
**Information technology — Multimedia
Middleware —**

**Part 5:
Component download**

*Technologies de l'information — Intergiciel multimédia —
Partie 5: Téléchargement de composant*



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2008

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

Page

Foreword	iv
Introduction.....	v
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
3.1 Specification terms and definitions.....	1
3.2 Realization terms and definitions	6
4 Abbreviated terms	7
5 Overview of interface suites.....	7
6 Download interface suites	8
6.1 Interface suites for receiving components	8
6.2 Interfaces suites for initiation of component transfer.....	13
7 Realization overview	18
8 Download realization.....	19
8.1 Structure view.....	19
8.2 Behavior view	22
8.3 Deployment	38
Annex A (informative) Deployment example.....	39
Annex B (informative) Meta data for security parameters	46
Bibliography.....	61

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

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

ISO/IEC 23004-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23004 consists of the following parts, under the general title *Information technology — Multimedia Middleware*:

- *Part 1: Architecture*
- *Part 2: Multimedia application programming interface (API)*
- *Part 3: Component model*
- *Part 4: Resource and quality management*
- *Part 5: Component download*
- *Part 6: Fault management*
- *Part 7: System integrity management*

Introduction

The Download Framework has been developed as part of M3W to fulfill the requirements concerning upgrading and extension of operational M3W-based systems. During the period that a CE device is owned by a user, there is often a need to improve and/or extend the device's software in order to extend the economic lifetime of the device.

In the Component lifecycle, the Download Framework is responsible for transferring M3W Components from a Repository to a Target (see Figure 1).

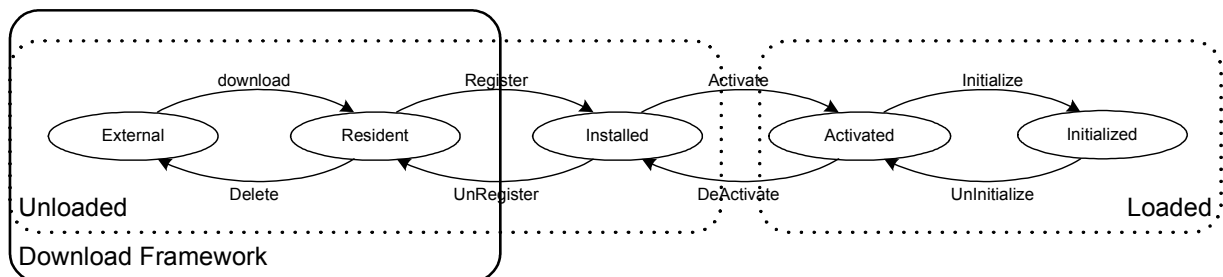


Figure 1 — Scope Download Framework

The download process takes a Component in its "installable Component" stage and downloads it to the Target, which brings the Component in its "resident" state. This is a very macroscopic view on the download process which, when elaborated, reveals the following constituent activities of the download process.

- **Identify need for download:** This activity describes the part in which the need for the presence of a given Component on a given M3W system is identified.
- **Entities are located:** After the need for download is realized, the place from where the Component can be retrieved, as well as its destination, must be identified. Also, the place where the decision about the feasibility of the download will be established must be identified, in case it is an entity other than the sender or the receiver.
- **Connection establishment:** The sender and the receiver in the context of the download for a specific Component establish a connection between them in order to proceed with the download process.
- **Security checks:** Using the connection between them, the server and the receiver run a number of security checks (e.g. mutual authentication, authorization for the download, payment guarantees to cover the costs of the Component as well as the cost of the download process).
- **Feasibility analysis:** This activity includes a number of checks and verifications to ensure that the Component can be installed and activated at its destination while it still satisfies its specifications and without putting in danger the rest of the M3W system at its destination.
- **Transmission properties:** A number of negotiations regarding the security and reliability characteristics of the data transmission as well as providing certain necessary information (e.g. the payment token like credit card number, payment certificate).
- **Component transmission:** This activity comprises the actual transfer of the data that constitute the Component and which includes at least the data corresponding to the representation of the executable model of a Component.

