
**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*



STANDARDSISO.COM : Click to view the full PDF of ISO 16100-6:2018



COPYRIGHT PROTECTED DOCUMENT

© ISO 2018

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

	Page
Foreword	v
Introduction	vi
1 Scope	1
2 Normative references	1
3 Terms and definitions	2
4 Symbols and abbreviated terms	3
5 Service provider interface services	4
5.1 Service sets	4
5.2 ESI service set	5
5.3 Dictionary Import Service Interface	6
5.3.1 Parts libraries imported to a repository	6
5.3.2 Relationship between parts libraries and MDDs	6
5.3.3 Mapping from a PLIB element to an MDD	7
6 Extended Service Interface	7
6.1 CPTI Group services	7
6.1.1 Scenarios handled by the CPTI Group	7
6.1.2 Capability profile template creation	7
6.1.3 The <i>accessTemplate</i> service	10
6.1.4 The <i>modifyTemplate</i> service	11
6.1.5 The <i>validateTemplate</i> service	12
6.1.6 The <i>deleteTemplate</i> service	12
6.2 The Extended CPI Group	13
6.2.1 Scenarios handled by the Extended CPI Group	13
6.2.2 Capability profile creation	14
6.2.3 The <i>accessProfile</i> service	15
6.2.4 The <i>modifyProfile</i> service	16
6.2.5 The <i>validateProfile</i> service	17
6.2.6 The <i>deleteProfile</i> service	18
6.3 The CCSI Group	19
6.3.1 Scenarios handled by the CCSI Group	19
6.3.2 Capability class structure creation	19
6.3.3 The <i>accessCCS</i> service	21
6.3.4 The <i>modifyCCS</i> service	22
6.3.5 The <i>validateCCS</i> service	22
6.3.6 The <i>deleteCCS</i> service	23
6.4 The Extended Matcher Group	24
6.4.1 Scenarios handled by the Extended Matcher Group	24
6.4.2 The <i>ExtendedMatcher</i> service	24
6.4.3 Matcher conformance testing	25
7 Formal ESI protocol description	28
7.1 The Extended Matcher Group	28
7.2 CPTI Group service protocol	29
7.2.1 Capability profile template creation	29
7.2.2 Capability profile template access	30
7.2.3 Capability profile template modification	30
7.2.4 Capability profile template conformance test	31
7.2.5 Capability profile template deletion	32
7.3 Extended CPI Group service protocols	32
7.3.1 Capability profile creation	32
7.3.2 Capability profile access	33
7.3.3 Capability profile modification	34
7.3.4 Capability profile conformance test	35

7.3.5	Capability profile deletion.....	35
7.4	CCSI Group service protocols.....	36
7.4.1	Capability class structure creation.....	36
7.4.2	Capability class structure access.....	37
7.4.3	Capability class structure modification.....	37
7.4.4	Capability class structure conformance test.....	38
7.4.5	Capability class structure deletion.....	38
7.5	Extended Matcher Group service protocols.....	39
8	Dictionary import service and protocol.....	40
8.1	The <i>DictionaryImporting</i> service.....	40
8.2	The <i>DictionaryImporting</i> protocol.....	40
Annex A	(informative) The capability model with MDDs.....	42
Annex B	(informative) Simplified matching of capability profile templates.....	47
Annex C	(informative) The profiles based on the capability profile templates.....	56
Annex D	(informative) The procedure for generating a capability class structure.....	59
Annex E	(informative) Mapping Parts Library (PLIB) to MDDs.....	60
Annex F	(informative) Mapping OTD to MDDs.....	63
Annex G	(informative) The procedure for matching two profiles.....	66
Bibliography	71

STANDARDSISO.COM : Click to view the full PDF of ISO 16100-6:2018

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document 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*.

This second edition cancels and replaces the first edition (ISO 16100-6:2011), of which it constitutes a minor revision. The changes compared to the previous edition are as follows:

- in 5.2 b) 5) and b) 6), changed “a requirement capability profile or a requirement capability profile” to “an MSU capability profile or a requirement capability profile”;
- in 6.2.2.2.2, changed “*createTemplate* service” in the second sentence to “*createProfile* service”;
- in Figure 8, NOTE 2, changed “unique template identifiers” to “unique profile identifiers”;
- in 6.2.5, changed “*returnTestingResult* services” to “*returnTestResult* services”;
- in 7.3.5, changed “The *deleteTemplate* service deletes an existing template” to “The *deleteProfile* service deletes an existing profile”;
- in 7.4.1.2, changed “generates a template” to “generates a CCS”;
- in 7.4.5, changed “deletes an existing template” to “deletes an existing CCS”;
- in Annex A, changed “the class capability model” to “the capability class model”;
- in Annex G, added double quotation marks for object names;

A list of all parts in the ISO 16100 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

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

- a growing base of vendor-specific solutions;
- user difficulties in applying standards;
- need to move to modular sets of system integration tools;
- 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 document 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.

STANDARDSISO.COM : Click to view the full PDF of ISO 16100-6:2018

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 document 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 document into languages other than the official ISO languages. Accordingly, any implementations should be developed only on the basis of the texts in the official ISO languages.

1 Scope

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

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

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

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 16100-1:2009, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 1: Framework*

ISO 16100-2:2003, *Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 2: Profiling methodology*

ISO 16100-3:2005, *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 profile 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, ISO 16100-5 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

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 to entry: The role of an 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 to entry: In this document, 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 to entry: 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.

[SOURCE: ISO 16100-2:2003, 3.3, modified — The domain in angle brackets has been added; Note 1, 2 and 3 to entry have been added; the wording “software units role” has been changed to “software unit’s role”.]

3.2

capability class structure

hierarchy of capability classes

Note 1 to entry: 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

XML schema representing a capability class structure

[SOURCE: ISO 16100-5:2009, 3.2, modified — The full form for “XML” has been deleted; the wording “a hierarchy of capability classes” has been changed to “a capability class structure”.]

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 document that handle manufacturing domain data, manufacturing domain models, capability class structures, capability profiles and capability profile templates

Note 1 to entry: “Extended” refers to both the services specified in this document and the “basic” services specified in ISO 16100-3.

3.6**manufacturing domain data**

information, represented by a UML class, about manufacturing resources, manufacturing activities, or items exchanged among manufacturing resources within a particular manufacturing domain

[SOURCE: ISO 16100-5:2009, 3.3, modified — The wording “unified modeling language (UML) class representing information” has been changed to “information, represented by a UML class”.]

3.7**manufacturing domain data template**

XML schema representing a manufacturing domain data

[SOURCE: ISO 16100-5:2009, 3.4, modified — The full form for “XML” has been deleted.]

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

[SOURCE: ISO 16100-5:2009, 3.5]

3.9**parts library**

<manufacturing> collection of part descriptions or catalogue

Note 1 to entry: 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

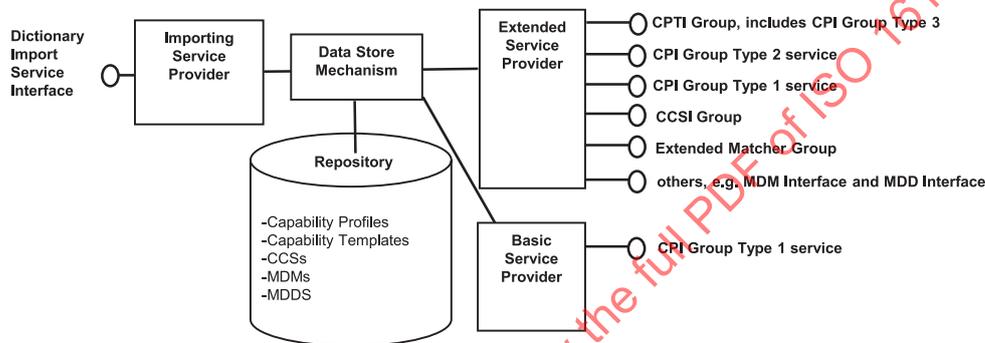
BSU	Basic Semantic Unit
CCS	Capability Class Structure
CCSI	Capability Class Structure Interface
CPI	Capability Profile Interface
CPTI	Capability Profile Template Interface
CSI	Conformance Statement for the Implementation
ESI	Extended Service Interface
ESP	Extended Service Provider
ICD	International 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 to handle 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 to handle 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 document.

