

SCA Developer's Guide

APPENDIX E

XML for a Sample Device

03/28/2002

Introduction

This appendix provides the XML for a fictitious device. The device serves no real purpose, other than to show the inter-relation between the various profile files. Each profile contained in this document is preceded by a brief discussion explaining what it contains and how its contents are organized. The explanations of this appendix assume familiarity with the contents of Appendix A, which explains some of the basic concepts of XML.

Due to the variability of the DTDs, there are numerous methods of defining a device. For example, in the software package descriptor, there can exist one-to-many author elements. However, within the author element, there can exist zero-to-many sub-elements (name, company, webpage). So, the sample XML provided here will not stray far from the elements that are shown as mandatory in the respective DTDs. The commentary in this appendix applies only to the options demonstrated in the example XML. No attempt is made to cover the various other options available, but using the other capabilities does follow the same patterns demonstrated in these examples.

Line numbers are provided for each XML file listed in this appendix. Line numbers are not part of XML structure, but they are included in this appendix as a means of readily identifying and/or locating the various components of the file.

Properties File - device_xyz.prf.xml

A properties file describes component attributes.

The purpose of this file is to specify one “readonly” property – the available capacity. This property is of type “allocation”, and is used to indicate whether the device is available.

The first two lines of this file are the prolog, which is explained on page A-6 of Appendix A. DTD file *properties.2.2.dtd* embodies the Properties Descriptor requirements (specified in section D.4 of Appendix D to the SCA 2.2 Specification).

Lines 4 and 21 provide the start tag and corresponding end tag that enclose the root element for the XML file. The following child elements are contained within the root element:

?? **description** {lines 6-8} is a character string of information about the file

?? A "simple" element is one way of defining a component attribute¹. *device_xyz.prf.xml* contains one simple element {lines 10-19 define "Available Capacity"}
Within the simple start tag, element attributes may be defined in any order.

- * **mode** {line 11} may be "readonly", "writeonly", or "readwrite".

- * **id** {line 12} is the formal identifier for the component attribute. Since this attribute is used in allocation {see line 18} the **id** is a DCE UUID.

- * **type** {line 13} denotes how the value is handled by the software.
A type of "boolean" means that the value must be either 0 or 1; in this setting, these values are interpreted as meaning "in use" or "available", respectively.

- * **name** {line 14} may be used, by a GUI for example, to identify the component attribute in a more user-friendly manner

A "simple" element may have child elements.

- * **description** {line 16} is a character string of information about the component attribute

- * **value** {line 17} provides an initial value for the component attribute;
as is noted above in the explanation of "type", the initial value of '1' means that the device is available initially.

- * **kind** {line 18} denotes how the component attribute is used;
"allocation" is the normal use of an attribute for a device component.

¹ If it is not "simple", an attribute is "simplesequence", "test", "struct", or "structsequence", but this device happens to use only "simple".

```
1 <?xml version="1.0" ?>
2 <!DOCTYPE properties SYSTEM "properties.2.2.dtd">
3
4 <properties>
5
6   <description>
7     This is the property file (.prf) called out by the XYZ device's SPD (device_xyz.spd.xml)
8   </description>
9
10  <simple
11    mode="readonly"
12    id="DCE:F364A632-5F0E-11d4-8164-00508B6A52E6 "
13    type="boolean"
14    name="Available Capacity">
15
16    <description>This property indicates the available capacity for this device</description>
17    <value>1</value>
18    <kind kindtype="allocation" />
19  </simple>
20
21 </properties>
```

Software Component Descriptor for Device - `device_xyz.scd.xml`

The software component descriptor describes the interfaces associated with an SCA-compliant component.

Each interface is uniquely identified by using a CORBA repository identifier, which consists of three parts: the prefix "IDL", a scoped type name, and a version number. Thus, for example, the CF::Device interface defined by version 2.2 of the SCA is identified as "IDL:CF/Device:2.2".

The first two lines of this file are the prolog, which is explained on page A-6 of Appendix A. DTD file *softwarecomponent.2.2.dtd* embodies the Software Component Descriptor requirements (specified in section D.5 of Appendix D to the SCA 2.2 Specification).

