



Solution Deployment Descriptor (SDD) Primer Version 1.0

Committee Draft

8 April 2008

Document URIs:

This Version:

<http://docs.oasis-open.org/sdd/v1.0/pr01/expository/sdd-primer-v1.0-cd01.doc>
<http://docs.oasis-open.org/sdd/v1.0/pr01/expository/sdd-primer-v1.0-cd01.pdf>
<http://docs.oasis-open.org/sdd/v1.0/pr01/expository/sdd-primer-v1.0-cd01.html>

Previous Version:

N/A

Latest Version:

<http://docs.oasis-open.org/sdd/v1.0/sdd-primer-v1.0.doc>
<http://docs.oasis-open.org/sdd/v1.0/sdd-primer-v1.0.pdf>
<http://docs.oasis-open.org/sdd/v1.0/sdd-primer-v1.0.html>

Technical Committee:

OASIS Solution Deployment Descriptor (SDD) TC

Chair(s):

Brent A. Miller, IBM Corp.

Editor(s):

Brent A. Miller, IBM Corp.

Related work:

This expository document replaces or supersedes:

- None

This expository document is related to:

- Solution Deployment Descriptor (SDD) Specification

Declared XML Namespace(s):

None

Abstract:

This expository document provides non-normative information to supplement the Solution Deployment Descriptor (SDD) specification and serves as a “getting started” guide.

Status:

This document was last revised or approved by the Solution Deployment Descriptor (SDD) Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/sdd/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/sdd/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/sdd/>.

Notices

Copyright © OASIS® 2007, 2008. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1	Introduction.....	5
1.1	Terminology	5
1.2	General Document Conventions	5
1.3	Normative References	5
1.4	Non-Normative References	5
2	Why Use SDD?	6
3	Getting Started	8
4	Examples.....	10
4.1	General Deployment Model.....	10
4.2	Simple JRE Example	11
4.2.1	PackageDescriptor	11
4.2.2	DeploymentDescriptor	12
4.3	Composite Application Example.....	15
4.3.1	Referenced PackageDescriptors and DeploymentDescriptors.....	15
4.3.2	Composite PackageDescriptor	25
4.3.3	Composite DeploymentDescriptor.....	25
5	Additional Considerations.....	30
6	Conclusion.....	32
A.	Complete Examples	33
B.	Acknowledgements	34

1 Introduction

This is a non-normative, expository document that supplements the *Solution Deployment Descriptor* (SDD) specification. This document provides a quick way to get started with the pragmatics of the SDD, but it does not replace the specification or schema, which need to be understood.

This Primer describes why and how to use SDD, making use of examples produced by the OASIS SDD technical committee. This initial version concentrates on the use of SDD for installation; in the future, new versions or additional modules may be created to describe other uses of SDD, such as configuration and localization.

1.1 Terminology

Terminology in this document is consistent with the SDD Specification **[SDDSP]** (see section 1.3).

1.2 General Document Conventions

This document contains cross-references. Such references appear as the referenced section number inside square brackets, for example, [4.5]. In electronic versions of this specification, the cross-references can act as links to the target section.

Within the XML snippets (excerpts from SDD examples), schema elements and attributes that serve as ids and references are highlighted in blue underscored text. Schema element and attribute values that are expected to be found in SDD profiles are identified with *red italic* text in the XML snippets.

1.3 Normative References

- | | |
|---------------------|---|
| [SDD] | OASIS, Solution Deployment Descriptor Specification v1.0, http://docs.oasis-open.org/sdd/v1.0/pr01/sdd-spec-v1.0-pr01-r01.doc . |
| [SDD_Schema] | OASIS, Solution Deployment Descriptor Specification v1.0, Full Schema, http://docs.oasis-open.org/sdd/v1.0/pr01/FullSchema/ . |

1.4 Non-Normative References

- | | |
|----------------|---|
| [SDDSP] | Solution Deployment Descriptor Starter Profile, http://docs.oasis-open.org/sdd/v1.0/pr01/expository/sdd-starter-profile-v1.0-cd01.zip . |
| [SDDEX] | Solution Deployment Descriptor Examples, http://docs.oasis-open.org/sdd/v1.0/pr01/expository/sdd-examples-v1.0-cd01.zip . |
| [SDDBP] | Solution Deployment Descriptor Best Practices wiki. |

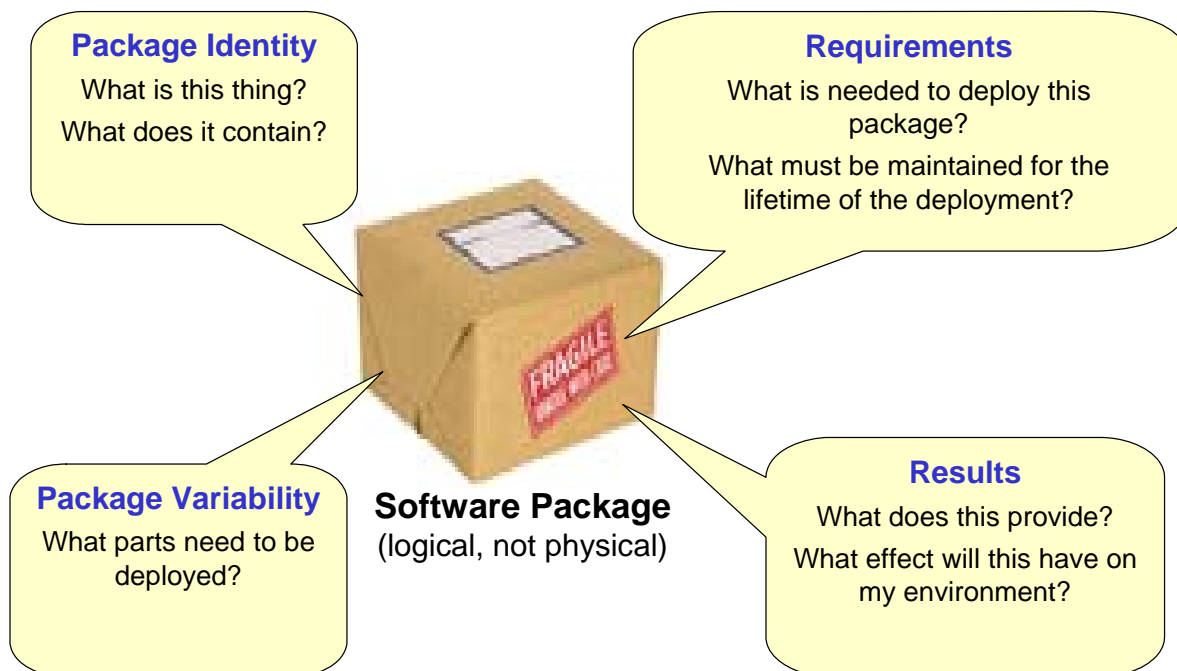
29

2 Why Use SDD?

30
31
32

Solution Deployment Descriptor [SDD] provides a standardized way to declare and externalize information about your software package and its deployment, as illustrated in Figure 1, with the explanation that follows.

Basic Information in a Solution Deployment Descriptor



33

34 **Figure 1: Overview of information provided in an SDD**

35
36
37
38
39

Consider some sort of software package as illustrated by the box in Figure 1. To successfully deploy this software, information about it is required. This information is analogous to what might be provided on the box for software purchased on a CD—it tells you what you need to know about this software. The SDD provides a way to describe such software in a standardized form that can be electronically recorded and programmatically processed, in addition to providing information useful to humans.

40
41
42
43
44
45
46
47
48
49

- **Package identity**: this portion of the SDD describes the name and source of the software and its logical package structure, including the content (executable files, license agreements, documentation and so on) that makes up the software.
- **Requirements**: this portion of the SDD describes what is necessary for the software to be successfully deployed, including requirements for disk space, CPU capacity, and other declarations about the required state of the deployment environment (for example, other pre-requisite software, configuration settings and so on).
- **Package variability**: this portion of the SDD describes options for deployment. Some parts of the software might not be used in every deployment. For example, the software package might contain software variations for two different operating systems; the condition of the deployment environment

50 (that is, the actual operating system type) determines which of the variations is used in a particular
51 deployment. The software package also might contain optional features that the installer chooses at
52 installation time. Also, different languages might be deployed for different installations.

- 53 • **Results:** this portion of the SDD describes what will happen once the software is deployed (for
54 example, installation of new software, update of existing software, configuration or localization of
55 existing software, and so on) and what effects it will have on the environment once it is deployed (for
56 example, new applications are created or existing applications are updated).

57 All of this information enables deployers to analyze and make pre-deployment decisions. SDD producers
58 can be developers, aggregators, service and maintenance support staff and others. SDD consumers can
59 include humans and tools that perform composition, perform pre-deployment planning, make pre-
60 deployment decisions and/or perform deployment operations.

61 The SDD can be used across a spectrum from simple packages to complex solutions. It enables lifecycle
62 management of software (installation, configuration, localization, fix application, update/upgrade and
63 uninstallation) to be more (if not fully) automated.

64 The standard representation provided by SDD enables solutions to be easily composed from existing
65 components. SDD does not require that you discard, replace or rewrite all of your installers; SDD provides
66 declarative metadata for software that can continue to be installed by existing deployment software.

3 Getting Started

67

68 SDD is represented as an XML schema [**SDD_Schema**]. The SDD specification [**SDD**] provides
69 semantics and other normative and non-normative information.

70 An SDD consists of two major parts: the *Package Descriptor* and the *Deployment Descriptor*. As noted in
71 the specification:

72 *The package descriptor defines package content which includes artifacts whose processing*
73 *results in deployment of the software package. The deployment descriptor defines metadata*
74 *associated with those artifacts. The SDD package descriptor defines the package identity, the*
75 *package content, and various other attributes of the package. Each SDD consists of exactly one*
76 *deployment descriptor and one package descriptor. The deployment descriptor is where the*
77 *topology, selectability, inputs, requirements, and conditions of the deployment are described.*

78 You describe your software by declaring things about it using XML. Referring back to Figure 1, the
79 information you can declare for each part illustrated there includes:

- 80 • **Package Identity** is part of the *PackageDescriptor*, and includes primarily the *PackageIdentity* and
81 *Contents* elements. *PackageIdentity* contains other elements and attributes that name and describe
82 the software package (including items such as version, human-readable descriptions and other
83 identifying information). *Contents* includes the “files” that make up the package, including their
84 purpose, where they can be found and other information, such as optional digital signatures.
- 85 • **Requirements** are contained in the *DeploymentDescriptor*, within the various types of content
86 elements (*InstallableUnit*, *ConfigurationUnit*, *LocalizationUnit*, *CompositeUnit* and
87 *CompositeInstallable*). The *Requirements* element describes resource constraints and dependencies
88 for internal content elements that must be met, including versions, relationships, property values,
89 capacity constraints and consumption constraints. Depending on the type of content element, the
90 presence of required base software might also be specified. The *Requisites* element allows the SDD
91 to incorporate other software that can help to satisfy some of these requirements if the deployment
92 environment doesn’t already satisfy them.
- 93 • **Package variability** can be accomplished within the *DeploymentDescriptor* with the
94 *SelectableContent* element that enables *Features* and *Groups* to be selected, as well as with the
95 *Condition* construct that can be applied to multiple elements in the SDD so that certain items can be
96 “conditioned” in or out of scope for a particular deployment.
- 97 • **Results** are contained in the *DeploymentDescriptor*, within the various types of content elements.
98 Depending on the type of content element, the *ResultingResource* or *ResultingChange* elements
99 describe the results of the deployment (what happens to the deployment environment as a result of
100 performing the deployment described by the SDD).