- **Connection termination:** This activity contains all actions necessary for the sender and the receiver to keep consistent information about the final status of the data transfer (and hence the download process so far). If this activity is completed successfully, the remaining parts of the download process necessitate only the participation of the receiver.

The remainder of this part of ISO/IEC 23004 contains the specification of the Download Framework. This specification describes the Download Framework from a number of perspectives.

- The decomposition into roles is described in the Structure View (8.1).
- The interaction between these roles is described in the Behavior View (8.2).
- The deployment of these roles is described in Deployment View (8.3).

Information technology — Multimedia Middleware —

Part 5: Component download

1 Scope

This part of ISO/IEC 23004 defines the MPEG Multimedia Middleware (M3W) technology Download Architecture. This definition contains the specification of the part of the M3W application programming interface (API) related to download as well as the realization. The M3W API specification provides a uniform view of the download functionality provided by M3W. The specification of the realization is relevant for those who are making an implementation of a download framework for M3W.

2 Normative references

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

ISO/IEC 23004-1, *Information technology — Multimedia Middleware — Part 1: Architecture*

ISO/IEC 23004-3, *Information technology — Multimedia Middleware — Part 3: Component model*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1 Specification terms and definitions

3.1.1

API specification

specification of a collection of software interfaces providing access to coherent streaming-related functionality

3.1.2

interface suite

collection of mutually related interfaces providing access to coherent functionality

3.1.3

logical component

coherent unit of functionality that interacts with its environment through explicit interfaces only

3.1.4

role

abstract class defining behavior only

3.1.5

role instance

object displaying the behavior defined by the role

3.1.6

attribute

instance variable associated with a role

NOTE Attributes are used to associate state information with roles.

3.1.7

signature

definition of the syntactic structure of a specification item such as a type, interface or function in IDL

NOTE For C functions, signature is equivalent to prototype.

3.1.8

specification item

entity defined in a specification

NOTE Data type, role, attribute, interface and function are examples of specification items.

3.1.9

qualifier

predefined keyword representing a property or constraint imposed on a specification item

3.1.10

constraint

restriction that applies to a specification item

3.1.11

execution constraint

constraint on multi-threaded behavior

3.1.12

model type

data type used for specification (modeling) purposes only

NOTE Set, map and entity are examples of model types.

3.1.13

model constant

constant used for specification (modeling) purposes only

3.1.14

enum element type

enumerated type whose values can be used to construct sets (bit vectors) of at most 32 values by logical or-ing

3.1.15

enum set type

32-bit integer data type representing sets of enumerated values

3.1.16

set type

data type whose values are mathematical sets of values of a specific type

NOTE Unlike enum sets, these sets may be infinite.

3.1.17

map type

data type whose values are tables mapping values of one type (the domain type) to values of another type (the range type)

NOTE Maps are a kind of generalized array.

3.1.18**entity type**

class of objects that may have attributes associated with them

3.1.19**interface-role model**

extended Unified Modeling Language class diagram showing the roles and interfaces associated with a logical component, and their mutual relations

3.1.20**logical component instance**

incarnation of a logical component: a configuration of objects displaying the behavior defined by the logical component

3.1.21**provides interface**

interface that is provided by a role or role instance

3.1.22**requires interface**

interface that is used by a role or role instance

3.1.23**specialization**

behavioral inheritance

definition, by a role, of behavior which implies the behavior defined by another role

NOTE A role S specializes a role R if the behavior defined by S implies the behavior defined by R, i.e. if S has more specific behavior than R.

3.1.24**diversity**

set of all parameters that can be set at instantiation time of a logical component, and that will not change during the lifetime of the logical component

3.1.25**mandatory interface**

provides interface of a role that should be implemented by each instance of the role

3.1.26**optional interface**

provides interface of a role that need not be implemented by each instance of the role

3.1.27**configurable item**

parameter that can be set at instantiation time of a logical component, usually represented by a role attribute

3.1.28**diversity attribute**

role attribute that represents a configurable item

3.1.29**instantiation**

process of creating an instance (an incarnation) of a role or logical component

3.1.30**initial state**

state of a role instance or logical component instance immediately after its instantiation

3.1.31**observable behavior**

behavior that can be observed at the external software and streaming interfaces of a logical component