NOTE 2 The boldfaced elements in this figure are specifically addressed in this document.

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

NOTE 4 The ESI access point is represented elsewhere in this document 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 following four service groups, 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 document.

- a) CPTI Group, that 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) 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 an MSU capability profile or a requirement capability profile;
 - 6) delete an MSU capability profile or a requirement capability profile.
- c) 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) Extended Matcher Interface 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.

Annex A provides information on the capability model with MDDs.

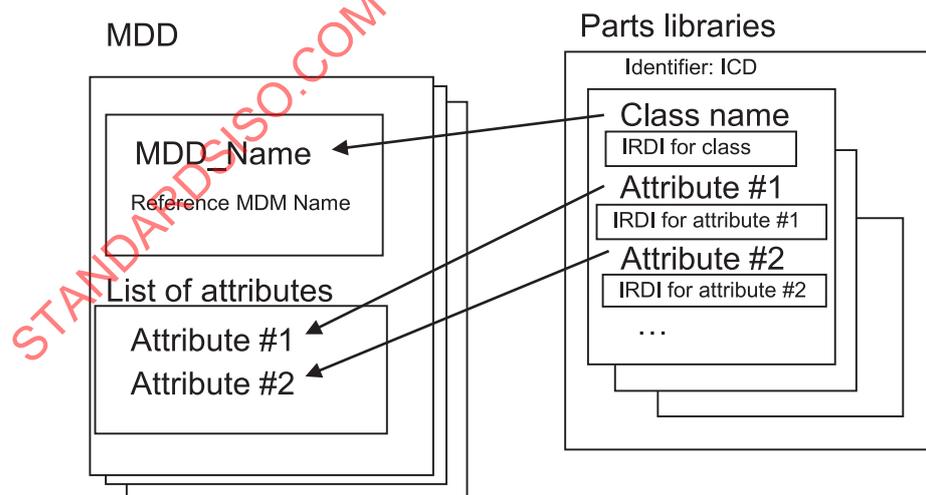
5.3 Dictionary Import Service Interface

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/TS 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 dictionary and a BSU, a code for each PLIB class and 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.

[Annex E](#) provides information on mapping PLIB to MDDs.

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.

[Annex B](#) provides information on the simplified matching of capability profile templates.

6.1.2 Capability profile template creation

6.1.2.1 The 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 to 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 element "Capability Profile Template";
- b) attribute "domain_Name" in element "Reference_MDM_Name";
- c) attribute "format_name" in 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 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.

[Annex B](#) provides information on the simplified matching of capability profile templates.

6.1.2.3 The *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

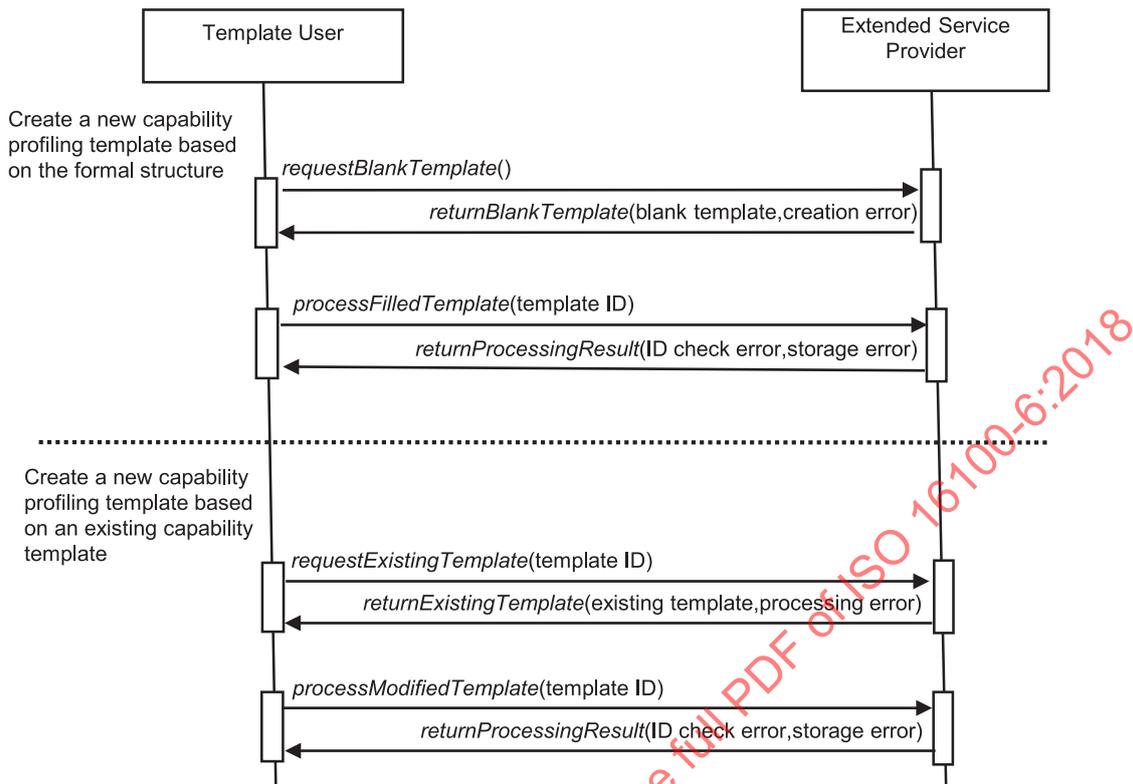
The *createTemplate* service shall allow a template user to create a template modified from an existing template. When creating a template modified from an existing template, the *createTemplate* service uses at a minimum the *requestExistingTemplate*, *returnExistingTemplate*, *processModifiedTemplate* 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.

STANDARDSISO.COM : Click to view the full PDF of ISO 16100-6:2018



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 — The createTemplate service

6.1.3 The 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.

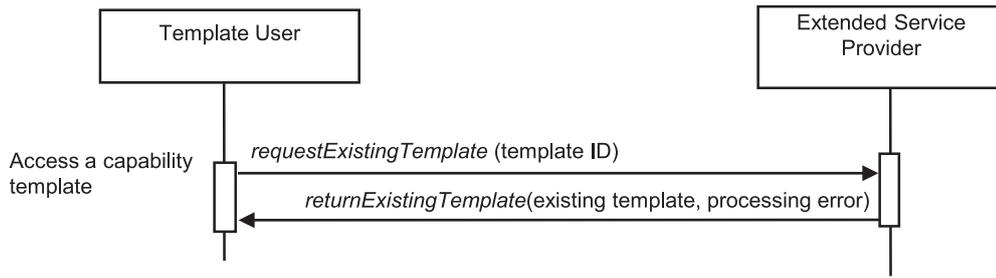


Figure 4 — The *accessTemplate* service

6.1.4 The *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.

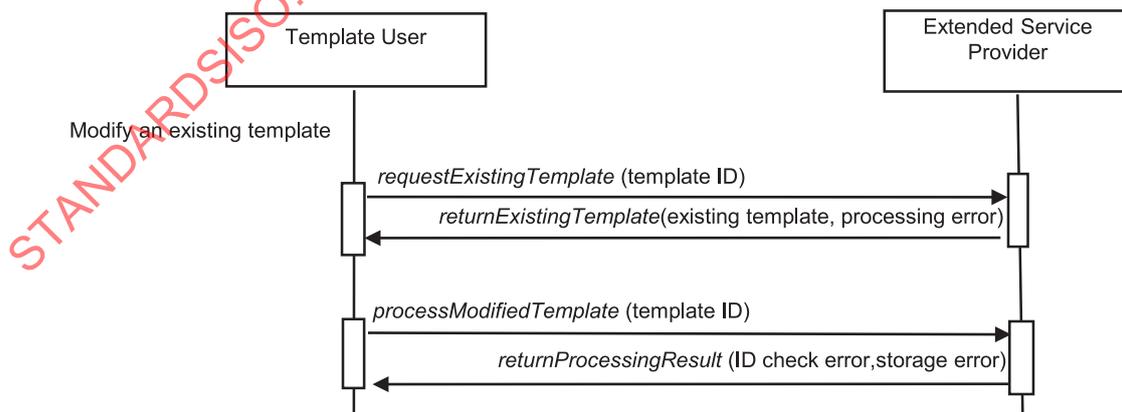


Figure 5 — The *modifyTemplate* service

6.1.5 The *validateTemplate* service

The *validateTemplate* service, using the *requestExistingTemplate*, *returnExistingTemplate*, *testTemplate* and *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, B.1.