101 In addition to the concepts illustrated in Figure 1, other items can be described in the SDD:

- 102 • **Topology:** The logical topology of the solution can be expressed in an SDD. The *Topology* element
103 describes all of the resources relevant for deployment, including resources that are required, created
104 or modified during deployment, as well as the relationships among these resources.
- 105 • **Artifacts:** These content files accomplish the deployment operations. Artifact files (for example, ZIP
106 files, RPM files or executable installation files) are processed during deployment to install, configure,
107 localize or perform other deployment operations. The *Artifact* element describes artifacts, along with
108 the inputs and outputs, including substitution values, used when processing those artifacts.
- 109 • **Atomic & composite content units:** Atomic content units define artifacts (just described); the three
110 atomic content unit elements are *InstallableUnit*, *ConfigurationUnit* and *LocalizationUnit* (this version
111 of the Primer focuses primarily on *InstallableUnits*). Atomic units suffice for many simple
112 deployments; when more complex solutions are deployed (for example, when multiple SDDs are
113 aggregated or when package variability exists in the form of selectable content), composite content
114 units are used. The three types of composite content units are *CompositeInstallable*, *CompositeUnit*

115 and *CompositeLocalizationUnit*. This version of the Primer focuses primarily on
116 *CompositeInstallables*).

- 117 • **Variables and parameters:** Variables provide a way to obtain and derive values from resource
118 properties and the deployment environment and human deployers. The variables can then be used in
119 various portions of the SDD to influence the deployment process, including the use of variables as
120 input arguments to artifacts, and values for resource constraints.
- 121 • **Operations:** Operations serve as the “verbs” for deployment described by an SDD. Operations
122 describe the deployment steps that are performed; some of the operations are create, update and
123 uninstall. This version of the Primer focuses primarily on the create operation that is associated with
124 installing software.
- 125 • **Display information:** Although the SDD enables programmatic processing and automated
126 deployment, it is also important to include descriptive information that humans can use. Many
127 deployment operations are interactive, requiring humans to make selections, confirm operations,
128 provide input, and so on. Many elements throughout the SDD support display information to provide
129 human-understandable descriptions; these display elements are translatable to support multiple
130 languages.

131 The remainder of this document uses examples to illustrate each of these aspects of the SDD and
132 addresses additional considerations for producing and consuming useful SDDs. This version of the
133 Primer focuses on concepts and practices for SDDs that perform installation; future publications may
134 expand this focus with examples and considerations for other SDD operations such as configuration and
135 localization (these latter aspects are introduced but addressed in less detail than installation).

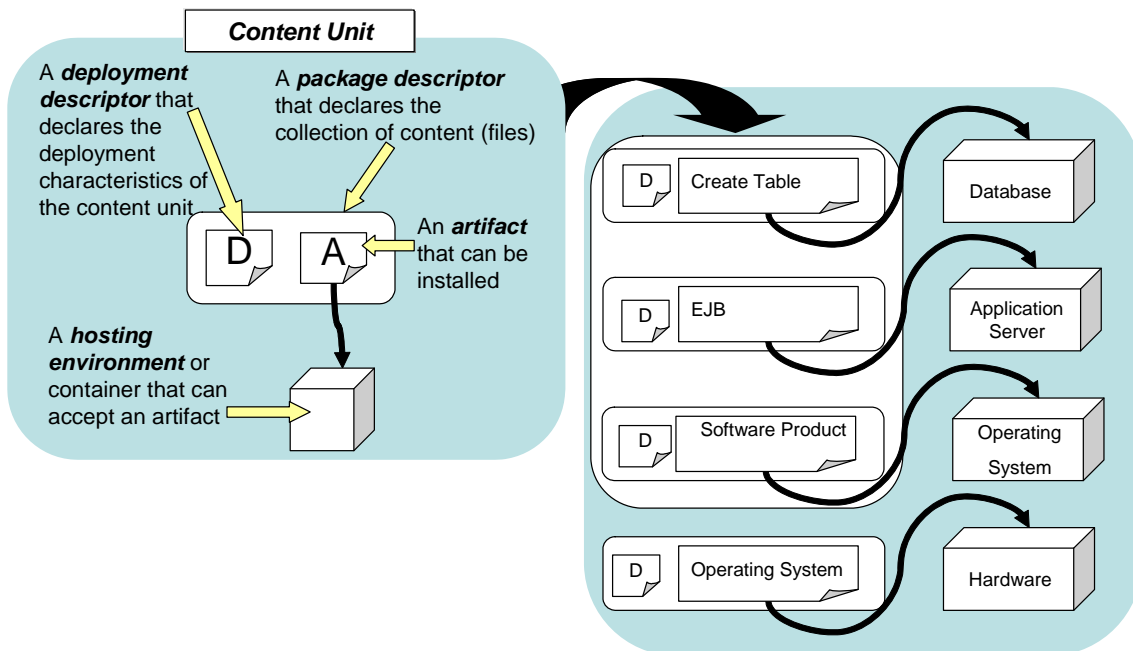
136 4 Examples

137 To further illustrate how the SDD can be used, we refer to the examples published by the SDD TC
138 [SDDEX] and describe those examples, along with information pertinent for SDD authors and consumers,
139 in the following sections.

140 4.1 General Deployment Model

141 Before examining the details of the examples, it is useful to understand the general deployment model
142 that underlies SDD; we call this the *installable unit/hosting environment design pattern*. This model is
143 illustrated in Figure 2 and described next.

Installable Unit / Hosting Environment Design Pattern



144

145 **Figure 2: Installable unit/hosting environment design pattern**

146 In general, things that are deployed can fit into the installable unit/hosting environment design pattern.
147 That is, as shown in the left side of Figure 2, an installable unit containing one or more artifacts can be
148 installed into a hosting environment. The figure also shows the other major portions of an SDD, namely a
149 descriptor for the installable unit and a package with the associated content.

150 As shown on the right side of Figure 2, this design pattern can be used at all levels of the resource stack.
151 For example, and operating system installable unit can be installed into a hardware hosting environment;
152 a software product (such as a native application) can be installed into an operating system hosting
153 environment, and so on.

154 This design pattern makes it possible to standardize many aspects of software that coordinates
155 deployment of diverse resource types, and it underlies the standardization of the declarative metadata for

156 the installable unit, including artifacts, package descriptors and deployment descriptors, embodied in the
157 SDD standard.
158 Understanding this design pattern facilitates understanding the examples, which are explained next.

159 4.2 Simple JRE Example

160 This example is a simple one that illustrates a basic *PackageDescriptor* and corresponding
161 *DeploymentDescriptor* for a Java™¹ runtime environment. For the complete example, refer to the files
162 `examples/CompositeApp/pkgs/JRE/jre.xml` and `examples/CompositeApp/pkgs/JRE/jre_pkg.xml` in
163 **[SDDEX]** (see Appendix [A]).

164 4.2.1 PackageDescriptor

- 165 • This simple *PackageDescriptor* contains *Identity* information, including *Description*, *ShortDescription*,
166 *Version*, *BuildInformation* and *Manufacturer* information. All of this information is provided by the
167 package author. The *Identity* information excerpt from this example follows.

```
168 <sdd-pd:PackageIdentity softwareID="2000-123" packageType="update">  
169   <sdd-common:Description translationKey="DESC">Lets you develop and deploy  
170   Java(TM) applications on desktops and servers, as well as today's Embedded and  
171   Realtime environments. Java SE includes classes that support the development  
172   of Java Web Services and provides the foundation for Java Platform, Enterprise  
173   Edition (Java EE).</sdd-common:Description>  
174   <sdd-common:ShortDescription translationKey="SHORT_DESC">Java(TM) Platform  
175   for software development</sdd-common:ShortDescription>  
176   <sdd-common:Name translationKey="PKG_DISPNAME">Java(TM) Platform, Standard  
177   Edition Runtime Environment</sdd-common:Name>  
178   <sdd-common:Version>1.5.0</sdd-common:Version>  
179   <sdd-common:BuildInformation buildID="09122006-193602" />  
180   <sdd-common:Manufacturer>  
181     <sdd-common:Name>Sun Microsystems, Inc.</sdd-common:Name>  
182     <sdd-common:Location>  
183       <sdd-common:Address>901 San Antonio Rd., Palo Alto, CA 94025</sdd-  
184   common:Address>  
185       <sdd-common:Country>U.S.A.</sdd-common:Country>  
186     </sdd-common:Location>  
187     <sdd-common:ContactInformation>support@sun.com</sdd-  
188   common:ContactInformation>  
189   </sdd-common:Manufacturer>  
190 </sdd-pd:PackageIdentity>
```

- 191 • The *Contents* in this package consist of the artifact, with the assigned *id* "JRE_RPM"; a language
192 bundled with the assigned *id* "EN_Bundle" and the corresponding *DeploymentDescriptor* (described
193 next), with the assigned *id* "DD". These ids are chosen by the package author and are used to
194 reference these content files from the deployment descriptor. These and other ids are highlighted in
195 [blue underscored](#) text in the following excerpt; their use as references is described in the next
196 section.

¹ Java is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

```

197 <sdd-pd:Contents>
198   <sdd-pd:Content pathname="jre-1_5_0_09-AIX-rpm.bin" id="JRE_RPM"
199   purpose="content" />
200   <sdd-pd:Content pathname="jre_update.xml" id="DD"
201   purpose="deploymentDescriptor" />
202   <sdd-pd:Content pathname="com/sun/java/jre_en.properties" id="EN_Bundle"
203   purpose="languageBundle" />
204 </sdd-pd:Contents>

```

205 4.2.2 DeploymentDescriptor

206 This simple *DeploymentDescriptor* is used with the preceding *PackageDescriptor*. It contains information
207 relevant to deploying the JRE that is described in that package descriptor.

- 208 • This simple *DeploymentDescriptor* illustrates the declaration of *Topology* and *Identity* for this
209 deployment, as well as the use of *Variables* during deployment operations (in this case, install) and
210 the declaration of *Requirements* (in this case, *ResourceConstraints*) necessary for deployment. It also
211 identifies the *Artifact* used during deployment.
- 212 • Several values in this *DeploymentDescriptor* are taken from a profile. These values are identified with
213 *red italic* text in the example excerpts included here. See [SDDSP] for the Starter Profile that
214 illustrates the definition of values such as these.