3.1.32

function behavior

behavior of the functions in the provides interfaces of a role

3.1.33

streaming behavior

input-output behavior of the streams associated with a role

3.1.34

active behavior

autonomous behavior that is visible at the provides and requires interfaces of a role

3.1.35

instantiation behavior

behavior of a role at instantiation time of a logical component

3.1.36

independent attribute

attribute whose value may be defined or changed independently of other attributes and entities

3.1.37

dependent attribute

attribute whose value is a function of the values of other attributes or entities

3.1.38

invariant

assertion about a role or logical component that is always true from an external observer's point of view

NOTE In reality, the assertion may temporarily be violated.

3.1.39

callback interface

interface provided by a client of a logical component whose functions are called by the logical component

NOTE A notification interface is an example of this, but there may be other call-back interfaces as well, e.g. associated with plug-ins.

3.1.40

callback-compliance

general constraint that the functions in a callback interface should not interfere with the behavior of the caller in an undesirable way, such as by blocking the caller or by delaying it too long

3.1.41

event notification

act of reporting the occurrence of events to 'interested' objects

3.1.42

event subscription

act of recording the types of event that should be notified to objects

3.1.43

cookie

special integer value that is used to identify an event subscription

NOTE Clients pass cookies to a logical component when subscribing to events. Logical components pass cookies back to clients when notifying the occurrence of the events.

3.1.44**event-action table**

table associating events that can occur to actions that will be performed in reaction to the events

NOTE This is used to specify event-driven behavior.

3.1.45**non-standard event notification**

event notification that is accompanied by other actions (such as state changes of the notifying logical component)

3.1.46**client role**

role modeling the users of a logical component

3.1.47**actor role**

role (usually a client role) whose active behavior consists of calling functions in interfaces without any a priori constraints on when these calls will occur

3.1.48**control interface**

interface provided by a logical component that allows the logical component's functionality to be controlled by a client

3.1.49**notification interface**

interface provided by a client of a logical component that is used by the logical component to report the occurrence of events to the client

3.1.50**specialized interface**

interface of a role R that is inherited from another role and is further constrained by R

3.1.51**precondition**

assertion that should be true immediately before a function is called

3.1.52**action clause**

part of an extended precondition and postcondition specification defining the abstract action performed by a function

NOTE The abstract action usually defines which variables are modified and/or which out-calls are made by the function.

3.1.53**out-call**

out-going function call of an object on an interface of another object

3.1.54**postcondition**

assertion that will be true immediately after a function has been called

3.1.55**asynchronous function**

function with a delayed effect

NOTE The effect of the function will occur some time after returning from the function call.

3.2 Realization terms and definitions

3.2.1

appliance

product as seen by the customer

NOTE It consists of the device, an operating system (OS), components and applications.

3.2.2

application

software entity that provides a set of functions to a user

3.2.3

component

unit of trading that conforms to the M3W component model

NOTE In the M3W component model, components are containers for models.

3.2.4

component model

specification of what constitutes a component

NOTE It defines *inter alia* the interaction mechanisms between components and between components and their environment.

3.2.5

device

physical part of the appliance, sometimes also used to denote an identifiable element of that appliance

3.2.6

executable component

model that is executable on a platform

NOTE The executable component is a container for services.

3.2.7

M3W system

system that conforms to the M3W specification

3.2.8

operational

state when an M3W system is fulfilling its required functions for a period of time

3.2.9

platform

operating system (OS) and hardware that executes the OS

3.2.10

system

combination of a platform, a set of executable components, a run-time environment and a set of applications that can provide a user with a set of functions

4 Abbreviated terms

API	Application Programming Interface
IDL	Interface Definition Language
M3W	Multimedia Middleware
OS	Operating System
UML	Unified Modeling Language

5 Overview of interface suites

This clause is informative and gives an overview of the interface suites in the M3W API for component download. These interface suites can be divided in the following categories:

- Interface suites for receiving M3W Components
- Interfaces for initiating M3W Component transfer

For receiving components this part of ISO/IEC 23004 specifies Target logical component (see 6.1.1). For initiating component transfer this part of ISO/IEC 23004 specifies the Initiator logical component (see 6.2.1). These logical components provide a uniform view (for the applications and other middleware) on the download functionality provided by M3W.