The tested template shall be registered into 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 into 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.

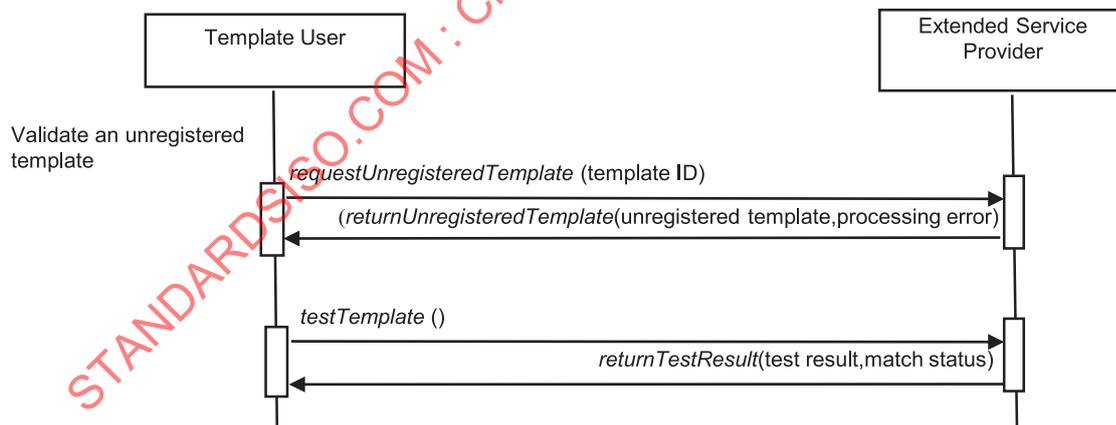


Figure 6 — The *validateTemplate* service

6.1.6 The *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.

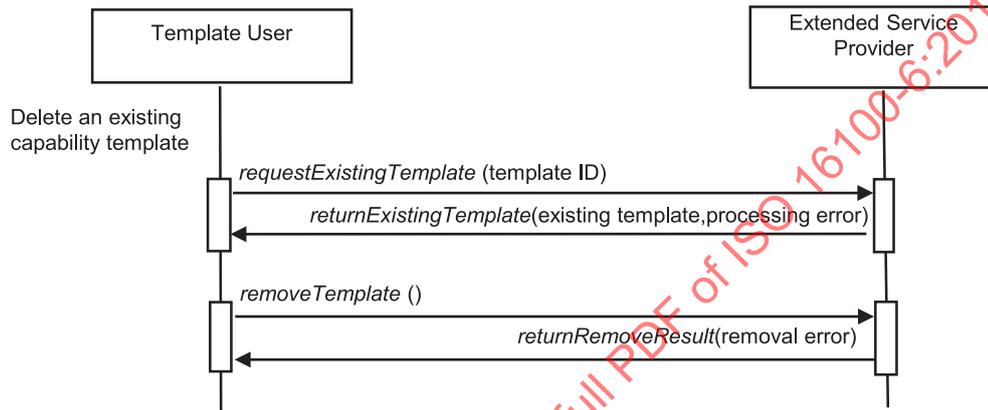


Figure 7 — The *DeleteTemplate* service

6.2 The 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 The creation process

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

[Annex C](#) provides information on the profiles based on the capability profile templates.

6.2.2.2 The *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.

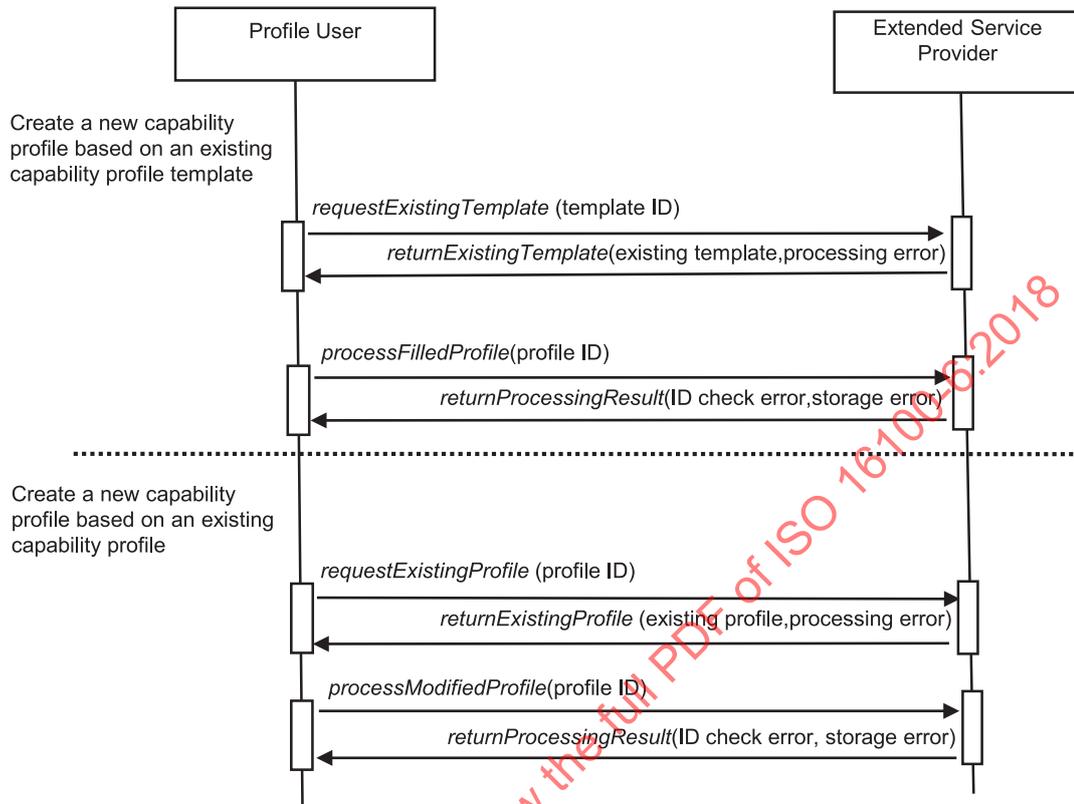
[Figure 8](#), above the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a profile from an existing template.

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 *createProfile* 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 MDMs 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 profile identifiers, which is outside the scope of ISO 16100.

Figure 8 — The createProfile service

6.2.3 The 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 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 *returnExistingProfile* services. The *accessProfile* service shall consist of the following steps:

- a) the profile user invokes the *requestExistingProfile* service of the MSU object, in which there are no parameters associated 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.

