## REVISION SUMMARY

<table>
<thead>
<tr>
<th>Version</th>
<th>Revision</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next &lt;Draft&gt;</td>
<td>Initial Draft Release</td>
<td>30 November 2010</td>
</tr>
<tr>
<td>Candidate Release</td>
<td>Initial Release</td>
<td>27 December 2011</td>
</tr>
<tr>
<td>4.0</td>
<td>ICWG Approved Release</td>
<td>28 February 2012</td>
</tr>
<tr>
<td>4.0.1</td>
<td>Incorporated transition to JTNC and applied SCA 4.0 Errata Sheet v1.0</td>
<td>01 October 2012</td>
</tr>
<tr>
<td>4.1&lt;DRAFT&gt;</td>
<td>Incorporating Naming Proposal Changes, Backwards Compatibility Changes,</td>
<td>31 December 2014</td>
</tr>
<tr>
<td></td>
<td>ScalableManagers, Device Registration</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>New :Process Collocation and Core Affinity Deployment Enhancement,</td>
<td>20 August 2015</td>
</tr>
<tr>
<td></td>
<td>Changes: Domain Late Registration, Allocation Properties, Domain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Component Type Uniformity, Deployment Data, DomainManager and DeviceManager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instance Level Property Value, Clarification and Consistency edits</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ICWG Approved</strong></td>
<td></td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

D-1.1 Scope...........................................................................................................11
D-1.2 Conformance ...............................................................................................11
D-1.3 Conventions .................................................................................................11
D-1.4 Normative References ..................................................................................11
D-1.5 Informative References .................................................................................11
D-1.6 Software Package Descriptor .......................................................................11
  D-1.6.1 Software Package ....................................................................................12
    D-1.6.1.1 title ..................................................................................................13
    D-1.6.1.2 author .............................................................................................13
    D-1.6.1.3 description ......................................................................................14
    D-1.6.1.4 propertyfile ....................................................................................14
      D-1.6.1.4.1 localfile ...................................................................................15
    D-1.6.1.5 descriptor .......................................................................................15
    D-1.6.1.6 implementation ..............................................................................15
      D-1.6.1.6.1 description ..............................................................................17
      D-1.6.1.6.2 propertyfile ..............................................................................17
      D-1.6.1.6.3 code ........................................................................................17
      D-1.6.1.6.4 compiler ...............................................................................19
      D-1.6.1.6.5 programminglanguage ..........................................................19
      D-1.6.1.6.6 humanlanguage ....................................................................19
    D-1.6.1.7 runtime .......................................................................................19
    D-1.6.1.8 os ..................................................................................................20
    D-1.6.1.9 processor ......................................................................................20
    D-1.6.1.10 dependency ...............................................................................20
      D-1.6.1.10.1 softpkgref ...........................................................................21
      D-1.6.1.10.2 propertyref ..........................................................................22
    D-1.6.1.11 usesdevice ...................................................................................22
      D-1.6.1.11.1 propertyref ..........................................................................23
    D-1.6.1.12 usesdevice ...................................................................................23
D-1.7 Device Package Descriptor .........................................................................24
D-1.7.1 Device Package ..............................................................................................................24
  D-1.7.1.1 title ..........................................................................................................................25
  D-1.7.1.2 author ......................................................................................................................25
  D-1.7.1.3 description ..............................................................................................................25
  D-1.7.1.4 hwdeviceregistration ............................................................................................25
    D-1.7.1.4.1 propertyfile ........................................................................................................26
    D-1.7.1.4.2 description ........................................................................................................27
    D-1.7.1.4.3 manufacturer .....................................................................................................27
    D-1.7.1.4.4 modelnumber ...................................................................................................27
    D-1.7.1.4.5 deviceclass ........................................................................................................27
    D-1.7.1.4.6 childhwdevice ..................................................................................................27
      D-1.7.1.4.6.1 hwdeviceregistration ..................................................................................28
      D-1.7.1.4.6.2 devicepkgref .............................................................................................28
D-1.8 Properties Descriptor ........................................................................................................29
  D-1.8.1 properties ..................................................................................................................29
    D-1.8.1.1 simple ....................................................................................................................30
      D-1.8.1.1.1 description .......................................................................................................31
      D-1.8.1.1.2 value ...............................................................................................................32
      D-1.8.1.1.3 units ................................................................................................................32
      D-1.8.1.1.4 range ................................................................................................................32
      D-1.8.1.1.5 enumerations ..................................................................................................32
      D-1.8.1.1.6 kind ................................................................................................................33
      D-1.8.1.1.7 action ..............................................................................................................34
    D-1.8.1.2 simplesequence .....................................................................................................34
    D-1.8.1.3 test .........................................................................................................................35
      D-1.8.1.3.1 inputValue .......................................................................................................36
      D-1.8.1.3.2 resultvalue ......................................................................................................36
    D-1.8.1.4 struct .....................................................................................................................37
      D-1.8.1.4.1 configurationkind ............................................................................................38
    D-1.8.1.5 structsequence .....................................................................................................38
D-1.9 Software Component Descriptor .......................................................................................40
**D-1.9.1** softwarecomponent .................................................................................40  
**D-1.9.1.1** componentrepid ..................................................................................41  
**D-1.9.1.2** componenttype ....................................................................................41  
**D-1.9.1.3** componentfeatures ..............................................................................41  
  **D-1.9.1.3.1** supportsinterface ............................................................................42  
  **D-1.9.1.3.2** ports ...............................................................................................42  
**D-1.9.1.4** interfaces ..............................................................................................43  
**D-1.9.1.5** propertyfile ..........................................................................................44  

**D-1.10 Software Assembly Descriptor** ..................................................................45  

**D-1.10.1** softwareassembly ..................................................................................45  
**D-1.10.1.1** description ..........................................................................................46  
**D-1.10.1.2** componentfiles ..................................................................................47  
  **D-1.10.1.2.1** componentfile ...............................................................................47  
**D-1.10.1.3** partitioning ..........................................................................................47  
  **D-1.10.1.3.1** componentplacement .....................................................................48  
    **D-1.10.1.3.1.1** componentfileref ......................................................................48  
    **D-1.10.1.3.1.2** componentinstantiation ..............................................................48  
      **D-1.10.1.3.1.2.1** componentproperties .............................................................49  
    **D-1.10.1.3.1.2.2** coreaffinity .............................................................................50  
    **D-1.10.1.3.1.2.3** deploymentdependencies .........................................................50  
  **D-1.10.1.3.2** hostcollocation ..............................................................................52  
    **D-1.10.1.3.2.1** componentplacement ...............................................................52  
  **D-1.10.1.3.3** assemblyplacement .......................................................................53  
    **D-1.10.1.3.3.1** assemblyinstantiation .................................................................53  
      **D-1.10.1.3.3.1.1** componentproperties .............................................................54  
    **D-1.10.1.3.3.1.2** deviceassignments .................................................................55  
    **D-1.10.1.3.3.1.3** deploymentdependencies .........................................................55  
    **D-1.10.1.3.3.1.4** executionaffinityassignments .....................................................55  
**D-1.10.1.4** deploymentdependencies ..................................................................56  
**D-1.10.1.5** assemblycontroller ..........................................................................57  
**D-1.10.1.6** connections .......................................................................................58
D-1.10.1.6.1 connectinterface

D-1.11 Device Configuration Descriptor
D-1.11.1 deviceconfiguration
D-1.11.1.1 description
D-1.11.1.2 devicemanagersofpkg
D-1.11.1.3 componentfiles
D-1.11.1.4 partitioning
D-1.11.1.4.1 componentplacement
D-1.11.1.4.1.1 componentfileref
D-1.11.1.4.1.2 deployondevice
D-1.11.1.4.1.3 compositepartofdevice
D-1.11.1.4.1.4 devicepkgfile
D-1.11.1.4.1.4.1 localfile
D-1.11.1.4.1.5 componentinstantiation
D-1.11.1.4.1.5.1 componentproperties
D-1.11.1.4.1.5.2 componentfactoryref
D-1.11.1.4.1.5.2.1 componentfactoryproperties
D-1.11.1.4.1.5.3 coreaffinity
D-1.11.2 Domain Manager Configuration Descriptor ........................................77
  D-1.12.1 domainmanagerconfiguration ..........................................................77
  D-1.12.1.1 description .................................................................................78
  D-1.12.1.2 domainmanagersoftpkg .................................................................78
  D-1.12.1.3 deploymentlayout .....................................................................79
  D-1.12.1.4 services .....................................................................................79
  D-1.12.1.4.1 service ....................................................................................79
D-1.13 Platform Deployment Descriptor ........................................................81
  D-1.13.1 deploymentplatform .....................................................................81
  D-1.13.1.1 description ................................................................................81
  D-1.13.1.2 channel ......................................................................................82
   D-1.13.1.2.1 devicelist ..............................................................................82
    D-1.13.1.2.1.1 deviceref ........................................................................82
   D-1.13.1.2.2 servicelist ............................................................................83
    D-1.13.1.2.2.1 serviceref .......................................................................83
D-1.14 Application Deployment Descriptor .......................................................84
  D-1.14.1 deploymentprecedence ................................................................84
  D-1.14.1.1 description ...............................................................................84
  D-1.14.1.2 deploymentoption ....................................................................84
   D-1.14.1.2.1 description ..........................................................................85
    D-1.14.1.2.2 channelref ........................................................................85
D-1.15 Attachments .........................................................................................86
D-1.16 Document Type definition (DTD) files .....................................................87
  D-1.16.1 softpkg .........................................................................................87
  D-1.16.2 devicepkg ....................................................................................90
  D-1.16.3 properties ...................................................................................92
  D-1.16.4 softwarecomponent ...................................................................95
  D-1.16.5 softwareassembly ....................................................................97
D-1.16.6 deviceconfiguration.................................................................102
D-1.16.7 domainmanagerconfiguration ..............................................107
D-1.16.8 platformdeployment...............................................................109
D-1.16.9 applicationdeployment..........................................................110
LIST OF FIGURES

Figure 1: softpkg Element Relationships ................................................................. 12
Figure 2: author Element Relationships ................................................................. 14
Figure 3: implementation Element Relationships ...................................................... 16
Figure 4: code Element Relationships ................................................................... 18
Figure 5: dependency Element Relationships .......................................................... 21
Figure 6: softpkgref Element Relationships ............................................................. 22
Figure 7: devicepkg Element Relationships ............................................................. 24
Figure 8: hwdeviceregistration Element Relationships ............................................. 26
Figure 9: childhwdevice Element Relationships ....................................................... 28
Figure 10: properties Element Relationships ........................................................... 29
Figure 11: simple Element Relationships ................................................................ 31
Figure 12: simplesequence Element Relationships ............................................... 35
Figure 13: test Element Relationships ..................................................................... 36
Figure 14: struct Element Relationships .................................................................. 37
Figure 15: structsequence Element Relationships ................................................. 39
Figure 16: softwarecomponent Element Relationships .......................................... 40
Figure 17: componentfeatures Element Relationships .......................................... 41
Figure 18: ports Element Relationships .................................................................. 43
Figure 19: softwareassembly Element Relationships .............................................. 46
Figure 20: partitioning Element Relationships ........................................................ 47
Figure 21: componentplacement Element Relationships ......................................... 48
Figure 22: componentinstantiation Element Relationships ..................................... 49
Figure 23: componentfactoryref Element Relationships ......................................... 50
Figure 24: componentfactoryproperties Element Relationships ............................ 51
Figure 25: assemblyplacement Element Relationships .......................................... 53
Figure 26: assemblyinstantiation Element Relationships ....................................... 54
Figure 27: componentproperties Element Relationships ........................................ 55
Figure 28: deploymentdependencies Element Relationships .................................. 57
Figure 29: assemblycontroller Element Relationships .......................................... 57
Figure 30: connectinterface Element Relationships .......................................................... 58
Figure 31: usesport Element Relationships ........................................................................ 59
Figure 32: providesport Element Relationships ................................................................. 63
Figure 33: componentsupportedinterface Element Relationships ..................................... 64
Figure 34: externalports Element Relationships ................................................................. 65
Figure 35: deviceconfiguration Element Relationships ...................................................... 68
Figure 36: componentplacement Element Relationships .................................................. 70
Figure 37: componentinstantiation Element Relationships .............................................. 72
Figure 38: componentproperties Element Relationships ................................................. 73
Figure 39: componentfactoryref Element Relationships .................................................. 74
Figure 40: componentfactoryproperties Element Relationships .................................... 75
Figure 41: domainmanagerconfiguration Element Relationships .................................... 77
Figure 42: services Element Relationships ....................................................................... 79
Figure 43: service Element Relationships ........................................................................ 80
Figure 44: deploymentplatform Element Relationships .................................................. 81
Figure 45: channel Element Relationships ..................................................................... 82
Figure 46: deploymentprecedence Element Relationships .............................................. 84
Figure 47: deploymentoption Element Relationships .................................................... 85
APPENDIX D-1 PLATFORM SPECIFIC MODEL (PSM) - DOCUMENT TYPE DEFINITION (DTD) FILES

D-1.1 SCOPE

This appendix defines the SCA Domain Profiles using XML Document Type Definition (DTD) files.

SCA501 DTD files are installed in the domain and shall have ".dtd" as their filename extension.