215 In the *Topology* section, the SDD author declares a resource of the type *CIM_OperatingSystem*; this
216 resource serves as a hosting environment (in the installable unit/hosting environment design pattern), and
217 its type comes from a profile. This operating system resource hosts a resource of the type
218 *CIM_FileSystem*; this file system serves as the actual hosting environment for the JRE that is deployed
219 with this SDD, and its type also comes from a profile. So, this SDD is declaring that the JRE, when
220 deployed, will be hosted by the file system (that in turn is hosted by an operating system). Moreover, it is
221 known that this JRE is destined for an AIX®² operating system, so a relevant property called *Root*, with
222 the value */usr*, is declared; this property declares the file system where the JRE will be installed. These
223 values also come from a profile that defines relevant properties for the AIX operating system. Finally, the
224 resource to be deployed (the JRE) is itself declared, with the type *CIM_InstalledProduct*, and that value
225 also comes from a profile. Note that each of these resources has an *id* (*os*, *UsrFilesys* and *JRE*,
226 respectively) that is used to refer to these resources elsewhere in the *DeploymentDescriptor*. The *os* and
227 *UsrFilesys* resources need to be defined because each has an associated *Requirement* defined later in
228 the *DeploymentDescriptor*³; the JRE resource needs to be defined because it is later referred to as a
229 resulting resource. These *Topology* elements are shown in the following example excerpt.

```

230 <sdd-dd:Topology>
231   <sdd-dd:Resource id="os" type="sp:CIM_OperatingSystem">
232     <sdd-dd:HostedResource id="UsrFilesys" type="sp:CIM_FileSystem">
233       <sdd-dd:Description>This is the /USR logical filesystem on AIX</sdd-
234 dd:Description>
235       <sdd-dd:Property>
236         <sdd-dd:PropertyName>Root</sdd-dd:PropertyName>

```

² AIX is a registered trademark of IBM in the United States, other countries, or both

³ In fact, the *os* and *UserFilesys* resources would need to be included in *Topology* even if *Requirements* were not defined for them, because of their relationships to another resource (in this case, the JRE) that is referred to elsewhere.

```

237     <sdd-dd:Value>/usr</sdd-dd:Value>
238   </sdd-dd:Property>
239 </sdd-dd:HostedResource>
240   <sdd-dd:HostedResource id="JRE" type="sp:CIM_InstalledProduct" />
241 </sdd-dd:Resource>
242 </sdd-dd:Topology>

```

- Because this *DeploymentDescriptor* describes the installation of the JRE, it declares an *InstallableUnit* whose *targetResourceRef* is the operating system that was previously declared with the *id* "os". This information tells the SDD consumer that the AIX operating system is capable of processing the JRE artifact file. In this example, the target resource is the operating system, whereas the hosting environment resource is the file system. In many cases, the hosting environment also serves as the target resource. These elements appear in the following excerpt.

```

249 <sdd-dd:InstallableUnit id="ID000026" targetResourceRef="os">
250 <sdd-dd:Identity softwareID="2000-123">

```

- This simple *DeploymentDescriptor* has a single *InstallableUnit*. That *InstallableUnit*'s *Identity* section happens to match the *Identity* information in the corresponding *PackageDescriptor*. However, the identity of the overall package (in the *PackageDescriptor*) and the identity of the content units (such as the *InstallableUnit* in this example) are distinct entities. In this case, the SDD author has determined that it is convenient for the *InstallableUnit Identity* and the overall package *Identity* to match; this enables identity information to be obtained from the *DeploymentDescriptor* without opening the *PackageDescriptor*. Note that these distinct identities need not (and often do not) match; see, for example, the composite application example in section [4.3], which has multiple *InstallableUnits*). The complete example contains this *Identity* information, which is similar to the *Identity* information in the *PackageDescriptor*.
- This *DeploymentDescriptor* defines a *Variable* of type *StringParameter* that represents the logging level to be used when the JRE is deployed. Valid and default values for this parameter (defined in a profile) are defined. This parameter's value will be set using a substitution (illustrated and described later), and the resulting value will be passed to the artifact when the artifact is invoked for processing. This *Variable* is shown in the following example excerpt.

```

266 <sdd-dd:Variables>
267   <sdd-dd:Parameters>
268     <sdd-dd:StringParameter sensitive="false" id="LoggingLevel"
269     defaultValue=".level=INFO">
270       <sdd-dd:Description>Default logging level for logging messages coming from
271       JRE</sdd-dd:Description>
272     <sdd-dd:ValidValue>FINEST</sdd-dd:ValidValue>
273     <sdd-dd:ValidValue>FINER</sdd-dd:ValidValue>
274     <sdd-dd:ValidValue>FINE</sdd-dd:ValidValue>
275     <sdd-dd:ValidValue>CONFIG</sdd-dd:ValidValue>
276     <sdd-dd:ValidValue>SEVERE</sdd-dd:ValidValue>
277     <sdd-dd:ValidValue>INFO</sdd-dd:ValidValue>
278     <sdd-dd:ValidValue>WARNING</sdd-dd:ValidValue>
279   </sdd-dd:StringParameter>
280 </sdd-dd:Parameters>
281 </sdd-dd:Variables>

```

- This *DeploymentDescriptor* defines *Requirements* that must be met for a successful deployment:
 - The first *Requirement* declares that the operating system type must be AIX and that the minimum version of the operating system is 5.1. This *Requirement* further declares that the "certified", or fully tested, versions for this JRE span AIX version 5.1 to 5.3. This expresses the SDD author's intent that the operating system type must be AIX and must be at least version 5.1; preferred versions are 5.1 to 5.3, so versions above 5.3 might be acceptable but probably should be tested or otherwise verified for this deployment. Note that some values for this *Requirement* come from a profile.
 - The second *Requirement* declares the amount of free disk space that must be available for a successful deployment. In this case, 2688 512-byte blocks (the unit of measure used for disk

292 space on this AIX operating system) must be available. Note that some values for this
293 *Requirement* come from a profile.

294 These *Requirements* appear in the following example excerpt.

```
295 <sdd-dd:Requirements>
296   <sdd-dd:Requirement id="OSLevel" operation="install use">
297     <sdd-dd:Description>This JRE requires an AIX Operating System at a version
298     of at least 5.1. Sun has certified all versions of AIX between 5.1 and
299     5.3</sdd-dd:Description>
300     <sdd-dd:ResourceConstraint id="osMinimumVersionRequirement"
301     resourceRef="os">
302       <sdd-dd:Description>This JRE requires a version of AIX or 5.1 or later.
303       It has been certified on versions of AIX between 5.1 and 5.3.</sdd-
304       dd:Description>
305       <sdd-dd:PropertyConstraint>
306         <sdd-dd:PropertyName>sp:CIM_OperatingSystem.OSType</sdd-
307         dd:PropertyName>
308         <sdd-dd:Value>AIX</sdd-dd:Value>
309       </sdd-dd:PropertyConstraint>
310       <sdd-dd:VersionConstraint>
311         <sdd-dd:Supported>
312           <sdd-dd:Range>
313             <sdd-dd:MinVersion>5.1</sdd-dd:MinVersion>
314           </sdd-dd:Range>
315         </sdd-dd:Supported>
316         <sdd-dd:Certified>
317           <sdd-dd:Range>
318             <sdd-dd:MinVersion>5.1</sdd-dd:MinVersion>
319             <sdd-dd:MaxVersion inclusive="true">5.3</sdd-dd:MaxVersion>
320           </sdd-dd:Range>
321         </sdd-dd:Certified>
322       </sdd-dd:VersionConstraint>
323     </sdd-dd:ResourceConstraint>
324   </sdd-dd:Requirement>
325   <sdd-dd:Requirement id="UsrDiskSpace" operation="install use">
326     <sdd-dd:Description>This JRE requires 2688 512-byte blocks of available
327     space on the /usr file system</sdd-dd:Description>
328     <sdd-dd:ResourceConstraint id="UsrDiskSpaceRequirement"
329     resourceRef="UsrFilesys">
330       <sdd-dd:ConsumptionConstraint>
331         <sdd-dd:PropertyName>sp:CIM_FileSystem.AvailableSpace</sdd-
332         dd:PropertyName>
333         <sdd-dd:Value unit="512-blocks">2688</sdd-dd:Value>
334       </sdd-dd:ConsumptionConstraint>
335     </sdd-dd:ResourceConstraint>
336   </sdd-dd:Requirement>
337 </sdd-dd:Requirements>
```

338 • The *DeploymentDescriptor* declares a *ResultingResource* that will result from successful deployment.
339 In this case, a new resource, an instance of the JRE version 1.5.0, will be created in the deployment
340 environment. This information is shown in the following excerpt.

```
341 <sdd-dd:ResultingResource resourceRef="JRE">
342   <sdd-dd:Description>An instance of Java(TM) Runtime Environment, Standard
343   Edition Version 5.0 is installed as a result of this deployment</sdd-
344   dd:Description>
345   <sdd-dd:Name>Java(TM) Runtime Environment, Standard Edition</sdd-dd:Name>
346   <sdd-dd:Version>1.5.0</sdd-dd:Version>
347 </sdd-dd:ResultingResource>
```

348 • Finally, the *DeploymentDescriptor* declares the *Artifact* that is processed to accomplish the
349 installation.

- 350 ○ This *InstallArtifact* element defines metadata about the artifact file ("JRE_RPM", as identified in the
351 *contentRef* attribute) that is used to create the JRE. This artifact is an RPM file (a typical type
352 used by installers in the AIX operating system), as identified by the *type* attribute.
- 353 ○ The argument value for the logging level in this example comes from a file, and the value in the
354 file requires a substitution before it can be used for artifact processing. The *contentRef* attribute
355 specifies the file in which the substitution will take place ("LoggingProperties"). For the
356 substitution, the *required* attribute indicates that the resulting value must be valid, and the *limit*
357 attribute specifies that only one substitution should occur (even if the *Pattern* occurs more than
358 once in the file). *Pattern* specifies the string in the file that will be replaced; *Value* specifies the
359 variable expression that is evaluated to determine the text that will replace (be literally substituted
360 for) *Pattern* in the file. In this example, "\${LoggingLevel}" indicates that the value of the
361 *LoggingLevel* variable (previously defined in the SDD and described earlier) is evaluated, and the
362 result is substituted for the string "INFO" in the file. The file (with the substituted value) is passed
363 with the artifact to the target resource for use when processing the artifact. The earlier declaration
364 of the parameter enables checking the value that is set against valid values (and also defines a
365 default value in case no value is set during the installation process). This information appears in
366 the following example excerpt.

```

367 <sdd-dd:Artifacts>
368   <sdd-dd:InstallArtifact type="rpm" contentRef="JRE_RPM">
369     <sdd-dd:AdditionalContent type="text" contentRef="LoggingProperties">
370       <sdd-dd:Substitution required="true" limit="1">
371         <sdd-dd:Pattern>INFO</sdd-dd:Pattern>
372         <sdd-dd:Value>${LoggingLevel}</sdd-dd:Value>
373       </sdd-dd:Substitution>
374     </sdd-dd:AdditionalContent>
375   </sdd-dd:InstallArtifact>
376 </sdd-dd:Artifacts>

```

377 4.3 Composite Application Example

378 In this section, a more complex example is provided. It is an aggregation of a set of other SDDs for the
379 deployment of the following packages:

- 380 • a 3-tier J2EE application with user interface, backend business logic and a database connection;
- 381 • an optional J2EE simple client (this client requires a JRE runtime with a minimum version);
- 382 • a JRE runtime of a version that satisfies the J2EE simple client requirement; and
- 383 • some optional German and French language packs for the J2EE simple client.