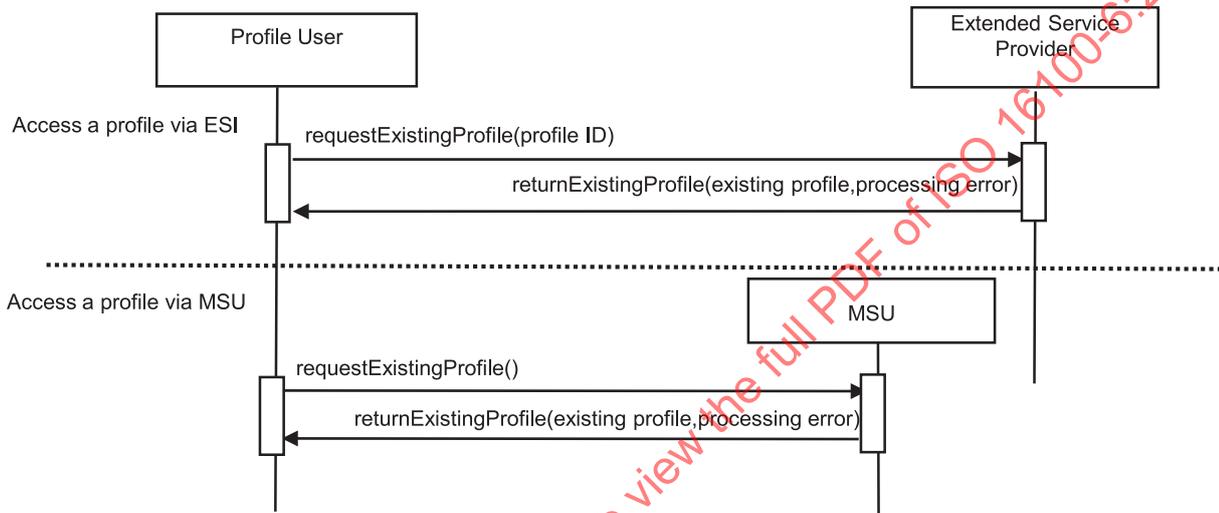


Figure 9 — The *accessProfile* service

6.2.4 The *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 MDMs 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:

- the profile user invokes the *requestExistingProfile* service of the MSU object, in which there are no parameters associated with the *requestExistingProfile* service;
- 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;
- the profile user modifies the existing profile using MDMs of the MDM and then invokes the *processModifiedProfile* service of the MSU object, in which there are no parameters associated with the *processModifiedProfile* service;
- 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.

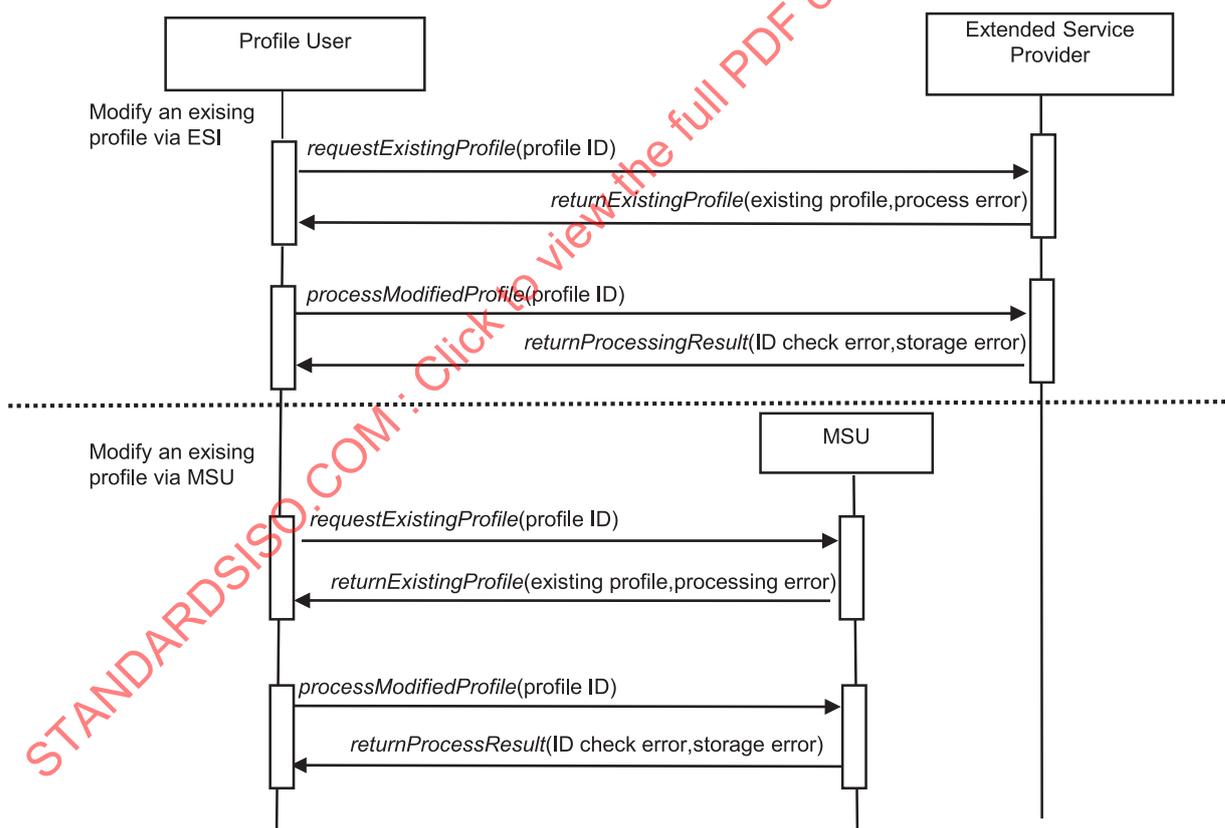


Figure 10 — The *modifyProfile* service

6.2.5 The *validateProfile* service

The *validateProfile* service, using the *requestExistingProfile*, *returnExistingProfile*, *testProfile* and *returnTestResult* 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.

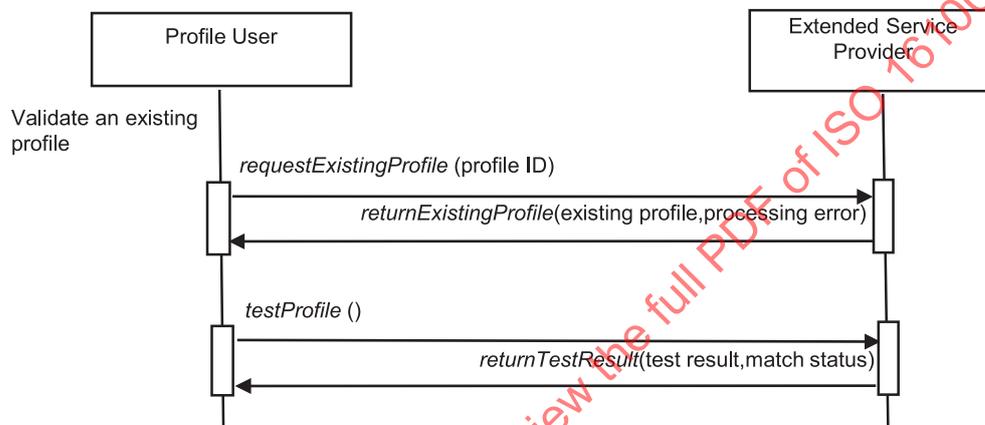


Figure 11 — The *validateProfile* service

6.2.6 The *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.

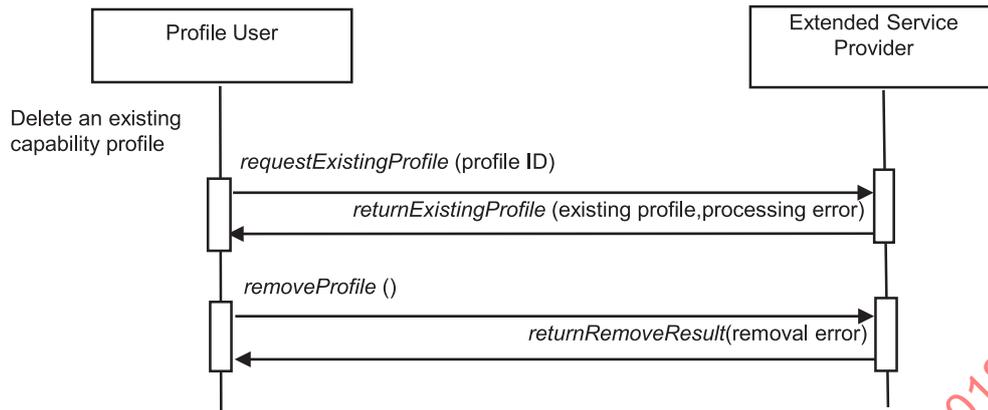


Figure 12 — The *deleteProfile* service

6.3 The CCSI Group

6.3.1 Scenarios handled by the CCSI Group

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

- 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;
- access a CCS from a repository by its CCS ID and then receive a CCS from a service provider;
- modify a CCS according to either user requirements or the results from a conformance test and receive a modified CCS from a service provider;
- test a CCS against the CCS conformance criteria and receive either a positive, negative or matching level response from a service provider;
- register a CCS into a repository and receive a “registered” status from a service provider;
- 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 The 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.