SCA502 All XML files shall have as the first two lines as an XML declaration (<?xml) and a document type declaration (DOCTYPE). The XML declaration specifies the XML version and whether the document is standalone. The document type declaration specifies the DTD for the document. Example declarations are as follows:

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE softwareassembly SYSTEM "softwareassembly.4.1.dtd”>
```

D-1.2 CONFORMANCE

See SCA Appendix D.

D-1.3 CONVENTIONS

N/A.

D-1.4 NORMATIVE REFERENCES

The following documents contain provisions or requirements which by reference constitute requirements of this appendix.


D-1.5 INFORMATIVE REFERENCES

N/A.

D-1.6 SOFTWARE PACKAGE DESCRIPTOR

The Software Package Descriptor (SPD) is used at deployment time to load a component and its various implementations. The information contained in the SPD will provide the basis for the domain management function to manage the component within the SCA architecture.

The SPD may contain various implementations of any given component. Within the specification of an SPD several other files are referenced including a component level propertyfile and a Software Component Descriptor (SCD) file. Within any given implementation there may be additional propertyfiles.
SCA503 A Software Package Descriptor file shall have a ".spd.xml" extension.

**D-1.6.1 Software Package**

The `softpkg` element (Figure 1) indicates an SPD definition. The `softpkg id` uniquely identifies the package. The `softpkg id` attribute definition guarantees uniqueness within an XML document, however an implementation specific approach must be utilized to maintain uniqueness within a Domain Profile. The `name` attribute is a user-friendly label for the `softpkg` element. The `type` attribute indicates whether or not the component implementation is SCA compliant. All files referenced by a software package are located in the same directory as the SPD file or a directory that is relative to the directory where the SPD file is located.

A software package requires at least one implementation. An implementation is a monolithic loadable/executable artifact. A single monolithic loadable/executable artifact is for a General Purpose Processor, Digital Signal Processor or Field Gate Array Processor.

![Figure 1: softpkg Element Relationships](image)
The set of properties to be used for a software package come from the union of these properties sources using the following precedence order:

1. SPD Implementation Properties
2. SPD level properties
3. SCD properties

Any duplicate properties having the same id are ignored. Duplicated properties must be the same property type, only the value can be over-ridden. The implementation properties are only used for the initial configuration and creation of a component by an ApplicationFactoryComponent and cannot be referenced by a Software Assembly Descriptor (SAD) componentinstantiation, componentproperties or componentfactoryproperties element.

<!ELEMENT softpkg ( title?, author+, description?, propertyfile?, descriptor?, implementation+, usesdevice*)>
<!ATTLIST softpkg id ID #REQUIRED
name CDATA #REQUIRED
type (sca_compliant | sca_non_compliant) "sca_compliant"
version CDATA #IMPLIED >

D-1.6.1.1 title
The title element is used to indicate the title of the software component being installed in accordance with the softpkg element.

<!ELEMENT title (#PCDATA)>

D-1.6.1.2 author
The author element (see Figure 2) is used to indicate the name of the person, company, and web page of the developer producing the component being installed into the system.
Figure 2: author Element Relationships

```xml
<!ELEMENT author
    ( name*
        , company?
        , webpage?
    )>
<!ELEMENT name (#PCDATA)>  
<!ELEMENT company (#PCDATA)>  
<!ELEMENT webpage (#PCDATA)> 
```

D-1.6.1.3 description

The description element is used to describe any pertinent information about the software component being delivered to the system.

```xml
<!ELEMENT description (#PCDATA)> 
```

D-1.6.1.4 propertyfile

The propertyfile element indicates the local filename of the Property Descriptor (PRF) file associated with the software package. The intent of the propertyfile will be to provide the definition of properties elements common to all component implementations being deployed in accordance with the software package (softpkg). PRF files may also contain properties elements that are used in definition of command and control id value pairs used by the SCA PropertySet configure() and query() operations. The format of the properties element is described in the PRF (section D-1.8.1).
<!ELEMENT propertyfile ( localfile )>
<!ATTLIST propertyfile
type CDATA #IMPLIED>

D-1.6.1.4.1 localfile

The localfile element is used to reference a file in the same directory as the SPD file or a
directory that is relative to the directory where the SPD file is located. When the name attribute
is a simple name, the file exists in the same directory as the SPD file. A relative directory
indication in the name attribute begins with either "./" meaning parent directory or "/." meaning
current directory. Multiple "./" and directory names can follow the initial "./" in the name
attribute. All name attributes must have a simple name at the end of the file name.

<!ELEMENT localfile EMPTY>
<!ATTLIST localfile
name CDATA #REQUIRED>

D-1.6.1.5 descriptor

The descriptor element points to the local filename of the SCD file used to document the
interface information for the component being delivered to the system. In the case of an SCA
component, the SCD will contain information about three aspects of the component (the
component type, message ports, and interface definitions). The SCD file is optional (see section
D-1.9 on SCD file).

<!ELEMENT descriptor (localfile )>
<!ATTLIST descriptor
name CDATA #IMPLIED>

D-1.6.1.6 implementation

The implementation element (see Figure 3) contains descriptive information about the particular
implementation template for a software component contained in the softpkg element. The
implementation element is intended to allow multiple component templates to be delivered to the
system in one software package. Each implementation element is intended to allow the same
component to support different types of processors, operating systems, etc. The implementation
element will also allow definition of implementation-dependent properties for use in
DeviceComponent or ApplicationComponent creation. The implementation element's id
attribute uniquely identifies a specific implementation of the component and an implementation
specific approach is required to maintain uniqueness within a Domain Profile. The compiler,
programminglanguage, humanlanguage, os, processor, and runtime elements are optional
dependency elements.
Figure 3: implementation Element Relationships
<!ELEMENT implementation ( description?, propertyfile?, code, compiler?, programminglanguage?, humanlanguage?, runtime?, ( os | processor | dependency )+, usesdevice*)>
<!ATTLIST implementation id ID #REQUIRED aepcompliance (aep_compliant | lw_aep_compliant | aep_non_compliant) "aep_compliant">

**D-1.6.1.6.1 description**

The `description` element is used to describe any pertinent information about the software component implementation that the software developer wishes to document within the software package profile.

<!ELEMENT description (#PCDATA)>

**D-1.6.1.6.2 propertyfile**

The `propertyfile` element is used to indicate the local filename of the PRF file associated with this component package described by the `implementation` element. Although the SCA does not restrict the specific use of the PRF file based on context, it is intended within the `implementation` element to provide component implementation specific `properties` elements for use in command and control id value pair settings to the `PropertySet configure()` and `query()` operations.  See section D-1.8.1 on the description of the `properties` element format in the PRF.

<!ELEMENT propertyfile ( localfile )>
<!ATTLIST propertyfile type CDATA #IMPLIED>

**D-1.6.1.6.3 code**

The `code` element (see Figure 4) is used to indicate the local filename of the code that is described by the `softpkg` element, for a specific implementation of the software component. The optional parameters `stacksize` and `priority` are used by the `ExecutableInterface::execute` operation. Data types for the values of the `stacksize` and `priority` options are unsigned long. The `stacksize` element provides the means to specify a stack size for the process/thread being created. The `priority` element provides the means to specify the scheduling priority for the process/thread
being created. The optional parameter *entrypoint* is used by the *ExecutableInterface::execute* operation. The *entrypoint* element allows the component being delivered to specify the name of its entry point. The data type for the value of the *entrypoint* option is a string. The *type* attribute of the *code* element indicates the type of file being delivered to the system. The valid values for the *type* attribute are: "Executable", "KernelModule", "SharedLibrary", and "Driver."

The meaning of the code *type* attribute:

1. Executable means to use *LoadableInterface::load* and *ExecutableInterface::execute* operations. This is a "main" process.
2. Driver and Kernel Module means load only.
4. Without a code *entrypoint* element means load only.
5. With a code *entrypoint* element means load and *ExecutableInterface::execute*.

![Figure 4: code Element Relationships](image-url)
<!ELEMENT code ( localfile, entrypoint?, stacksize?, priority?)>
<!ATTLIST code type CDATA #IMPLIED>
<!ELEMENT entrypoint (#PCDATA)>
<!ELEMENT stacksize (#PCDATA)>
<!ELEMENT priority (#PCDATA)>

D-1.6.1.6.4 compiler

The `compiler` element is used to indicate the compiler used to build the software component being described by the `softpkg` element. The required `name` attribute will specify the name of the compiler used, and the `version` attribute will contain the compiler version.

<!ELEMENT compiler EMPTY>
<!ATTLIST compiler
    name CDATA #REQUIRED
    version CDATA #IMPLIED>

D-1.6.1.6.5 programminglanguage

The `programminglanguage` element is used to indicate the type of programming language used to build the component implementation. The required `name` attribute will specify a language such as "c", "c++", or "java".

<!ELEMENT programminglanguage EMPTY>
<!ATTLIST programminglanguage
    name CDATA #REQUIRED
    version CDATA #IMPLIED>

D-1.6.1.6.6 humanlanguage

The `humanlanguage` element is used to indicate the human language for which the software component was developed.

<!ELEMENT humanlanguage EMPTY>
<!ATTLIST humanlanguage
    name CDATA #REQUIRED>

D-1.6.1.6.7 runtime

The `runtime` element specifies a runtime required by a component implementation. An example of the runtime is a Java VM.
The os element is used to indicate the operating system on which the software component is capable of operating. The required name attribute will indicate the name of the operating system and the version attribute will contain the operating system. The os attributes will be defined in a property file as an allocation property of string type and with names os_name and os_version and an action element value other than external. The os element is automatically interpreted as a dependency and compared against allocation properties with names of os_name and os_version. The os_name allocation property is defined in Attachment 1 of this appendix.

The processor element is used to indicate the processor and/or processor family on which this software component will operate. The processor name attribute will be defined in a property file as an allocation property of string type and with a name of processor_name and an action element value other than external. The processor element is automatically interpreted as a dependency and compared against an allocation property with a name of processor_name. The processor_name allocation property is defined in Attachment 1 of this appendix.

The dependency element (see Figure 5) is used to indicate the dependent relationships between the components being delivered and other components and devices, in an SCA compliant system. The softpkgref element is used to specify a software package file that must be resident within the system for the component, described by this softpkg element, to load without errors. The propertyref will reference a specific allocation property, using a unique identifier, and provide the value that will be used by a DeviceComponent capacity model.

A DomainManagerComponent and DeviceManagerComponent will use these dependency definitions to assure that the components and devices which are necessary for proper operation of
the implementation are present and available. The type attribute contains descriptive information indicating the type of dependency.

![Diagram of dependency element relationships]

**Figure 5: dependency Element Relationships**

```xml
<!ELEMENT dependency
  ( softpkgref | propertyref )>
<!ATTLIST dependency
type CDATA      #REQUIRED>
```

*D-1.6.1.6.10.1* `softpkgref`

The `softpkgref` element (see Figure 6) refers to a `softpkg` element contained in another SPD file and indicates a file-load dependency on that file. The other file is referenced by the `localfile` element. An optional `implref` element refers to a particular implementation unique identifier, within the SPD of the other file.
Figure 6: softpkgref Element Relationships

```xml
<!ELEMENT softpkgref
  ( localfile
   , implref?)>

<!ELEMENT implref EMPTY>
<!ATTLIST implref
  refid CDATA #REQUIRED>

D-1.6.1.6.10.2 propertyref

The propertyref element is used to indicate a unique refid attribute that references a simple allocation property, defined in the package, and a property value attribute used by the domain management function to perform the dependency check.

```xml
<!ELEMENT propertyref EMPTY>
<!ATTLIST propertyref
  refid CDATA #REQUIRED
  value CDATA #REQUIRED>

D-1.6.1.6.11 usesdevice

The usesdevice element describes any "uses" relationships this component has with a device in the system. The propertyref element references allocation properties, which indicate the DeviceComponent to be used, and/or the capacity needed from the DeviceComponent to be used.

```xml
<!ELEMENT usesdevice
  ( propertyref+ )>
<!ATTLIST usesdevice
  id ID #REQUIRED
  type CDATA #REQUIRED>
```
D-1.6.1.6.11.1 propertyref
See D-1.6.1.6.10.2 for a definition of the propertyref element.

D-1.6.1.7 usesdevice
See D-1.6.1.6.11 for a definition of the usesdevice element.
D-1.7 DEVICE PACKAGE DESCRIPTOR

The SCA Device Package Descriptor (DPD) is the part of a Device Profile that contains hardware device registration attributes, which are typically used by a Human Computer Interface application to display information about the DeviceComponent(s) resident in an SCA-compliant radio system. DPD information is intended to provide hardware configuration and revision information to a radio operator or to radio maintenance personnel. A DPD may be used to describe a single hardware element residing in a radio or it may be used to describe the complete hardware structure of a radio.

SCA504 A Device Package Descriptor File shall have a ".dpd.xml" extension.

D-1.7.1 Device Package

The devicepkg element (see Figure 7) is the root element of the DPD. The devicepkg id attribute uniquely identifies the package and an implementation specific approach is required to maintain uniqueness within a Domain Profile. The version attribute specifies the version of the devicepkg. The format of the version string is numerical major and minor version numbers separated by commas (e.g., "1,0,0,0"). The name attribute is a user-friendly label for the devicepkg.

![Figure 7: devicepkg Element Relationships](image-url)
<!ELEMENT devicepkg
    ( title?,
        author*,
        description?,
        hwdeviceregistration
    )>
<!ATTLIST devicepkg
    id ID #REQUIRED
    name CDATA #REQUIRED
    version CDATA #IMPLIED>

D-1.7.1.1 title
The title element is used to indicate the title of the hardware device being described by devicepkg.

<!ELEMENT title (#PCDATA)>

D-1.7.1.2 author
See D-1.6.1.2 for a description of the author element.

<!ELEMENT author
    ( name*,
        company?,
        webpage?
    )>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT webpage (#PCDATA)>

D-1.7.1.3 description
The description element is used to describe any pertinent information about the device implementation that the hardware developer wishes to document within the DPD.

<!ELEMENT description (#PCDATA)>

D-1.7.1.4 hwdeviceregistration
The hwdeviceregistration element (see Figure 8) provides device-specific information for a hardware device. The hwdeviceregistration id attribute uniquely identifies the device and an implementation specific approach is required to maintain uniqueness within a Domain Profile. The version attribute specifies the version of the hwdeviceregistration element. The format of the version string is a series of numerical major and minor version numbers separated by commas (e.g., "1,0,0,0"). The name attribute is a user-friendly label for the hardware device being registered. At a minimum, the hwdeviceregistration element must include a description, the manufacturer, the model number and the device’s hardware class(es).
Figure 8: `hwdeviceregistration` Element Relationships

```xml
<!ELEMENT hwdeviceregistration
  ( propertyfile?,
    description,
    manufacturer,
    modelnumber,
    deviceclass,
    childhwdevice*)>

<!ATTLIST hwdeviceregistration
  id ID #REQUIRED
  name CDATA #REQUIRED
  version CDATA #IMPLIED>
```

D-1.7.1.4.1 propertyfile

The `propertyfile` element is used to indicate the local filename of the property file associated with the `hwdeviceregistration` element. The format of a property file is described in the Properties Descriptor (PRF) (see section D-1.8).

The intent of the property file is to provide the definition of properties elements for the hardware device being deployed and described in the device package (`devicepkg`) or `hwdeviceregistration` element.
D-1.7.1.4.2 description
See D-1.7.1.3 for the definition of the description element.

D-1.7.1.4.3 manufacturer
The manufacturer element is used to convey the name of manufacturer of the device being installed.

D-1.7.1.4.4 modelnumber
The modelnumber element is used to indicate the manufacture's model number, for the device being installed.

D-1.7.1.4.5 deviceclass
The deviceclass element is used to identify one or more hardware classes that make up the device being installed.

D-1.7.1.4.6 childhwdevice
The childhwdevice element (see Figure 9) indicates additional device-specific information for hardware devices that make up the root or parent hardware device registration. An example of childhwdevice would be a radio's RF module that contains receiver and exciter functions. In this case, a DeviceComponent representing the RF module itself would be a parent device with its DPD, and the receiver and exciter are child devices to the module. The parent / child
relationship implies that when the RF module is removed from the system, the receiver and exciter devices will also be removed.

![Diagram of childhwdevice Element Relationships]

```xml
<!ELEMENT childhwdevice
 ( hwdeviceregistration | devicepkgref )>

D-1.7.1.4.6.1 hwdeviceregistration

The `hwdeviceregistration` element provides device-specific information for the child hardware device. See D-1.7.1.4 for definition of the `hwdeviceregistration` element.

