ISO/IEC 14543-5-11

Edition 1.0 2018-03

INTERNATIONAL STANDARD

colour inside

Information technology – Home electronic system (HES) architecture – Part 5-11: Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Remote user interface

ISO/IEC 14543-5-11:2018-03(en)

ISO/IEC 14543-5-11

Edition 1.0 2018-03

INTERNATIONAL STANDARD

colour inside

Information technology – Home electronic system (HES) architecture – Part 5-11: Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Remote user interface

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 35.240.67

ISBN 978-2-8322-5531-5

CONTENTS

INFORMATION TECHNOLOGY – HOME ELECTRONIC SYSTEM (HES) ARCHITECTURE –

Part 5-11: Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Remote user interface

FOREWORD

1) 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.

2) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees and ISO member bodies.

3) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC National Committees and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO, IEC or ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.

5) ISO and IEC do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. ISO or IEC are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC National Committees or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

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

International Standard ISO/IEC 14543-5-11 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The list of all currently available parts of the ISO/IEC 14543 series, under the general title *Information technology – Home electronic system (HES) architecture*, can be found on the IEC and ISO websites.

In this document, the following print types are used:

- CAPITAL LETTERS: for special functions or terms
- *italics*: for abstract entity names in the IGRS RUI model

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The ISO/IEC 14543-5 series of standards specifies the services and protocol of the application layer for Intelligent Grouping and Resource Sharing (IGRS) devices and services in the Home Electronic System.

The ISO/IEC 14543-5 series includes the following parts.

- IGRS Part 5-1: Core protocol
  - Specifies the TCP/IP protocol stack as the basis and the HTTP protocol as the message-exchange framework among devices.
  - Specifies a series of device and service interaction/invocation standards, including device and service discovery protocol, device and service description, service invocation, security mechanisms, etc.
  - Specifies core protocols for a type of home network that supports streaming media and other high-speed data transports within a home.
- IGRS Parts 5-2#: Application profile
  - Based on the IGRS Core Protocol.
  - Specifies a device and service interaction mechanism, as well as application interfaces used in IGRS basic applications.
  - Multiple application profiles are specified, including:
    i) Part 5-21: AV profile
    ii) Part 5-22: File profile
- IGRS Part 5-3: Basic application
  - Includes an IGRS basic application list.
  - Specifies a basic application framework.
  - Specifies operation details (device grouping, service description template, etc.), function definitions and service invocation interfaces.
- IGRS Part 5-4: Device validation
  - Defines a standard method to validate an IGRS-compliant device.
- IGRS Part 5-5: Device type
  - Specifies IGRS Device types used in IGRS applications.
- IGRS Part 5-6: Service type
  - Specifies basic service types used in IGRS applications.
- IGRS Part 5-7: Remote access system architecture
  - Specifies the architecture and framework for the remote access of IGRS devices and services in the Home Electronic System. The remote access communications protocol and application profiles are specified in the following parts of ISO/IEC 14543-5:
    i) ISO/IEC 14543-5-8: Remote access core protocol
    ii) ISO/IEC 14543-5-9: Remote access service platform
    iii) ISO/IEC 14543-5-101: Remote media access profile
    iv) ISO/IEC 14543-5-102: Remote universal management profile
    v) ISO/IEC 14543-5-11: Remote user interface
    vi) ISO/IEC 14543-5-12: Remote access test and verification
  - The relationships among these parts are specified in Part 5-7.

- IGRS Part 5-8: Remote access core protocol
  - Provides detailed system components, system function modules, basic concepts of IGRS remote access elements and their relationships, message exchange mechanisms and security related specifications.
  - Specifies interfaces between IGRS Remote Access (RA) client and service platforms. Defines co-operative procedures among IGRS RA clients.
- IGRS Part 5-9: Remote access service platform
  - Specifies the IGRS RA service platform (IRSP) architectures and interfaces among servers in the service platforms.
  - Based on Part 5-8: Remote access core protocol.
- IGRS Part 5-10#: Remote access application profiles
  - Defines a device and service interaction mechanism for various applications
  - Based on Part 5-8: Remote access core protocol
  - The following profile is under development:
    - i) Part 5-101: Remote media access profile. [1] This part defines the common requirements for IGRS RA media users and devices in IGRS networks.
  - Remote universal management profile will form the subject of a future Part 5-102. This part will specify a mechanism for integrating devices with both relatively high and low processing capabilities into IGRS networks. It will also specify universal remote device discovery and a management framework.
  - Additional application profiles will be specified in the future.
- IGRS Part 5-11: Remote user interface
  - Specifies adaptive user interface generation and remote device control mechanisms suitable for different remote access applications and devices.
- IGRS Part 5-12: Remote access test and verification[2]
  - Defines a standard method to test and verify IGRS-RA compliant device and service interfaces.

[1] Under preparation. Stage at the time of publication: ISO/IEC DIS 14543-5-101:2017.

[2] Under preparation. Stage at the time of publication: ISO/IEC DIS 14543-5-12:2017.

INFORMATION TECHNOLOGY – HOME ELECTRONIC SYSTEM (HES) ARCHITECTURE –

Part 5-11: Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Remote user interface

1 Scope

This part of ISO/IEC 14543-5 specifies a remote user interface (RUI) for the ISO/IEC 14543-5 series on IGRS for HES Class 2 and Class 3. It defines the mechanisms necessary for allowing an adaptive user interface to be displayed on and controlled by devices or control points from a remote location.

This document is applicable to IGRS local and remote access (RA) devices.

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/IEC 14543-5-1:2010, *Information technology – Home electronic system (HES) architecture – Part 5-1: Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Core protocol*

ISO/IEC 14543-5-8, *Information technology – Home electronic system (HES) architecture – Part 5-8: Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Remote access core protocol*

ISO/IEC 15045 (all parts), *Information technology – Home electronic system (HES) gateway*

IETF RFC 2045, *Multipurpose Internet Mail Extensions (MIME) – Part 1: Format of Internet Message Bodies*

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*

IETF RFC 4648, *The Base16, Base32, and Base64 Data Encodings*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

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

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

3.1.1
remote user interface
RUI
interface that is used for remote interaction with IGRS users

Note 1 to entry: The IGRS RUI can display information, play multimedia contents to the users or receive users' inputs.

3.1.2
RUI Client
RUIC
logical device in an IGRS network that possesses RUI retrieval, rendering capabilities and provides a user input interface to acquire messages for the RUIS

Note 1 to entry: An RUIC device may be physically a PC, a TV, a mobile phone, etc.

3.1.3
RUI Server
RUIS
logical device in an IGRS network that possesses capabilities for RUI contents storage and for providing RUI contents to the RUICs

Note 1 to entry: An RUIS device may be physically a PC, a set-top-box, a server, etc.

3.1.4
RUI Control Point
RUICP
logical device in an IGRS network that possesses RUI discovery and control capabilities.

Note 1 to entry: An RUICP device may be physically integrated into an RUIC or RUIS, or may be a stand-alone device.

3.2 Abbreviated terms

DC	Delivery Context
IGRS	Intelligent Grouping and Resource Sharing
RA	Remote Access
RDL	RUI Description Language
RUI	Remote User Interface
RUIS	RUI Server
RUIC	RUI Client
RUICP	RUI Control Point
SSDP	Simple Service Discovery Protocol
UI	User Interface

4 Conformance

For conformance to this document the following applies.

- The IGRS RUI system architecture shall conform to Clause 5.
- The device and service types of RUI Server (RUIS) and RUI Client (RUIC) shall conform to Clause 6.
- The abstract entities in IGRS RUI shall conform to Clause 7.
- The RUI Description Language (RDL) shall conform to Clause 8.

5 IGRS RUI overview

5.1 IGRS RUI features

The IGRS RUI supports the following features.

a) Discovery of an RUI: the RUIS lists available UIs for the user and exposes capabilities that it requires and supports.

b) Connecting to an RUI: the RUIC can connect to an RUIS and retrieve a matching RUI. A separate RUI Control Point (RUICP) can also set up or control this connection.

c) Presenting RUI content: an RUIS presents RUI contents to an RUIC. The condition of an RUIC is described in RDL. This enables the RUI device to adjust the RUI contents according to the condition of the RUIC.

5.2 RUI configuration models

5.2.1 Overview

Three basic models for RUI configuration are defined:

a) Internet RUI configuration model;

b) 2-tier RUI configuration model;

c) 3-tier RUI configuration model.

5.2.2 Internet RUI configuration model

In this configuration, the RUIS is on the Internet. An RUIC may be discoverable (stand-alone RUICP) or non-discoverable (RUICP and RUIC in the same box). The Internet RUI configuration model is shown in Figure 1.