[Annex D](#) provides information on the procedure for generating a capability class structure.

6.3.2.2 The *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.

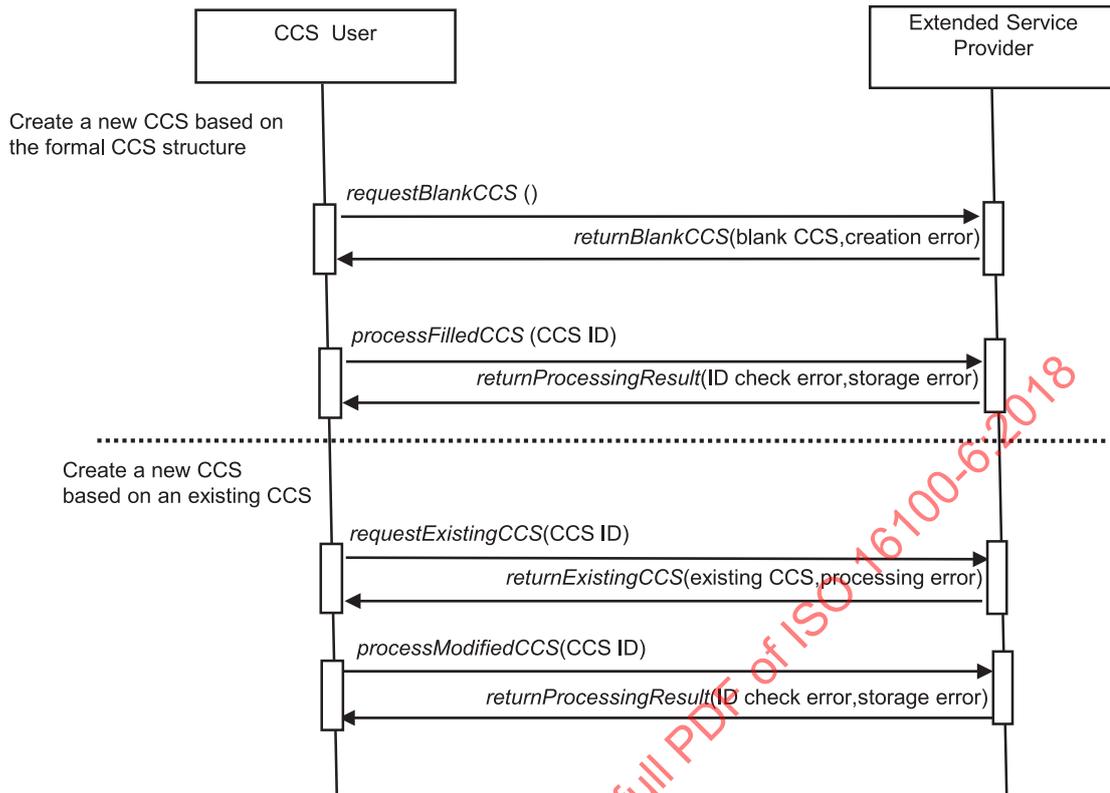
Figure 13, above the dotted line, provides a UML sequence diagram of the mandatory steps for the creation of a CCS from a formal CCS structure.

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 — The *createCCS* service

6.3.3 The *accessCCS* service

The *accessCCS* service, using the *requestExistingCCS* and *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.

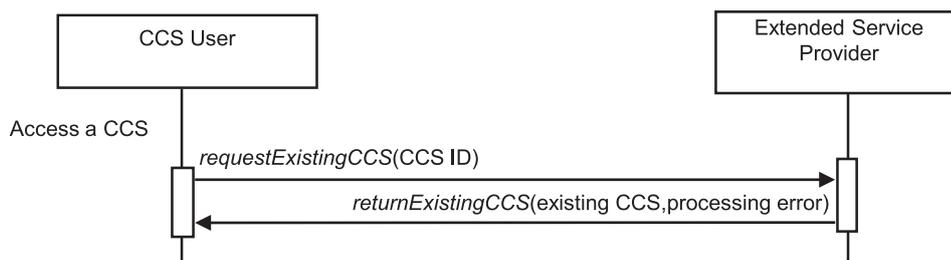


Figure 14 — The *accessCCS* service

6.3.4 The *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.

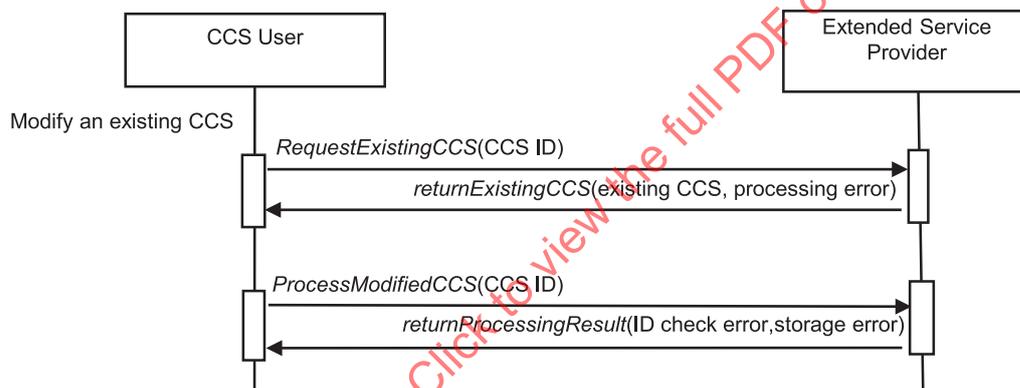


Figure 15 — The *modifyCCS* service

6.3.5 The *validateCCS* service

The *validateCCS* service, using the *requestExistingCCS*, *returnExistingCCS*, *testCCS* and *returnTestingResult* 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.

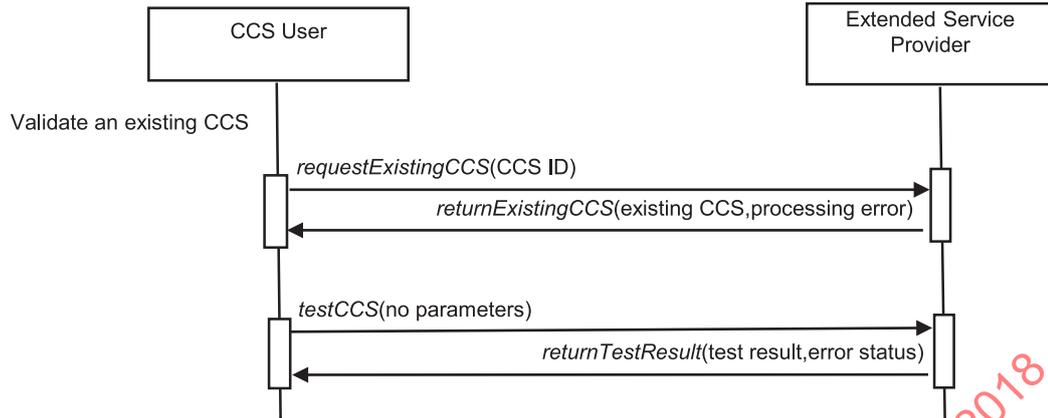


Figure 16 — The *validateCCS* service

6.3.6 The *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 parameter of the *returnRemoveResult* service is removal error.