384 These constituent SDDs are referred to as *referenced packages* in the aggregating SDD. Note that the
385 individual constituent packages themselves are self-sufficient; that is, they are complete SDD packages
386 and may be used to deploy their respective content independently, in addition to being used to deploy the
387 aggregated solution that is described in this section.

388 A composite package descriptor-deployment descriptor pair combines these constituent SDDs into a
389 composite application SDD.

390 For the complete set of example files described throughout this section, refer to the
391 `examples/CompositeApp` folder in [SDDEX] (see Appendix [A]).

392 4.3.1 Referenced PackageDescriptors and DeploymentDescriptors

393 The complete solution consists of several SDDs. The Composite *PackageDescriptor* and
394 *DeploymentDescriptor* sections describe the components that constitute the aggregated SDD. In this
395 section, more details are provided for the individual referenced packages (the ones that are aggregated
396 by the composite SDD), including:

- 397 • the 3-tier J2EE application package;
- 398 • the Simple J2EE client;

- the JRE runtime package; and
- the German and French language packs.

4.3.1.1 J2EE Application Package

The J2EE application package (*SC_pkg*) has a *PackageDescriptor* located in `pkgs\Composite\SimpleCompositeApp_pkg.xml` and a *DeploymentDescriptor* located in `pkgs\Composite\SimpleCompositeApp.xml` in [SDDEX] (see Appendix [A]).

- The *PackageDescriptor* for the 3-tier J2EE application simply lists the identity of the package and the contents contained in the package. Note that the contents listed in the *Content* section are of different types; hence, their deployment processing will differ as described in the corresponding *DeploymentDescriptor* (for example, the J2EE client content has a different file type and format than the database content, so their respective installable unit processing varies).
- The *DeploymentDescriptor* first describes a *Topology* with nested resources for the components to be installed. There are three top level resources described with *Resource* elements. The first is *J2eeServletServer*, of type *CIM_J2eeServer*, hosting hosted resource *SimpleJ2eeServlet*, of type *CIM_J2eeServlet*. The second top-level resource is *appServer_os*, of type *CIM_OperatingSystem*, and it hosts resource *J2eeAppServer*, of type *CIM_J2eeServer*, which in turn hosts resource *SimpleJ2eeApp*, of type *CIM_J2eeApplication*. The third top-level resource is *os*, of type *CIM_OperatingSystem*, and it hosts resource *DatabaseServer*, of type *CIM_DatabaseServer*, which in turn hosts resource *SimpleDatabase*, of type *CIM_CommonDatabase*. Note that declaring *appServer_os* and *os* as separate resources does not imply that they are separate operating systems (or separate hardware hosts), but rather allows for distributed deployment—for example, *J2eeAppServer* and *DatabaseServer* could reside on separate operating system instances. If, instead, *J2eeAppServer* and *DatabaseServer* were both hosted by the same hosting resource of type *CIM_OperatingSystem*, then it would be implied that they both reside on the same operating system instance (and hardware host). As shown later in the deployment descriptor, the *J2eeServletServer*, *J2eeAppServer* and *DatabaseServer* are used in the *Requirements* sections to declare requirements for the deployment. The *SimpleJ2eeServlet*, *SimpleJ2eeApp* and *SimpleDatabase* are used to declare the resulting resources for this SDD, and the *J2eeAppServer*, *SimpleJ2eeApp*, *DatabaseServer* and *SimpleDatabase* are used to declare the resource relationships.

```

428 <sdd-dd:Topology>
429   <sdd-dd:Resource id="J2eeServletServer" type="sp:CIM_J2eeServer">
430     <sdd-dd:HostedResource id="SimpleJ2eeServlet" type="sp:CIM_J2eeServlet" />
431   </sdd-dd:Resource>
432   <sdd-dd:Resource id="appServer_os" type="sp:CIM_OperatingSystem">
433     <sdd-dd:HostedResource id="J2eeAppServer" type="sp:CIM_J2eeServer">
434       <sdd-dd:HostedResource id="SimpleJ2eeApp" type="sp:CIM_J2eeApplication"
435     />
436     </sdd-dd:HostedResource>
437   </sdd-dd:Resource>
438   <sdd-dd:Resource id="os" type="sp:CIM_OperatingSystem">
439     <sdd-dd:HostedResource id="DatabaseServer" type="sp:CIM_DatabaseSystem">
440       <sdd-dd:HostedResource id="SimpleDatabase" type="sp:CIM_CommonDatabase"
441     />
442     </sdd-dd:HostedResource>
443   </sdd-dd:Resource>
444 </sdd-dd:Topology>

```

- The *DeploymentDescriptor* defines the *InstallableUnits* associated with this component of the composite application and specifies their different deployment characteristics based on *Requirements*. The *InstallableUnits* are combined in *CompositeInstallable*. Note the *Parameters* element within the *Variables* element. It declares two input parameters that apply to the entire *CompositeInstallable*: *JDBC_User* and *JDBC_Password*.

```

450 <sdd-dd:CompositeInstallable id="CompApp01" operation="install">
451   <sdd-dd:Identity softwareID="5000-123">
452     ...

```



```

453 </sdd-dd:Identity>
454 <sdd-dd:Variables>
455   <sdd-dd:Parameters>
456     <sdd-dd:StringParameter id="JDBC_User" />
457     <sdd-dd:StringParameter id="JDBC_Password" sensitive="true" />
458   </sdd-dd:Parameters>
459 </sdd-dd:Variables>
460 <sdd-dd:BaseContent>
461   <sdd-dd:InstallableUnit id="SimpleJ2eeAppUnit"
462 targetResourceRef="appServer_os">
463     ...
464   </sdd-dd:InstallableUnit>
465   <sdd-dd:InstallableUnit id="SimpleJ2eeServletUnit"
466 targetResourceRef="J2eeServletServer">
467     ...
468   </sdd-dd:InstallableUnit>
469   <sdd-dd:InstallableUnit id="OracleSimpleDatabaseUnit"
470 targetResourceRef="DatabaseServer">
471     ...
472   </sdd-dd:InstallableUnit>
473   <sdd-dd:InstallableUnit id="DB2SimpleDatabaseUnit"
474 targetResourceRef="DatabaseServer">
475     ...
476   </sdd-dd:InstallableUnit>
477 </sdd-dd:BaseContent>
478 </sdd-dd:CompositeInstallable>

```

- 479 • Each individual constituent *InstallableUnit* follows the pattern already described in the *Simple JRE*

480 *example*. For example, there is an *InstallableUnit* called *SimpleJ2EEServletUnit*. This component is

481 packaged into a Web archive (war) format. The *Requirements* element in the *InstallableUnit* defines

482 the pre-requisite for this component by using the *ResourceConstraint* and *RelationshipConstraint*.

483 The *ResourceConstraint* in the example specifies that the servlet package must be deployed with an

484 Apache Tomcat server pre-installed and the version of Apache Tomcat must be 5.5.20. The

485 requirement section also specifies a *RelationshipConstraint* that specifies that there must be a J2EE

486 application server running the RMI-IIOP protocol. Once the requirement for the *InstallableUnit* is met,

487 the files specified in the *InstallArtifact* element can be deployed. In this case, the war file as specified

488 in the *SC_WAR* reference is deployed as the Servlet unit. This *InstallableUnit* and its *Requirements*

489 are shown in the following example excerpt.
- 490 • An additional concept not illustrated in the Simple JRE example is shown here: the *InstallArtifact* has

491 a *weight* attribute. One of the objectives of the SDD is to enable information to be included that can

492 be used by runtimes to monitor and display the progress of deployment operations. Mechanisms for

493 progress tracking could vary widely and results in various environments also are likely to differ. In

494 addition, including precise information about deployment times in an SDD is impractical, especially

495 because multiple packages from multiple sources can be aggregated and the processing of an SDD

496 can vary from deployment to deployment (based on conditions, features, characteristics of the

497 deployment environment and so on). Nevertheless, the SDD specification does include *weight*

498 attributes for *Artifacts* and *ReferencedPackages*; these weights are *relative times* that SDD authors

499 can specify and SDD consumers can use for estimation and coarse deployment progress

500 expectations and tracking.

501 The key to the use of *weight* in the SDD is to understand that the weight for a particular *Artifact* or

502 *ReferencedPackage* is an estimate of the time required to process that artifact or package relative to

503 other artifacts and packages within that SDD. Weights are not absolute and do not apply across

504 SDDs, only within a single SDD. In this example, the *SimpleJ2eeServletUnit's Artifact* has a *weight*

505 value of 2, as shown in the following example excerpt. In the complete *SimpleCompositeApp*

506 example, the other *InstallableUnits* also have *weight* values: the *SimpleJ2eeAppUnit's Artifact* has a

507 *weight* value of 3; the *OracleSimpleDatabaseUnit's Artifact* has a *weight* value of 4; and the

508 *DB2SimpleDatabaseUnit's Artifact* has a *weight* value of 4.

509 So, in this example, the expectation is that the *SimpleJ2eeAppUnit Artifact (weight=3)* will take

510 approximately 1.5 times as long to process as the *SimpleJ2eeServletUnit Artifact (weight=2)*; and

511 either database *InstallableUnit Artifact* (both *weight=4*) will take approximately twice as long to
 512 process as the *SimpleJ2eeServletUnit Artifact* (*weight=2*). (Similarly, either database *InstallableUnit*
 513 *Artifact* (both *weight=4*) is expected to take approximately 1.3 times as long to process as the
 514 *SimpleJ2eeAppUnit Artifact* (*weight=3*). The *weight* values have no meaning in isolation and do not
 515 describe actual time elapsed. They simply provide an estimate, or hint, of relative time.

```

516 <sdd-dd:InstallableUnit id="SimpleJ2eeServletUnit"
517 targetResourceRef="J2eeServletServer">
518   <sdd-dd:Identity>
519     <sdd-common:Description>The user interface for the Simple Composite
520 Application.</sdd-common:Description>
521     <sdd-common:Name>Simple Application Servlets</sdd-common:Name>
522     <sdd-common:Version>1.0</sdd-common:Version>
523   </sdd-dd:Identity>
524   <sdd-dd:Requirements>
525     <sdd-dd:Requirement id="ServletServer.reqt" operation="install use">
526       <sdd-dd:ResourceConstraint id="J2eeServletServer.check"
527 resourceRef="J2eeServletServer">
528         <sdd-dd:Name>Apache Tomcat</sdd-dd:Name>
529         <sdd-dd:VersionConstraint>
530           <sdd-dd:Supported>
531             <sdd-dd:Value>
532               <sdd-dd:Version>5.5.20</sdd-dd:Version>
533             </sdd-dd:Value>
534           </sdd-dd:Supported>
535         </sdd-dd:VersionConstraint>
536       <sdd-dd:RelationshipConstraint relatedResourceRef="J2eeAppServer"
537 type="connects">
538         <sdd-dd:Property>
539           <sdd-dd:PropertyName>Protocol</sdd-dd:PropertyName>
540           <sdd-dd:Value>RMI-IIOP</sdd-dd:Value>
541         </sdd-dd:Property>
542       </sdd-dd:RelationshipConstraint>
543     </sdd-dd:ResourceConstraint>
544   </sdd-dd:Requirement>
545 </sdd-dd:Requirements>
546   <sdd-dd:ResultingResource resourceRef="SimpleJ2eeServlet">
547     <sdd-dd:Name>Simple Application Servlet</sdd-dd:Name>
548     <sdd-dd:Relationship relatedResourceRef="SimpleJ2eeApp" type="uses">
549       <sdd-dd:Property>
550         <sdd-dd:PropertyName>Protocol</sdd-dd:PropertyName>
551         <sdd-dd:Value>RMI-IIOP</sdd-dd:Value>
552       </sdd-dd:Property>
553     </sdd-dd:Relationship>
554   </sdd-dd:ResultingResource>
555   <sdd-dd:Artifacts>
556     <sdd-dd:InstallArtifact type="war" contentRef="SC_WAR" weight="2" />
557   </sdd-dd:Artifacts>
558 </sdd-dd:InstallableUnit>

```

559 Other *InstallableUnits* in the J2EE Application package follow patterns similar to the preceding
 560 example.

561 Note that there are two installable units for database installation that have the same resulting
562 resource reference: *SimpleDatabase*. These two installable units, however, have different conditions—
563 the first requires database server Oracle®⁴ and the second requires database server DB2®⁵.
564 Therefore, only one of these two units will be processed during a particular deployment, resulting in
565 only one *SimpleDatabase* resulting resource.