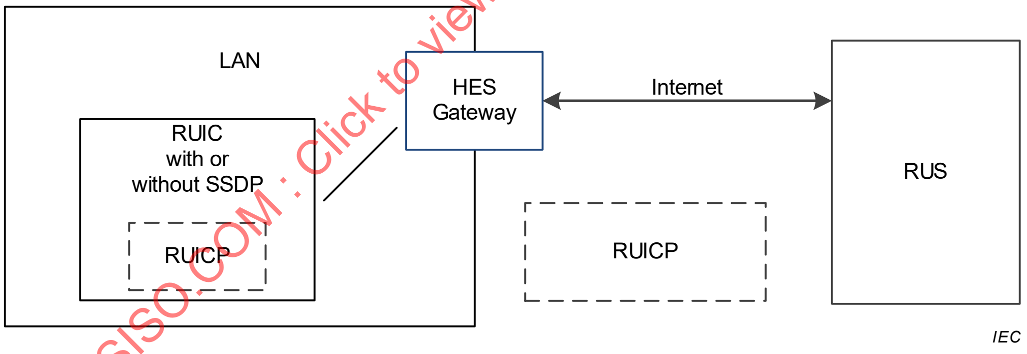


Figure 1 – Internet RUI configuration model

The HES gateway is specified in the ISO/IEC 15045 series (ISO/IEC 15045-1 and ISO/IEC 15045-2 are published). The HES gateway shall be used to connect the RUIC on the LAN and the Internet.

NOTE ISO/IEC 15045-3, under development, is expected to address privacy and security of data passing through the HES gateway.

5.2.3 2-tier RUI configuration model

In this configuration, all the RUI entities are on the same LAN, and the RUIS is discoverable. The RUIC is not discoverable in this configuration, so it needs to have a RUICP in it. The 2-tier RUI configuration model is shown in Figure 2.

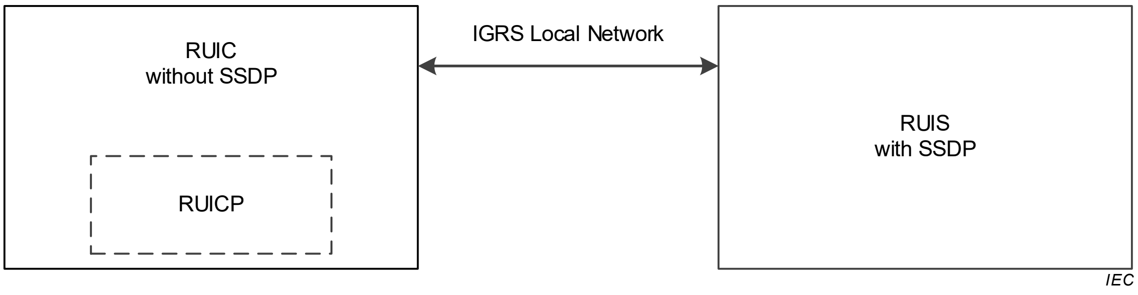


Figure 2 – 2-tier RUI configuration model

5.2.4 3-tier RUI configuration model

In this configuration, all the RUI entities are on the same LAN. The RUIS and RUIC are both discoverable. The RUICP can be stand-alone or reside in the RUIS or RUIC. The 3-tier RUI configuration model is shown in Figure 3.

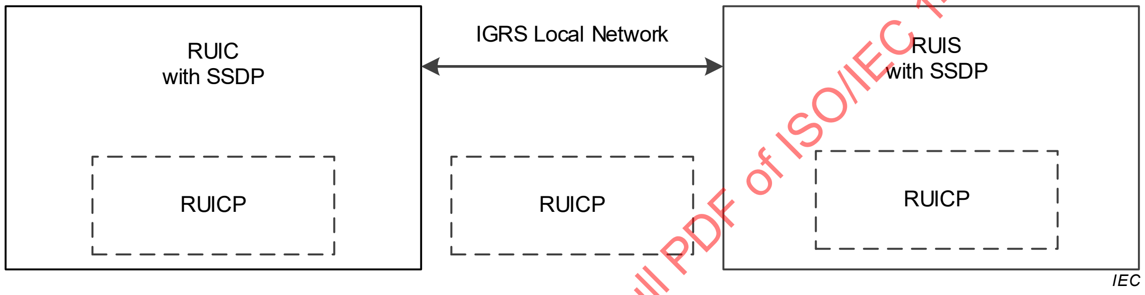


Figure 3 – 3-tier RUI configuration model

5.3 RUIS and RUIC types

Two types are defined for RUIS and RUIC.

a) Internet-type: the RUIC or RUIS is not discoverable on an IGRS network.

b) IGRS-type: the RUIC or RUIS is discoverable on an IGRS network, and shall be controlled by an RUICP using the IGRS invocation interface specified in ISO/IEC 14543-5-1 and ISO/IEC 14543-5-8.

5.4 RUI architecture

5.4.1 Detailed RUI architecture

The detailed RUI architecture is shown in Figure 4.

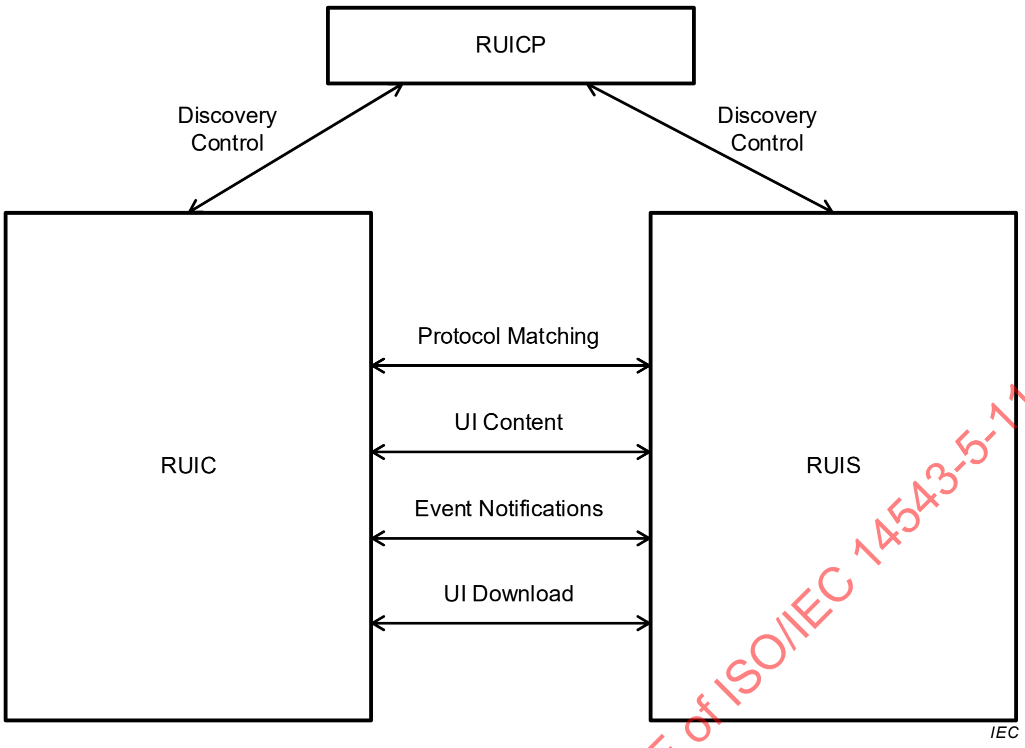


Figure 4 – Detailed RUI architecture

The IGRS RUI consists of one RUICP and two RUI devices: RUIC and RUIS. The RUIS provides RUI contents, which are described using RDL. The RUIC retrieves RDL or HTML documents from the RUIS, renders them and provides a UI.

5.4.2 RUIC architecture

The architecture of an RUIC is shown in Figure 5 and consists of following components.

a) Discovery Manager: Optional component – Supports the IGRS discovery protocol (see ISO/IEC 14543-5-1:2010, Clause 9).

b) IGRS RUI Browser: Retrieves and renders RUI contents from server. These contents may be described in HTML or RDL. If described in RDL, an internal RDL Parser parses the RDL document and generates HTML content.

c) IGRS Control: Optional component – Controls other IGRS devices according to the user action on the RUI.