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

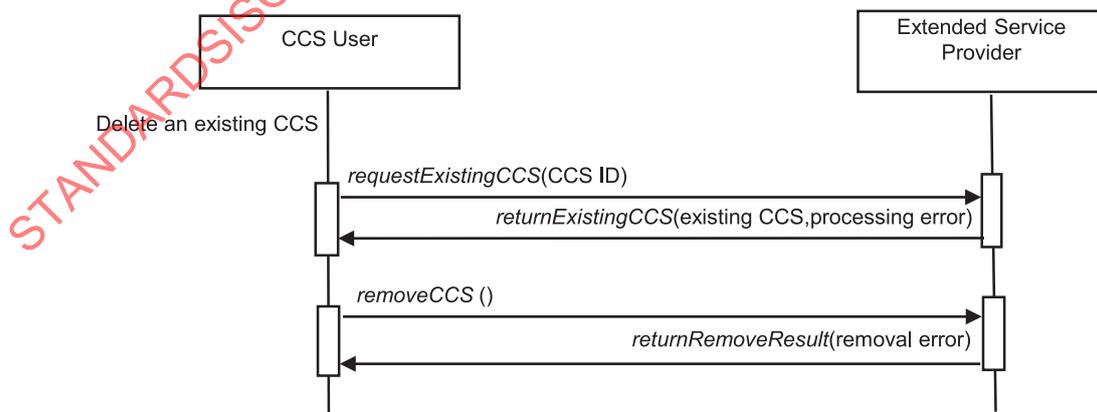


Figure 17 — The *deleteCCS* service

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.

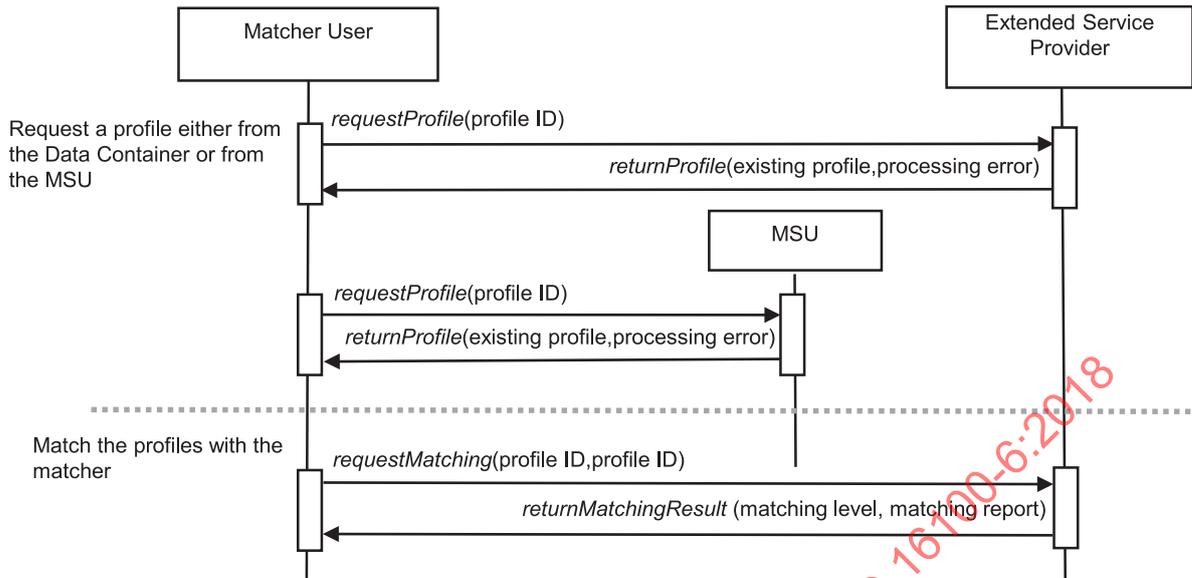
6.4.2 The *ExtendedMatcher* service

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* 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 — The *ExtendedMatcher* services

6.4.3 Matcher conformance testing

The conformance methodology specified in ISO 16100-4 applies to this document, which extends the CSIs for a capability profile matcher shown in ISO 16100-4:2006, Table 9. Table 1 and Table 2 shall be used for conformance testing.

Table 1 — CSIs for matcher performance

Conformance point and set number	Conformance point description	Specification reference	Conformance point type	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	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"

Table 1 (continued)

Conformance point and set number	Conformance point description	Specification reference	Conformance point type	Abstract test criteria
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"
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 the elements "Set_of_MDD_Objects" in the two capability definitions	ISO 16100-5:2009, Clause 7	A	

Table 1 (continued)

Conformance point and set number	Conformance point description	Specification reference	Conformance point type	Abstract test criteria
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

Table 2 — CSIs for matcher reporting

Conformance point and set number	Conformance point description	Specification reference	Conformance point type	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	DetailedListReport	ISO 16100-5:2009, 7.2	A	Verify matched functions and the unmatched functions
Index_3	CompareMatchingLevel&DetailedReport MatchingLevel			
Index_3.1	ForCompleteMatch			Verify that both sets of functions in the two profiles are fully equivalent
Index_3.2	ForAllMandatory-Match			Verify that both sets of mandatory functions in the two profiles are fully equivalent
Index_3.3	ForSomeMandatory-Match			Verify the list of equivalent mandatory functions in the two profiles and the list of non-equivalent mandatory functions
Index_3.4	ForNoMandatory-Match			Present that both sets of functions in the two profiles are fully non-equivalent

7 Formal ESI protocol description

7.1 The Extended Matcher Group

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

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 to accept 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 to accept 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 to accept 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 to accept 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 to accept 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 to accept 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 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 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 *deleteProfile* service deletes 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 *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 to accept 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 CCS 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 to accept 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 to accept 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 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 *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".

[Annex G](#) provides information on the procedure for matching two profiles.

8 Dictionary import service and protocol

8.1 The DictionaryImporting service

The *DictionaryImporting* service, using the *requestImportDictionaries*, *returnImportedDictionaries*, *requestDictionaries* and *returnDictionaries* services, shall allow a user to import a parts library to a Repository and to review the contents in the Repository.

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

- a) the dictionary user invokes the *requestImportDictionaries* service of the *ImportServicePoint* object; in which the parameter associated with the *requestImportDictionaries* 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 *requestDictionaries* service of the *ImportServicePoint* object, in which the parameter of the *requestDictionaries* 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.

Figure 19 provides a UML sequence diagram of the mandatory steps for importing a parts library to a Repository and reviewing the contents.

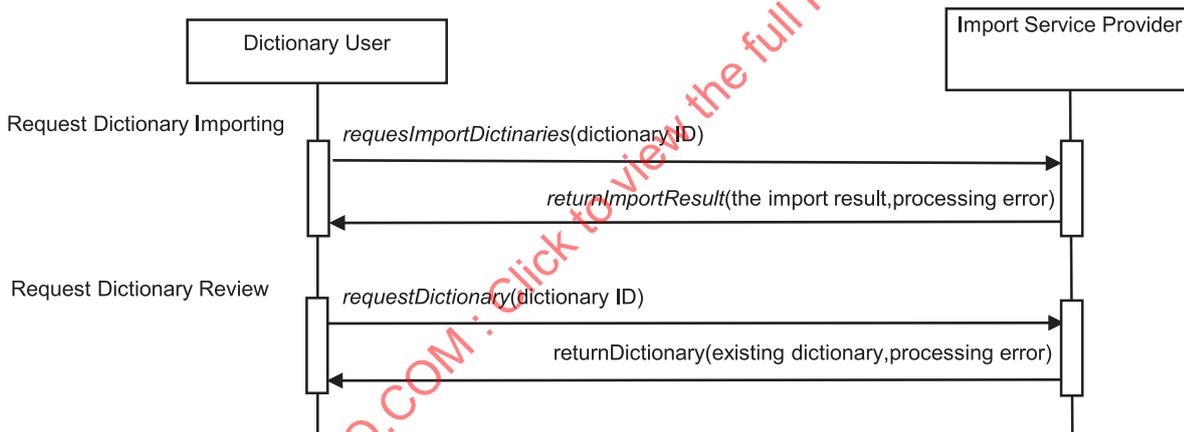


Figure 19 — The *DictionaryImporting* service

Annex F provides information on mapping OTD to MDDs.