566 The J2EE application also illustrates the *Condition* construct, which is useful to enable flexible
567 processing within an SDD by declaring which aspects of an SDD are applicable (or can be ignored) in
568 certain circumstances. *Conditions* can be applied to *Content* (to determine if a content element is
569 applicable), *Variables* (to choose values), *Features* (to determine when a feature is applicable),
570 *ResultingResources* (to determine when a particular result is applicable) and *CompletionActions* (to
571 determine if a particular completion action is necessary). The J2EE example illustrates one use of the
572 *Condition* element to provide content for different deployment environments.

573 In this example, the SDD describes some content (a *CompositeInstallableUnit* for the overall J2EE
574 application) that consists of software that is common to all deployments, plus some content that
575 varies. The variable content is the database for the 3-tier application, and it is based on a *Condition*.
576 The database for the 3-tier application can be either an Oracle 9iAS database or an IBM®⁶ DB2 9
577 database. This *DeploymentDescriptor* describes an *InstallableUnit* that consists of an Oracle
578 database. The *Condition* checks *DatabaseServer* for the value "Oracle9iAS", and if the condition is
579 met, the SDD specifies additional information that applies for the Oracle database (in this case,
580 Oracle database-specific *Requirements* in addition to those specified for the common content, as well
581 as the particular Oracle database *ResultingResource* and *Artifact*). The SDD then describes another
582 *InstallableUnit* that consists of a DB2 database. The *Condition* checks *DatabaseServer* for the value
583 "DB2 9", and if the condition is met, the SDD specifies additional information that applies for the DB2
584 database (in this case, DB2 database-specific *Requirements* in addition to those specified for the
585 common content, as well as the particular DB2 database *ResultingResource* and *Artifact*).

586 These *Conditions* are shown in the following example excerpt.

```
587 <sdd-dd:InstallableUnit id="OracleSimpleDatabaseUnit "  
588 targetResourceRef="DatabaseServer ">  
589   <sdd-dd:Identity>  
590     <sdd-common:Description>The RDBMS Database structure for the Simple  
591 Composite Application.</sdd-common:Description>  
592     <sdd-common:Name>Simple Application Database</sdd-common:Name>  
593     <sdd-common:Version>1.0</sdd-common:Version>  
594   </sdd-dd:Identity>  
595   <sdd-dd:Condition>  
596     <sdd-dd:ResourceConstraint id="OracleDatabaseServer.condition "  
597 resourceRef="DatabaseServer ">  
598       <sdd-dd:Name>Oracle 9iAS</sdd-dd:Name>  
599     </sdd-dd:ResourceConstraint>  
600   </sdd-dd:Condition>  
601
```

⁴ Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

⁵ DB2 is a registered trademark of IBM Corporation in the United States.

⁶ IBM is a registered trademark of IBM Corporation in the United States.

```

602     ... [Oracle database-specific Requirements, ResultingResource, Artifact and InstallableUnit]
603
604     <sdd-dd:InstallableUnit id="DB2SimpleDatabaseUnit"
605     targetResourceRef="DatabaseServer">
606         <sdd-dd:Identity>
607             <sdd-common:Description>The RDBMS Database structure for the Simple
608             Composite Application.</sdd-common:Description>
609             <sdd-common:Name>Simple Application Database</sdd-common:Name>
610             <sdd-common:Version>1.0</sdd-common:Version>
611         </sdd-dd:Identity>
612         <sdd-dd:Condition>
613             <sdd-dd:ResourceConstraint id="DB2DatabaseServer.condition"
614             resourceRef="DatabaseServer">
615                 <sdd-dd:Name>DB2_9</sdd-dd:Name>
616             </sdd-dd:ResourceConstraint>
617         </sdd-dd:Condition>
618
619     ... [DB2 database-specific Requirements, ResultingResource, Artifact and InstallableUnit]

```

620 This example illustrates conditional *Content*, *Conditions* also can be applied to other elements within
621 the SDD (*Variables*, *Features*, *ResultingResources* and *CompletionActions*).

622 4.3.1.2 J2EE Client

623 The second referenced package is a simple client application with *PackageDescriptor* located in
624 `pkgs\Client\SimpleAppClient_pkg.xml` and *DeploymentDescriptor* located in
625 `pkgs\Client\SimpleAppClient.xml` in **[SDDEX]** (see Appendix [A]).

- 626 • The *PackageDescriptor* for the application client is relatively simple and similar to previously
627 described *PackageDescriptors*. This *PackageDescriptor* contains a jar file for the installer and a
628 separate one for the uninstaller.
- 629 • The *DeploymentDescriptor* contains several key elements that are critical for deploying this simple
630 client.
- 631 • The first is the *Topology* section. Three top-level resources are described with *Resource* elements.
632 The first is *JRE*, of type *sp:CIM_InstalledProduct*. The second is *Filesys*, of type *sp:CIM_FileSystem*
633 and it hosts resource *InstallDir*, of type *sp:CIM_Directory*; the name of this directory is the value of the
634 *InstallLocation* variable. The third top-level resource is *SimpleAppClient*, of type *sp:CIM_Application*.
635 As shown later in the deployment descriptor, the *JRE* and *Filesys* are used in the *Requirements*
636 section to declare the requirements for the deployment. The *SimpleAppClient* and *InstallDir* are used
637 to declare the resulting resources for this SDD.

```

638     <sdd-dd:Topology>
639         <sdd-dd:Resource id="os" type="sp:CIM_OperatingSystem">
640             <sdd-dd:HostedResource id="JRE" type="sp:CIM_InstalledProduct" />
641             <sdd-dd:HostedResource id="Filesys" type="sp:CIM_FileSystem">
642                 <sdd-dd:HostedResource id="InstallDir" type="sp:CIM_Directory">
643                     <sdd-common:Name>$(InstallLocation)</sdd-common:Name>
644                 </sdd-dd:HostedResource>
645             </sdd-dd:HostedResource>
646             <sdd-dd:HostedResource id="SimpleAppClient" type="sp:CIM_Application" />

```

```
647 </sdd-dd:Resource>
648 </sdd-dd:Topology>
```

- 649 • In the *Requirements* section, there are three requirements that must be met to deploy this package.
650 These requirements include an operating system requirement, a Java Runtime requirement and a
651 disk space requirement. The operating system requirement uses an *Alternative* element to specify
652 that the package can be deployed either to an AIX operating system with version 5.3 or to a
653 Windows™⁷ operating system with a version higher than 5.1.2600. The *OperatingSystem* type used
654 here is an *OSType* of the *CIM_OperatingSystem* class defined in the Starter Profile **[SDDSP]**. The
655 JRE requirement specifies that the minimum supported version of JRE is 1.4.1. The disk space
656 requirement specifies that the minimum available disk space is 2000 blocks at the size of 512 bytes
657 per block. The element that specifies disk space is *ConsumptionConstraint* with a property defined as
658 *AvailableSpace* for a *CIM_FileSystem* class. These *Requirements* for the simple client are shown in
659 the following example excerpt.

```
660 <sdd-dd:InstallableUnit id="App01" targetResourceRef="os">
661   <sdd-dd:Identity softwareID="3000-123">
662     <sdd-common:Description>This is a local client interface for the simple
663     application.</sdd-common:Description>
664     <sdd-common:Name>Simple Application Client</sdd-common:Name>
665     <sdd-common:Version>1.0</sdd-common:Version>
666     <sdd-common:Manufacturer>
667       <sdd-common:Name>IBM Corporation</sdd-common:Name>
668       <sdd-common:Location>
669         <sdd-common:Address>1133 Westchester Avenue White Plains, New York
670         10604</sdd-common:Address>
671         <sdd-common:Country>U.S.A.</sdd-common:Country>
672       </sdd-common:Location>
673     </sdd-common:Manufacturer>
674   </sdd-dd:Identity>
675   <sdd-dd:Variables>
676     <sdd-dd:Parameters>
677       <sdd-dd:URIParameter id="InstallLocation">
678         <sdd-dd:Description>Root of the directory into which Simple
679         Application should be installed</sdd-dd:Description>
680         <sdd-dd:ShortDescription>Install location for Simple Application
681         Client</sdd-dd:ShortDescription>
682       </sdd-dd:URIParameter>
683     </sdd-dd:Parameters>
684   </sdd-dd:Variables>
685   <sdd-dd:Requirements>
686     <sdd-dd:Requirement id="OS.reqt" operation="install use">
687       <sdd-dd:Description>Simple Application Client is supported on AIX V5.3
688       and Windows XP Professional</sdd-dd:Description>
689       <sdd-dd:Alternative id="AIX.alt">
690         <sdd-dd:Description>Requirements on AIX</sdd-dd:Description>
691         <sdd-dd:ResourceConstraint id="AIX.requirement" resourceRef="os">
692           <sdd-dd:PropertyConstraint>
693             <sdd-dd:PropertyName>sp:CIM_OperatingSystem.OSType</sdd-
694             dd:PropertyName>
695             <sdd-dd:Value>AIX</sdd-dd:Value>
```