D-1.7.1.4.6.2 devicepkgref

The `devicepkgref` element is used to indicate the local filename of a DPD file pointed to by a DPD (e.g., a `devicepkg` within a `devicepkg`).

```xml
<!ELEMENT devicepkgref
 ( localfile )>
<!ATTLIST devicepkgref
type CDATA #IMPLIED>
D-1.8 PROPERTIES_DESCRIPTOR

The Properties Descriptor (PRF) file details component and device attribute settings. For purposes of the SCA, PRF files will contain *simple*, *simplesequence*, *test*, *struct* or *structsequence* elements. These elements will be used to describe attributes of a component that will be used for dependency checking.

These elements will also be used to provide values used by the *PropertySet::configure*, *PropertySet::query*, and *PropertySet::runTest* operations.

SCA494 A Properties Descriptor shall have a ".prf.xml" extension.

**D-1.8.1 properties**

The *properties* element (see Figure 10) is used to describe property attributes that will be used in the *configure* and *query* operations of SCA components that realize the *PropertySet* interface and for definition of attributes used during dependency checking. The *properties* element can also be used by the *TestableInterface::runTest* operation to configure tests and provide test results.

![Diagram of properties Element Relationships](image)

*Figure 10: properties Element Relationships*
<!ELEMENT properties
   ( description?,
   (simple | simplesequence | test | struct | structsequence )+)
>

D-1.8.1.1 simple

The *simple* element (see Figure 11) supports the definition of properties which include unique *id*, *type*, *name* and *mode* attributes that will be used by the *PropertySet configure()* and *query()* operations for indication of component capabilities, or by the *TestableInterface runTest* operation. The *simple* element is specifically designed to support id-value pair definitions. A *simple* property *id* attribute corresponds to the id of the id-value pair. The *value* and *range* of a *simple* property correspond to the value of the id-value pair. The optional *enumerations* element allows for the definition of a label-to-value for a particular property. The *mode* attribute defines whether the *properties* element is "readonly", "writeonly" or "readwrite". The *id* attribute is an identifier for the *simple* property element. The *id* attribute for a *simple* property that is an allocation type requires an implementation specific approach to maintain uniqueness within a Domain Profile. The *id* attribute for all other *simple* property elements can be any valid XML ID type. The *mode* attribute is only meaningful when the type of the *kind* element is "configure". The *simple* element maps to the types for the CF Primitive types and CORBA module, defined in SCA Appendix C and the OMG CORBA Specification version 3.2 [1], based upon the type attribute.
<!ELEMENT simple
( description?, value?, units?, range?, enumerations?, kind*, action? )>
<!ATTLIST simple
id ID #REQUIRED
 type ( boolean | char | double | float | short | long | objref | octet | string | ulong | ushort ) #REQUIRED
name CDATA #IMPLIED
mode ( readonly | readwrite | writeonly ) "readwrite">

**D-1.8.1.1.1 description**

The *description* element is used to provide a description of the *properties* element that is being defined.

<!ELEMENT description (#PCDATA)>
D-1.8.1.1.2 value

The value element is used to provide a value for the properties element.

```xml
<!ELEMENT value (#PCDATA)>  
```

D-1.8.1.1.3 units

The units element describes the intended practical data representation to be used for the properties element.

```xml
<!ELEMENT units (#PCDATA)>  
```

D-1.8.1.1.4 range

The range element describes the specific min and max values that are legal for the simple element. The intent of the range element is to provide a means to perform range validation. This element is not used by ApplicationFactoryComponents or ApplicationManagerComponents.

```xml
<!ELEMENT range EMPTY>
<!ATTLIST range
  min CDATA #REQUIRED
  max CDATA #REQUIRED>  
```

D-1.8.1.1.5 enumerations

The enumerations element is used to specify one or more enumeration elements.

```xml
<!ELEMENT enumerations
  ( enumeration+ )>  
```

The enumeration element is used to associate a value attribute with a label attribute. Enumerations are legal for various integer type properties elements. Enumeration values are implied; if not specified by a developer, the initial implied value is 0 and subsequent values are incremented by 1.

Note: The advantage of the enumeration element over the sequence element is that the enumeration element provides a mechanism to associate a value of a property to a label. The sequence element does not allow association of values (only lists of sequences).
<!ELEMENT enumeration EMPTY>
<!ATTLIST enumeration
    label CDATA #REQUIRED
    value CDATA #IMPLIED>

D-1.8.1.1.6 kind

The kind element's kindtype attribute is used to specify the kind of property. The types of kindtype attributes are:

1. configure, which is used in the configure and query operations of the PropertySet interface. The ApplicationFactoryComponent will use the configure kind of properties to build the CF Properties input parameter to the configure operation that is invoked on the ApplicationControllerComponent during application creation. The DeviceManagerComponent will use the configure kind of properties to build the CF Properties input parameter to the configure operation that is invoked on components implementing the PropertySet interface, during device and service creation. The ApplicationFactoryComponent and DeviceManagerComponent will also use the configure kind of properties for ComponentFactory create options parameters. When the mode is readonly, only the query behavior is supported. When the mode is writeonly, only the configure behavior is supported. When the mode is readwrite, both configure and query are supported.

2. test, which is used in the runTest operation of the TestableInterface interface. The test kind of properties will be used as the testValues parameter to the runTest operation that is invoked on components that realize the TestableInterface interface.

3. allocation, which is used in the allocateCapacity and deallocateCapacity operations of the CapacityManagement interface. The ApplicationFactoryComponent uses a component's simple dependency properties with a kindtype of allocation to build the input capacities parameter to the allocateCapacity operation when the action element of those properties is external. The ApplicationFactoryComponent uses a component's simple dependency properties with a kindtype of allocation to perform the dependency check when the action element is not external. A DeviceManagerComponent processes allocation properties the same as an ApplicationFactoryComponent when deploying a platform components. The only distinction between the two approaches is that a DeviceComponent may have either simple or simplesequence properties for an allocation property with an action element that is not external. When a simplesequence property is used, the dependency check is successful when one value of the sequence passes the dependency check.

4. execparam, which is used in the execute operations of the ExecutableInterface interface. The ApplicationFactoryComponent and DeviceManagerComponent will use the execparam kind of properties to build the CF Properties input parameter to the execute operation that is invoked on ExecutableDeviceComponents during component creation. Only simple elements can be used as execparam types.
5. factoryparam, which is used in the `createComponent` operation of the `ComponentFactory` interface. The ApplicationFactoryComponent and DeviceManagerComponent will use the factoryparam type of properties to build the CF Properties input parameter to the `createComponent` operation.

A property can have multiple `kind` elements and the default `kindtype` is `configure`.

```
<!ELEMENT kind EMPTY>
<!ATTLIST kind
  kindtype ( allocation | configure | test | execparam | factoryparam ) "configure">
```

### D-1.8.1.1.7 action

The `action` element is used to define the type of comparison used to compare an SPD property value to a device property value, during the process of checking SPD dependencies. The `kindtype` attribute of the `action` element, will determine the type of comparison to be made (e.g., equal, not equal, greater than, etc.). The default value for `kindtype` is `external`.

In principle, the `action` element defines the operation executed during the comparison of the allocation property value, provided by an SPD `dependency` element, to the associated allocation property value of a DeviceComponent. The allocation property is on the left side of the action and the dependency value is on the right side of the action. This process allows for the allocation of appropriate objects within the system based on their attributes, as defined by their dependent relationships.

For example, if a DeviceComponent's properties file defines a DeviceKind allocation property whose `action` element is set to "equal", then at the time of dependency checking a valid DeviceKind property is checked for equality. If a software component implementation is dependent on a DeviceKind property with its value set to "NarrowBand", then the component's SPD dependency `propertyref` element will reference the `id` of the DeviceKind allocation property with a value of "NarrowBand". At the time of dependency checking, the ApplicationFactoryComponent and DeviceManagerComponent will check DeviceComponents whose `properties kind` element is set to "allocation" and property `id` is DeviceKind for equality against a "NarrowBand" value.

```
<!ELEMENT action EMPTY>
<!ATTLIST action
  type ( eq | ne | gt | lt | ge | le | external ) "external">
```

### D-1.8.1.2 simplesequence

The `simplesequence` element (see Figure 12) is used to specify a list of `properties` with the same characteristics (e.g., type, range, units, etc.). The `simplesequence` element definition is similar to the `simple` element definition except that it has a list of values instead of one value. The `simplesequence` element maps to the sequence types for the CF Primitive sequence types defined in SCA Appendix C, based upon the `type` attribute.
Figure 12: `simplesequence` Element Relationships

```xml
<!ELEMENT simplesequence
  ( description?, values?, units?, range?, kind*, action? )>

<!ATTLIST simplesequence
  id ID #REQUIRED
  type ( boolean | char | double | float | short | long | objref | octet | string | ulong | ushort ) #REQUIRED
  name CDATA #IMPLIED
  mode (readonly | readwrite | writeonly) "readwrite">

D-1.8.1.3 test

The `test` element (see Figure 13) is used to specify a list of test properties for executing the `runTest` operation in order to perform a component specific test. This element contains `inputvalue` and `resultvalue` elements and it has an `id` attribute for grouping test properties to a specific test. The `id` attribute will be represented by a numeric value. The optional `label` attribute allows a meaningful label to be associated with a `test` element. `Inputvalues` are used to configure
the test to be performed (e.g., frequency and RF power output level). When the test has completed, *resultvalues* contain the results of the testing (e.g., pass or a fault code/message)

![Diagram showing test element relationships](image)

**Figure 13: test Element Relationships**

```xml
<!ELEMENT test ( description , inputvalue? , resultvalue )>
<!ATTLIST test id CDATA #REQUIRED
label CDATA #IMPLIED>

D-1.8.1.3.1 inputvalue

The *inputvalue* element is used to provide test configuration properties. The *simple* properties it contains must have a *kindtype* value of test.

```xml
<!ELEMENT inputvalue ( simple+ )>
```

D-1.8.1.3.2 resultvalue

The *resultvalue* element is used to specify the desired results of the *runTest* operation. The *simple* properties it contains must have a *kindtype* value of test.
D-1.8.1.4 struct

The **struct** element (see Figure 14) is used to group properties with different characteristics (i.e., similar to a structure or record entry). Each item in the **struct** element can be a different **simple** type (e.g., short, long, etc.). The **struct** element corresponds to the CF **Properties** type where each **struct** item (ID, value) corresponds to a **properties** element list item. The **properties** element list size is based on the number of **struct** items.

![Diagram showing the relationships between struct elements](image-url)

**Figure 14:** struct Element Relationships
<!ELEMENT struct
 ( description?,
  simple+,
  configurationkind?)>

<!ATTLIST struct
 id ID #REQUIRED
 name CDATA #IMPLIED
 mode (readonly | readwrite | writeonly) "readwrite">

D-1.8.1.4.1 configurationkind

The configurationkind element's kindtype attribute is used to specify the kind of property. The kindtypes are:

1. configure, which is used in the configure and query operations of the PropertySet interface. The ApplicationFactoryComponent and DeviceManagerComponent will use the configure kind of properties to build the CF Properties input parameter to the configure operation that is invoked on SCA components that realize the PropertySet interface during application creation. When the mode is readonly, only the query behavior is supported. When the mode is writeonly, only the configure behavior is supported. When the mode is readwrite, both configure and query are supported.

2. factoryparam, which is used in the createComponent operations of the ComponentFactory interface. The ApplicationFactoryComponent and DeviceManagerComponent will use the factoryparam kind of properties to build the CF Properties input parameter to the createComponent() operation. A property can have multiple configurationkind elements and their default kindtype is configure.

3. typedef, which is used when this definition is to be used a type definition for a structsequence. A property of this type will not have an identity independent of the element that references it (i.e. it would not be possible to use the configure or query operations on this element).

<!ELEMENT configurationkind EMPTY>
<!ATTLIST configurationkind
 kindtype (configure | factoryparam | typedef) "configure">

D-1.8.1.5 structsequence

The structsequence element (see Figure 15) is used to specify a list of properties with the same struct characteristics. The structsequence element maps to a properties element having the CF Properties type. Each item in the CF Properties type will be the same struct definition as referenced by the structrefid attribute. Any values specified within the struct definition are ignored and values for the sequence are provided by the structvalue element.
Figure 15: structsequence Element Relationships

```xml
<!ELEMENT structsequence
  ( description?,
    structvalue*,
    configurationkind?)>

<!ATTLIST structsequence
  id ID #REQUIRED
  structrefid CDATA #REQUIRED
  name CDATA #IMPLIED
  mode (readonly | readwrite | writeonly) "readwrite">

<!ELEMENT structvalue
  ( simpleref+ )>

<!ELEMENT simpleref EMPTY>
<!ATTLIST simpleref
  refid CDATA #REQUIRED
  value CDATA #REQUIRED>
```
D-1.9 SOFTWARE COMPONENT DESCRIPTOR

The SCA components ManageableApplicationComponent, DeviceComponent, LoadableDeviceComponent, ExecutableDeviceComponent, BaseFactoryComponent, and ServiceComponents that are described by the Software Component Descriptor (SCD) are based on the SCA specification, and this section concentrates on defining the elements necessary for describing the ports and interfaces of these components.

SCA495 A Software Component Descriptor file shall have a ".scd.xml" extension.

D-1.9.1 softwarecomponent

The softwarecomponent element (see Figure 16) is the root element of the SCD file. For use within the SCA the sub-elements that are supported include:

1. componentrepid – is the repository id of the component,
2. componenttype – identifies the type of software component object,
3. componentfeatures – provides the supported message ports for the component,
4. interface – describes the component unique id and name for supported interfaces,
5. propertyfile – provides the link to the component's SCD level properties.