8.2 The DictionaryImporting protocol

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

- a) the *requestImportDictionaries* service requests importing dictionaries and shall have the service type `<service-type>="requestImportingDictionaries"` with the corresponding attribute `dictionary_ID="the_dictionary_id"`;
- b) the *returnImportingDictionaries* 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".
```

STANDARDSISO.COM : Click to view the full PDF of ISO 16100-6:2018

Annex A (informative)

The capability model with MDDs

A.1 The 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.

An application activity tree has a one-to-one mapping to a capability class tree. The activity model can be mapped to the capability class model 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).

STANDARDSISO.COM : Click to view the full PDF of ISO 16100-6:2018

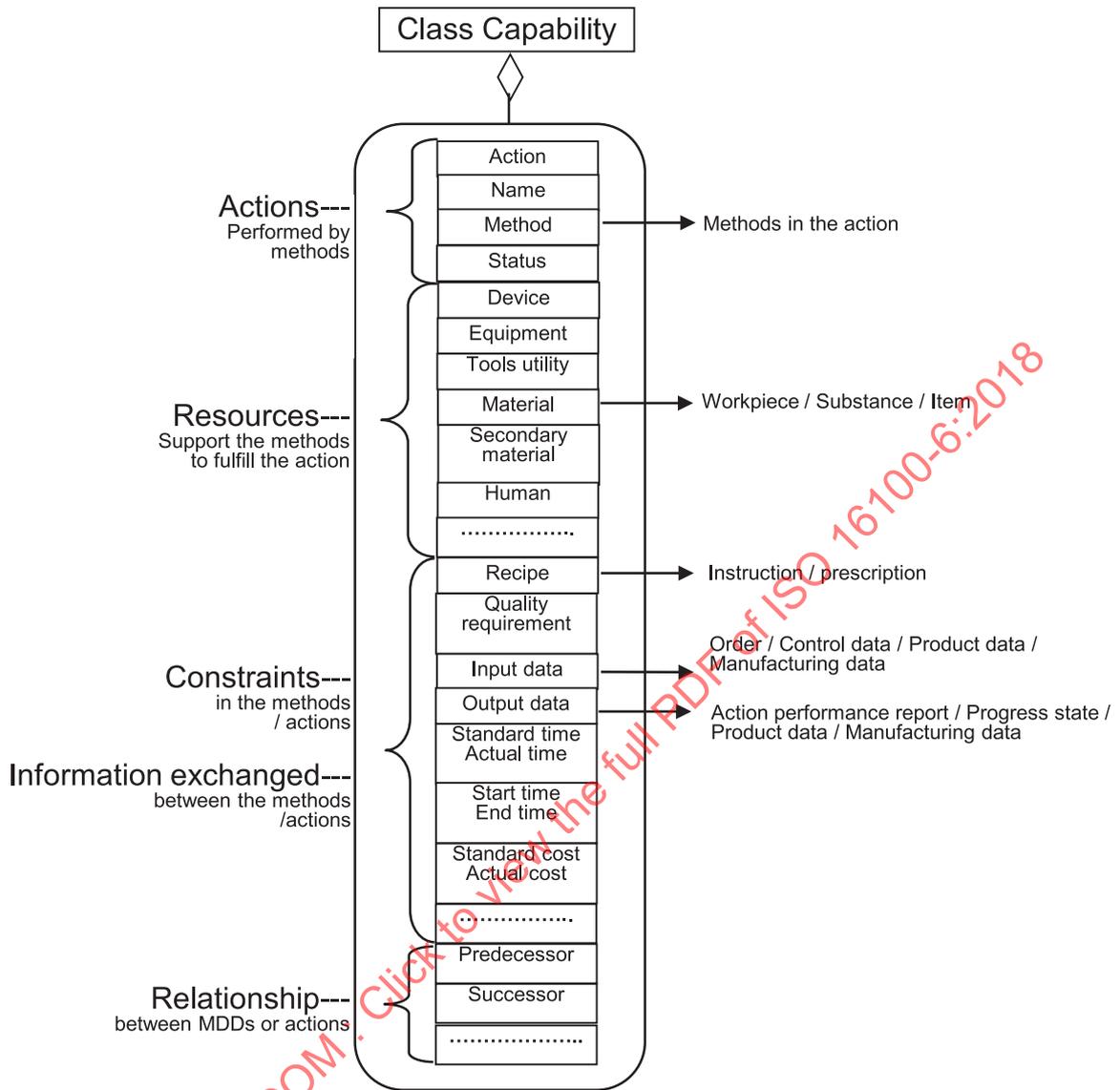


Figure A.1 Mapping a capability model to an activity model using MDDs

A.2 The 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_informatio">
                    <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>
                </xs:choice>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:sequence>
      </xs:complexType>
    </xs:sequence>
  </xs:complexType>

```

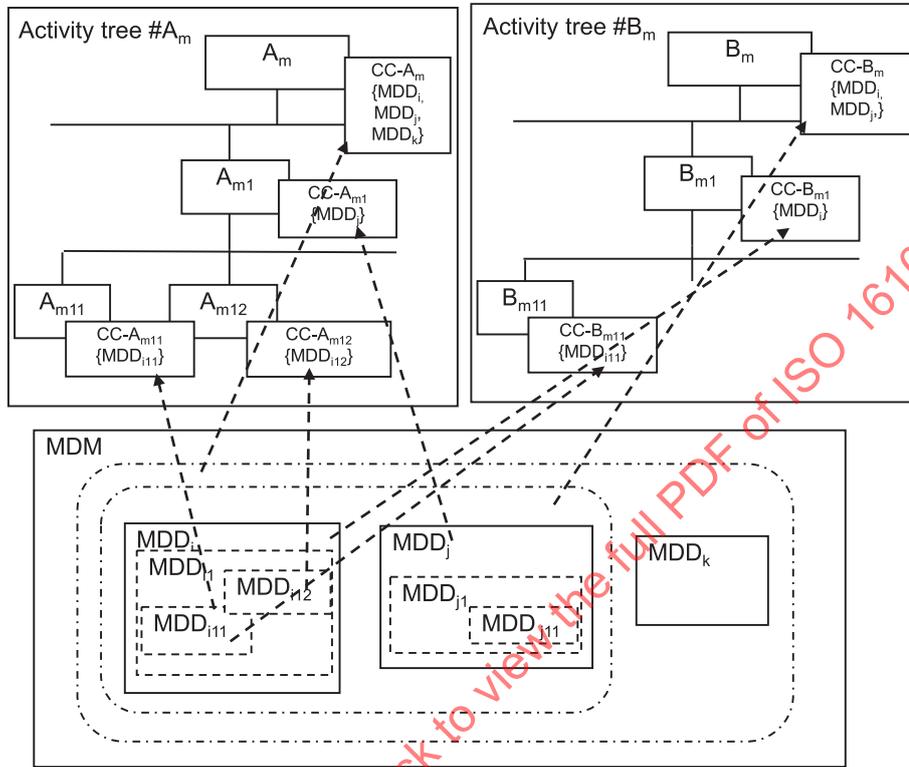
```

        </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>
            <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 The relationships of MDDs and the 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 ($MDD_i \in MDM \mid i \in [1... n]$). 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, 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: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="Conatraits">
          .....
          .....
        </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="Conatrains">
        .....
    </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, 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="RreceiveManufacturingInstruction" />
                    </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:schema>

```

```

        </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: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:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