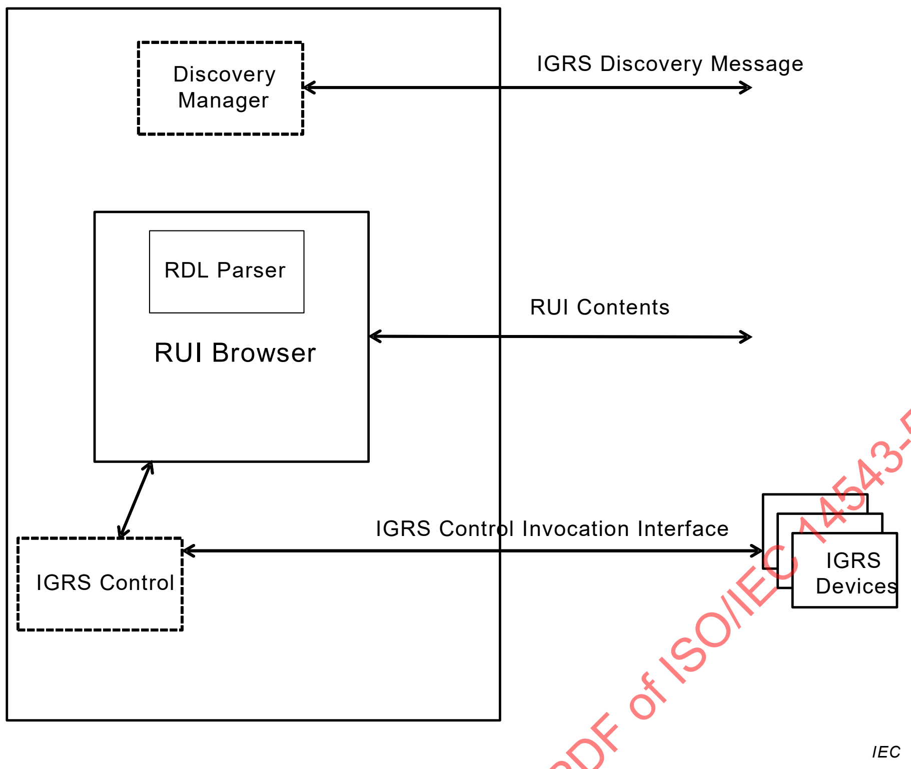


Figure 5 – RUIC architecture

5.4.3 RUI Server architecture

The architecture of an RUIS is shown in Figure 6 and consists of following components:

a) Discovery Manager: Optional component – Supports the IGRS discovery protocol.

b) IGRS RUIS: Serves RUI contents.

c) RDL Contents Storage: Repository for contents described in RDL.

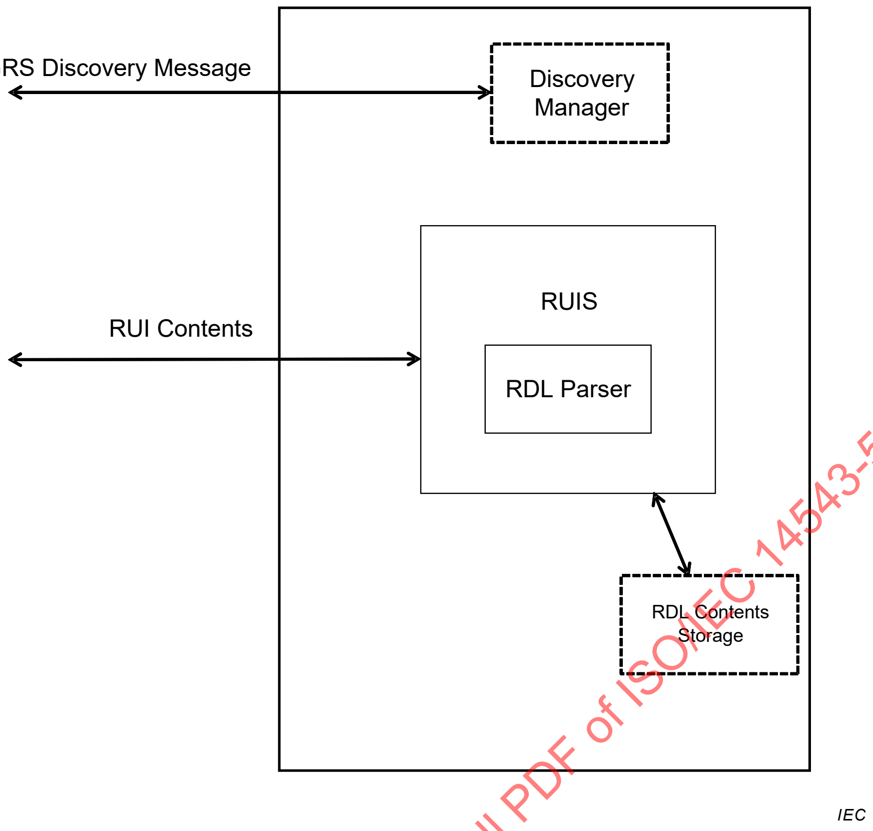


Figure 6 – RUIS architecture

6 IGRS type definitions for RUIS and RUIC

6.1 Overview

Clause 6 describes IGRS type definitions for RUIS and RUIC, including the device type definition and service type definition. The functions of the IGRS RUIS and RUIC are to find and match the most suitable RDL UI between RUIS and RUIC. RDL information exchange is processed on the RDL UI, and is not within the scope of Clause 6.

6.2 IGRS device types for RUIS and RUIC

The device types for RUIS and RUIC are shown in Table 1.

Table 1 – RUI device type definitions

Device type name	Functional device type Identifier	Field explanation
RUIServer	urn:IGRS:Device:DeviceType:RUIServerDevice	Device that provides UI contents
RUIClient	urn:IGRS:Device:DeviceType:RUIClientDevice	Device that renders UI contents

6.3 IGRS service types for RUIS and RUIC

The service types for RUIS and RUIC are shown in Table 2.

Table 2 – RUI service type definitions

Service type name	Service type identifier	Field explanation
RUIS Service	urn:IGRS:Service:ServiceType:RUIServer:1	Service that provides UI contents
RUI Client Service	urn:IGRS:Service:ServiceType: RUICelinet:1	Service that renders UI contents

6.4 IGRS invocation interfaces for RUIC service

There are two invocation interfaces for RUIC service:

a) Interface name: Connect
   Description: Connect to an RUIS

b) Interface name: Disconnect
   Description: Close connection to an RUIS

6.5 IGRS invocation interface for RUIS service

There is only one invocation interface specified for RUIS service:

Interface name: GetRuiUri
Description: Returns URI of RUI Content Location

6.6 IGRS RUI operation scenarios

6.6.1 Overview

Subclause 6.6 shows IGRS RUI scenarios based on an IGRS invocation interface, and these are valid for IGRS devices for 2-tier or 3-tier RUI configuration. The purpose of the IGRS specification for an RUI is to find a server and to match a suitable UI (application) between an RUIS and RUIC. For Internet-type devices, the RUIC knows information about the RUIS and does not require an IGRS invocation interface.

6.6.2 Discovery and retrieval of server information

RUISs and RUICs each issue a Simple Service Discovery Protocol (SSDP) packet for the corresponding server.

6.6.3 Connecting

The RUICP may display a list of RUICs and RUISs. If the user selects RUIS–UI–RUIC, the RUICP issues a Connect() invocation interface to the RUIC. In this case, the argument of the Connect() invocation interface is the URL of the selected UI.

Also, matching can be done on the RUIC. In this case, the RUICP issues a Connect() invocation interface with the URL of the RUIS. When the RUIC receives this invocation interface, it chooses the most suitable UI from the list and connects.

This scenario is shown on Figure 7. For the RUIC to receive the RUI from the RUIS, it needs the address of the RUI content. This RUI content address can be set by using the internal user interface of RUI (e.g. address bar of Browser of a smart TV), or by using the RUICP. Once the address of the RUI content is given by either an internal UI or RUICP, the RUIC retrieves the contents and renders it on the device.