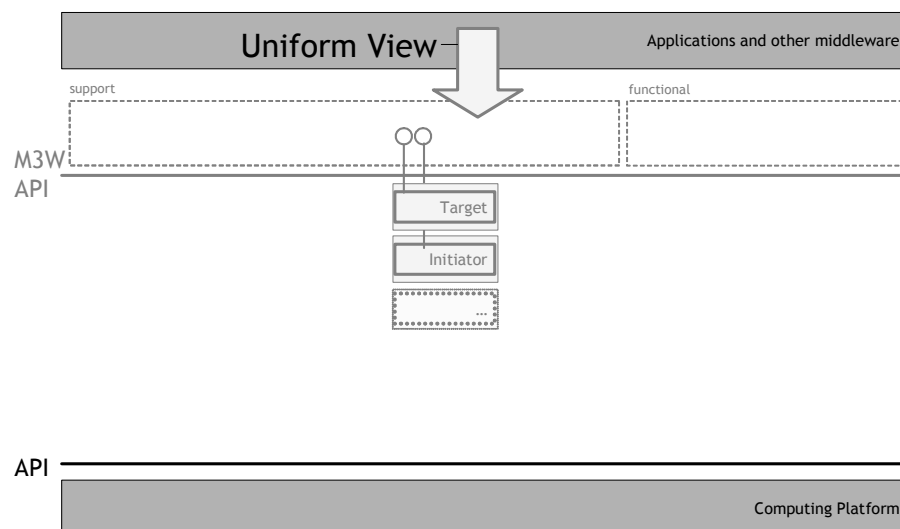


Figure 2 — Overview of interfaces suites (logical components) for component download

The download framework contains more roles than just those used for initiating component transfer and receiving components. The download framework is suitable for transfer of components according to ISO/IEC 23004-3 specification of M3W components. However, these roles only expose interfaces that are internal to the download framework and are therefore not part of the M3W API specification. These other roles (Locator, Decider and Repository) are discussed in the clause on realization of the Download framework (see clause 8). The integration in the overall architecture shall be done according to ISO/IEC 23004-1.

6 Download interface suites

6.1 Interface suites for receiving components

6.1.1 Target

6.1.1.1 Concepts

The Target is one of the roles in the Download framework. The Download framework enables the transfer of components to a device. The Target is responsible for exposing the target profile (used for accessing feasibility of component download), receiving components and storing the received components in non-volatile storage on the device.

6.1.1.2 Types and Constants

6.1.1.2.1 Public Types and Constants

6.1.1.2.1.1 Error Codes

Signature

```
const rcResult RC_ERR_TARGET_LOCATOR_NOT_FOUND = 0x00000001
const rcResult RC_ERR_TARGET_CANNOT_STORE_COMPONENT = 0x00000002
```

Qualifiers

— Error-codes

Description

The non-standard error codes that can be returned by functions of this logical component

Constants

Table 1

Name	Description
RC_ERR_TARGET_LOCATOR_NOT_FOUND	This error is returned when the target is unable to connect to the locator
RC_ERR_TARGET_CANNOT_STORE_COMPONENT	This error is returned when the target is unable to store the downloaded component, for example due to write permissions or low storage space

6.1.1.2.2 Model Types and Constants

None

6.1.1.3 Logical Component

6.1.1.3.1 Interface Role Model

Figure 3 — Interface-Role Model, depicts the interface-role model of the Target (grey box). It shows the interfaces and the roles involved with this component.