⁷ Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

```

696     </sdd-dd:PropertyConstraint>
697     <sdd-dd:VersionConstraint>
698         <sdd-dd:Supported>
699             <sdd-dd:Range>
700                 <sdd-dd:MinVersion>5.3</sdd-dd:MinVersion>
701                 <sdd-dd:MaxVersion inclusive="false">5.4</sdd-dd:MaxVersion>
702             </sdd-dd:Range>
703         </sdd-dd:Supported>
704     </sdd-dd:VersionConstraint>
705 </sdd-dd:ResourceConstraint>
706 </sdd-dd:Alternative>
707 <sdd-dd:Alternative id="Windows.alt">
708     <sdd-dd:Description>Requirements on WindowsXP</sdd-dd:Description>
709     <sdd-dd:ResourceConstraint id="Windows.requirement" resourceRef="os">
710         <sdd-dd:PropertyConstraint>
711             <sdd-dd:PropertyName>sp:CIM_OperatingSystem.OSType</sdd-
712 dd:PropertyName>
713             <sdd-dd:Value>Windows XP</sdd-dd:Value>
714         </sdd-dd:PropertyConstraint>
715         <sdd-dd:VersionConstraint>
716             <sdd-dd:Supported>
717                 <sdd-dd:Range>
718                     <sdd-dd:MinVersion>5.1.2600</sdd-dd:MinVersion>
719                 </sdd-dd:Range>
720             </sdd-dd:Supported>
721         </sdd-dd:VersionConstraint>
722     </sdd-dd:ResourceConstraint>
723 </sdd-dd:Alternative>
724 </sdd-dd:Requirement>
725     <sdd-dd:Requirement id="JRE01.reqt" operation="install use">
726         <sdd-dd:Description>The installation of Simple Application Client
727 requires the a J2SE version 1.4.1 or later</sdd-dd:Description>
728         <sdd-dd:ResourceConstraint id="JRE01.check" resourceRef="JRE">
729             <sdd-dd:Name>Java(TM) Runtime Environment, Standard Edition</sdd-
730 dd:Name>
731             <sdd-dd:VersionConstraint>
732                 <sdd-dd:Supported>
733                     <sdd-dd:Range>
734                         <sdd-dd:MinVersion>1.4.1</sdd-dd:MinVersion>
735                     </sdd-dd:Range>
736                 </sdd-dd:Supported>
737             </sdd-dd:VersionConstraint>
738         </sdd-dd:ResourceConstraint>
739     </sdd-dd:Requirement>
740     <sdd-dd:Requirement id="DiskSpace.reqt" operation="install use">
741         <sdd-dd:Description>The installation of Simple Application Client
742 requires 2000 512-Bytes of available space in the file system where the
743 application is installed.</sdd-dd:Description>
744         <sdd-dd:ResourceConstraint id="DiskSpace.check" resourceRef="Filesys">
745             <sdd-dd:ConsumptionConstraint>
746                 <sdd-dd:PropertyName>sp:CIM_FileSystem.AvailableSpace</sdd-
747 dd:PropertyName>
748                 <sdd-dd:Value unit="512-blocks">2000</sdd-dd:Value>
749             </sdd-dd:ConsumptionConstraint>
750         </sdd-dd:ResourceConstraint>
751     </sdd-dd:Requirement>
752 </sdd-dd:Requirements>

```

- Note the *Variables* section. It declares an input parameter, *InstalledLocation*, for the J2EE Client deployment descriptor. The input parameter is declared as *URIParameter*, which implicitly declares the value to be a valid Uniform Resource Identifier.

```

756     <sdd-dd:Variables>
757         <sdd-dd:Parameters>

```



```

758     <sdd-dd:URIParameter id="InstallLocation">
759       <sdd-dd:Description>Root of the directory into which Simple Application
760       should be installed</sdd-dd:Description>
761       <sdd-dd:ShortDescription>Install location for Simple Application Client
762     </sdd-dd:ShortDescription>
763     </sdd-dd:URIParameter>
764   </sdd-dd:Parameters>
765 </sdd-dd:Variables>

```

- Note the *RelationshipConstraint* element. It defines a relationship between two resources within the scope of a *ResourceConstraint*. The first resource is declared with the *resourceRef* attribute of the *ResourceConstraint*; the second is declared with *relatedResourceRef* in the *RelationshipConstraint*. The relationship is declared with the *type* attribute of the *RelationshipConstraint* (in this case, the type is "connects"). The deployment runtime determines the semantics associated with the particular *RelationshipConstraint*. The *RelationshipConstraint* is shown in the following example excerpt.

```

772 <sdd-dd:ResourceConstraint id="J2eeAppServer.check"
773 resourceRef="J2eeAppServer">
774   ...
775   <sdd-dd:RelationshipConstraint relatedResourceRef="DatabaseServer"
776   type="connects">
777     <sdd-dd:Property>
778       <sdd-dd:PropertyName>Protocol</sdd-dd:PropertyName>
779       <sdd-dd:Value>JDBC</sdd-dd:Value>
780     </sdd-dd:Property>
781   </sdd-dd:RelationshipConstraint>
782 </sdd-dd:ResourceConstraint>

```

783 The first *ResultingResource* element declares that deployment of the J2EE Client will result in a resource
 784 called *SimpleAppClient*, with a property named *Directory* whose value will be equal to the value of the
 785 *InstallLocation* variable.

786 The second *ResultingResource* element declares that deployment of the J2EE Client will result in a
 787 directory type resource. As specified by the *resourceRef* attribute, this is the installation directory that was
 788 also declared earlier in *Topology* as a *HostedResource* (see the "Topology" description earlier in this
 789 section). Together, the *Topology* and *ResultingResource* information indicates that the *InstallDir* resource
 790 is a resulting resource that is hosted by the file system and takes its name from the value of the
 791 *InstallLocation* variable. This directory *ResultingResource* is declared because it is necessary to bring the
 792 installation directory resource into scope and to specify that this installation directory is, in fact, the one
 793 that was declared in *Topology* and it must be hosted by the file system that was declared in *Topology*.

794 The *ResultingResources* are shown in the following example excerpt.

```

795 <sdd-dd:ResultingResource resourceRef="SimpleAppClient">
796   <sdd-dd:Name>Simple Application Client</sdd-dd:Name>
797   <sdd-dd:Version>1.0</sdd-dd:Version>
798   <sdd-dd:Property>
799     <sdd-dd:PropertyName>Directory</sdd-dd:PropertyName>
800     <sdd-dd:Value>$(InstallLocation)</sdd-dd:Value>
801   </sdd-dd:Property>
802 </sdd-dd:ResultingResource>
803 <sdd-dd:ResultingResource resourceRef="InstallDir" />

```

- Once the requirements are met, the *Artifacts* element defines the files that need to be processed for the deployment operation. Besides the normal *InstallArtifact*, this example also has an *UninstallArtifact* that describes how to uninstall an existing simple application client. In this example, only the *InstallArtifact* is used in the context of the composite application, because the *install* operation declared by the *CompositeInstallable* for the composite application effectively "selects" the *InstallArtifact* in the *SimpleAppClient* (the *UninstallArtifact* might be used in other deployment contexts). Note also that the *SimpleAppClient* SDD itself is a Conformance Level 1 (CL1) SDD (it has no *CompositeInstallable*), even though in the context of this example it is being aggregated into a Conformance Level 2 (CL2) solution (see the "Conformance" chapter of the specification [see Section

813 1.3 of this document] for additional information). These *Artifacts* are shown in the following example
814 excerpt.

```
815 <sdd-dd:Artifacts>
816   <sdd-dd:InstallArtifact type="jar" contentRef="SAC_InstallArtifact">
817     <sdd-dd:Arguments>
818       <sdd-dd:Argument name="$(InstallLocation)" />
819     </sdd-dd:Arguments>
820     <sdd-dd:AdditionalContent contentRef="SAC_JAR" />
821   </sdd-dd:InstallArtifact>
822   <sdd-dd:UninstallArtifact type="jar" contentRef="SAC_UninstallArtifact">
823     <sdd-dd:Arguments>
824       <sdd-dd:Argument name="$(InstallLocation)" />
825     </sdd-dd:Arguments>
826   </sdd-dd:UninstallArtifact>
827 </sdd-dd:Artifacts>
```

828 4.3.1.3 JRE Runtime

829 The third referenced package is a Java runtime that is a *Requisite* of the composite package. The JRE's
830 *PackageDescriptor* is located in `pkgs\JRE\jre.xml` and the *DeploymentDescriptor* is located in
831 `pkgs\JRE\jre_pkg.xml` in [SDDEX] (see Appendix [A]).

- 832 • The JRE example is a simple package that was described in Section [4.2].
- 833 • This *ReferencedPackage* is associated with resource constraints via the properties of the resulting
834 resources that are declared in its *ResultingResourceMap*. In this example, the *Name* and *Version* of
835 the JRE are compared with constraints on the *Name* and *Version* of the JRE that are declared in
836 *Requirements* in SDDs elsewhere in the composite application hierarchy (for example, the
837 `JRE01.req` *Requirement* that is defined in *SimpleAppClient*—see section [Error! Reference source
838 not found.]).
- 839 • The resulting resource from the installation of this package has an *id* value of "JRE" and a *type* of
840 "sp:CIM_InstalledProduct".

841 4.3.1.4 Language Packs

842 The final referenced package for this composite example are the language packages located in the
843 `pkgs\Languages` directory in [SDDEX] (see Appendix [A]).

- 844 • These language packages are similar to other simple packages already described. Note that in these
845 packages, the language package can be installed or updated. For the install case, the *InstallArtifact* is
846 used; whereas in the update case, the *UpdateArtifact* is used.
- 847 • Note that both language package deployment descriptors declare, in their respective *Variables*
848 section, a required resource property with the name "Directory" from the resource *SimpleApp1*.
849 This is shown in the following example excerpt from the `pkgs\Languages\French Language`
850 `Pack.xml`.

```
851 <sdd-dd:Variables>
852   <sdd-dd:ResourceProperty resourceRef="SimpleApp1" id="InstallLocation"
853   propertyName="Directory">
854     <sdd-dd:Description>Install location for Simple Demonstration
855     Application</sdd-dd:Description>
856     <sdd-dd:ShortDescription>Install location for Simple Demonstration
857     Application</sdd-dd:ShortDescription>
858   </sdd-dd:ResourceProperty>
859 </sdd-dd:Variables>
```

860

861 4.3.2 Composite PackageDescriptor

862 The complete composite *PackageDescriptor* example for a J2EE application and its associated
863 components is in the file `examples/CompositeApp/CompositeApp_pkg.xml` in **[SDDEX]** (see Appendix
864 [A]).

- 865 • Compared with the simple *PackageDescriptor* as shown in the previous example, the composite
866 *PackageDescriptor* has similar identity information, including *Description*, *ShortDescription*, *Version*,
867 *BuildInformation* and *Manufacturer* information. All of this information is provided by the package
868 author.
- 869 • The composite *PackageDescriptor* differs from the simple descriptor in the content section. The
870 content section contains constituent and referenced *PackageDescriptors* that identify the components
871 to be installed. These components include:
 - 872 ○ Three-tier J2EE application package (*SC_pkg*)
 - 873 ○ Simple J2EE client (*SAC_pkg*)
 - 874 ○ JRE runtime package (*JRE_pkg*)
 - 875 ○ German and French language packs (*DE_Lang_pkg* and *FR_Lang_Pkg*)

876 These individual packages, shown in the following example excerpt, are associated with the
877 corresponding composite *DeploymentDescriptor* (described next) that describes the characteristics of
878 these packages and their relationships and dependencies.

```
879 <sdd-pd:Contents>  
880   <sdd-pd:Content pathname="CompositeApp.xml" id="CAP_DD"  
881   purpose="deploymentDescriptor" />  
882   <sdd-pd:Content pathname="pkgs/Composite/SimpleComposite_pkg.xml"  
883   id="SC_pkg" purpose="packageDescriptor" />  
884   <sdd-pd:Content pathname="pkgs/Client/SimpleAppClient_pkg.xml" id="SAC_pkg"  
885   purpose="packageDescriptor" />  
886   <sdd-pd:Content pathname="pkgs/JRE/JRE_pkg.xml" id="JRE_pkg"  
887   purpose="packageDescriptor" />  
888   <sdd-pd:Content pathname="pkgs/Languages/German Language Pack_pkg.xml"  
889   id="DE_Lang_pkg" purpose="packageDescriptor" />  
890   <sdd-pd:Content pathname="pkgs/Languages/French Language Pack_pkg.xml"  
891   id="FR_Lang_pkg" purpose="packageDescriptor" />  
892 </sdd-pd:Contents>
```

893 4.3.3 Composite DeploymentDescriptor

894 The complete composite *DeploymentDescriptor* example for a J2EE application and its associated
895 components is in the file `examples/CompositeApp/CompositeApp.xml` in **[SDDEX]** (see Appendix [A]).