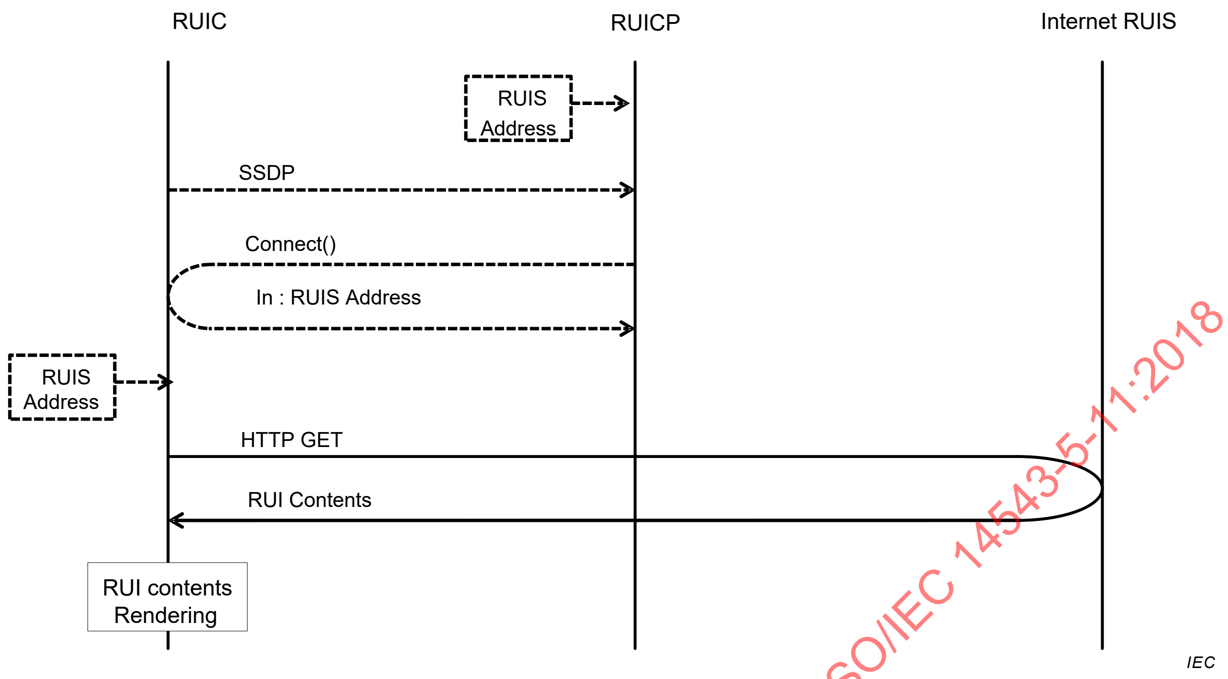


Figure 7 – RUI retrieval from Internet RUIS

6.6.4 Controlling IGRS devices using IGRS RUI

To control IGRS devices using IGRS RUI, IGRS RUIC shall have an IGRS Control module that is bound to an IGRS CP. Figure 8 shows a configuration example for controlling an IGRS device using an IGRS RUI, where the RUIC is set as an Internet RUI configuration model. The RUI Browser in the RUIC handles RUI retrieval, rendering and user interaction. The IGRS Control handles IGRS control messages between the RUIC and IGRS device. The RUI Browser and IGRS Control are bound to each other using JavaScript.

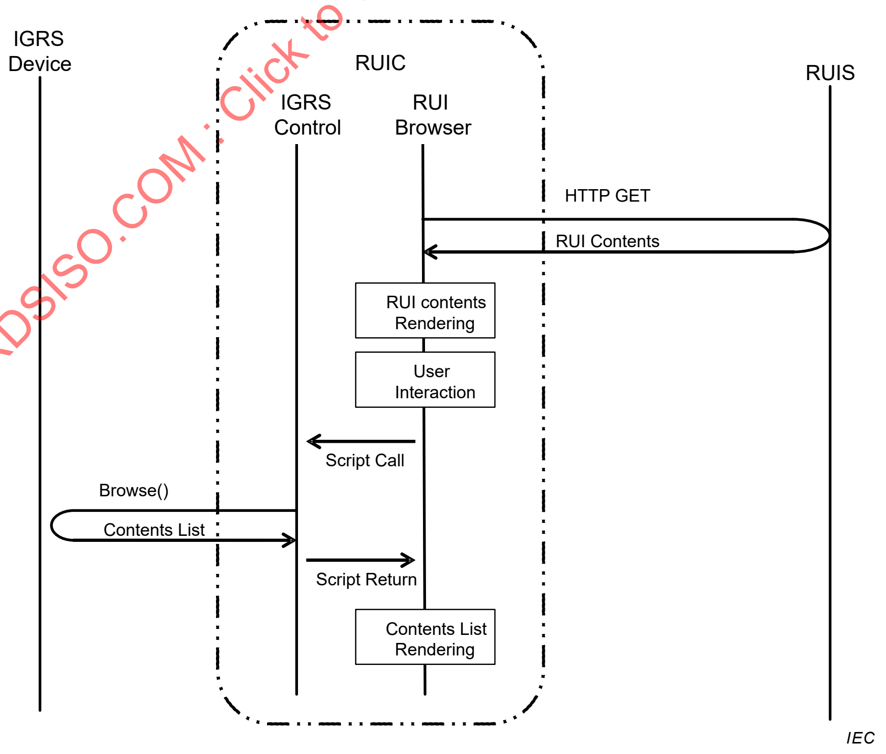


Figure 8 – Example configuration of controlling IGRS device with RUI

The detailed scenario is as follows:

a) The RUI Browser retrieves the RUI (which is prepared for an IGRS device) from the RUIS.

b) The RUI Browser renders the RUI contents. If information about the IGRS device is needed, it calls IGRS Control using JavaScript binding.

c) The RUI Browser receives user commands using RUI contents. If it needs to control an IGRS device, it calls IGRS Control using JavaScript binding.

7 Abstract entity in IGRS RUI

16 abstract entities in an IGRS RUI model are defined in italics in Table 3.

Table 3 – Definitions of abstract entities in an IGRS RUI

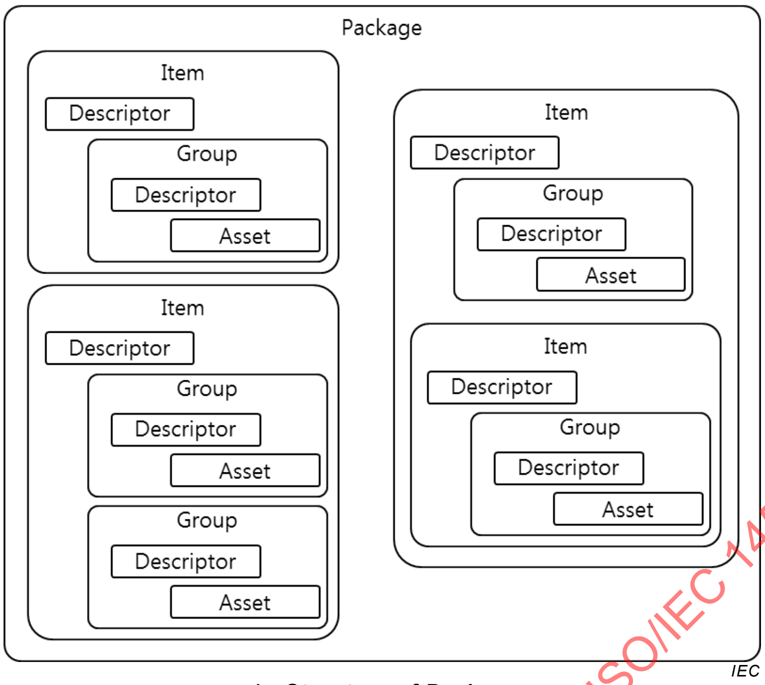
Entity name	Definition
Package	A structure that allows *Items* to be grouped. These groupings of *Items* can be used to form logical *Packages* (for organization).
LayoutContainer	A structure that allows *Layouts* to be grouped. These groupings of *Layouts* can be used to form logical *Packages*.
SceneContainer	A structure that allows *Scenes* to be grouped. These groupings of scenes can be used to form logical packages.
Item	A group or groups of sub-items that are bound to relevant *Descriptors*. *Descriptors* contain information about the *Item*, as a representation of an *Asset*. *Items* may contain *Choices*, which allow them to be customized or configured. *Items* may be conditional.
Layout	A *Layout* is a grouping of sub-layouts and/or layouts that are bound to relevant *ItemRefs*. *Layouts* may contain *Conditions* and *Choices* which allow them to be customized or configured.
Scene	A *Scene* is the binding of scene itself and corresponding *LayoutRef*.
Group	A *Group* is the binding of an *Asset* to a set of *Descriptors*. These descriptors are information concerning all or part of the specific *Asset* instance. Such descriptors will typically contain control or structural information about the *Asset* (such as bit rate, character set, start points or encryption information) but not information describing the "content" within.
Asset	An *Asset* is an individually identifiable *Asset* such as a video or audio clip, an image, or a textual *Asset*. An *Asset* may also potentially be a physical object.
SceneNavigation	A *SceneNavigation* associates information about navigation within a *Scene*.
Annotation	An *Annotation* associates brief description with the enclosing entity.
Descriptor	A *Descriptor* associates information with the enclosing entity. This information may be a *Group* (such as a thumbnail of an image, or a text component), or a textual statement.
Condition	A *Condition* describes the enclosing entity as being optional and links it to the *Selection(s)* that affect its inclusion. Multiple predicates within a *Condition* are combined as a conjunction (an AND relationship). Any predicate may be negated within a *Condition*. Multiple *Conditions* associated with a given entity are combined as a disjunction (an OR relationship) when determining whether to include the entity.
Choice	A *Choice* describes a set of related *Selections* that can affect the configuration of an *Item*. The *Selections* within a choice are either exclusive (choose exactly one) or inclusive (choose any number, including all or none).
Selection	A *Selection* describes a specific decision that will affect one or more *Conditions* somewhere within an *Item*. If the *Selection* is chosen, its predicate becomes true; if it is not chosen, its predicate becomes false; if it is left unresolved, its predicate is undecided.
Statement	A *Statement* is a literal textual value that contains information.
DCCondition	A *DCCondition* describes *Conditions* using Delivery Context (DC) that can influence the scalability of Scalable Remote User Interfaces and the adaptation of services. It use the Reverse Polish Notation, which can be easily implemented using stack-based function to express complex conditions based on mathematical expressions such as Boolean expressions, comparison expressions or arithmetic expressions.

Figure 9 shows the structures of the major entities in an IGRS RUI model. The major entities include *Package*, *LayoutContainer* and *SceneContainer*.