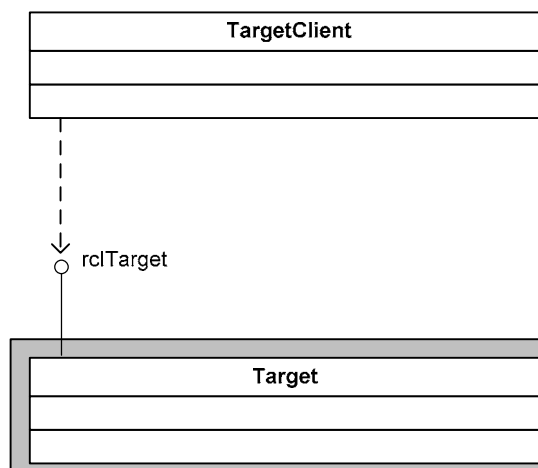


Figure 3 — Interface-Role Model

A Target Logical Component contains the Target role that provides the `rcITarget` interface. This interface can be used by a client to start a download task. The download task is responsible for participating in the interaction between the download roles. More specifically the download task is responsible for receiving and storing components.

6.1.1.3.2 Diversity

6.1.1.3.2.1 Provided Interfaces

Table 2

Role	Interface	Presence
Target	<code>rcITarget</code>	Mandatory

6.1.1.3.2.2 Configurable Items

The behavior of the Target role only depends on the parameters of the operation `downloadTask`. The operation `downloadTask` has a number of parameters that influence the behavior of the download task created.

6.1.1.3.2.3 Constraints

None.

6.1.1.3.3 Instantiation**6.1.1.3.3.1 Objects Created**

When the Target logical component is created an instance of the Target role is created. Multiple instances of the Target logical component can be created on a device. The different targets will need to listen to different ports and have different names.

Table 3

Type	Object	Multiplicity
Target	Target	1

6.1.1.3.3.2 Initial State

The following constraints apply to the initial state of a logical component instance:

— No download task has been started.

6.1.1.3.4 Execution Constraints

The Target logical component is thread safe.

6.1.1.4 Roles**6.1.1.4.1 Target****Signature**

```
role Target {
}
```

Qualifiers

— Root

Description

A Target enables the reception of components by a device. The target can be used to start a download task, which is a thread responsible for providing the target profile to other download roles and receiving and storing the transferred components.

Independent Attributes

— None

Invariants

— None

Instantiation

The Target role is always instantiated as part of the Target logical component.

Active Behavior

The Target role has no active behavior. Upon invocation of the `downloadTask` operation, the client must create a new thread. The active behavior of this task (thread) is providing the target profile to other download roles, receipt of components and storage of these components.

NOTE The transfer of components is initiated by a different role (initiator) which is not part of this logical component.

6.1.1.5 Interfaces

6.1.1.5.1 rclTarget

Qualifiers

— None

Description

This interface is used for activation of the target. Activation means that the target starts listening to a specific port for attempts to transfer components.

Interface ID

{bf56a0f9-0d34-4925-88b3-11631628e3bf}

6.1.1.5.1.1 DownloadTask

Signature

```
RcResult downloadTask([in] String locatorAddress,
                      [in] String targetName,
                      [in] String targetAddress,
                      [in] Int32 targetPort,
                      [in] String downloadDirectory);
};
```

Qualifiers

— Thread Safe

Description

By invoking this operation the target starts listening to the specified port for attempts to transfer components to this device.

Parameters**Table 4**

Name	Description
targetName	Name used to identify the target
targetAddress	Network address of the target. This address is registered at the locator, such that the other roles participating in the download process are able to locate this target.
targetPort	The port to which the target is listening for requests for component transfers.
downloadDirectory	Location to store the downloaded components (absolute path)

Return Values

Only the non-standard error codes are listed.

Table 5

Name	Description
RC_ERR_TARGET_LOCATOR_NOT_FOUND	The target is unable to connect to the specified locator.
RC_ERR_TARGET_CANNOT_STORE_COMPONENT	The target is unable to store received components.

Precondition

True

Action

Start thread listening to the `targetPort` for component transfer attempts.

Postcondition

Thread that is listening to the `targetPort` for component transfer attempts is started.

6.2 Interfaces suites for initiation of component transfer

6.2.1 Initiator

6.2.1.1 Concepts

The Initiator is one of the roles in the Download framework. The Download framework enables the transfer of components to a device. The Initiator is responsible for initiating and coordinating the component transfer process. Initiating means that this role will need to find all the entities that will be involved in the component transfer process and coordinate the interaction between these entities. For the client this logical component provides an operation to request for the transfer of a component to a device.