```xml
<!ELEMENT softwarecomponent (
    componentrepid,
    componenttype,
    componentfeatures,
    interfaces,
    propertyfile?)>
```

Figure 16: softwarecomponent Element Relationships
D-1.9.1.1 componentrepid

The componentrepid uniquely identifies the principal interface that the component is implementing. The componentrepid may be referred to by the componentfeatures element. The componentrepid is the IDL repository Id of the component’s principal interface.

```xml
<!ELEMENT componentrepid EMPTY>
<!ATTLIST componentrepid repid CDATA #REQUIRED>
```

D-1.9.1.2 componenttype

The componenttype describes properties of the component. For SCA components, the component types include APPLICATION_COMPONENT, DEVICE_COMPONENT, LOADABLE_DEVICE_COMPONENT, EXECUTABLE_DEVICE_COMPONENT, MANAGEABLE_SERVICE_COMPONENT, SERVICE_COMPONENT, DEVICE_MANAGER_COMPONENT, DOMAIN_MANAGER_COMPONENT, APPLICATION_MANAGER_COMPONENT, APPLICATION_FACTORY_COMPONENT, APPLICATION_COMPONENT_FACTORY_COMPONENT, and PLATFORM_COMPONENT_FACTORY_COMPONENT.

```xml
<!ELEMENT componenttype (#PCDATA)>
```

D-1.9.1.3 componentfeatures

The componentfeatures element (see Figure 17) is used to describe a component with respect to the interfaces that it inherits, either directly or indirectly, from its principal interface which are represented by its provides and uses ports. If a component extends any SCA interfaces, then all of the inherited interfaces are depicted as supportsinterface elements.

![Figure 17: componentfeatures Element Relationships](image_url)
D-1.9.1.3.1 supportsinterface

The supportsinterface element is used to identify an interface definition that the component supports. These are all of the interfaces inherited either directly or indirectly by the component's principal interface, which is defined by the componentrepid element. One can widen the component's interface to be a supportsinterface. The repid is used to refer to the interface element (see interfaces section D-1.9.1.4).

<!ELEMENT supportsinterface EMPTY>
<!ATTLIST supportsinterface
  repid   CDATA   #REQUIRED
  supportsname  CDATA   #REQUIRED>

D-1.9.1.3.2 ports

The ports element (see Figure 18) describes what interfaces a component provides and uses.

The provides elements represent interfaces that are not part of a component's interface but are independent interfaces known as facets (i.e. a provides port at the end of a path, like I/O Device or Modem Device). The uses element is—uses port at the start of a path (i.e. I/O Device) that is connected to a provides or supportsinterfaces interface. Any number of uses and provides elements can be given in any order. Each ports element has a name and references an interface by repid (see interfaces section D-1.9.1.4). The port names are used in the SAD to connect ports together. The maxconnections attribute allows the developer to specify how many simultaneous connections are allowed to be made using that port. A ports element also has an optional porttype element that allows for identification of port classification. Values for porttype include "data", "control", "responses", and "test". If a porttype is not given then "control" is assumed.
Figure 18: ports Element Relationships

<!ELEMENT ports
   ( provides | uses )*>

<!ELEMENT provides
   ( porttype* )>
<!ATTLIST provides
   repid   CDATA   #REQUIRED
   providesname  CDATA   #REQUIRED
   maxconnections  CDATA   #REQUIRED>

<!ELEMENT uses
   ( porttype* )>
<!ATTLIST uses
   repid   CDATA   #REQUIRED
   usesname   CDATA   #REQUIRED
   maxconnections  CDATA   #REQUIRED>

<!ELEMENT porttype EMPTY>
<!ATTLIST porttype
   type ( data | control | responses | test ) #REQUIRED>

D-1.9.1.4 interfaces

The interfaces element is made up of one to many interface elements.

<!ELEMENT interfaces
   ( interface+ )>

The interface element describes an interface that the component, either inherits directly or indirectly, provides or uses. The name attribute is the character-based, non-qualified name of the interface. The repid attribute is the unique IDL repository id [1] of the interface. The repid is
also used to reference an interface element elsewhere in the SCD, for example from the inheritsinterface element.

```xml
<!ELEMENT interface ( inheritsinterface*) >
<!ATTLIST interface repid CDATA #REQUIRED
name CDATA #REQUIRED>
<!ELEMENT inheritsinterface EMPTY>
<!ATTLIST inheritsinterface repid CDATA #REQUIRED>
```

**D-1.9.1.5 propertyfile**

The propertyfile element is used to indicate the local filename of the PRF file associated with the software component. The description of the propertyfile element can be found in section D-1.6.1.4. Within the SCD, the localfile sub-element of the propertyfile element is a relative pathname referencing a file in the same directory as the SCD or in a directory that is relative to the directory where the SCD file is located.

```xml
<!ELEMENT propertyfile ( localfile )>
<!ATTLIST propertyfile type CDATA #IMPLIED>
<!ELEMENT localfile EMPTY>
<!ATTLIST localfile name CDATA #REQUIRED>
```
**D-1.10 SOFTWARE ASSEMBLY DESCRIPTOR**

This section describes the XML elements of the Software Assembly Descriptor (SAD) XML file; the `softwareassembly` element (see Figure 19). The intent of the software assembly is to provide the means of describing the assembled functional application and the interconnection characteristics of the SCA components within that application. Created applications are assembled from a combination of one or more component instantiations and nested assembly instantiations that are interconnected with each other and to platform devices and services.

SCA496 A Software Assembly Descriptor file shall have a ".sad.xml" extension.

**D-1.10.1 softwareassembly**

The installation of an application into the system includes the installation of a main SAD file and one or more SPD and SAD files. The SAD file references component SPD files to obtain deployment information for these components, as well SAD files for nested applications. The `softwareassembly` element's `name` attribute uniquely identifies the assembly, requiring an implementation specific approach to maintain uniqueness within a Domain Profile. The `softwareassembly` element's `version` attribute is the version of the application. The `softwareassembly` element's `sca_version` attribute is the targeted SCA version (i.e., V4.1) of the application.

The SAD `deploymentdependencies` are merged with and overridden by lower-level `deploymentdependencies` defined within the `componentinstantiation` and `assemblyinstantiation` elements.
Figure 19: softwareassembly Element Relationships

```xml
<!ELEMENT softwareassembly
  ( description?,
    componentfiles,
    partitioning,
    deploymentdependencies?,
    assemblycontroller,
    connections?,
    externalports?,
    deploymentprefs?)>

<!ATTLIST softwareassembly
  name ID #REQUIRED
  sca_version CDATA "V4.1"
  version CDATA #IMPLIED>

D-1.10.1.1 description

The description element of the component assembly may be used to describe any information the developer would like to provide about the assembly.

```xml
description (#PCDATA)```
**D-1.10.1.2 componentfiles**

The `componentfiles` element is used to indicate that an assembly is made up of 1…n component files and / or subassemblies. The `componentfile` element contains a reference to a local file, which can be an SPD or SAD file.

```xml
<!ELEMENT componentfiles ( componentfile+ )>
```

**D-1.10.1.2.1 componentfile**

The `componentfile` element is a reference to a local file. See section D-1.6.1.4.1 for the definition of the `localfile` element. The `type` attribute is "Software Package Descriptor" or "Software Assembly Descriptor".

```xml
<!ELEMENT componentfile ( localfile )>
<!ATTLIST componentfile
  id ID #REQUIRED
  type CDATA #IMPLIED>
```

**D-1.10.1.3 partitioning**

A component `partitioning` element (see Figure 20) specifies a deployment pattern of components and their components-to-hosts relationships as well as nested sub-applications. A component instantiation is captured inside a `componentplacement` element. The `hostcollocation` element allows the components to be placed on a common device. When the `componentplacement` is by itself and not inside a `hostcollocation`, it has no collocation constraints. An assembly instantiation (nested sub-application) is captured within an `assemblyplacement` element.

![Diagram](image-url)

**Figure 20: partitioning Element Relationships**
<!ELEMENT partitioning  
   ( componentplacement | hostcollocation | 
    assemblyplacement )+> 

D-1.10.1.3.1 componentplacement

The componentplacement element (see Figure 21) defines a particular deployment of a component. The component can be deployed either directly or by using a ComponentFactory.

Figure 21: componentplacement Element Relationships

<!ELEMENT componentplacement  
   ( componentfileref , componentinstantiation+ 
    )>

D-1.10.1.3.1.1 componentfileref

The componentfileref element is used to reference a particular SPD or a SAD file. The componentfileref element's refid attribute corresponds to the componentfile element's id attribute.

<!ELEMENT componentfileref EMPTY>
<!ATTLIST componentfileref 
   refid CDATA #REQUIRED>

D-1.10.1.3.1.2 componentinstantiation

The componentinstantiation element (see Figure 22) is intended to describe a particular instantiation of a component relative to a componentplacement element. The componentinstantiation's id attribute is an implementation specific value that uniquely identifies the component within a Domain Profile. The componentinstantiation element's id may be referenced by the usesport and providesport elements within the SAD file. It is the component name for the instantiation and not the application name. The componentinstantiation element's stringifiedobjectref attribute, when specified, is the component instantiation object reference that requires dynamic connections. The componentinstantiation element's optional
processcollocation attribute indicates a specific logical process in which the component instance must be executed. The processcollocation attribute is used within the options parameter of the ExecutableInterface::execute operation.

Figure 22: componentinstantiation Element Relationships

<!ELEMENT componentinstantiation ( componentproperties?, coreaffinity*, deploymentdependencies?, componentfactoryref? )>
<!ATTLIST componentinstantiation
   id ID #REQUIRED
   processcollocation CDATA #IMPLIED
   stringifiedobjectref CDATA #IMPLIED>

D-1.10.1.3.1.2.1 componentproperties

The optional componentproperties element (see D-1.10.1.3.3.1.1) is a list of configure, factoryparam, and/or execparam properties values that are used in creating the component or for the initial configuration of the component.

The following sources will be searched in the given precedence order for initial values of properties with a kindtype of "execparam" or "configure" and a mode attribute of "readwrite" or "writeonly":

1. The SAD partitioning : componentplacement : componentinstantiation element,
2. The value or default value, if any, from the SPD using the properties precedence stated in D-1.6.1.

If no values are found in the sources above, the property is discarded.

The following sources will be searched in the given precedence order for initial values of properties with a kindtype of "factoryparam":
1. The SAD partitioning : componentplacement : componentinstantiation : componentfactoryref : componentfactoryproperties element,

2. The SAD partitioning : componentplacement : componentinstantiation : componentproperties element,

3. The value or default value, if any, from the SPD using the properties precedence stated in D-1.6.1.

If no values are found in the sources above, the property is discarded.

**D-1.10.1.3.1.2.2 coreaffinity**

The optional coreaffinity element indicates a preferred processor core for execution of a component instance. The coreaffinity element is used within the options parameter of the ExecutableInterface::execute operation.

The data type of the coreaffinity element is an unsigned long and its value is platform dependent.

```
<!ELEMENT coreaffinity (#PCDATA)>  
```

**D-1.10.1.3.1.2.3 deploymentdependencies**

The optional deploymentdependencies element (described generically in section D-1.10.1.3.1.4) overrides a componentinstantiation’s SPD and SAD dependencies.

**D-1.10.1.3.1.2.4 componentfactoryref**

The optional componentfactoryref element (see Figure 23) refers to a particular ApplicationComponentFactoryComponent componentinstantiation element found in the SAD, which is used to obtain an ApplicationComponent reference for this componentinstantiation element. The refid attribute refers to a unique componentinstantiation id attribute.

**Figure 23: componentfactoryref Element Relationships**
<!ELEMENT componentfactoryref ( componentfactoryproperties? )>
<!ATTLIST componentfactoryref refid CDATA #REQUIRED>

D-1.10.1.3.1.2.4.1 componentfactoryproperties

The optional componentfactoryproperties element (see Figure 24) specifies the properties "qualifiers", for the ComponentFactory::createComponent call.

![Figure 24: componentfactoryproperties Element Relationships](image_url)
<!ELEMENT componentfactoryproperties ( simpleref | simplesequenceref | structref | structsequenceref )+>
<!ELEMENT simpleref EMPTY>
<!ATTLIST simpleref
    refid CDATA #REQUIRED
    value CDATA #REQUIRED>
<!ELEMENT simplesequenceref ( values )>
<!ATTLIST simplesequenceref
    refid CDATA #REQUIRED>
<!ELEMENT structref ( simpleref+ )>
<!ATTLIST structref
    refid CDATA #REQUIRED>
<!ELEMENT structsequenceref ( structvalue+ )>
<!ATTLIST structsequenceref
    refid CDATA #REQUIRED>
<!ELEMENT structvalue ( simpleref+ )>
<!ELEMENT values ( value+ )>
<!ELEMENT value (#PCDATA)>

D-1.10.1.3.2 hostcollocation

The hostcollocation element specifies a group of component instances that are to be deployed together on a single host. For purposes of the SCA, the componentplacement element will be used to describe the 1…n components that will be collocated on the same host platform. Within the SCA specification, a host platform will be interpreted as a single device. The id and name attributes are optional but may be used to uniquely identify a set of collocated components within a SAD file.

<!ELEMENT hostcollocation ( componentplacement )+>
<!ATTLIST hostcollocation
    id ID #IMPLIED
    name CDATA #IMPLIED>

D-1.10.1.3.2.1 componentplacement

See componentplacement, section D-1.10.1.3.1.


**D-1.10.1.3.3 assemblyplacement**

The *assemblyplacement* element (see Figure 25) defines a particular deployment of a nested subassembly. It references the SAD file for that nested subassembly and an *assemblyinstantiation* element defining its creation.