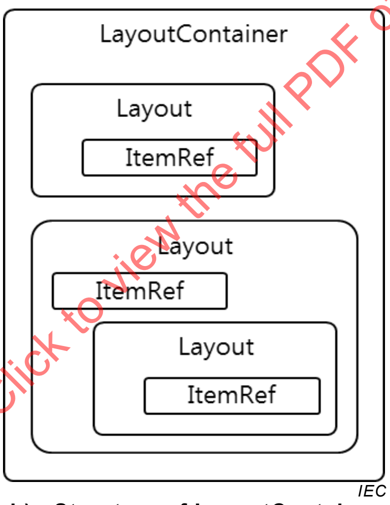
Figure 9 a) shows the structure of a *Package*, which is the top entity and is composed of a combination of *Items*. An *Item* can be composed of sub-Items and/or *Groups*. A *Group* is a binding of an *Asset* to a set of *Descriptors*. Each *Item* and/or *Group* shall have a *Descriptor* to describe itself.

Figure 9 b) shows the structure of a *LayoutContainer*, which is composed of multiple *Layouts*. A *Layout* is composed of *ItemRefs* and/or sub-Layouts, where the *ItemRef* is a pointer to an *Item*.

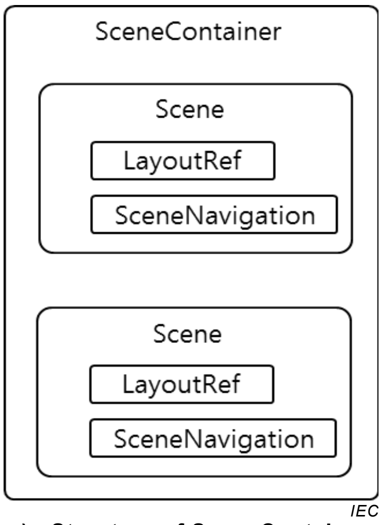
Figure 9 c) shows the structure of a *SceneContainer*, which is composed of multiple *Scenes*. A *Scene* is composed of *LayoutRefs* and *SceneNavigations*, where the *LayoutRef* is a pointer to a *Layout* and the *SceneNavigation* contains navigational information between *Scenes*.



a) **Structure of Package**



b) **Structure of LayoutContainer**



c) **Structure of SceneContainer**

Figure 9 – Structures of major entities in an IGRS RUI application

8 RUI Description Language (RDL)

8.1 Overview

RDL is a description language in the form of an XML schema. RDL describes functions of selecting and filtering RUI documents, fragments of an RUI document, or multimedia contents of the IGRS service based on the factors such as device capabilities, user information, user viewing environment and service policies. In addition, RDL supports efficient version management of the RUI documents and fragments of an RUI document or multimedia contents of the IGRS service, which are defined as digital Items of an RUI. The RDL schema is specified in Annex A. The elements of this schema are explained in the following sub-clauses.

8.2 RDL element

An RDL element is the root element of the RDL instance document. An RDL root element can be either an RDL or an RDLPackage. RDL root elements have a Declarations element and/or more than one Package, LayoutContainer or SceneContainer. RDL elements can be used for moving or transporting Package, LayoutContainer and SceneContainer, which are described in an RDL document.

An RDL element shall include the namespace declaration to declare the RDL namespace of the RDL element and its contents. Applications are responsible for recognizing RDL documents. The namespace can be described as either “default namespace declaration” or “prefix-specific namespace declaration”.

The RDL namespace declaration is given in Figure 10. The definition of the RDL element is shown in Figure 11.

```
<!--**********************************************
Default namespace declaration
************************************************-->
<RDL xmlns=“urn:saf:2012:06-RDL-NS”>
...
</RDL>

or

<!--**********************************************
Prefix-specific namespace declaration
************************************************-->
<rdl:RDL xmlns:rdl="urn:saf:2012:06-RDL-NS">
...
</rdl:RDL>
```

IEC

Figure 10 – RDL namespace declaration

```
<element name="RDL" type="rdl:RDLType"/>
<complexType name="RDLType">
   <sequence>
      <element name="RDLInfo" type="rdl:RDLInfoType" minOccurs="0"
                     maxOccurs="unbounded"/>
      <element name="Declarations" type="rdl:DeclarationsType"
                   minOccurs="0"/>
      <element name="Package" type="rdl:PackageType" minOccurs="0"/>
      <element name="LayoutContainer" type="rdl:LayoutContainerType"
                   minOccurs="0"/>
      <element name="SceneContainer" type="rdl:SceneContainerType"
                   minOccurs="0"/>
   </sequence>
   <attribute name="RDLDocumentId" type="anyURI"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

IEC

Figure 11 – RDL element definition

8.3 RDLPackage element

An RDLPackage root element has a Declarations element and contains at least one of each Package, LayoutContainer and SceneContainer. The RDLPackage element presents one RUI using Package, LayoutContainer or SceneContainer, which is contained in an RDLPackage document.

An RDLPackage element shall include a namespace declaration to declare the RDL namespace such as an RDL element defined in 8.2. The definition of the RDLPackage element is shown in Figure 12.

```
<element name="RDLPackage" type="rdl:RDLPackageType"/>
<complexType name="RDLPackageType">
      <sequence>
            <element name="RDLInfo" type="rdl:RDLInfoType"
                        minOccurs="0" maxOccurs="unbounded"/>
            <element name="Declarations" type="rdl:DeclarationsType"
                        minOccurs="0"/>
            <element name="Package" type="rdl:PackageType"/>
            <element name="LayoutContainer" type="rdl:LayoutContainerType"/>
            <element name="SceneContainer" type="rdl:SceneContainerType"/>
      </sequence>
      <attribute name="RDLPackageId" type="anyURI"/>
      <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

IEC

Figure 12 – RDLPackage element definition

An RDLPackage element is configured as a combination of Packages, LayoutContainer and SceneContainer. An example of an RDLPackage is shown in Figure 13.

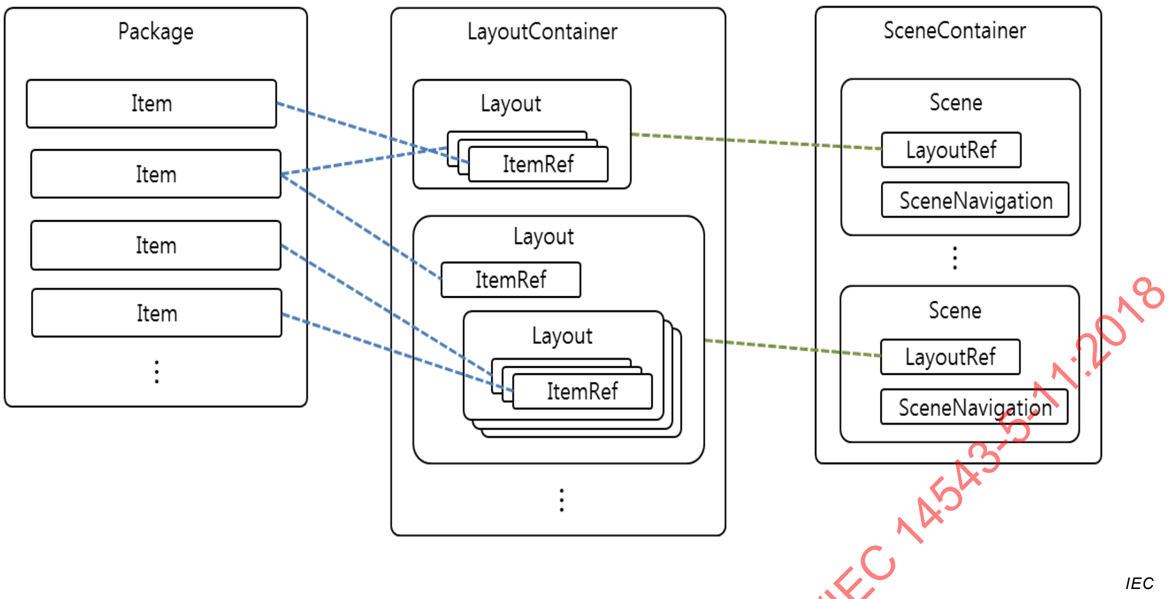


Figure 13 – Example of an RDLPackage element

8.4 RDLInfo element

An RDLInfo element describes information applicable to an RDL document. RDLInfoType type is declared in any type. If an application cannot interpret information of any type, it ignores this element. The definition of the RDLInfo element is shown in Figure 14.

```
<element name="RDLInfo" type="rdl:RDLInfoType"/>
<complexType name="RDLInfoType">
    <sequence>
        <any namespace="##any" processContents="lax"/>
    </sequence>
</complexType>
```