- 896 • The composite *DeploymentDescriptor* first describes a *Topology* with nested resources for the
897 components to be installed. The *Resource* element is used to describe the top level resources (in this
898 case, the operating systems in which each of the components of the composite application will be
899 installed). The type of each operating system resource is described by using the
900 *CIM_OperatingSystem* class defined in the Starter Profile **[SDDSP]**. Each operating system then
901 contains *HostedResources* that are the components of the composite application: the servlet,
902 application server, database and client portions, respectively. These *nestedResource* types, in turn,
903 are defined by the types defined in the Starter Profile, namely, *CIM_J2eeServer*, *CIM_J2eeServlet*,
904 *CIM_J2eeApplication*, *CIM_DatabaseSystem*, *CIM_CommonDatabase*, *CIM_InstalledProduct* and
905 *CIM_Application*. Note that there are three top-level resources of type *cim:CIM_OperatingSystem*.
906 The first declares the hosting resource where the J2EE application will be installed; the second
907 declares the hosting resource where database will be installed; and the third declares the hosting
908 resource where both the JRE and J2EE client will be installed. Because the JRE is required by the
909 J2EE client, they are both hosted by the same resource. The three operating system resources do
910 not explicitly declare that they represent three separate physical resources. Instead, they implicitly
911 declare that these resources may represent different physical resources. Later in the deployment

912 descriptor, these resources are mapped to the resources in the topologies in the referenced SDDs.
913 This *Topology* is shown in the following example excerpt.

```
914 <sdd-dd:Topology>
915   <sdd-dd:Resource id="servlet_os" type="sp:CIM_OperatingSystem">
916     <sdd-dd:HostedResource id="J2eeServletServer" type="sp:CIM_J2eeServer">
917       <sdd-dd:HostedResource id="SimpleJ2eeServlet" type="sp:CIM_J2eeServlet"
918     />
919     </sdd-dd:HostedResource>
920   </sdd-dd:Resource>
921   <sdd-dd:Resource id="appServer_os" type="sp:CIM_OperatingSystem">
922     <sdd-dd:HostedResource id="J2eeAppServer" type="sp:CIM_J2eeServer">
923       <sdd-dd:HostedResource id="SimpleJ2eeApp" type="sp:CIM_J2eeApplication"
924     />
925     </sdd-dd:HostedResource>
926   </sdd-dd:Resource>
927   <sdd-dd:Resource id="os" type="sp:CIM_OperatingSystem">
928     <sdd-dd:HostedResource id="DatabaseServer" type="sp:CIM_DatabaseSystem">
929       <sdd-dd:HostedResource id="SimpleDatabase" type="sp:CIM_CommonDatabase"
930     />
931     </sdd-dd:HostedResource>
932   </sdd-dd:Resource>
933   <sdd-dd:Resource id="Client_OS" type="sp:CIM_OperatingSystem">
934     <sdd-dd:HostedResource id="JRE" type="sp:CIM_InstalledProduct" />
935     <sdd-dd:HostedResource id="SimpleAppClient" type="sp:CIM_Application" />
936   </sdd-dd:Resource>
937 </sdd-dd:Topology>
```

- 938
- The *CompositeInstallable* element then describes the deployment of the aggregated packages and the associations among these packages. The content of the aggregating composite application *DeploymentDescriptor* in the example contains *BaseContent*, *SelectableContent*, and *LocalizationContent*.
- 942
- The *Variables* element defines the input parameter for the composite deployment descriptor.

```
943 <sdd-dd:Variables>
944   <sdd-dd:Parameters>
945     <sdd-dd:URIParameter id="ClientInstallLocation">
946       <sdd-dd:Description>Root of the directory into which Simple Application
947     Client should be installed</sdd-dd:Description>
948     </sdd-dd:URIParameter>
949     <sdd-dd:StringParameter id="DatabaseUserName" />
950     <sdd-dd:StringParameter id="DatabaseUserPassword" sensitive="true" />
951   </sdd-dd:Parameters>
952 </sdd-dd:Variables>
```

- 953
- The *BaseContent* hierarchically defines the non-optional content for the deployment operation by using the *ContainedPackage* element. The *ContainedPackage* in this example is the 3-tier J2EE application package (*SC_Pkg*) as described in the *PackageDescriptor*. The *ContainedPackage* element in this case can pass arguments to the referenced package of *SC_Pkg* through the *Argument* element, such as *JDBC_User* and *JDBC_Password* (note that the *DeploymentDescriptor* also contains *Parameter Variables* corresponding to these *Arguments*). *ContainedPackage* also maps the resources in the composite *DeploymentDescriptor* to the referenced *DeploymentDescriptor* (the *Topology* in the referenced J2EE application SDD, *SC_pkg*) through *ResultingResourceMap* and *RequiredResourceMap*. The *BaseContent* is shown in the following example excerpt.

```
962 <sdd-dd:BaseContent>
963   <sdd-dd:ContainedPackage id="SimpleCompositeApp_PKG" contentRef="SC_pkg">
964     <sdd-dd:Arguments>
965       <sdd-dd:Argument name="JDBC_User" value="$(DatabaseUserName)" />
966       <sdd-dd:Argument name="JDBC_Password" value="$(DatabaseUserPassword)" />
967     </sdd-dd:Arguments>
968     <sdd-dd:ResultingResourceMap resourceRef="SimpleJ2eeApp"
969   foreignId="SimpleJ2eeApp">
```

```

970     <sdd-dd:Name>Simple Application</sdd-dd:Name>
971   </sdd-dd:ResultingResourceMap>
972   <sdd-dd:ResultingResourceMap resourceRef="SimpleJ2eeServlet "
973   foreignId="SimpleJ2eeServlet ">
974     <sdd-dd:Name>Simple Application Servlet</sdd-dd:Name>
975   </sdd-dd:ResultingResourceMap>
976   <sdd-dd:ResultingResourceMap resourceRef="SimpleDatabase "
977   foreignId="SimpleDatabase ">
978     <sdd-dd:Name>Simple Application Database</sdd-dd:Name>
979   </sdd-dd:ResultingResourceMap>
980   <sdd-dd:RequiredResourceMap resourceRef="J2eeServletServer "
981   foreignId="J2eeServletServer " />
982   <sdd-dd:RequiredResourceMap resourceRef="J2eeAppServer "
983   foreignId="J2eeAppServer " />
984   <sdd-dd:RequiredResourceMap resourceRef="DatabaseServer "
985   foreignId="DatabaseServer " />
986 </sdd-dd:ContainedPackage>
987 </sdd-dd:BaseContent>

```

- 988 • The *SelectableContent* defines contents that are selectable by feature. The *Feature* element within *SelectableContent* defines features that are associated with content elements that are defined within *SelectableContent*. In this example, only one feature is defined: the client application is considered to be an add-on feature and the client application package is defined as a *ContainedPackage* in *SelectableContent*. Note the mapping between resources in the composite deployment descriptor and resources in the topology of the referenced J2EE client SDD, *SAC_pkg*. The *SelectableContent* is shown in the following example excerpt.

```

995 <sdd-dd:SelectableContent>
996   <sdd-dd:Features>
997     <sdd-dd:Feature id="ClientFeature" addOn="true">
998       <sdd-dd:DisplayName>Thick Client for Simple Application</sdd-
999   dd:DisplayName>
1000     <sdd-dd:Multiplicity multiplesAllowed="true" />
1001     <sdd-dd:ContentElement contentElementRef="SimpleAppClient_PKG" />
1002   </sdd-dd:Feature>
1003 </sdd-dd:Features>
1004 <sdd-dd:ContainedPackage id="SimpleAppClient_PKG" contentRef="SAC_pkg">
1005   <sdd-dd:Arguments>
1006     <sdd-dd:Argument name="InstallLocation" value="$(ClientInstallLocation)"
1007   />
1008   </sdd-dd:Arguments>
1009   <sdd-dd:Requirements>
1010     <sdd-dd:Requirement id="solutionJRE_req" operation="install use">
1011       <sdd-dd:ResourceConstraint id="SolutionJRE.check" resourceRef="JRE">
1012         <sdd-dd:VersionConstraint>
1013           <sdd-dd:Supported>
1014             <sdd-dd:Range>
1015               <sdd-dd:MinVersion>1.4.1</sdd-dd:MinVersion>
1016             </sdd-dd:Range>
1017           </sdd-dd:Supported>
1018           <sdd-dd:Certified>
1019             <sdd-dd:Value>
1020               <sdd-dd:Version>1.5.0</sdd-dd:Version>
1021             </sdd-dd:Value>
1022           </sdd-dd:Certified>
1023         </sdd-dd:VersionConstraint>
1024       </sdd-dd:ResourceConstraint>
1025       <sdd-dd:ResourceConstraint id="client_connectivity"
1026   resourceRef="Client_OS">
1027         <sdd-dd:RelationshipConstraint relatedResourceRef="appServer_os"
1028   type="connects">
1029           <sdd-dd:Property>
1030             <sdd-dd:PropertyName>Protocol</sdd-dd:PropertyName>
1031             <sdd-dd:Value>TCP/IP</sdd-dd:Value>

```

```

1032         </sdd-dd:Property>
1033     </sdd-dd:RelationshipConstraint>
1034 </sdd-dd:ResourceConstraint>
1035 </sdd-dd:Requirement>
1036 </sdd-dd:Requirements>
1037 <sdd-dd:ResultingResourceMap resourceRef="SimpleAppClient"
1038 foreignId="SimpleAppClient">
1039     <sdd-dd:Relationship relatedResourceRef="SimpleJ2eeApp" type="connects">
1040         <sdd-dd:Property>
1041             <sdd-dd:PropertyName>Protocol</sdd-dd:PropertyName>
1042             <sdd-dd:Value>HTTPS</sdd-dd:Value>
1043         </sdd-dd:Property>
1044     </sdd-dd:Relationship>
1045 </sdd-dd:ResultingResourceMap>
1046 <sdd-dd:RequiredResourceMap resourceRef="Client\_OS" foreignId="os" />
1047 <sdd-dd:RequiredResourceMap resourceRef="JRE" foreignId="JRE" />
1048 </sdd-dd:ContainedPackage>
1049 </sdd-dd>SelectableContent>