```
<!ELEMENT assemblyplacement
   ( componentfileref , assemblyinstantiation+)
```

**Figure 25: assemblyplacement Element Relationships**

```
D-1.10.1.3.3.1 assemblyinstantiation
```

The *assemblyinstantiation* element (see Figure 26) describes an instantiation of an assembly as a nested sub-application relative to an *assemblyplacement* element. This *assemblyinstantiation* (which itself can potentially contain other nested sub-applications) can then be connected and controlled through the *assemblycontroller* and *externalports* elements defined in the referenced SAD file. *Assemblyinstantiation* creation can be viewed as essentially similar to a normal application created by an ApplicationFactoryComponent, with the omission of the registration of the sub-application's ApplicationManagerComponent with the DomainManagerComponent.

The *assemblyinstantiation*'s *id* attribute is an implementation specific value that uniquely identifies the assembly within a Domain Profile. The *assemblyinstantiation* element's *id* may be referenced by the *usesport* and *providesport* elements within the enclosing SAD file when connecting to/from ports listed in the sub-application's SAD *externalports* element.

The *assemblyinstantiation* element contains a number of sub-elements used by the core framework in the creation, deployment and configuration of the sub-application. Most of these elements appear only as sub-elements of the *assemblyinstantiation* element.
Figure 26: `assemblyinstantiation` Element Relationships

```xml
<!ELEMENT assemblyinstantiation
  (componentproperties?,
   deviceassignments?,
   deploymentdependencies?,
   executionaffinityassignments?)>

<!ATTLIST assemblyinstantiation
  id ID #REQUIRED>
```

**D-1.10.1.3.1.1 componentproperties**

The optional `componentproperties` element (see Figure 27) is a list of configure, factoryparam, and/or execparam property values that are used in creating and / or initially configuring the components of the sub-application. For valid properties with a `kindtype` of "execparam" or "factoryparam", or "configure" properties with a `mode` attribute of "readwrite" or "writeonly", values will supplement or (if of the same name), override sub-application values following the given precedence order:

1. The outer SAD partitioning: `assemblyplacement : assemblyinstantiation` element,
2. The nested SAD `componentinstantiation / assemblyinstantiation` element,
3. (N/A for factoryparam values) The value or default value, if any, from the component's SPD using the properties precedence stated in D-1.6.1.
D-1.10.1.3.3.1.2 deviceassignments
The optional *deviceassignments* element provides a list of *deviceassignment* elements which are used when deploying the sub-application's components.

In a *deviceassignment* element, the *componentid* attribute refers to the *componentinstantiationref* within the scope of the sub-application being created, while the *assignedDeviceid* refers a device identifier (Device Configuration Descriptor (DCD) *componentinstantiation id*) in the domain.

D-1.10.1.3.3.1.3 deploymentdependencies
The *deploymentdependencies* element (described generically in section D-1.10.1.3.3.1.4) overrides referenced SAD dependencies.

D-1.10.1.3.3.1.4 executionaffinityassignments
The optional *executionaffinityassignments* element provides a list of *executionaffinityassignment* elements which are used when deploying the sub-application's components.
The `executionaffinityassignment` element `componentid` attribute refers to a `componentinstantiationref` element within the scope of the sub-application being created. The optional `processcollocation` attribute indicates the logical process in which the component instance must be executed. The `processcollocation` attribute is used within the options parameter of the `ExecutableInterface::execute` operation. The optional `coreaffinity` element indicates a preferred processor core for execution of the sub-application component instances. The `coreaffinity` element is used within the options parameter of the `ExecutableInterface::execute` operation.

```xml
<!ELEMENT executionaffinityassignments (executionaffinityassignment+)
>
<!ELEMENT executionaffinityassignment (coreaffinity*)
>
<!ATTLIST executionaffinityassignment
    componentid CDATA #REQUIRED
    processcollocation CDATA #IMPLIED
>
```

### D-1.10.1.4 deploymentdependencies

The `deploymentdependencies` element (see Figure 28) is cited in multiple places within the SAD. It supplies, within its governing scope, overriding values for like-named dependencies defined within the scope. This allows the addition of scope-dependent deployment information, e.g. specification of a particular radio channel or security domain for the deployed application, sub-application or component. Each `dependencies` element is a `propertyref` referencing a property of `kind "allocation"`, and overrides any values from

- narrower-scope `deploymentdependencies`,
- SPD-defined `dependency propertyref` elements (see section D-1.6.1.6.10.2),
- SPD-defined `usesdevice propertyref` elements (see section D-1.6.1.6.11.1).

When, for any `componentinstantiation`, there are no matching dependencies (same id) specified in the SPD file, then the `deploymentdependencies`-supplied `propertyref` is not used to constrain deployment. In this way, dependencies can be overridden where they are specified, while not imposing new dependencies where they are not intended.

Deployment dependencies precedence order in order of highest to lowest is:

1. `Application Factory::create` deploymentdependencies parameter,
2. `Assemblyinstantiation deploymentdependencies`,
3. `Componentinstantiation deploymentdependencies`,
4. SAD deployment `deploymentdependencies`,
5. `Componentinstantiation SPD dependency` and `usesdevice`. 
The `assemblycontroller` element (see Figure 29) indicates the `componentinstantiation` and / or `assemblyinstantiation(s)` that form the control point(s) for the assembly. The ApplicationManagerComponent delegates it's `configure`, `query`, `start`, `stop`, and `runTest` operations to the elements identified by the `assemblycontroller` element.
<!ELEMENT assemblycontroller
    (( componentinstantiationref | assemblyinstantiationref),
    assemblyinstantiationref*)>

**D-1.10.1.6 connections**

The *connections* element is a child element of the *softwareassembly* element. The *connections* element is intended to provide the connection map between components in the assembly.

<!ELEMENT connections
    ( connectinterface* )>

**D-1.10.1.6.1 connectinterface**

The *connectinterface* element (see Figure 30) is used when application components are being assembled to describe connections between their port interfaces. The *connectinterface* element consists of a *usesport* element and a *providesport* or *componentsupportedinterface* element. These elements are intended to connect two compatible components. The *connectinterface* element's *id* attribute is a unique identifier for the connection interface definition within the Domain Profile.

![Diagram of connectinterface Element Relationships](image-url)

**Figure 30: connectinterface Element Relationships**
<!ELEMENT connectinterface
    ( usesport
        , ( providesport | componentsupportedinterface )
    )>
<!ATTLIST connectinterface
    id ID #IMPLIED>

D-1.10.1.6.1.1 usesport

The usesport element (see Figure 31) identifies, using the identifier element, the component port that is using the provided interface from the providesport element. A BaseComponent may be referenced by one of five elements: componentinstantiationref, assemblyinstantiationref, domainfinder, devicethatloadedthiscomponentref, and deviceusedbythiscomponentref.

Figure 31: usesport Element Relationships
<!ELEMENT usesport (identifier , ( componentinstantiationref | assemblyinstantiationref | devicethatloadedthiscomponentref | deviceusedbythiscomponentref | domainfinder ))>

D-1.10.1.6.1.1.1 identifier

The *identifier* element identifies which "uses port" on the component is to participate in the connection relationship. This identifier will correspond with an *id* for one of the component ports specified in the SCD.

<!ELEMENT identifier (#PCDATA)>

D-1.10.1.6.1.1.2 componentinstantiationref

The *componentinstantiationref* element refers to the *id* attribute of the *componentinstantiation* element within the SAD file. The *refid* attribute will correspond to the unique *componentinstantiation id* attribute.

<!ELEMENT componentinstantiationref EMPTY>
<!ATTLIST componentinstantiationref refid CDATA #REQUIRED>

D-1.10.1.6.1.1.3 assemblyinstantiationref

The *assemblyinstantiationref* element refers to the *id* attribute of the *assemblyinstantiation* element within the SAD file. The *refid* attribute will correspond to the unique *assemblyinstantiation id* attribute.

<!ELEMENT assemblyinstantiationref EMPTY>
<!ATTLIST assemblyinstantiationref refid CDATA #REQUIRED>

D-1.10.1.6.1.1.4 devicethatloadedthiscomponentref

The *devicethatloadedthiscomponentref* element refers to a specific component found in the assembly, which is used to obtain the DeviceComponent that was used to load the referenced component from the ApplicationFactoryComponent. The DeviceComponent obtained is then associated with this component instance.

<!ELEMENT devicethatloadedthiscomponentref EMPTY>
<!ATTLIST devicethatloadedthiscomponentref refid CDATA #REQUIRED>
D-1.10.1.6.1.1.5 deviceusedbythiscomponentref

The deviceusedbythiscomponentref element refers to a specific component, within the assembly, which is used to obtain the DeviceComponent (e.g., logical device) that is being used by the specific component from the ApplicationFactoryComponent. This relationship is needed when a component is pushing or pulling data and/or commands to another component that exists in the system such as an audio device.

<!ELEMENT deviceusedbythiscomponentref EMPTY>
<!ATTLIST deviceusedbythiscomponentref
    refid CDATA #REQUIRED
    usesrefid CDATA #REQUIRED>

D-1.10.1.6.1.1.6 domainfinder

The domainfinder element is used to indicate to the ApplicationFactoryComponent the necessary information to find an object reference that is of specific type and may also be known by an optional name within the domain. The valid type attributes are "filesystem", "eventchannel", "application", "servicename", and "servicetype".

For "filesystem" type when name attribute is not supplied then the closest FileSystemComponent proximity-wise (e.g. file system residing on the same physical device as the component on the other end of the connection would be used first) is provided.

The type attribute value of "eventchannel" is used to specify the event channel to be used in the OE's Event Service for producing or consuming events. If the name attribute is not supplied and the type attribute has a value of "eventchannel" then the Incoming domain management event channel is used.

For "application" type the name attribute must be specified. For "application" type the name attribute format is ApplicationFactoryComponent name followed by forward slash "/" followed by Application name (e.g. "the_applicationfactory_name/the_application_name"). The options for "application" type name and meaning are:

1. When only the application name is specified then any existing ApplicationManagerComponent in the domain with that name can be used;
2. When both the Application Factory name and Application name (e.g. "the_applicationfactory_name/the_application_name") is specified the ApplicationManagerComponent with that name created by the specified Application FactoryComponent is returned;
3. When only the Application Factory name followed by a forward slash is specified then any ApplicationManagerComponent created by the specified ApplicationFactoryComponent with that name can be used.

If "servicename" or "servicetype" is specified then name must be specified. Both values correspond to a service identified in a DeviceMangerComponent's DCD. The DCD usagename element contains a value in an "identifier\type" format for a component service type. For "servicename" the name corresponds to the "identifier" portion of the usagename element. For
"servicetype" the name corresponds to the "type" portion of the usagename element. The type attribute value of "servicename" is used to locate registered ServiceComponents on a per name basis. The type attribute value of "servicetype" is used to locate registered ServiceComponents on a per type basis.

<!ELEMENT domainfinder EMPTY>
<!ATTLIST domainfinder
type (filesystem | eventchannel | application | servicename | servicetype) #REQUIRED
name CDATA #IMPLIED>

D-1.10.1.6.1.2 providesport

The providesport element (see Figure 32) identifies, using the identifier element, the component port that is provided to a usesport within the connectinterface element. A BaseComponent may be referenced by one of five elements: componentinstantiationref, assemblyinstantiationref, domainfinder, devicethatloadedthiscomponentref, and deviceusedbythiscomponentref. The domainfinder element by itself is used when the component is not manageable via the CF interfaces. The providesport element's stringifiedobjectref attribute, when specified, is the component instantiation provides port object reference that does not require component registration but still requires dynamic connections.
Figure 32: providesport Element Relationships

`<!ELEMENT providesport (identifier , ( componentinstantiationref | assemblyinstantiationref | devicethatloadedthiscomponentref | deviceusedbythiscomponentref | domainfinder ))>`

`<!ATTLIST providesport stringifiedobjectref CDATA #IMPLIED>`

**D-1.10.1.6.1.2.1 identifier**

The *identifier* element identifies which "provides port" on the component is to participate in the connection relationship. This identifier will correspond with a *repid* attribute for one of the component ports elements, specified in the SCD.

`<!ELEMENT identifier (#PCDATA)>`

**D-1.10.1.6.1.2.2 componentinstantiationref**

See D-1.10.1.6.1.2 for a description of the *componentinstantiationref* element.

**D-1.10.1.6.1.2.3 assemblyinstantiationref**

See D-1.10.1.6.1.3 for a description of the *assemblyinstantiationref* element.

**D-1.10.1.6.1.2.4 devicethatloadedthiscomponentref**

See D-1.10.1.6.1.4 for a description of the *devicethatloadedthiscomponentref* element.

**D-1.10.1.6.1.2.5 deviceusedbythiscomponentref**

See D-1.10.1.6.1.5 for a description of the *deviceusedbythiscomponentref* element.

**D-1.10.1.6.1.2.6 domainfinder**

See section D-1.10.1.6.1.6 for a description of the *domainfinder* element.

**D-1.10.1.6.1.3 componentsupportedinterface**

The *componentsupportedinterface* element (see Figure 33) specifies a component, which has a *supportsinterface* element that can satisfy an interface connection to a port specified by the *usesport* element, within a *connectinterface* element. A component within the assembly may be referenced by one of four elements: *componentinstantiationref*, *domainfinder*, *devicethatloadedthiscomponentref*, and *deviceusedbythiscomponentref*. The *componentinstantiationref* identifies a component within the assembly. The *domainfinder* element points to an existing component that can be found within a DomainManagerComponent.
Figure 33: *componentsupportedinterface* Element Relationships

<!ELEMENT componentsupportedinterface
 (identifier
  , ( componentinstantiationref | devicethatloadedthiscomponentref |
              deviceusedbythiscomponentref | domainfinder)
 )>

*D-1.10.1.6.1.3.1 identifier*

The *identifier* element identifies which supported interface on the component is to participate in the connection relationship. This identifier will correspond with the *repid* attribute of one of the component's *supportsinterface* elements, specified in the SCD.

<!ELEMENT identifier (#PCDATA)>

*D-1.10.1.6.1.3.2 componentinstantiationref*

See section D-1.10.1.6.1.1.2 for a description of the *componentinstantiationref* element.

*D-1.10.1.6.1.3.3 domainfinder*

See section D-1.10.1.6.1.1.6 for a description of the *domainfinder* element.

*D-1.10.1.7 externalports*

The optional *externalports* element (see Figure 34) is a child element of the *softwareassembly* element (see Figure 19). The *externalports* element is used to identify the visible ports for the
software assembly. The ApplicationManager::getProvidesPorts operation is used to access the assembly's obtainable provides ports.

The usesidentifier element identifies which supported interface of the software assembly is to participate in the connection relationship. This identifier will correspond with an identifier element from one of the assembly component's usesport definitions.

The providesidentifier element identifies which supported interface of the software assembly is to participate in the connection relationship. This identifier will correspond with an identifier element from one of the assembly component's providesport definitions.

The supportedidentifier element identifies which supported interface of the software assembly is to participate in the connection relationship. This identifier will correspond with an identifier element from one of the assembly component's componentsupportedinterface definitions.

Figure 34: externalports Element Relationships
<!ELEMENT externalports ( port+ )>
<!ELEMENT port ( description?, ( usesidentifier | providesidentifier | supportedidentifier), (componentinstantiationref | assemblyinstantiationref) )>

<!ELEMENT usesidentifier (#PCDATA)>
<!ELEMENT providesidentifier (#PCDATA)>
<!ELEMENT supportedidentifier (#PCDATA)>
D-1.10.1.8 deploymentprefs

The optional deploymentprefs element is a reference to a local file. See section D-1.6.1.4.1 for the definition of the localfile element. The file refers to an Application Deployment Descriptor (ADD) file.

<!ELEMENT deploymentprefs
 ( localfile
 )>

D-1.11 DEVICE CONFIGURATION DESCRIPTOR

This section describes the XML elements of the Device Configuration Descriptor (DCD) XML file; the deviceconfiguration element (see Figure 35). The DCD is based on the SAD (e.g., componentfiles, partitioning, etc.) DTD. The intent of the DCD is to provide the means of describing the components that are initially started on the DeviceManagerComponent node, how to obtain the DomainManagerComponent reference, connections of services to components (DeviceComponents, DeviceManagerComponent), and the characteristics (file system names, etc.) for a DeviceManagerComponent. The componentfiles and partitioning elements are optional; if not provided, that means no components are started up on the node, except for a DeviceManagerComponent. If the partitioning element is specified then a componentfiles element has to be specified also.

SCA497 A Device Configuration Descriptor file shall have a ".dcd.xml" extension.

D-1.11.1 deviceconfiguration

The deviceconfiguration element's id attribute is a unique identifier within the domain for the device configuration. This id attribute is a unique identifier within the Domain Profile for the device configuration. The name attribute is the user-friendly name for the DeviceManagerComponent's label attribute. The corba_provider, log_capable, log_producer, devicemgr_deployment_data and platformcomponentfactorydeployment attributes represent the optional Units of Functionality (UOF) supported by the DeviceManagerComponent. The oe_profile attribute is the operating environment profile supported by the DeviceManagerComponent.
Figure 35: deviceconfiguration Element Relationships