6.2.1.2 Types and Constants

6.2.1.2.1 Public Types and Constants

6.2.1.2.1.1 Error Codes

Signature

```
const rcResult RC_ERR_INITIATOR_LOCATOR_NOT_FOUND = 0x00000001
const rcResult RC_ERR_INITIATOR_TARGET_NOT_FOUND = 0x00000002
const rcResult RC_ERR_INITIATOR_REPOSITORY_NOT_FOUND = 0x00000004
const rcResult RC_ERR_INITIATOR_DECIDER_NOT_FOUND = 0x00000008
const rcResult RC_ERR_INITIATOR_COMPONENT_NOT_FOUND = 0x00000010
const rcResult RC_ERR_INITIATOR_DOWNLOAD_NOT_FEASIBLE = 0x00000020
```

Qualifiers

— Error codes

Description

The non-standard error codes that can be returned by functions of this logical component

Constants

Table 6

Name	Description
RC_ERR_INITIATOR_LOCATOR_NOT_FOUND	This error is returned when the locator is unable to connect to the locator
RC_ERR_INITIATOR_TARGET_NOT_FOUND	This error is returned when the target cannot be found (using the locator)
RC_ERR_INITIATOR_REPOSITORY_NOT_FOUND	This error is returned when no repository can be found (using the locator)
RC_ERR_INITIATOR_DECIDER_NOT_FOUND	This error is returned when no suitable decider can be found (using the locator)
RC_ERR_INITIATOR_COMPONENT_NOT_FOUND	This error is returned when the requested component is not available in the registered repositories (at the locator)
RC_ERR_INITIATOR_DOWNLOAD_NOT_FEASIBLE	This error is returned when the download is not feasible (decided by the decider)

6.2.1.2.2 Model Types and Constants

None.

6.2.1.3 Logical Component

6.2.1.3.1 Interface Role Model

Figure 4 — Interface-Role Model, depicts the interface-role model of the Initiator (grey box). It shows the interfaces and the roles involved with this component.

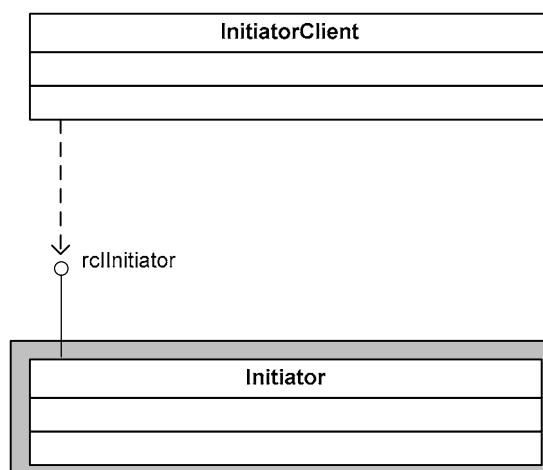


Figure 4 — Interface-Role Model

An Initiator Logical Component contains the Initiator role that provides the `rcIInitiator` interface. This interface can be used by a client to initiate the transfer of a specific component to a specific target. The Initiator role locates all of the involved entities (using the locator, see subclause 8.1.2) and also coordinates the download process.

6.2.1.3.2 Diversity

6.2.1.3.2.1 Provided Interfaces

Table 7

Role	Interface	Presence
Initiator	<code>rcIInitiator</code>	Mandatory

6.2.1.3.2.2 Configurable Items

The behavior of the Initiator role only depends on the parameters of the operation `getComp`. The operation `getComp` has a number of parameters that influence which component is to be transferred, to which target and the decider that is used.

6.2.1.3.2.3 Constraints

None.

6.2.1.3.3 Instantiation

6.2.1.3.3.1 Objects Created

When the Initiator logical component is created, an instance of the Initiator role is also created. Multiple instances of the Initiator logical component can be created on a device.

Table 8

Type	Object	Multiplicity
Initiator	Initiator	1

6.2.1.3.3.2 Initial State

The following constraints apply to the initial state of a logical component instance:

- No component transfers have been initiated from this initiator.
- Initiator is ready to initiate a component transfer

6.2.1.3.4 Execution Constraints

This logical component is Thread-Safe.