```



```

        </xs:element>

        <xs:element name="Resources">
            .....
            .....
        </xs:element>
        <xs:element name="Conatraits">
            .....
            .....
        </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:sequence>
    </xs:complexType>
</xs:element>

    <xs:element name="Resources">
        .....
        .....
    </xs:element>
    <xs:element name="Conatraits">
        .....
        .....
    </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, 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:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Resources">
            .....
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</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 The procedure for generating a capability profile template

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

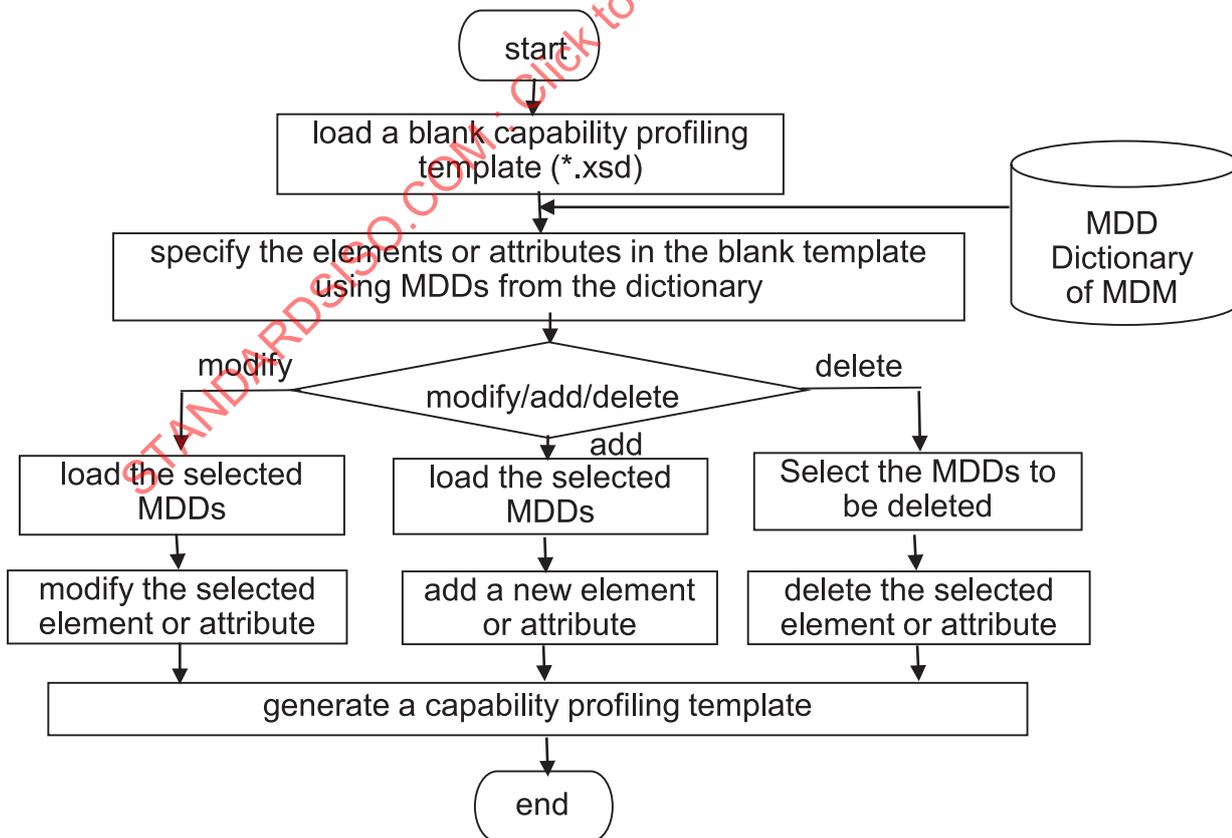
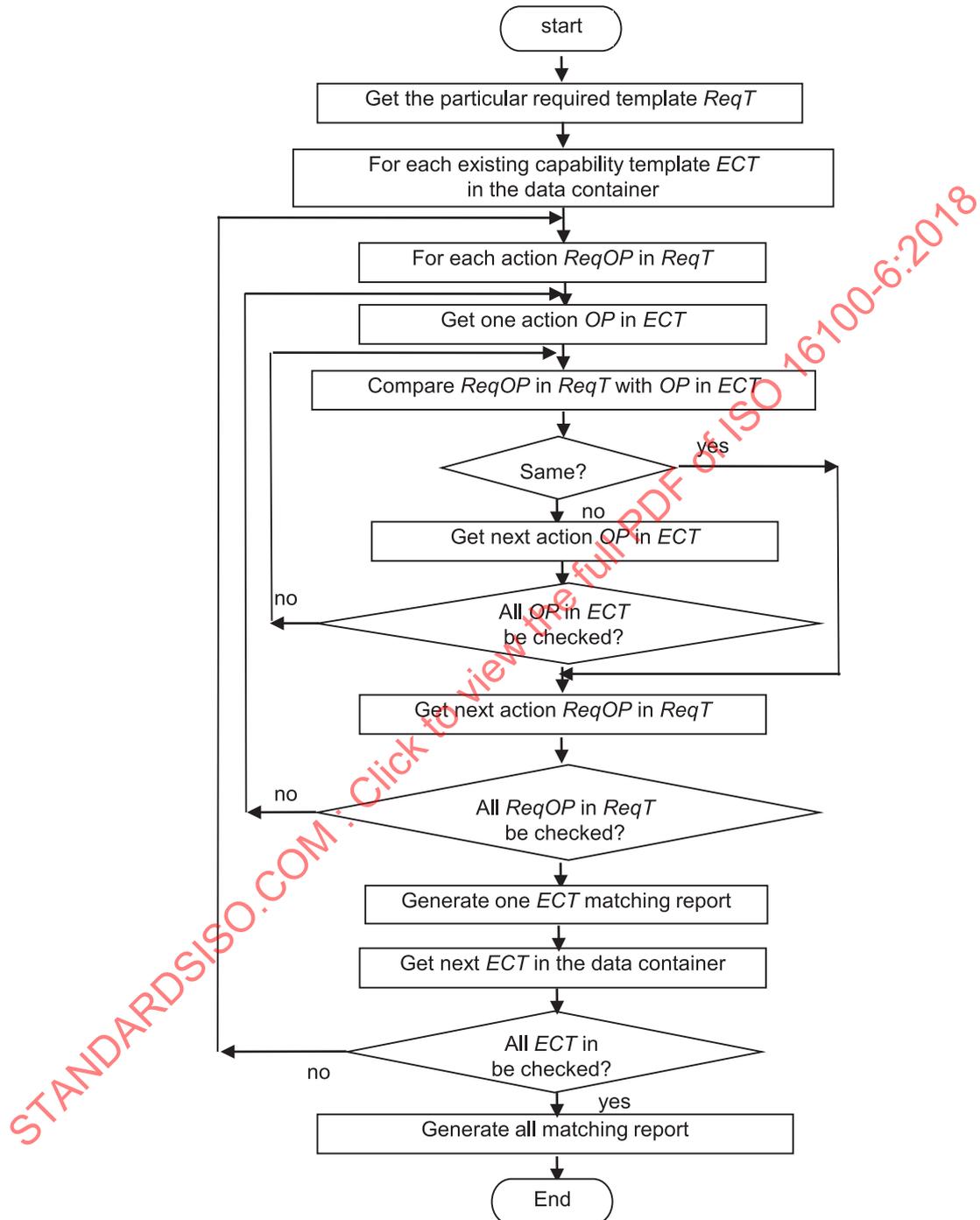


Figure B.1 — The procedure for generating a capability profile template

B.3 The process for selecting a proper capability profile template

The process for selecting a proper capability profile template in the repository according to the required template is shown in [Figure B.2](#).



Key

- ECT* Existing Capability Profile Template
- ReqT* Required Template
- ReqOP* Required Operation
- OP* Operation

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

Annex C (informative)

The profiles based on the capability profile templates

C.1 The profile of “getOperationMethod”

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