```xml
<!ELEMENT deviceconfiguration
  ( description?,
    devicemanagersoftpkg,
    componentfiles?,
    partitioning?,
    connections?,
    domainmanager,
    filesystemnames?)>

<!ATTLIST deviceconfiguration
  id ID #REQUIRED
  name CDATA #IMPLIED
  corba_provider (false | true) "true"
  log_capable (false | true) "false"
  log_producer (false | true) "false"
  oe_profile (lightweight | medium | full) "medium"
  devicemgr_deployment_data (false | true) "false"
  platformcomponentfactorydeployment (false | true) "true">
```
**D-1.11.1.1 description**

The optional *description* element, of the *deviceconfiguration* element, may be used to provide information about the device configuration.

```xml
<!ELEMENT description (#PCDATA)>
```

**D-1.11.1.2 devicemanagersofpkg**

The *devicemanagersofpkg* element refers to the SPD for the DeviceManagerComponent that corresponds to this DCD. The SPD file is referenced by a *localfile* element. The SPD describes the DeviceManagerComponent and can be used to specify the *usesports* for the services (e.g. Log(s), etc.) that the DeviceManagerComponent uses. See section D-1.6.1.4.1 for a description of the *localfile* element. The optional *componentproperties* element is a list of configure and/or execparam properties values that are used in creating the DeviceManagerComponent. The following sources will be searched in the given precedence order for initial values for *simple* properties with a kindtype of "execparam" or "configure" and a mode attribute of "readwrite" or "writeonly":

1. The DCD *devicemanagersofpkg* element,
2. The value, if any, from the SPD using the properties precedence stated in D-1.6.1.

If no values are found in the sources above, the property is discarded.

```xml
<!ELEMENT devicemanagersofpkg ( localfile , componentproperties?)>
```

**D-1.11.1.3 componentfiles**

The optional *componentfiles* element is used to reference deployment information for components that are started up on the device. The *componentfile* element references a SPD. The SPD, for example, can be used to describe DeviceComponents, a DeviceManagerComponent, a DomainManagerComponent, and FileSystemComponents. The XML definition of the DCD's *componentfiles* element is the same as the one provided in the SAD's section D-1.10.1.2, see section D-1.10.1.2 for the XML definition of the *componentfiles* element.

**D-1.11.1.4 partitioning**

The optional *partitioning* element consists of a set of *componentplacement* elements. A component instantiation is captured inside a *componentplacement* element.

```xml
<!ELEMENT partitioning ( componentplacement )*>
```

**D-1.11.1.4.1 componentplacement**

The *componentplacement* element (see Figure 36) is used to define a particular deployment of a component. The *componentfileref* element identifies the component to be deployed. The
**componentinstantiation** element identifies the actual component created. Multiple components of the same kind can be created within the same **componentplacement** element.

The optional **deployondevice** element indicates the device on which the **componentinstantiation** element is deployed. The optional **compositepartofdevice** element indicates the parent device of the **componentinstantiation** element. When the component is a logical device, the optional **devicepkgfile** element indicates the hardware device information for the logical device.

![Figure 36: componentplacement Element Relationships](image)

```xml
<!ELEMENT componentplacement
   ( componentfileref?, deployondevice?, compositepartofdevice?, devicepkgfile?, componentinstantiation+ )>

D-1.11.1.4.1.1 componentfileref

The **componentfileref** element is used to reference a **componentfile** element within the **componentfiles** element. The **componentfileref** element's **refid** attribute corresponds to a **componentfile** element's **id** attribute.

```xml
<!ELEMENT componentfileref EMPTY>
<!ATTLIST componentfileref
   refid CDATA #REQUIRED>
```

D-1.11.1.4.1.2 deployondevice

The **deployondevice** element is used to reference a **componentinstantiation** element on which this **componentinstantiation** is deployed.
<!ELEMENT deployondevice EMPTY>
<!ATTLIST deployondevice
    refid CDATA #REQUIRED>

D-1.11.1.4.1.3 compositepartofdevice

The compositepartofdevice element is used when a parent-child relationship exists between devices to reference the componentinstantiation element that describes the parent device when this device's componentinstantiation element describes the child device.

<!ELEMENT compositepartofdevice EMPTY>
<!ATTLIST compositepartofdevice
    refid CDATA #REQUIRED>

D-1.11.1.4.1.4 devicepkgfile

The devicepkgfile element is used to refer to a device package file that contains the hardware device definition.

<!ELEMENT devicepkgfile
 ( localfile )>
<!ATTLIST devicepkgfile
type CDATA #IMPLIED>

D-1.11.1.4.1.4.1localfile

See D-1.6.1.4.1 for a definition of the localfile element.

D-1.11.1.4.1.5 componentinstantiation

The componentinstantiation element (see Figure 37) is intended to describe a particular instantiation of a component relative to a componentplacement element. The componentinstantiation's id attribute is an implementation specific value that uniquely identifies the component within the domain. The componentinstantiation element's stringifiedobjectref attribute, when specified, is the component instantiation object reference that requires dynamic connections. The usagename element contains a readable name for the component.

The usagename element is required for a component service (e.g., Log Service implementation), and is not used by other platform components. For ServiceComponents, usagename must be provided in an "identifier|type" format. The "identifier" portion of the name must be unique for each service instantiation within a domain. The "type" value is common across all instantiations of the same service. The "type" value should be representative of the service that is being provided such as the name or the interfaces. The value "log" represents the type for a Log Service implementation. The componentinstantiation element's optional processcollocation attribute indicates a specific logical process in which the component instance must be executed. The processcollocation attribute is used within the options parameter of the ExecutableInterface::execute operation.
Figure 37: componentinstantiation Element Relationships

```xml
<!ELEMENT componentinstantiation
    ( usagename?,
      componentproperties?,
      componentfactoryref?,
      coreaffinity*)>

<!ATTLIST componentinstantiation
    id ID #REQUIRED
    processcollocation CDATA #IMPLIED
    stringifiedobjectref CDATA #IMPLIED>

<!ELEMENT usagename (#PCDATA)>
```

D-1.11.1.4.1.5.1 componentproperties

The optional componentproperties element (see Figure 38) is a list of configure, factoryparam, and/or execparam properties values that are used in creating the component or for the initial configuration of the component.

The following sources will be searched in the precedence order below for initial values of properties with a kindtype of "execparam", "configure" or "allocation" and a mode attribute of "readwrite" or "writeonly":

1. The DCD partitioning : componentplacement : componentinstantiation element,
2. The value, if any, from the SPD using the properties precedence stated in D-1.6.1.

If no values are found in the sources above, the property is discarded.

The following sources will be searched in the given precedence order for initial values of properties with a kindtype of "factoryparam":

1. The DCD partitioning: componentplacement: componentinstantiation: componentfactoryref: componentfactoryproperties element,
2. The DCD partitioning: componentplacement: componentinstantiation: componentproperties element,

3. The value, if any, from the SPD using the properties precedence stated in D-1.6.1.

If no values are found in the sources above, the property is discarded.

Figure 38: componentproperties Element Relationships

```xml
<!ELEMENT componentproperties ( simpleref | simplesequenceref | structref | structsequenceref )+ >
```

D-1.11.1.4.1.5.2 componentfactoryref

The optional componentfactoryref element (see Figure 39) refers to a particular PlatformComponentFactoryComponent componentinstantiation element found in the DCD, which is used to obtain a DeviceComponent or ServiceComponent for this componentinstantiation element. The refid attribute refers to a unique componentinstantiation id attribute. The optional componentfactoryref element should be specified only when a PlatformComponentFactoryComponent is used to create DeviceComponent or ServiceComponents.
The optional `componentfactoryproperties` element (see Figure 40) specifies the properties "qualifiers", for the PlatformComponentFactoryComponent `createComponent` call.
Figure 40: componentfactoryproperties Element Relationships

```xml
<!ELEMENT componentfactoryproperties ( simpleref | simplesequenceref | structref | structsequenceref )+ >
<!ELEMENT simpleref EMPTY>
<!ATTLIST simpleref refid CDATA #REQUIRED
value CDATA #REQUIRED>
<!ELEMENT simplesequenceref ( values )>
<!ATTLIST simplesequenceref refid CDATA #REQUIRED>
<!ELEMENT structref ( simpleref+ )>
<!ATTLIST structref refid CDATA #REQUIRED>
<!ELEMENT structsequenceref ( structvalue+ )>
<!ATTLIST structsequenceref refid CDATA #REQUIRED>
<!ELEMENT structvalue ( simpleref+ )>
<!ELEMENT values ( value+ )>
<!ELEMENT value (#PCDATA)>
```
**D-1.11.1.4.1.5.3 coreaffinity**

The optional `coreaffinity` element indicates a preferred processor core for execution of a component instance. The `coreaffinity` element is used within the options parameter of the `ExecutableInterface::execute` operation.

The data type of the `coreaffinity` element is an unsigned long and its value is platform dependent.

```xml
<!ELEMENT coreaffinity (#PCDATA)>```

**D-1.11.1.5 connections**

The `connections` element in the DCD is the same as the `connections` element in the SAD in section D-1.10.1.6. The `connections` element in the DCD is used to indicate the services (Log, etc.) instances that are used by the DeviceManagerComponent and DeviceComponent in the DCD. To establish connections to a DeviceManagerComponent, the DCD's `deviceconfiguration` element's `id` attribute value is used for the SAD's `usesport` element's `componentinstantiationref` element's `refid` attribute value.

**D-1.11.1.6 domainmanager**

The `domainmanager` element indicates how to register to the DomainManagerComponent. The value of the `name` attribute is based upon the `type` attribute, which could be the name of a file. The `type` attribute indicates the type of mechanism used for obtaining the DomainManagerComponent registration reference such as the `localfile` that contains an ior, ior_string, or resolve_initial_reference.

```xml
<!ELEMENT domainmanager EMPTY>
<!ATTLIST domainmanager
  name CDATA #REQUIRED
  type CDATA #REQUIRED>
```

**D-1.11.1.7 filesystemnames**

The optional `filesystemnames` element indicates the mounted file system names for DeviceManagerComponent's FileManagerComponent.

The optional `filesystemnames` element indicates the names for file systems mounted within a DeviceManagerComponent's FileManagerComponent. The `mountname` attribute contains a file system name that uniquely identifies a mount point. The `deviceid` attribute is the unique identifier for a specific component, within the DCD, which represents the device hosting this file system. The use of the `deviceid` attribute value is implementation dependent.