IEC

Figure 14 – RDLInfo element definition

8.5 Declaration element

A Declaration element declares elements such as Item, Descriptor, Group, DCCondition and Anchor. A declared element can be used in an RDL document later. A Declaration element includes Item, Descriptor, DCCondition, Group and Anchor elements. Multiple numbers of elements can be defined in a single Declaration element. The definition of the Declaration element is shown in Figure 15.

```
<element name="Declaration" type="rdl:DeclarationType"/>
<complexType name="DeclarationsType">
    <choice maxOccurs="unbounded">
        <element name="Item" type="rdl:ItemType"/>
        <element name="Layout" type="rdl:LayoutType"/>
        <element name="Descriptor" type="rdl:DescriptorType"/>
        <element name="DCCondition" type="rdl:DCConditionType"/>
        <element name="Group" type="rdl:GroupType"/>
        <element name="Choice" type="rdl:ChoiceType"/>
    </choice>
</complexType>
```

IEC

Figure 15 – Declaration element definition

8.6 Package element

The Package element represents a container. It is a grouping of Items bound with a set of Descriptors that contains descriptive information about the container. A Package has any attribute from other namespaces. Such attributes provide an additional representation of a descriptor with descriptive information about the container by means of an attribute. The definition of the Package element is shown in Figure 16.

```
<element name="Package" type="rdl:PackageType"/>
<complexType name="PackageType">
   <complexContent>
      <extension base="rdl:SDIDBaseType">
         <sequence>
            <element name="Annotation" type="rdl:AnnotationType"
                     minOccurs="0" maxOccurs="unbounded"/>
            <element name="Item" type="rdl:ItemType"
                     minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attributeGroup ref="rdl:ID_ATTRS"/>
         <attributeGroup ref="rdl:VERSION_ATTRS"/>
         <anyAttribute namespace="##other" processContents="lax"/>
      </extension>
   </complexContent>
</complexType>
```

IEC

Figure 16 – Package element definition

8.7 LayoutContainer element

A LayoutContainer element can be composed of Layouts, and is a grouping of Layouts to construct a logical package. LayoutContainer elements include the anyAttribute attribute to convey information about LayoutContainer. An Annotation element includes a brief description of the Layout element. A LayoutContainer type contains a Layout element and an Annotation element. The definition of the LayoutContainer element is shown in Figure 17.

```
<element name="LayoutContainer" type="rdl:LayoutContainerType"/>
<complexType name="LayoutContainerType">
   <complexContent>
      <extension base="rdl:SDIDBaseType">
         <sequence>
            <element name="Annotation" type="rdl:AnnotationType"
                     minOccurs="0" maxOccurs="unbounded"/>
            <element name="Layout" type="rdl:LayoutType"
                     minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attributeGroup ref="rdl:ID_ATTRS"/>
         <attributeGroup ref="rdl:VERSION_ATTRS"/>
         <anyAttribute namespace="##other" processContents="lax"/>
      </extension>
   </complexContent>
</complexType>
```

IEC

Figure 17 – LayoutContainer element definition

8.8 SceneContainer element

A SceneContainer element can be composed of Scenes, and is a grouping of Scenes to construct a logical package. SceneContainer elements include the anyAttribute attribute to convey information about a SceneContainer. An Annotation element has a brief description of the SceneContainer element. A SceneContainer type contains an Item element and an Annotation element. The definition of the SceneContainer element is shown in Figure 18.

```
<element name="SceneContainer" type="rdl:SceneContainerType"/>
<complexType name="SceneContainerType">
   <complexContent>
       <extension base="rdl:SDIDBaseType">
           <sequence>
               <element name="Annotation" type="rdl:AnnotationType"
                               minOccurs="0" maxOccurs="unbounded"/>
               <element name="Scene" type="rdl:SceneType"
                               minOccurs="0" maxOccurs="unbounded"/>
           </sequence>
           <attributeGroup ref="rdl:ID_ATTRS"/>
           <attributeGroup ref="rdl:VERSION_ATTRS"/>
           <anyAttribute namespace="##other" processContents="lax"/>
       </extension>
   </complexContent>
</complexType>
```

IEC

Figure 18 – SceneContainer element definition

8.9 Item element

An Item element represents an Item. It is a grouping of possible sub-Items and/or Groups bound to a set of relevant Annotations containing descriptive information about the item. In addition, an Item can be made conditional via a set of DCCondition and Condition child elements, which are made configurable via a set of Choice elements.

An Item may have any attribute from other namespaces. Such attributes provide additional representations of a descriptor with descriptive information about the Item by means of an attribute. The IdRef attribute contains the pre-defined ID of an Item. The definition of the Item element is shown in Figure 19.

```
<element name="Item" type="rdl:ItemType"/>
<complexType name="ItemType">
   <complexContent>
     <extension base="rdl:SDIDBaseType">
       <sequence>
         <element name="Annotation" type="rdl:AnnotationType"
                                 minOccurs="0" maxOccurs="unbounded"/>
         <element name="Condition" type="rdl:ConditionType"
                                 minOccurs="0" maxOccurs="unbounded"/>
         <element name="Choice" type="rdl:ChoiceType" minOccurs="0"
                                 maxOccurs="unbounded"/>
         <element name="Descriptor" type="rdl:DescriptorType"
                                 minOccurs="0" maxOccurs="unbounded"/>
         <choice minOccurs="0" maxOccurs="unbounded">
             <element name="Item" type="rdl:ItemType"/>
             <element name="Group" type="rdl:GroupType"/>
         </choice>
       </sequence>
       <attributeGroup ref="rdl:ID_ATTRS"/>
       <attributeGroup ref="rdl:IDREF_ATTRS"/>
       <attributeGroup ref="rdl:VERSION_ATTRS"/>
       <anyAttribute namespace="##other" processContents="lax"/>
     </extension>
   </complexContent>
</complexType>
```

IEC

Figure 19 – Item element definition

8.10 Layout element

A Layout element contains the description of the layout of an RUI. A Layout element can be composed of a sub-Layout and/or ItemRef, which designates the Item related to the Layout. The Layout element may contain an Annotation for a brief description and/or a Descriptor for a full description.

Conditions to construct a Layout element are described in the Layout element using a Condition element and a Choice element. A Layout element should contain horizontalSize and verticalSize attributes to describe the size of a Layout element, and may contain a subLayoutOrder attribute for the order of each sub-Layout. The SubLayoutOrder starts from the top-left of the user interface, and continues from left to right and from top to bottom. The definition of the Layout element is shown in Figure 20.

```
<element name="Layout" type="rdl:LayoutType"/>
<complexType name="LayoutType">
   <complexContent>
        <extension base="rdl:DIDBaseType">
             <sequence>
                  <element name="Annotation" type="rdl:AnnotationType"
                                  minOccurs="0" maxOccurs="unbounded"/>
                  <element name="Condition" type="rdl:ConditionType"
                                  minOccurs="0" maxOccurs="unbounded"/>
                  <element name="Choice" type="rdl:ChoiceType" minOccurs="0"
                                  maxOccurs="unbounded"/>
                  <element name="Descriptor" type="rdl:DescriptorType"
                                  minOccurs="0" maxOccurs="unbounded"/>
                  <choice minOccurs="0" maxOccurs="unbounded">
                       <element name="Layout" type="rdl:LayoutType"/>
                       <element name="ItemRef" type="rdl:ItemRefType"/>
                  </choice>
             </sequence>
             <attributeGroup ref="rdl:ID_ATTRS"/>
             <attributeGroup ref="rdl:IDREF_ATTRS"/>
             <attributeGroup ref="rdl:VERSION_ATTRS"/>
             <attributeGroup ref="rdl:LAYOUT_ATTRS"/>
             <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
   </complexContent>
</complexType>
```

IEC

Figure 20 – Layout element definition

8.11 Scene element

A Scene element can be composed of LayoutRefs, which designate the Layouts related to the Scene. Navigational information between Scenes is described in the SceneNavigation. A Scene element may contain an Annotation conveying a brief description. The definition of the Scene element is shown in Figure 21.

```
<element name="Scene" type="rdl:SceneType"/>
<complexType name="SceneType">
   <complexContent>
        <extension base="rdl:DIDBaseType">
             <sequence>
                  <element name="Annotation" type="rdl:AnnotationType"
                                  minOccurs="0" maxOccurs="unbounded"/>
                  <element name="LayoutRef" type="rdl:LayoutRefType"/>
                  <element name="SceneNavigation" type="rdl:SceneNavigationType"
                                  minOccurs="0" maxOccurs="unbounded"/>
             </sequence>
             <attributeGroup ref="rdl:ID_ATTRS"/>
             <attributeGroup ref="rdl:VERSION_ATTRS"/>
             <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
   </complexContent>
</complexType>
```

IEC

Figure 21 – Scene element definition

8.12 Group element

A Group element represents a group. It groups an Asset element with a set of Descriptors containing descriptive information about the Asset. The Group, being a logical union of an Asset with relevant descriptive data and anchors, is intended to be the basic unit of digital content within an RDL document.