Lines 4 and 63 provide the start tag and corresponding end tag that enclose the root element for the XML file. The following child elements are contained within the root element:

?? **corbaversion** {line 6} indicates the version of CORBA that the delivered component supports.

?? **componentrepid** {line 8} identifies which interface the component implements. A device must implement CF::Device, an application must implement CF::Resource, and an application factory must implement CF::ResourceFactory. Thus, the *repid* for this element always identifies CF::Device, CF::Resource, CF::ResourceFactory, or an interface which inherits one of these interfaces², even though the component may implement other interfaces also.

?? **componenttype** {line 10} for the fictitious device **device_xyz** is "device". Other valid values are "resource", "resourcefactory", "domainmanager", "log", "filesystem", "filemanager", "devicemanager", "eventservice", and "namingservice".

?? **componentfeatures** {lines 12-42} has a child element named **ports** {lines 14-40}, which has a "provides" child element for each provides port and a "uses" child element for each uses port.

Each **provides** port {lines 16-20 and 22-26} has two element attributes and one optional child element

repid is the name that uniquely identifies the interface realized at that port (it should match the *repid* of the corresponding **interfaces** element - see lines 44-61)

providesname is the port identifier (it should match some **providesidentifier** child element of a **connectinterface** element in the appropriate SAD file).

² For this device, **componentrepid** identifies CF::ExecutableDevice, which inherits from CF::Device.

Optional child element **porttype** can be "control", "data", "responses" or "test".

Each **uses** port {lines 28-32 and 34-38} has two element attributes and one optional child element

repid is the name that uniquely identifies the interface realized at that port (it should match the *repid* of the corresponding **interfaces** element - see lines 44-61)

usesname is the port identifier (it should match some **usesidentifier** child element of a **connectinterface** element in the appropriate SAD file).

Optional child element **porttype** can be "data", "control", "responses" or "test".

For example, the *repid* on line 17 matches the *repid* on line 47.

For both **provides** and **uses** ports, **porttype** defaults to "control" if the element is omitted. In general, **porttype** corresponds to interface types specified by the API Supplement³ - "control" corresponds to an API "B" (non-real-time) interface and "data" corresponds to an API "A" (real-time) interface.

?? **interfaces** {lines 44-61} has an **interface** child element for each interface that the component provides, uses, or supports. *repid* is the name that uniquely identifies the interface. *name* provides a character-based non-qualified (i.e., "user-friendly") way of identifying the interface.

³ see Section 3 of the SCA Developer's Guide

```
1 <?xml version="1.0" ?>
2 <!DOCTYPE softwarecomponent SYSTEM "softwarecomponent.2.2.dtd">
3
4 <softwarecomponent>
5
6     <corbaversion>2.2</corbaversion>
7
8     <componentrepid repid="IDL:CF/ExecutableDevice:2.2" />
9
10    <componenttype>device</componenttype>
11
12    <componentfeatures>
13
14        <ports>
15
16            <provides
17                repid="IDL:CF/ExecutableDevice:2.2"
18                providesname="device_in_port" />
19                <porttype type="control"/>
20            </provides>
21
22            <provides
23                repid="IDL:IOAPI/UserProvider_AnalogAudio:2.2"
24                providesname="audio_in_port" />
25                <porttype type="data"/>
26            </provides>
27
28            <uses
29                repid="IDL:IOAPI/UserProvider_AnalogAudio:2.2"
30                providesname="audio_out_port" />
31                <porttype type="data"/>
32            </uses>
33
34            <uses
35                repid="IDL:LogService/Log:2.2"
36                usesname="log_out_port" />
37                <porttype type="data"/>
38            </uses>
39
40        </ports>
41
42    </componentfeatures>
43
44    <interfaces>
45
46        <interface
47            repid="IDL:CF/ExecutableDevice:2.2"
48            name="ExecutableDevice">
49        </interface>
50
51        <interface
52            repid="IDL:LogService/Log:2.2"
53            name="Log">
```

```
54     </interface>
55
56     <interface
57         repid=" IDL:IOAPI/UserProvider_AnalogAudio:2.2"
58         name=" AudioUserProvider">
59     </interface
60
61 </interfaces>
62
63 </softwarecomponent>
```

Software Package Descriptor for Device - `device_xyz.spd.xml`

This software package descriptor is used to load an SCA-compliant component.

