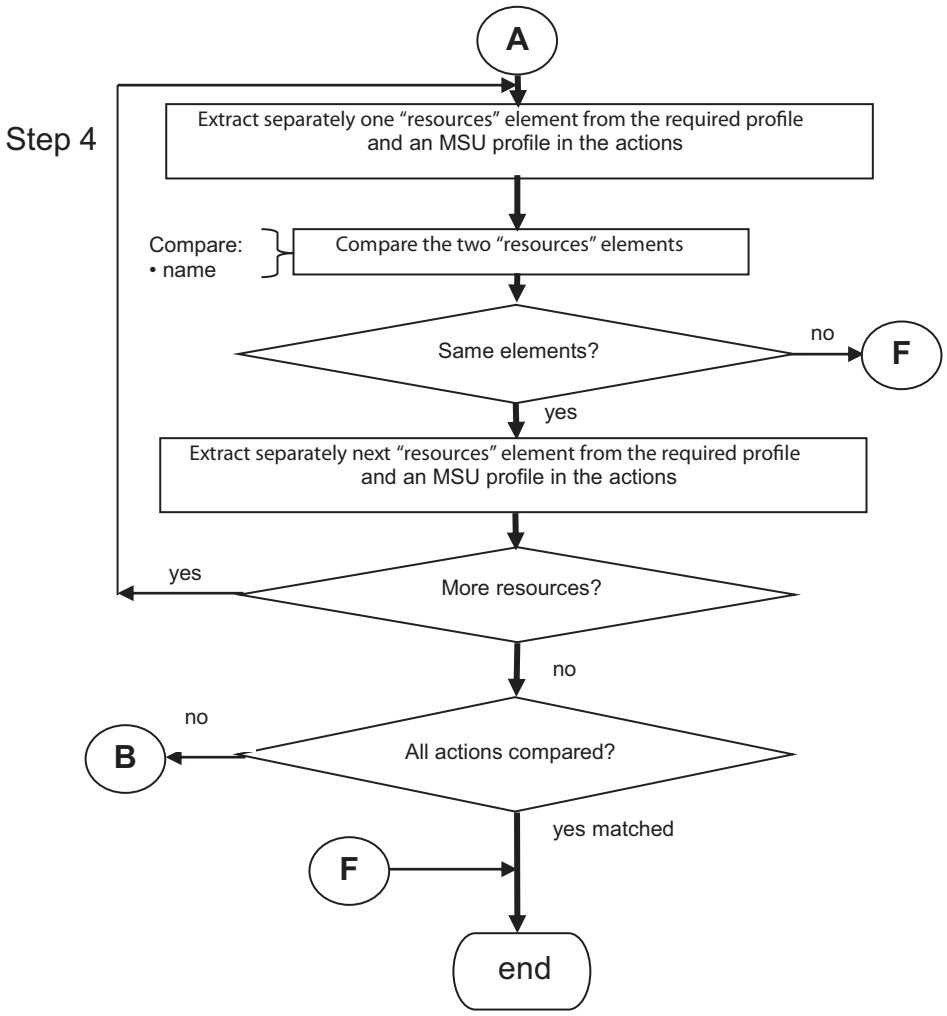


Figure G.1 — Procedure for “compare MDD_description” in a Type 2 matcher

Bibliography

[1] ISO/IEC 11179-5, *Information technology — Metadata registries (MDR) — Part 5: Naming and identification principles*

[2] ISO 13584-24, *Industrial automation systems and integration — Parts library — Part 24: Logical resource: Logical model of supplier library*

[3] ISO 13584-26, *Industrial automation systems and integration — Parts library — Part 26: Logical resource: Information supplier identification*

[4] ISO 13584-42, *Industrial automation systems and integration — Parts library — Part 42: Description methodology: Methodology for structuring parts families*

[5] ISO 15745-1, *Industrial automation systems and integration — Open systems application integration framework — Part 1: Generic reference description*

[6] ISO/IEC 19501, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[7] ISO 22745-1, *Industrial automation systems and integration — Open technical dictionaries and their application to master data — Part 1: Overview and fundamental principles*

[8] ISO 22745-2, *Industrial automation systems and integration — Open technical dictionaries and their application to master data — Part 2: Vocabulary*

[9] ISO 22745-13, *Industrial automation systems and integration — Open technical dictionaries and their application to master data — Part 13: Identification of concepts and terminology*

[10] ISO/TS 22745-35, *Industrial automation systems and integration — Open technical dictionaries and their application to master data — Part 35: Query for characteristic data*

[11] ISO/TS 29002-5, *Industrial automation systems and integration — Exchange of characteristic data — Part 5: Identification scheme*

[12] RFC 2276, *Architectural Principles of Uniform Resource Name Resolution*, IETF (Internet Engineering Task Force), ed. K. Sollins, 1998

ICS 25.040.01

Price based on 70 pages