A Group may have any attribute from other namespaces. Such attributes provide an additional representation of a descriptor with descriptive information about the Asset by means of an attribute of the Group. If multiple Asset children are present, they are considered bit equivalent and any one of them may be used. The definition of the Group element is shown in Figure 22.

```
<element name="Group" type="rdl:GroupType"/>
<complexType name="GroupType">
   <complexContent>
     <extension base="rdl:DIDBaseType">
         <sequence>
             <element name="Annotation" type="rdl:AnnotationType"
                              minOccurs="0" maxOccurs="unbounded"/>
             <element name="Condition" type="rdl:ConditionType"
                              minOccurs="0" maxOccurs="unbounded"/>
             <element name="Descriptor" type="rdl:DescriptorType"
                              minOccurs="0" maxOccurs="unbounded"/>
             <element name="Asset" type="rdl:AssetType"
                              maxOccurs="unbounded"/>
         </sequence>
         <attributeGroup ref="rdl:ID_ATTRS"/>
         <attributeGroup ref="rdl:IDREF_ATTRS"/>
         <attributeGroup ref="rdl:VERSION_ATTRS"/>
         <anyAttribute namespace="##other" processContents="lax"/>
     </extension>
   </complexContent>
</complexType>
```

IEC

Figure 22 – Group element definition

8.13 Asset element

An Asset element represents an asset. It defines an individually identifiable Asset such as a video or audio clip, an image, a textual Asset, etc. Normally, an Asset is defined in an Asset element by reference, which specifies the URI of the Asset in the ref attribute. The URI identifies the Asset for the purpose of allowing an application to retrieve the contents of the Asset. This URI can be a traditional URL, which gives the explicit physical location from which to retrieve the contents, or a more abstract identifier, such as a URN, which identifies the Asset contents independent of location.

The data type of the Asset is identified by the mimeType attribute, which is a concatenation of MIME media-type, sub-type and parameters as specified in IETF RFC 2045 (e.g. 'video/mpeg').

The mimeType attribute identifies the data type of the Asset before any content-encodings specified in the contentEncoding attribute are applied to the Asset. The MIME media-type shall be modified by the presence of the contentEncoding attribute, which specifies the content-encoding as specified in IETF RFC 2616. When present, the value of the contentEncoding attribute indicates what additional content-encodings have been applied to the Asset, and thus what decoding mechanisms to apply in order to obtain the MIME media-type identified by the mimeType attribute.

The encoding attribute specifies the encoding format for including the Asset by value. If the Asset is included by value and no such encoding is applied, then the encoding attribute shall be omitted. If the encoding attribute is present, the value shall be set to “base64” and the Asset shall be provided by value using a base64 encoding, as specified in IETF RFC 4648.

An Asset may have any attribute from other namespaces. Such attributes may provide additional information about the Asset. The definition of the Asset element is shown in Figure 23.

```
<element name="Asset" type="rdl:AssetType"/>
<complexType name="AssetType" mixed="true">
 <complexContent mixed="true">
   <extension base="rdl:DIDBaseType">
     <sequence>
       <any namespace="##any" processContents="lax" minOccurs="0"/>
     </sequence>
     <attribute name="mimeType" type="string" use="required"/>
     <attribute name="ref" type="anyURI"/>
     <attribute name="encoding" type="string"/>
     <attribute name="contentEncoding" type="NMTOKENS"/>
     <anyAttribute namespace="##other" processContents="lax"/>
   </extension>
 </complexContent>
</complexType>
```

IEC

Figure 23 – Asset element definition

8.14 ItemRef element

An ItemRef is a child element of a Layout element, and contains the ID of the Item related to parent Layout element. The internalItem_ID describes an Item in the same document, while externalItem_ID describes an Item in another document. The definition of the ItemRef element is shown in Figure 24.

```
<element name="ItemRef" type="rdl:ItemRefType"/>
<complexType name="ItemRefType">
  <complexContent>
    <extension base="rdl:DIDBaseType">
      <attribute name="internalItem_ID" type="IDREF" use="optional"/>
      <attribute name="externalItem_ID" type="anyURI" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

IEC

Figure 24 – ItemRef element definition

8.15 LayoutRef element

A LayoutRef is a child element of a Scene element, and contains the ID of the Layout related to the parent Scene element. internalItem_ID describes an Item in the same document, while externalItem_ID describes an item in another document. The definition of the ItemRef element is shown in Figure 25.

```
<element name="LayoutRef" type="rdl:LayoutRefType"/>
<complexType name="LayoutRefType">
   <complexContent>
        <extension base="rdl:DIDBaseType">
           <attribute name="internalItem_ID" type="IDREF" use="optional"/>
           <attribute name="externalItem_ID" type="anyURI" use="optional"/>
        </extension>
   </complexContent>
</complexType>
```

IEC

Figure 25 – LayoutRef element definition

8.16 SceneNavigation element

A SceneNavigation element describes navigational information from one Scene to another Scene. Navigational information is described in the Descriptor element, and the destination is described in the refScene_ID attribute. The definition of the SceneNavigation element is shown in Figure 26.

```
<element name="SceneNavigation" type="rdl:SceneNavigationType"/>
<complexType name="SceneNavigationType">
   <complexContent>
        <extension base="rdl:DIDBaseType">
             <sequence>
                  <element name="Annotation" type="rdl:AnnotationType"
                                      minOccurs="0" maxOccurs="unbounded"/>
                  <element name="Descriptor" type="rdl:DescriptorType"
                                      minOccurs="0" maxOccurs="unbounded"/>
             </sequence>
             <attribute name="refScene_ID" type="IDREF" use="required"/>
             <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
   </complexContent>
</complexType>
```

IEC

Figure 26 – SceneNavigation element definition

8.17 Annotation element

An Annotation element contains brief information of its parent element, and only a text value is allowed. The definition of the Annotation element is shown in Figure 27.

```
<element name="Annotation" type="rdl:AnnotationType"/>
<complexType name="AnnotationType" mixed="true">
   <complexContent mixed="true">
        <extension base="rdl:DIDBaseType">
              <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
   </complexContent>
</complexType>
```

IEC

Figure 27 – Annotation element definition

8.18 Descriptor element

A Descriptor element represents a Descriptor. It associates information with its parent element. This information may be contained in a Group element, or a Statement element.

Typically, a Descriptor element associates descriptive data with its parent element. Descriptive data may be Group or Statement type. An example of a Group element containing descriptive data is a thumbnail version of a photographic image. An example of a Statement element containing descriptive data is a simple textual description, or meta-data, such as the title and author of an Asset. The definition of the Descriptor element is shown in Figure 28.

```
<element name="Descriptor" type="rdl:DescriptorType"/>
<complexType name="DescriptorType">
   <complexContent>
        <extension base="rdl:DIDBaseType">
             <sequence>
                  <element name="Annotation" type="rdl:AnnotationType"
                                      minOccurs="0" maxOccurs="unbounded"/>
                  <element name="Condition" type="rdl:ConditionType"
                                      minOccurs="0" maxOccurs="unbounded"/>
                  <element name="Descriptor" type="rdl:DescriptorType"
                                      minOccurs="0" maxOccurs="unbounded"/>
                  <element name="Statement" type="rdl:StatementType"
                                      minOccurs="0" maxOccurs="unbounded"/>
             </sequence>
             <attributeGroup ref="rdl:ID_ATTRS"/>
             <attributeGroup ref="rdl:IDREF_ATTRS"/>
             <attributeGroup ref="rdl:VERSION_ATTRS"/>
             <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
   </complexContent>
 </complexType>
```

IEC

Figure 28 – Descriptor element definition

8.19 Condition element

A Condition element represents a Condition. It denotes the parent element as being conditional on a set of predicate tests. The require attribute lists the set of predicates that are required to become true, and the except attribute lists the set of predicates that are required to become false in order for the condition to be satisfied. Each predicate is identified by the value of the select_id attribute in a Selection element.

A set of Condition elements defines a Boolean combination of predicate tests. Multiple tests within a Condition are combined as a conjunction (an AND relationship). Multiple Condition elements within a given parent are combined as a disjunction (an OR relationship). The definition of the Condition element is shown in Figure 29.

```
<element name="Condition" type="rdl:ConditionType"/>
<complexType name="ConditionType">
  <complexContent>
    <extension base="rdl:DIDBaseType">
      <attribute name="require" type="IDREFS"/>
      <attribute name="except" type="IDREFS"/>
    </extension>
  </complexContent>
</complexType>
```

IEC

Figure 29 – Condition element definition

8.20 Choice element

A Choice element represents a Choice. It encapsulates a set of related Selections that can affect the configuration of an Item and/or Layout. The optional minSelections and maxSelections attributes specify the number of Selections that are required to be made for a Choice to be validly resolved. For example, if minSelections and maxSelections are omitted, then the Choice is multiple, meaning that any number of Selections may be made, including zero. If the minSelections and maxSelections attributes are both set to ‘1’, then there is only one Choice, that is, exactly one Selection is required to be chosen. The definition of the Choice element is shown in Figure 30.