```xml
<!ELEMENT filesystemnames
  ( filesystemname+ )>
<!ELEMENT filesystemname EMPTY>
<!ATTLIST filesystemname
  mountname CDATA #REQUIRED
  deviceid CDATA #REQUIRED>
```
D-1.12 DOMAIN MANAGER CONFIGURATION DESCRIPTOR

This section describes the XML elements of the Domain Manager Configuration Descriptor (DMD) XML file.

SCA498 A DomainManager Configuration Descriptor file shall have a ".dmd.xml" extension.

D-1.12.1 domainmanagerconfiguration

The domainmanagerconfiguration element (see Figure 41) id attribute is an implementation specific value that uniquely identifies the DomainManagerComponent within the domain. The app_backwards_compatible, app_deployment_data, app_installable, app_releasable, corba_provider, channel_extension, event_channel, log_producer, and nested_deployment attributes represent the optional UOFs supported by the DomainManagerComponent. The accardinality attribute indicates whether or not the domain supports multiple application controllers. The oe_profile attribute indicates the operating environment profile supported by the DomainManagerComponent.

![Diagram showing the domainmanagerconfiguration Element Relationships]

Figure 41: domainmanagerconfiguration Element Relationships
<!--ELEMENT domainmanagerconfiguration (> description?, domainmanagersoftpkg?, deploymentlayout?, services?) -->
<!ATTLIST domainmanagerconfiguration
id ID #REQUIRED
name CDATA #REQUIRED
accardinality (single | multiple) "single"
app_backwards_compatible (false | true) "false"
app_deployment_data (false | true) "true"
app_installable (false | true) "true"
app_releasable (false | true) "true"
corba_provider (false | true) "true"
channel_extension (false | true) "false"
event_channel (false | true) "false"
log_producer (false | true) "false"
nested_deployment (false | true) "false"
oe_profile (lightweight| medium | full) "medium" >

D-1.12.1.1 description
The optional description element of the DMD may be used to provide information about the configuration.

<!ELEMENT description (#PCDATA)> 

D-1.12.1.2 domainmanagersoftpkg
The domainmanagersoftpkg element refers to the SPD for the DomainManagerComponent. The SPD file is referenced by a localfile element. This SPD can be used to describe the DomainManagerComponent implementation and to specify the usesports for the services (Log(s), etc.) used by the DomainManagerComponent. See section D-1.6.1.4.1 for description of the localfile element. The optional componentproperties element is a list of configure and/or execparam properties values that are used in creating the DomainManagerComponent. The following sources will be searched in the given precedence order for initial values for simple properties with a kindtype of "execparam" or "configure" and a mode attribute of "readwrite" or "writeonly":
1. The DMD domainmanagersoftpkg element,
2. The value, if any, from the SPD using the properties precedence stated in D-1.6.1.
If no values are found in the sources above, the property is discarded.
<!ELEMENT domainmanagersoftpkg
     ( localfile
     , componentproperties?) >

**D-1.12.1.3 deploymentlayout**

The optional deploymentlayout element is a reference to a local file. See section D-1.6.1.4.1 for the definition of the localfile element. The file refers to a Platform Deployment Descriptor (PDD) file.

<!ELEMENT deploymentlayout
     ( localfile )>

**D-1.12.1.4 services**

The optional services element (see Figure 42) in the DMD is used by the DomainManagerComponent to determine which service (e.g. Log, etc.) instances to use; it makes use of the service element.

```
<!ELEMENT services
     ( service+ ) >
```

**D-1.12.1.4.1 service**

The service element (see Figure 43) defines the service instance. See section D-1.10.1.6.1.1.6 for a description of the domainfinder element. See section D-1.10.1.6.1.1.1 for a description of the identifier element.

«DTDSequenceGroup»
services_grp1
«DTDElement»
services
«DTDElement»
service
1..*
Figure 43: *service* Element Relationships

```xml
<!ELEMENT service
 ( identifier
 , domainfinder)>
```
D-1.13 PLATFORM DEPLOYMENT DESCRIPTOR

This section describes the XML elements of the Platform Deployment Descriptor (PDD) XML file; the deploymentplatform element. The intent of the PDD is to provide a means of describing the collection of services and devices that are associated with a virtual channel. The knowledge of the channel composition can be utilized as part of an overall systems engineering strategy to control the allocation of applications to system resources. Another use of the information could be to improve the efficiency of application deployment as the channel elements would be used to constrain the search space for the allocation of individual application components. The use of the PDD is optional within a system, a system designer is free to use allocation properties or other approaches to manage the allocation of application components to platform resources.

SCA499 A Platform Deployment Descriptor file shall have a ".pdd.xml" extension.

D-1.13.1 deploymentplatform

The deploymentplatform element (see Figure 44) contains the layout of the virtual channels within a platform domain.

```
<!ELEMENT deploymentplatform
   ( description?, channel+ ) >
```

Figure 44: deploymentplatform Element Relationships

D-1.13.1.1 description

The optional description element of the PDD may be used to provide information about the platform domain.

```
<!ELEMENT description (#PCDATA)>
```
**D-1.13.1.2 channel**

The channel element (see Figure 45) in the PDD defines the collections of devices and services that are used by an ApplicationFactoryComponent as target resource pools for application deployment. The channel element's name attribute contains the identifier for the channel that is used by the ApplicationFactoryComponent and the ADD.

![Diagram showing channel element relationships]

```xml
<!ELEMENT channel
    ( devicelist?, servicelist?) >
<!ATTLIST channel
    name ID #REQUIRED>
```

**D-1.13.1.2.1 devicelist**

The optional devicelist element in the PDD defines the collection of devices for a given channel that are used by the ApplicationFactoryComponent as target resource pools for application deployment.

```xml
<!ELEMENT devicelist
    ( deviceref+)>
```

**D-1.13.1.2.1.1 deviceref**

The deviceref element is used to reference a componentinstantiation element which is part of the channel. The refid attribute points to a componentinstantiation identifier for a device that has registered with the platform.
<!ELEMENT deviceref EMPTY>
<!ATTLIST deviceref
    refid CDATA #REQUIRED>

D-1.13.1.2.2 servicelist

The optional servicelist element in the PDD defines the collection of services for a given channel that are used by the ApplicationFactoryComponent as target resource pools for application deployment.

<!ELEMENT servicelist
    ( serviceref+)
>
D-1.13.1.2.2.1 serviceref

The serviceref element identifies a service which is part of the channel. The servicename attribute is identical to the "identifier" portion usagename identifier for a service that has registered with the platform (for a component service type usagename is provided in an "identifier\type" format).

<!ELEMENT serviceref EMPTY>
<!ATTLIST serviceref
    servicename CDATA #REQUIRED>
D-1.14 APPLICATION DEPLOYMENT DESCRIPTOR

This section describes the XML elements of the Application Deployment Descriptor (ADD) XML file; the deploymentprecedence element. The intent of the ADD is to provide prioritized lists of deployment alternatives for application instances.

SCA500 An Application Deployment Descriptor file shall have an "add.xml" extension.

D-1.14.1 deploymentprecedence

The deploymentprecedence element (see Figure 46) contains the relationship between application instances and their candidate virtual channels.

```xml
<!ELEMENT deploymentprecedence
    ( description?, deploymentoption+)>
```

Figure 46: deploymentprecedence Element Relationships

D-1.14.1.1 description

The optional description element of the ADD may be used to provide information about the application.

```xml
<!ELEMENT description (#PCDATA)>
```

D-1.14.1.2 deploymentoption

The deploymentoption element (see Figure 47) in the ADD identifies the ordered list of channels that provide deployment alternatives for a specific application instance. The deployedname attribute corresponds to a named application instance (e.g. the name parameter passed to the ApplicationFactory::create operation).
Figure 47: deploymentoption Element Relationships

```xml
<!ELEMENT deploymentoption
    ( description?,
    channelref+)
>
<!ATTLIST deploymentoption
    deployedname CDATA #REQUIRED>
```

**D-1.14.1.2.1 description**

The optional `description` element may be used to provide information about the application instance.

```xml
<!ELEMENT description (#PCDATA)> 
```

**D-1.14.1.2.2 channelref**

The `channelref` element is used to reference a `channel` element from the PDD which provides a deployment alternative. The `refname` attribute points to a `channel` element `name` attribute that identifies a channel.