The first two lines of this file are the prolog, which is explained on page A-6 of Appendix A. DTD file *softpkg.2.2.dtd* embodies the Software Package Descriptor requirements (specified in section D.2 of Appendix D to the SCA 2.2 Specification).

Lines 4 and 53 provide the start tag and corresponding end tag that enclose the root element for the XML file. The *id* element attribute {line 5} is a DCE UUID that uniquely identifies the component, and is used elsewhere in the system to locate this particular component. The *name* element attribute {line 6} is a user-friendly means of identifying the component, and the *type* element attribute {line 7} specifies that the device does meet SCA requirements.

The following child elements are contained within the root element:

- ?? **author** {lines 9-11} is used to provide information about the person(s) who wrote the XML; no requirements limit what can be entered in the child elements of **author**.
- ?? **description** {line 13} is a character string of information about the file.
- ?? **propertyfile** {lines 15-17} specifies which particular property file should be used with this device; line 16 specifies that the file presented on pages E-2 and E-3 of this appendix should be used with the implementation described in this file.
- ?? **descriptor** {lines 19-21} specifies which Software Component Descriptor provides interface information for the component being loaded; line 20 specifies that the file presented on pages E-4 through E-7 of the appendix provides interface for the component loaded through the use of this file.
- ?? **implementation** {lines 23-51} describes a particular implementation of the device. The *id* element attribute {line 24} is a DCE UUID that uniquely identifies the particular implementation of the component, and is used elsewhere in the system to locate this implementation. The *aepcompliance* element attribute {line 25} states that the software for this implementation does meet the requirements specified in Appendix B to the SCA 2.2 Specification.⁴

The **description** child element {lines 27-29} is a character string of information about the implementation.

The **code** child element {lines 31-35} describes the instructions that provide the functionality of this implementation. The *type* element attribute {line 32} can be "Executable" (a main process to be loaded and executed), "KernelModule" (load only),

⁴ Software which meets the requirements specified in "Appendix B: SCA Application Environment Profile" attain a desired level of portability.

"SharedLibrary" (dynamically linked) or "Driver" (load only). Child element **loadfile** {line 29} specifies the file containing the actual code, and **entrypoint** {line 30} specifies where execution should begin (if that information is needed in the context in which the code will be used⁵).

A **dependency** child element describes other resources that are needed by this device. If either of these dependencies cannot be satisfied, the rest of the device load will not be carried out.

The first **dependency** {lines 37-42} refers to the system resource "deployment MIPS"; the name implies that it is processing power, but it could be anything. The *propertyref* child element references a simple allocation property, such as the one described on lines 10-19 of file **device_xyz.prf.xml** on page E-3 of this Appendix. The property is identified by matching the DCE UUID specified by *refid* {line 40}, and *value* {line 41} reports how many units of the property are needed. This **dependency** will cause the specified resource to be allocated to this device.

The second **dependency** {lines 44-49} refers to additional software. The *localfile name* {line 47} within the *softpkgref* references another software package descriptor (in this case, it is the file presented on pages E-11 and E-12 of this appendix) that will describe the software. This **dependency** will cause the specified software to be loaded.