```
<element name="Choice" type="rdl:ChoiceType"/>
<complexType name="ChoiceType">
    <complexContent>
        <extension base="rdl:DIDBaseType">
            <sequence>
                <element name="Annotation" type="rdl:AnnotationType"
                                minOccurs="0" maxOccurs="unbounded"/>
                <element name="Selection" type="rdl:SelectionType"
                                maxOccurs="unbounded"/>
            </sequence>
            <attribute name="minSelections" type="nonNegativeInteger"/>
            <attribute name="maxSelections" type="positiveInteger"/>
            <attribute name="default" type="IDREFS" use="required"/>
            <attributeGroup ref="rdl:ID_ATTRS"/>
            <attributeGroup ref="rdl:IDREF_ATTRS"/>
            <attributeGroup ref="rdl:VERSION_ATTRS"/>
            <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
    </complexContent>
</complexType>
```

IEC

Figure 30 – Choice element definition

8.21 Selection element

A Selection element represents a Selection. It defines a specific decision about a particular Choice. The select_id attribute value identifies the predicate embodied by the Selection, and relates it to one or more Conditions somewhere within an Item. At configuration time, if the Selection is chosen, its predicate becomes true; if it is rejected, its predicate becomes false; if it is left unresolved, its predicate is left undecided.

A Selection may have any attribute from other namespaces. Such attributes provide an additional representation of a Descriptor with descriptive information about the Selection by means of an attribute. The definition of the Selection element is shown in Figure 31.

```
<element name="Selection" type="rdl:SelectionType"/>
<complexType name="SelectionType">
    <complexContent>
        <extension base="rdl:DIDBaseType">
            <sequence>
                <element name="Annotation" type="rdl:AnnotationType"
                                minOccurs="0" maxOccurs="unbounded"/>
                <element name="DCCondition" type="rdl:DCConditionType"
                                minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attributeGroup ref="rdl:ID_ATTRS"/>
            <attributeGroup ref="rdl:VERSION_ATTRS"/>
            <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
    </complexContent>
</complexType>
```

IEC

Figure 31 – Selection element definition

8.22 Statement element

A Statement defines a textual value that contains information, but not an Asset.

A Statement may contain any data format, including plain text and various machine-readable formats such as well-formed XML. The data type of the Statement is identified by the mimeType attribute, which is a concatenation of MIME media-type, sub-type and parameters, as specified in IETF RFC 2045 (e.g. ‘text/xml’).

The mimeType attribute identifies the data type of the Statement before any content-encodings specified in the contentEncoding attribute are applied to the statement.

The MIME media-type may be modified by the presence of the contentEncoding attribute, which specifies the content-encoding as specified in IETF RFC 2616. When present, the value of the contentEncoding attribute indicates what additional content-encodings have been applied to the Statement, and thus what decoding mechanisms shall be applied to obtain the MIME media-type identified by the mimeType attribute.

Content-encoding is primarily used for allowing a document to be compressed without losing the identity of its underlying MIME media-type. If multiple content-encodings are applied to the Statement, each content-encoding shall be listed in the value of the contentEncoding attribute in a space-delimited list in the order in which they were applied. The definition of the Statement element is shown in Figure 32.

```
<element name="Statement" type="rdl:StatementType"/>
<complexType name="StatementType" mixed="true">
 <complexContent mixed="true">
    <extension base="rdl:DIDBaseType">
       <sequence>
          <any namespace="##any" processContents="lax" minOccurs="0"/>
       </sequence>
       <attribute name="mimeType" type="string" use="required"/>
       <attribute name="ref" type="anyURI"/>
       <attribute name="encoding" type="string"/>
       <attribute name="contentEncoding" type="NMTOKENS"/>
       <anyAttribute namespace="##other" processContents="lax"/>
    </extension>
 </complexContent>
</complexType>
```

IEC

Figure 32 – Statement element definition

8.23 DCCondition element

A DCCondition element represents a DCCondition. A DCCondition describes conditions for selecting and filtering Digital Items to present an adapted application. For minimizing the number of letters and the depth of the XML tree of the condition description, it uses the Reverse Polish Notation (RPN), which can be easily implemented using stack-based functions to express complex conditions based on mathematical expressions such as Boolean expressions, comparison expressions or arithmetic expressions. A DCCondition element can contain StackEntry elements to describe conditional expressions for selecting and filtering Digital Items as the RPN form. The definition of the DCCondition element is shown in Figure 33.

```
<element name="DCCondition" type="rdl:DCConditionType"/>
<complexType name="DCConditionType">
 <complexContent>
    <extension base="rdl:DIDBaseType">
       <sequence>
          <element name="StackEntry" type="rdl:StackEntryType"
                   minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
    </extension>
 </complexContent>
</complexType>
```

IEC

Figure 33 – DCCondition element definition

Annex A
(normative)

RDL schema

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:rdl="urn:saf:2013:01-RDL-NS"
targetNamespace="urn:saf:2013:01-RDL-NS" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0">

<!-- **************************************************************************************
       DID: Digital Item Declaration
********************************************************************************************
-->
<complexType name="DIDBaseType" abstract="true"/>

<!-- **************************************************************************************
    Definition of Attribute Groups
********************************************************************************************
-->
<attributeGroup name="ID_ATTRS">
       <attribute name="id" type="ID" use="required"/>
</attributeGroup>
<attributeGroup name="IDREF_ATTRS">
       <attribute name="idRef" type="IDREF" use="optional"/>
</attributeGroup>
<attributeGroup name="VERSION_ATTRS">
       <attribute name="version" type="string" use="optional"/>
</attributeGroup>
<attributeGroup name="LAYOUT_ATTRS">
       <attribute name="horizontalSize" type="positiveInteger" use="required"/>
       <attribute name="verticalSize" type="positiveInteger" use="required"/>
       <attribute name="subLayoutOrder" type="IDREFS" use="optional"/>
</attributeGroup>

<!-- **************************************************************************************
    Definition of Elements
********************************************************************************************
-->

<!--================================================================
Definition of RDL
================================================================-->
<element name="RDL" type="rdl:RDLType"/>
<complexType name="RDLType">
       <sequence>
              <element name="RDLInfo" type="rdl:RDLInfoType" minOccurs="0"
maxOccurs="unbounded"/>
              <element name="Declarations" type="rdl:DeclarationsType" minOccurs="0"/>
              <element name="Package" type="rdl:PackageType" minOccurs="0"/>
              <element name="LayoutContainer" type="rdl:LayoutContainerType" minOccurs="0"/>
              <element name="SceneContainer" type="rdl:SceneContainerType" minOccurs="0"/>
       </sequence>
       <attribute name="RDLDocumentId" type="anyURI"/>
       <anyAttribute namespace="##other" processContents="lax"/>
</complexType>

<!--================================================================
Definition of RDLPackage
================================================================-->
<element name="RDLPackage" type="rdl:RDLPackageType"/>
```

```
<complexType name="RDLPackageType">
      <sequence>
            <element name="RDLInfo" type="rdl:RDLInfoType" minOccurs="0"
maxOccurs="unbounded"/>
            <element name="Declarations" type="rdl:DeclarationsType" minOccurs="0"/>
            <element name="Package" type="rdl:PackageType"/>
            <element name="LayoutContainer" type="rdl:LayoutContainerType"/>
            <element name="SceneContainer" type="rdl:SceneContainerType"/>
      </sequence>
      <attribute name="RDLPackageId" type="anyURI"/>
      <anyAttribute namespace="##other" processContents="lax"/>
</complexType>

<!--============================================================
Definition of RDLInfo
============================================================-->
<element name="RDLInfo" type="rdl:RDLInfoType"/>
<complexType name="RDLInfoType">
      <sequence>
            <any namespace="##any" processContents="lax"/>
      </sequence>
</complexType>

<!--============================================================
Definition of Declarations
============================================================-->
<element name="Declaration" type="rdl:DeclarationType"/>
<complexType name="DeclarationsType">
      <choice maxOccurs="unbounded">
            <element name="Item" type="rdl:ItemType"/>
            <element name="Layout" type="rdl:LayoutType"/>
            <element name="Descriptor" type="rdl:DescriptorType"/>
            <element name="DCCondition" type="rdl:DCConditionType"/>
            <element name="Group" type="rdl:GroupType"/>
            <element name="Choice" type="rdl:ChoiceType"/>
      </choice>
</complexType>

<!--============================================================
Definition of Package
============================================================-->
<element name="Package" type="rdl:PackageType"/>
<complexType name="PackageType">
      <complexContent>
            <extension base="rdl:DIDBaseType">
                  <sequence>
                        <element name="Annotation" type="rdl:AnnotationType" minOccurs="0"
maxOccurs="unbounded"/>
                        <element name="Item" type="rdl:ItemType" minOccurs="0"
maxOccurs="unbounded"/>
                  </sequence>
                  <attributeGroup ref="rdl:ID_ATTRS"/>
                  <attributeGroup ref="rdl:VERSION_ATTRS"/>
                  <anyAttribute namespace="##other" processContents="lax"/>
            </extension>
      </complexContent>
</complexType>

<!--============================================================
Definition of LayoutContainer
============================================================-->
<element name="LayoutContainer" type="rdl:LayoutContainerType"/>
```