```xml
<!ELEMENT channelref EMPTY>
<!ATTLIST channelref
    refname CDATA #REQUIRED>
```
D-1.15 ATTACHMENTS

This appendix includes the following:

- Attachment 1: Common Properties Definitions
D-1.16 DOCUMENT TYPE DEFINTION (DTD) FILES

D-1.16.1 softpkg

<!-- softpkg.4.1.dtd -->

<!ELEMENT softpkg
   ( title?,
     author+,
     description?,
     propertyfile?,
     descriptor?,
     implementation+, 
     usesdevice*)
>
<!ATTLIST softpkg
   id ID #REQUIRED
   name CDATA #REQUIRED
   type (sca_compliant | sca_non_compliant) "sca_compliant"
   version CDATA #IMPLIED >

<!ELEMENT title (#PCDATA)>
<!ELEMENT author
   ( name*,
     company?,
     webpage?)
>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT webpage (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT propertyfile
   ( localfile
   )>
<!ATTLIST propertyfile
   type CDATA #IMPLIED>
<!ELEMENT localfile EMPTY>
<!ATTLIST localfile
   name CDATA #REQUIRED>
<!ELEMENT descriptor
   (localfile
   )>
<!ATTLIST descriptor
   name CDATA #IMPLIED>
<!ELEMENT implementation
   ( description?
name CDATA #REQUIRED>
</!ELEMENT dependency
  ( softpkgref | propertyref )>
<!ATTLIST dependency
type CDATA #REQUIRED>
<!ELEMENT softpkgref
  ( localfile
    , implref?
   )>
<!ELEMENT implref EMPTY>
<!ATTLIST implref
  refid CDATA #REQUIRED>
<!ELEMENT propertyref EMPTY>
<!ATTLIST propertyref
  refid CDATA #REQUIRED
  value CDATA #REQUIRED>
<!ELEMENT usesdevice
  ( propertyref+ )>
<!ATTLIST usesdevice
  id ID #REQUIRED
  type CDATA #REQUIRED>
D-1.16.2  devicepkg

<!-- devicepkg.4.1.dtd -->

<!ELEMENT devicepkg
  ( title?,
    author+,
    description?,
    hwdeviceregistration
  )>
<!ATTLIST devicepkg
  id    ID    #REQUIRED
  name  CDATA  #REQUIRED
  version CDATA  #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author
  ( name*,
    company?,
    webpage?
  )>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT webpage (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT hwdeviceregistration
  ( propertyfile?,
    description,
    manufacturer,
    modelnumber,
    deviceclass,
    childhwdevice*
  )>
<!ATTLIST hwdeviceregistration
  id    ID    #REQUIRED
  name  CDATA  #REQUIRED
  version CDATA  #IMPLIED>
<!ELEMENT propertyfile
  ( localfile
  )>
<!ATTLIST propertyfile
  type CDATA  #IMPLIED>
<!ELEMENT localfile EMPTY>
<!ATTLIST localfile
  name CDATA  #REQUIRED>
<!ELEMENT manufacturer (#PCDATA)>
<!ELEMENT modelnumber (#PCDATA)>
<!ELEMENT deviceclass
    ( class+)
>
<!ELEMENT class (#PCDATA)>
<!ELEMENT childhwdevice
    ( hwdeviceregistration | devicepkgref )>
<!ELEMENT devicepkgref
    ( localfile )>
<!ATTLIST devicepkgref
type CDATA #IMPLIED>
D-1.16.3 properties

<!-- properties.4.1.dtd -->

<!ELEMENT properties ( description?, (simple | simplesequence | test | struct | structsequence )* )>

<!ELEMENT simple ( description?, value?, units?, range?, enumerations?, kind*, action?)>

<!ATTLIST simple id ID #REQUIRED
 type ( boolean | char | double | float | short | long |
       objref | octet | string | ulong | ushort ) #REQUIRED
 name CDATA #IMPLIED
 mode (readonly | readwrite | writeonly) "readwrite">

<!ELEMENT description (#PCDATA)>

<!ELEMENT value (#PCDATA)>

<!ELEMENT units (#PCDATA)>

<!ELEMENT range EMPTY>

<!ATTLIST range min CDATA #REQUIRED
 max CDATA #REQUIRED>

<!ELEMENT enumerations ( enumeration+ )>

<!ELEMENT enumeration EMPTY>

<!ATTLIST enumeration label CDATA #REQUIRED
 value CDATA #IMPLIED>

<!ELEMENT kind EMPTY>

<!ATTLIST kind kindtype ( allocation | configure | test | execparam |
       factoryparam) "configure">

<!ELEMENT action EMPTY>

<!ATTLIST action type ( eq | ne | gt | lt | ge | le | external ) "external">

<!ELEMENT simplesequence
( description?,
  values?,
  units?,
  range?,
  kind*,
  action? )>

<!ATTLIST simplesequence
  id ID #REQUIRED
  type ( boolean | char | double | float | short | long |
  objref | octet | string | ulong |ushort ) #REQUIRED
  name CDATA #IMPLIED
  mode (readonly | readwrite | writeonly) "readwrite">

<!ELEMENT values
  ( value+ )>

<!ELEMENT test
( description
 , inputvalue?
 , resultvalue
 )>

<!ATTLIST test
  id CDATA #REQUIRED
  label CDATA #IMPLIED>

<!ELEMENT inputvalue
( simple+ )>

<!ELEMENT resultvalue
( simple+ )>

<!ELEMENT struct
( description?
 , simple+
 , configurationkind?
 )>

<!ATTLIST struct
  id ID #REQUIRED
  name CDATA #IMPLIED
  mode (readonly | readwrite | writeonly) "readwrite">

<!ELEMENT configurationkind EMPTY>

<!ATTLIST configurationkind
  kindtype (configure | factoryparam | typedef) "configure">

<!ELEMENT structsequence
( description?
 , structvalue*
 , configurationkind?
 )>

<!ATTLIST structsequence
id | ID | #REQUIRED
structrefid | CDATA | #REQUIRED
name | CDATA | #IMPLIED
mode (readonly | readwrite | writeonly) "readwrite"

<!ELEMENT structvalue
   ( simpleref+ )>

<!ELEMENT simpleref EMPTY>
<!ATTLIST simpleref
   refid | CDATA | #REQUIRED
   value | CDATA | #REQUIRED>
D-1.16.4 softwarecomponent

<!-- softwarecomponent.4.1.dtd -->

<!ELEMENT softwarecomponent
  ( componentrepid
    , componenttype
    , componentfeatures
    , interfaces
    , propertyfile?
  )>

<!ELEMENT componentrepid EMPTY>
<!ATTLIST componentrepid
  repid CDATA #REQUIRED>

<!ELEMENT componenttype (#PCDATA)>

<!ELEMENT componentfeatures
  ( supportsinterface*
    , ports
  )>

<!ELEMENT supportsinterface EMPTY>
<!ATTLIST supportsinterface
  repid CDATA #REQUIRED
  supportsname CDATA #REQUIRED>

<!ELEMENT ports
  ( provides | uses )*>
( inheritsinterface*) >
<!ATTLIST interface
    repid CDATA #REQUIRED
    name CDATA #REQUIRED>
<!ELEMENT inheritsinterface EMPTY>
<!ATTLIST inheritsinterface
    repid CDATA #REQUIRED>
<!ELEMENT propertyfile
    ( localfile )>
<!ATTLIST propertyfile
    type CDATA #IMPLIED>
<!ELEMENT localfile EMPTY>
<!ATTLIST localfile
    name CDATA #REQUIRED>
D-1.16.5  softwareassembly

<!-- softwareassembly.4.1.dtd -->

<!ELEMENT softwareassembly
  ( description?,
    componentfiles
    , partitioning
    , deploymentdependencies?,
    assemblycontroller
    , connections?
    , externalports?
    , deploymentprefs?)>

<!ATTLIST softwareassembly
  name ID #REQUIRED
  sca_version CDATA "V4.1"
  version CDATA #IMPLIED>

<!ELEMENT description (#PCDATA)>
<!ELEMENT componentfiles
  ( componentfile+ )>

<!ELEMENT componentfile
  ( localfile )>

<!ATTLIST componentfile
  id ID #REQUIRED
  type CDATA #IMPLIED>

<!ELEMENT localfile EMPTY>
<!ATTLIST localfile
  name CDATA #REQUIRED>

<!ELEMENT partitioning
  ( componentplacement | hostcollocation | assemblyplacement )+>

<!ELEMENT componentplacement
  ( componentfileref
    , componentinstantiation+ )>

<!ELEMENT componentfileref EMPTY>
<!ATTLIST componentfileref
  refid CDATA #REQUIRED>

<!ELEMENT componentinstantiation
  ( componentproperties?
    , coreaffinity*
    , deploymentdependencies?
    , componentfactoryref? )>
<!ATTLIST componentinstantiation
  id ID #REQUIRED
  processcollocation CDATA #IMPLIED
  stringifiedobjectref CDATA #IMPLIED>
<!ELEMENT coreaffinity (#PCDATA)>
<!ELEMENT componentfactoryref
  ( componentfactoryproperties? )>
<!ATTLIST componentfactoryref
  refid CDATA #REQUIRED>
<!ELEMENT componentfactoryproperties
  ( simpleref | simplesequenceref | structref |
    structsequenceref )+ >
<!ELEMENT simpleref EMPTY>
<!ATTLIST simpleref
  refid CDATA #REQUIRED
  value CDATA #REQUIRED>
<!ELEMENT simplesequenceref
  ( values )>
<!ATTLIST simplesequenceref
  refid CDATA #REQUIRED>
<!ELEMENT structref
  ( simpleref+ )>
<!ATTLIST structref
  refid CDATA #REQUIRED>
<!ELEMENT structsequenceref
  ( structvalue+ )>
<!ATTLIST structsequenceref
  refid CDATA #IMPLIED>
<!ELEMENT structvalue
  ( simpleref+ )>
<!ATTLIST structvalue
  refid CDATA #IMPLIED>
<!ELEMENT values
  ( value+ )>
<!ELEMENT value (#PCDATA)>
<!ELEMENT hostcollocation
  ( componentplacement )+>
<!ATTLIST hostcollocation
  id ID #IMPLIED
  name CDATA #IMPLIED>
<!ELEMENT assemblyplacement
  ( componentfileref
    , assemblyinstantiation+ )>
<!ELEMENT assemblyinstantiation
  ( componentproperties?
    , deviceassignments? )>
<!ATTLIST assemblyinstantiation id ID #REQUIRED>
<!ELEMENT componentproperties ( simpleref | simplesequenceref | structref | structsequenceref )+ >
<!ELEMENT deviceassignments ( deviceassignment )+>
<!ELEMENT deviceassignment EMPTY>
<!ATTLIST deviceassignment componentid CDATA #REQUIRED assignedDeviceid CDATA #REQUIRED>
<!ELEMENT deploymentdependencies (propertyref )+>
<!ELEMENT executionaffinityassignments ( executionaffinityassignment+ )>
<!ELEMENT executionaffinityassignment ( coreaffinity* )>
<!ATTLIST executionaffinityassignment componentid CDATA #REQUIRED processcollocation CDATA #IMPLIED>
<!ELEMENT propertyref EMPTY>
<!ATTLIST propertyref refid CDATA #REQUIRED value CDATA #REQUIRED>
<!ELEMENT assemblycontroller (( componentinstantiationref | assemblyinstantiationref), assemblyinstantiationref* )>
<!ELEMENT connections ( connectinterface* )>
<!ELEMENT connectinterface ( usesport , ( providesport | componentsupportedinterface ) )>
<!ATTLIST connectinterface id ID #IMPLIED>
<!ELEMENT usesport (identifier , ( componentinstantiationref | assemblyinstantiationref | devicethatloadedthiscomponentref | deviceusedbythiscomponentref |
domainfinder )
>
<!ELEMENT identifier (#PCDATA)>
<!ELEMENT componentinstantiationref EMPTY>
<!ATTLIST componentinstantiationref
 refid CDATA #REQUIRED>
<!ELEMENT assemblyinstantiationref EMPTY>
<!ATTLIST assemblyinstantiationref
 refid CDATA #REQUIRED>
<!ELEMENT devicethatloadedthiscomponentref EMPTY>
<!ATTLIST devicethatloadedthiscomponentref
 refid CDATA #REQUIRED>
<!ELEMENT deviceusedbythiscomponentref EMPTY>
<!ATTLIST deviceusedbythiscomponentref
 refid CDATA #REQUIRED
 usesrefid CDATA #REQUIRED>
<!ELEMENT domainfinder EMPTY>
<!ATTLIST domainfinder
 type (filesystem | eventchannel | application | servicename
 | servicetype) #REQUIRED
 name CDATA #IMPLIED>
<!ELEMENT providesport
 (identifier
 , ( componentinstantiationref |
 assemblyinstantiationref |
 devicethatloadedthiscomponentref |
 deviceusedbythiscomponentref |
 domainfinder )
 )>
<!ATTLIST providesport
 stringifiedobjectref CDATA #IMPLIED>
<!ELEMENT componentsupportedinterface
 (identifier
 , ( componentinstantiationref |
 devicethatloadedthiscomponentref |
 deviceusedbythiscomponentref |
 domainfinder)
 )>
<!ELEMENT externalports
 ( port+
 )>
<!ELEMENT port
 ( description?
 , ( usesidentifier | providesidentifier |
 supportedidentifier)
 , (componentinstantiationref | assemblyinstantiationref)
 )>
<!ELEMENT usesidentifier (#PCDATA)>
<!ELEMENT providesidentifier (#PCDATA)>
<!ELEMENT supportedidentifier (#PCDATA)>
<!ELEMENT deploymentprefs ( localfile )>
D-1.16.6  **deviceconfiguration**

<!-- deviceconfiguration.4.1.dtd -->

<!ELEMENT deviceconfiguration
  ( description?,
    devicemanagersoftpkg,
    componentfiles?,
    partitioning?,
    connections?,
    domainmanager,
    filesystemnames?)>

<!ATTLIST deviceconfiguration
  id     ID   #REQUIRED
  name   CDATA  #IMPLIED
  corba_provider (false | true)  "true"
  log_capable (false | true)  "false"
  log_producer (false | true)  "false"
  oe_profile (lightweight | medium | full)  "medium"
<!ATTLIST componentinstantiation
   id ID #REQUIRED
   processcollocation CDATA #IMPLIED
   stringifiedobjectref CDATA #IMPLIED >
</!ELEMENT usagename (#PCDATA)>
<!ELEMENT componentproperties
   ( simpleref | simplesequenceref | structref |
     structsequenceref )+ >
</!ELEMENT componentfactoryref
   ( componentfactoryproperties? )>
</!ATTLIST componentfactoryref
   refid CDATA #REQUIRED>
</!ELEMENT componentfactoryproperties
   ( simpleref | simplesequenceref | structref | structsequenceref )+ >
</!ELEMENT simpleref EMPTY>
</!ATTLIST simpleref
   refid CDATA #REQUIRED
   value CDATA #REQUIRED>
</!ELEMENT simplesequenceref
   ( values )>
</!ATTLIST simplesequenceref
   refid CDATA #REQUIRED>
</!ELEMENT structref
   ( simpleref+ )>
</!ATTLIST structref
   refid CDATA #REQUIRED>
</!ELEMENT structsequenceref
   ( structvalue+ )>
</!ATTLIST structsequenceref
   refid CDATA #REQUIRED>
</!ELEMENT structvalue
   ( simpleref+ )>
</!ATTLIST values
   ( value+ )>
</!ELEMENT value (#PCDATA)>
</!ELEMENT coreaffinity (#PCDATA)>
</!ELEMENT connections
   ( connectinterface* )>
</!ELEMENT connectinterface
   ( usesport
     , ( providesport | componentssupportedinterface )
   )>
</!ATTLIST connectinterface
id  ID  #IMPLIED>
<!ELEMENT usesport
  (identifier
   , ( componentinstantiationref | assemblyinstantiationref | devicethatloadedthiscomponentref | deviceusedbythiscomponentref | domainfinder )
  )>
<!ELEMENT identifier (#PCDATA)>
<!ELEMENT componentinstantiationref EMPTY>
<!ATTLIST componentinstantiationref refid CDATA #REQUIRED>
<!ELEMENT assemblyinstantiationref EMPTY>
<!ATTLIST assemblyinstantiationref refid CDATA #REQUIRED>
<!ELEMENT devicethatloadedthiscomponentref EMPTY>
<!ATTLIST devicethatloadedthiscomponentref refid CDATA #REQUIRED>
<!ELEMENT deviceusedbythiscomponentref EMPTY>
<!ATTLIST deviceusedbythiscomponentref refid CDATA #REQUIRED
  usesrefid CDATA #REQUIRED>
<!ELEMENT domainfinder EMPTY>
<!ATTLIST domainfinder
  type (filesystem | eventchannel | application | servicename | servicetype) #REQUIRED
  name CDATA #IMPLIED>
<!ELEMENT providesport
  (identifier
   , ( componentinstantiationref | assemblyinstantiationref | devicethatloadedthiscomponentref | deviceusedbythiscomponentref | domainfinder )
  )>
<!ATTLIST providesport stringifiedobjectref CDATA #IMPLIED>
<!ELEMENT componentsupportedinterface
  (identifier
   , ( componentinstantiationref | devicethatloadedthiscomponentref | deviceusedbythiscomponentref | domainfinder)
  )>
<!ELEMENT domainmanager EMPTY>
<!ATTLIST domainmanager
  name CDATA #REQUIRED>
type CDATA  #REQUIRED>
</!ELEMENT filesystemnames
  ( filesystemname+  )>
</!ELEMENT filesystemname EMPTY>
<!ATTLIST filesystemname
  mountname CDATA  #REQUIRED
  deviceid  CDATA  #REQUIRED>
D-1.16.7  domainmanagerconfiguration

<!-- domainmanagerconfiguration.4.1.dtd -->

<!ELEMENT domainmanagerconfiguration
 ( description?,
  domainmanagersoftpkg?
  , deploymentlayout?
  , services? )>

<!ATTLIST domainmanagerconfiguration
 id ID #REQUIRED
 name CDATA    #REQUIRED
 accardinality (single | multiple) "single"
 app_backwards_compatible (false | true) "false"
 app_deployment_data (false | true) "true"
 app_installable (false | true) "true"
 app_releasable (false | true) "true"
 corba_provider (false | true) "true"
 channel_extension (false | true) "false"
 event_channel (false | true) "false"
 log_producer (false | true) "false"
 nested_deployment (false | true) "false"
 oe_profile (lightweight | medium | full) "medium" >

<!ELEMENT description (#PCDATA)>  
<!ELEMENT domainmanagersoftpkg
 ( localfile
  , componentproperties? )>

<!ELEMENT simpleref EMPTY>
<!ATTLIST simpleref
 refid CDATA    #REQUIRED
 value CDATA    #REQUIRED>

<!ELEMENT simplesequenceref
 ( values )>
<!ATTLIST simplesequenceref
 refid CDATA    #REQUIRED>

<!ELEMENT structref
 ( simpleref+ )>
<!ATTLIST structref
 refid CDATA    #REQUIRED>

<!ELEMENT structsequenceref
 ( structvalue+ )>
<!ATTLIST structsequenceref
 refid CDATA    #REQUIRED>
<!ELEMENT structvalue     
    ( simpleref+ )>
<!ELEMENT values         
    ( value+ )>
<!ELEMENT value (#PCDATA)>  
<!ELEMENT componentproperties>
    ( simpleref | simplesequenceref | structref | structsequenceref )+ >
<!ELEMENT localfile EMPTY>
<!ATTLIST localfile    
    name CDATA #REQUIRED>
<!ELEMENT deploymentlayout  
    ( localfile )>
<!ELEMENT services     
    ( service+ ) >
<!ELEMENT service       
    ( identifier , domainfinder)>
<!ELEMENT domainfinder EMPTY>
<!ATTLIST domainfinder    
    type (filesystem | eventchannel | application | servicename 
    | servicetype) #REQUIRED 
    name CDATA #IMPLIED>
<!ELEMENT identifier (#PCDATA)>
D-1.16.8  platformdeployment

<!-- platformdeployment.4.1.dtd -->

<!ELEMENT deploymentplatform (
    description?,
    channel+)
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT channel (devicelist?,
    servicelist?)>
<!ATTLIST channel name ID #REQUIRED>
<!ELEMENT devicelist (deviceref+)
>
<!ELEMENT deviceref EMPTY>
<!ATTLIST deviceref refid CDATA #REQUIRED>
<!ELEMENT servicelist (serviceref+)
>
<!ELEMENT serviceref EMPTY>
<!ATTLIST serviceref servicename CDATA #REQUIRED>
D-1.16.9 applicationdeployment

<!-- applicationdeployment.4.1.dtd -->

<!ELEMENT deploymentprecedence 
  ( description?, 
    deploymentoption+ 
  )>

<!ELEMENT description (#PCDATA)> 

<!ELEMENT deploymentoption 
  ( description?, 
    channelref+ 
  )>

<!ATTLIST deploymentoption 
  deployedname CDATA #REQUIRED>

<!ELEMENT channelref EMPTY> 

<!ATTLIST channelref 
  refname CDATA #REQUIRED>