```

- The *LocalizationContent* in the *CompositeInstallable* defines the localization information and resources. In this example, the *ContainedLocalizationPackages* define the packages whose contents enable resources to be localized for German and French. These *ContainedLocalizationPackages* point to individual *PackageDescriptors* (*DE_Lang_PKG* and *FR_Lang_PKG*) as defined in the composite *PackageDescriptor*. The *LocalizationContent* is shown in the following example excerpt.

```

1055 <sdd-dd:LocalizationContent>
1056     <sdd-dd:ContainedLocalizationPackage id="DE\_Language\_Pack\_PKG"
1057 contentRef="DE\_Lang\_pkg">
1058         <sdd-dd:Languages>
1059             <sdd-dd:Language type="de-DE" />
1060         </sdd-dd:Languages>
1061     </sdd-dd:ContainedLocalizationPackage>
1062     <sdd-dd:ContainedLocalizationPackage id="FR\_Language\_Pack\_PKG"
1063 contentRef="FR\_Lang\_pkg">
1064         <sdd-dd:Languages>
1065             <sdd-dd:Language type="fr-FR" />
1066             <sdd-dd:Language type="fr-CA" />
1067         </sdd-dd:Languages>
1068     </sdd-dd:ContainedLocalizationPackage>
1069 </sdd-dd:LocalizationContent>

```

- Note the *Languages* element in the composite deployment descriptor. It declares as mandatory the language "en-US", which is the default (implicit) language of the J2EE Client. Two optional languages, *French* and *German*, are declared; the first with a *LanguageSet* element that declares two languages, "fr-FR" and "fr-CA" (French and French Canadian) for the set French; the second with a *Language* element that declares a single language "de-DE" for language *German*. These languages map to the related *ContainedLocalizationPackage* element.

```

1076 <sdd-dd:Languages>
1077     <sdd-dd:Mandatory>
1078         <sdd-dd:Language type="en-US" />
1079     </sdd-dd:Mandatory>
1080     <sdd-dd:Optional>
1081         <sdd-dd:LanguageSet>
1082             <sdd-dd:Description>French</sdd-dd:Description>
1083             <sdd-dd:Language type="fr-FR" />
1084             <sdd-dd:Language type="fr-CA" />
1085         </sdd-dd:LanguageSet>
1086         <sdd-dd:Language type="de-DE">
1087             <sdd-dd:Description>German</sdd-dd:Description>
1088         </sdd-dd:Language>
1089     </sdd-dd:Optional>
1090 </sdd-dd:Languages>

```

- 1091 • Besides *ContainedPackage* and *ContainedLocalizationPackage*, a *Requisites* element can also be
1092 used in the aggregation of SDDs. The *Requisites* element is used to identify an SDD package that
1093 can be deployed, if necessary, to satisfy a resource constraint. In this example, the *Requisite* uses a
1094 *ReferencedPackage* element to refer to a JRE package that can be deployed to satisfy the
1095 requirement for the client application if no JRE that satisfies the requirement is present in the
1096 deployment environment. *ResultingResourceMap* is used to map the resource *JRE* from the
1097 composite deployment descriptor to the resource *JRE* in the referenced JRE SDD, *JRE_pkg*. The
1098 *Requisites* element is shown in the following example excerpt.

1099
1100
1101
1102
1103
1104
1105
1106
1107

```
<sdd-dd:Requisites>
  <sdd-dd:ReferencedPackage id="JRE_Requisite" contentRef="JRE_pkg">
    <sdd-dd:ResultingResourceMap resourceRef="JRE" foreignId="JRE">
      <sdd-dd:Name>Java(TM) Runtime Environment, Standard Edition</sdd-
dd:Name>
      <sdd-dd:Version>1.5.0</sdd-dd:Version>
    </sdd-dd:ResultingResourceMap>
  </sdd-dd:ReferencedPackage>
</sdd-dd:Requisites>
```

5 Additional Considerations

1108

1109 The preceding examples illustrate basic concepts of the SDD for software installation and are intended to
1110 serve as “getting started” content. Future publications may detail other uses of SDD beyond installation,
1111 such as updates, configuration and localization. A few other considerations for SDDs that deal primarily
1112 with installation but do not appear in the examples are described next.

1113 • **Monolithic artifacts:** The composite application example described in Section [4.3] showed how
1114 multiple artifacts can be present in a single *DeploymentDescriptor*, enabling individual components to
1115 be appropriately installed to deploy the complete solution.

1116 Another form of an artifact that might be present in a *DeploymentDescriptor* is a *monolithic* artifact. A
1117 monolithic artifact is a single *Artifact* for a single target that has content variability (selectable content)
1118 and that a particular runtime understands how to process. A monolithic artifact exposes selectable
1119 content that can be deployed from the single artifact (rather than selecting particular artifacts for
1120 deployment, as was illustrated in the preceding *Conditions* example). SDDs with monolithic artifacts
1121 do not differ substantially from other SDDs; the *DeploymentDescriptor* can still describe *Topology*,
1122 *Identity*, *Requirements*, *Variables*, and so on. The example in
1123 `examples/Monolithic/Monolithic.xml` in [SDDEX] (see Appendix [A]) contains a
1124 *DeploymentDescriptor* that illustrates this concept.

1125 In that example, the *DeploymentDescriptor* contains a single artifact with selectable content described
1126 as two *Features*—a representative application feature and help files—that optionally can be deployed.
1127 The SDD was designed to support either model; this accommodates multiple kinds of artifacts,
1128 including existing or legacy artifacts (this design choice contributed to the decision to have two
1129 different conformance levels for SDD; see the “Conformance” chapter of the specification [see
1130 Section 1.3 of this document] for additional information). Existing artifacts (monolithic artifacts and
1131 others) can still benefit from use of the SDD by “wrapping” the artifacts in the declarative SDD
1132 metadata that adds valuable deployment information. This then enables existing runtimes to benefit
1133 from the SDD for deployment planning and enhanced deployment.

1134 **One operation/one artifact rule:** As described in the SDD specification [see Section 1.3 of this
1135 document], restrictions of a single operation per *CompositeInstallable* and a single artifact per
1136 *InstallableUnit* apply for aggregated SDDs. Here we explain this rule in more detail, including its
1137 rationale.

1138 A *CompositeInstallable* aggregates multiple artifacts that together support the application of one
1139 operation to the overall software. The SDD specification stipulates that each *CompositeInstallable*
1140 defines only a single operation and that all *InstallableUnits* in a *CompositeInstallable* define only a
1141 single artifact. These rules prevent ambiguity that would otherwise arise because some of the
1142 elements of *CompositeInstallable* and *InstallableUnit* are defined per operation; whereas other
1143 elements have implicit meanings that vary based on the operation or typically apply only to a
1144 particular operation.

1145 Allowing only one operation per *CompositeInstallable* prevents ambiguity about the meaning of
1146 *Feature*, *ResultingResource* and *ResultingChange* elements defined in the *CompositeInstallable*.
1147 *Features* are mechanisms to select portions of content for a particular deployment and can be used
1148 with any operation. *Features* do not have long-lived identity; rather, they exist to enable choosing the
1149 content of the deployment package. A *Feature* defined for an install operation is likely to be
1150 unambiguous (typically representing a component or capability that optionally can be included during
1151 the installation). A *Feature* defined for an update operation, however, could be equivocal (for
1152 example, is the *Feature* a particular, optional, portion of the update, similar to the install case, or is it
1153 rather an update to an existing *Feature* that was previously selected during install?). For SDD, a
1154 single definition must be chosen, and the SDD specification opts for the former (that is, *Features*
1155 select a particular, optional portion of the content for the operation that is specified). Hence, a
1156 *CompositeInstallable* defines only one operation because it can define only one set of *Features*.

1157 When applying the one operation defined in the *CompositeInstallable* to the overall software, it may
1158 be necessary to perform a variety of operations on individual resources. For example, to update a
1159 database product, it might be necessary to create (install) a new component. The overall operation for
1160 this *CompositeInstallable* is *update*, but it contains—somewhere in the aggregation—an *InstallArtifact*
1161 for the new component. Limiting the *CompositeInstallable* to a single operation (update in this
1162 example) prevents uncertainty about contained artifacts that specify different operations—if the artifact
1163 is defined (even though, in this example, it is an *InstallArtifact*), it is intended to support the one
1164 overall operation (update in this example) of the *CompositeInstallable*.

1165 Allowing only one artifact in an *InstallableUnit* that is defined within a *CompositeInstallable* also
1166 prevents ambiguity in the interpretation of *RequiredBase*, *ResultingResource* and *ResultingChange*
1167 elements. If multiple artifacts were permitted in a single *InstallableUnit*, then it could be unclear which
1168 artifact's results were described by *ResultingResource* or *ResultingChange*; it also could be unclear
1169 as to which artifact(s) require(s) the *RequiredBase*.

1170 6 Conclusion

1171 This Primer has offered “getting started” information that illustrates the value, rationale and use of SDD
1172 for software deployment, focusing mainly on software installation. Several of the published SDD
1173 examples were described.

1174 As previously stated, this Primer does not take the place of the SDD specification, schema and profiles
1175 (see sections [1.3] and [1.4]). The Starter Profile published by the OASIS SDD Technical Committee
1176 includes types and values used in the examples, but does not include all types and values that might be
1177 used in particular SDDs. However, existing profiles can be extended and new profiles generated to define
1178 such values, using the pattern established by the Starter Profile, as described in **[SDDSP]**.

1179 Finally, as implementation experience with the SDD standard is gained, Best Practices will be developed.
1180 The OASIS SDD Technical Committee has established a wiki **[SDDBP]** to capture these best practices;
1181 use of the wiki by implementers is encouraged.

1182 **A. Complete Examples**

1183 Example SDDs **[SDDEX]** showing the use of the schema can be found at the following address.

1184 <http://docs.oasis-open.org/sdd/v1.0/pr01/expository/sdd-examples-v1.0-cd01.zip>

1185 These examples include those excerpted and described elsewhere in this document.

1186 **B. Acknowledgements**

1187 The following individuals have participated in the creation of this specification and are gratefully
1188 acknowledged:

1189 **Participants:**

- 1190 Dr. Howard Abrams, CA
- 1191 Mr. Lazar Borissov, SAP AG
- 1192 Ms. Debra Danielson, CA
- 1193 Mr. Robert DeMason, SAS Institute, Inc.
- 1194 Mr. Robert Dickau, Macrovision Corporation
- 1195 Mr. Quenio dos Santos, Macrovision Corporation
- 1196 Mr. Adrian Dunston, SAS Institute, Inc.
- 1197 Mr. Randy George, IBM
- 1198 Mr. Nico Groh, SAP AG
- 1199 Ms. Merri Jensen, SAS Institute, Inc.
- 1200 Mr. Jason Losh, SAS Institute, Inc.
- 1201 Ms. Julia McCarthy, IBM
- 1202 Mr. Brent Miller, IBM
- 1203 Mr. Ed Overton, SAS Institute, Inc.
- 1204 Mr. Chris Robsahm, SAP AG
- 1205 Mr. Thomas Studwell, Dell
- 1206 Dr. Weijia (John) Zhang, Dell