⁵ For example, if the operating system is LynxOS, the entrypoint is understood to be 'main', so this information need not be provided in an SPD that will be used with LynxOS.

```
1 <?xml version="1.0" ?>
2 <!DOCTYPE softpkg SYSTEM "softpkg.2.2.dtd">
3
4 <softpkg
5   id="DCE:F364A623-5F0E-11d4-8164-00508B6A52E6"
6   name="Logical XYZ Device"
7   type="sca_compliant">
8
9   <author>
10     <company>XYZ Company</company>
11   </author>
12
13   <description>Software Package for the logical XYZ Audio device</description>
14
15   <propertyfile>
16     <localfile name=" device_xyz.prf.xml " />
17   </propertyfile>
18
19   <descriptor>
20     <localfile name=" device_xyz.scd.xml " />
21   </descriptor>
22
23   <implementation
24     id="DCE:F364A636-5F0E-11d4-8164-00508B6A52E6 "
25     aepcompliance="aep_compliant">
26
27     <description>This is an implementation for a logical component on a xyz processor using xyz
28       operating system
29     </description>
30
31     <code
32       type="Executable">
33       <localfile name="xyz_device.out" />
34       <entrypoint>xyzDeviceServer</entrypoint>
35     </code>
36
37     <dependency
38       type="deployment MIPS">
39       <propertyref
40         refid="DCE:F364A630-5F0E-11d4-8164-00508B6A52E6 "
41         value="50" />
42     </dependency>
43
44     <dependency
45       type="software dependency">
46       <softpkgref>
47         <localfile name=" device_xyz_dep1.spd.xml " />
48       </softpkgref>
49     </dependency>
50
51   </implementation>
52 </softpkg>
```

Software Package Descriptor for Dependency - device_xyz_dep1.spd.xml

This software package descriptor is used to load a component that is required by the component that was presented on pages E-8 through E-10 of this appendix.

The first two lines of this file are the prolog, which is explained on page A-6 of Appendix A. DTD file *softpkg.2.2.dtd* embodies the Software Package Descriptor requirements (specified in section D.2 of Appendix D to the SCA 2.2 Specification).

Lines 4 and 32 provide the start tag and corresponding end tag that enclose the root element for the XML file. The *id* element attribute {line 5} is a DCE UUID that uniquely identifies the component. The *name* element attribute {line 6} is a user-friendly means of identifying the component, and the *type* element attribute {line 7} specifies that the software does not meet SCA requirements.

The following child elements are contained within the root element:

?? **title** {line 9} provides a user-friendly way of identifying the interface.

?? **author** {lines 11-13} is used to provide information about the person(s) who wrote the XML; no requirements limit what can be entered in the child elements of **author**.

?? **implementation** {lines 15-30} describes a particular implementation of the software. The *id* element attribute {line 16} is a DCE UUID that uniquely identifies the implementation of the software. The *aeptcompliance* element attribute {line 17} states that the software for this implementation does meet the requirements specified in Appendix B to the SCA 2.2 Specification.⁶

The **code** child element {lines 19-22} describes the instructions that provide the functionality of this implementation. The *type* element attribute {line 20} can be "Executable" (a main process to be loaded and executed), "KernelModule" (load only), "SharedLibrary" (dynamically linked) or "Driver" (load only). Child element **localfile** {line 21} specifies which file contains the actual code..

The **dependency** child element {lines 24-28} refers to the system resource "deployment MIPS"; the name implies that it is processing power, but it could be anything.

The *propertyref* child element references a simple allocation property, such as the one described on lines 10-19 of file **device_xyz.prf.xml** on page E-3 of this Appendix. The property is identified by matching the DCE UUID specified by *refid* {line 26}, and *value* {line 27} reports how many units of the property are needed. . This **dependency** will cause the specified resource to be allocated to this software. The software will not be loaded if the dependencies cannot be satisfied.

⁶ Software which meets the requirements specified in "Appendix B: SCA Application Environment Profile" attain a desired level of portability.

```
1 <?xml version="1.0" ?>
2 <!DOCTYPE softpkg SYSTEM "softpkg.2.2.dtd">
3
4 <softpkg
5   id="DCE:8BF71881-790B-11d4-816B-00508B6A52E6 "
6   name="XYZ Dependency File"
7   type="sca_non_compliant">
8
9   <title>Dependency Component</title>
10
11   <author>
12     <company>Devices-R-Us</company>
13   </author>
14
15   <implementation
16     id="DCE:8BF71882-790B-11d4-816B-00508B6A52E6 "
17     aepcompliance="aep_compliant">
18
19     <code
20       type="Driver">
21       <localfile name="xyz_dependency1.o" />
22     </code>
23
24     <dependency type=" deployment MIPS">
25       <propertyref
26         refid="DCE:F364A630-5F0E-11d4-8164-00508B6A52E6 "
27         value="10" />
28     </dependency>
29
30   </implementation>
31 </softpkg>
```