App-V OSD Reference Book

Tim Mangan

TMurgent Technologies

```
<?xml version="1.0" standalone="no"?>
<SOFTPKG GUID="CB2CFE37-5346-4E70-968E-81E4C7853995" NAME="Micro
  <IMPLEMENTATION>
    <CODEBASE HREF="RTSP://%SFT_SOFTGRIDSERVER%:554/M
    <VIRTUALENV TERMINATECHILDREN="FALSE">
      <DEPENDENCIES/>
      <POLICIES>
        <LOCAL_INTERACTION_ALLOWED>FALSE</L
      </POLICIES>
      <ENVLIST/>
    </VIRTUALENV>
    <WORKINGDIR/>
    <VM VALUE="Win32">
      <SUBSYSTEM VALUE="windows"/>
    </VM>
  </IMPLEMENTATION>
  <COMMENT>Sequenced by TMurgent Technologie
  <PACKAGE NAME="Microsoft_OfficeViewers20
  <ABSTRACT/>
  <MGMT_SHORTCUTLIST>
    <SHORTCUTLIST>
      <SHORTCUT LOCATION="%CSI
    </SHORTCUTLIST>
  </MGMT_SHORTCUTLIST>
  <MGMT_FILEASSOCIATIONS>
    <PROGIDLIST>
      <PROGI
```

This is the App-V OSD Reference Book

Updated November 7, 2010
A Service of TMurgent Technologies LLP

TMurgent Technologies, LLP
Your Destination for Application Virtualization

This reference book allows you to explore the xml syntax of the OSD file. The content of this book has been derived from the xsd schema, personal experience in working with OSD files, and feedback that I have received from Microsoft over the years for my online tool The OSD Illustrated.

The book has two table of contents, one alphabetical and one ordered by usage. Send Feedback via email to Tim Mangan. Updated for Microsoft Application Virtualization (App-V) version 4.6.

*What's an **OSD** file?* **O**pen **S**oftware **D**escription: a borrowed proposed standard format created by Marimba and Microsoft as part of an obscure proposal to database software applications using XML. This format was discovered by Softricity long before their Microsoft relationship began. Softricity used this as a base and extended the DTD to meet their needs. An OSD formatted file is currently used by App-V as the target of shortcuts which describe, to an App-V client, how to access an App-V enabled virtual application. These files use names that end in ".OSD" as the extension. In addition to this tool you may be interested in the most recent updated SoftGrid OSD schema definition file in the Tools section of our website.

The History of the OSD

When we were first building the SoftGrid platform (later renamed as App-V after Microsoft acquired Softricity), we wanted a way to describe applications using XML. Looking around, we discovered an abandoned and forgotten proposal for using XML to describe publishing information for software.

This proposal, called ***O**pen **S**oftware **D**escription*, or OSD for short, was a 1997 joint proposal to a working group in the W3C by Microsoft and Marimba to aid in the automation of software delivery. This would allow publishers to describe their software in a format that would allow it to be distributed over the Internet, and checked to ensure that the latest version is being used. This was during the heyday of "Push" (or as they preferred to call it, "Smart-Pull"). Other than this one draft proposal, we never found any evidence of public work on this specification. Whether the proposal was roundly rejected or if maybe the forces driving the two companies to work together changed we might never know.

As the original specification did not completely meet our needs, we extended the original proposal to meet our needs.

For the curious, the original specification may be found here.

Finally, we note that the term OSD is also now being used for ***O**bject-based **S**torage **D**evice*. Geeks love acronyms, so it is not too surprising.

OSD XML Syntax Guidance

The OSD is an XML based file. XML can be explained as a "self describing data format", where all of the data within the file is decorated by descriptors (sometimes called metadata) that explain what the data is. Metadata comes in the form of Elements and Attributes. While XML describes rules for well formatted XML data, use of XML requires a schema. The schema explains what the valid Elements and Attributes are, as well as rules for what form particular data values must follow, and also what combinations are allowed.

Examples of these rules include:

- A CODEBASE element may contain a HREF element, but not the other way around.
- An IMPLEMENTATION element may contain multiple CODEBASE elements, while the CODEBASE may contain only one HREF element.
- The VALUE attribute of the VM element may only contain values consisting of one of the strings "Win16", "Win32", or "Win64".

Microsoft does not publish the OSD schema. The product does contain an xsd file (used by popular XML tools to understand the schema), however the file is not correct and does not seem to be used by the most important parts of the client for processing the OSD. In this book, we describe the OSD syntax from a practical, derived standpoint.

Elements

Well formed XML consists of elements and sub-elements. Elements have a beginning and an end, and these beginning and ends always match up. This example:

```
<ELEMENT>
    <SUBELEMENT> Some data </SUBELEMENT>
</ELEMENT>
```

is well formed XML with matching beginning and ending tags. There is an alternate, short, form for an element that has no sub-elements or separate data. Here is an example:

```
<ELEMENT ATTRIBUTE="value"/>
```

White Space

In XML, white space is always optional. White space is any combination of space, tab, and new line characters. So adding in spacer lines, and tabbing things over to improve your readability is just a nicety. The XML reader doesn't care and ignores everything after the first white space until it sees more content.

Uppercase

While not required for XML in general, the OSD is very specific when it comes to using case. Pretty much everything must be in UPPERCASE letters. Certainly all Element names and all Attribute names **must** be in uppercase letters. Most attribute values must also be in upper case (for example, VIRTUALENV TERMINATECHILDREN="true" is not supported; the "true" must be in upper case). The few cases where you can use lowercase or mixed case letters include:

- Comments
- Things inside quotation marks
- File paths and file names
- The values for attributes name VALUE when in the VM, SUBSYSTEM, and OS elements.

Silently Ignores the Unknown

When parsing XML, it is normal to *silently ignore* any elements, attributes, and values, that are not understood or legal under the schema -- **as long as the XML is well formed**. This is true of the client, as long as it can find sufficient valid information in the file for the task at hand. This is why we continue to document outdated elements from versions of the product many years back. The client silently ignores them today, but (as long as the SFT being pointed to is still valid) the app can still run with those old elements.

The First Line of the OSD

The first line of the OSD is important only in that it must be there.

<?xml version="1.0" standalone="no"? >

Basically, this line just tells the reader that this is an xml based file. It is the xml equivalent of a "magic number" placed at the beginning of a binary data file to identify the format.

Comments in XML

Although the sequencer does not add comments, these are allowed in XML, and in theory you could use them to manually document your work.

<!--

The four character sequence above (Less-than, Bang, minus, minus) marks the start of a comment. The comment can have multiple lines if desired. The comment ends with these three characters:

-->

The main use I have for the XML Comment is to temporarily disable a section of the OSD when testing. This might be used to comment out a script or policy in the OSD, rather than deleting the entry and then having to type it back in.

Alphabetical Listing of Elements

The remaining pages detail the various elements, presented in a logical order. Below, on this page, is an alphabetical list of these elements with links to ease finding the one you are interested in. (Hint: bookmark this page in your e-reader) .

- **SXS_32_ENABLED**
- **TITLE**
- **VIRTUAL_SERVICES_DISABLED**
- **VIRTUAL_FILE_SYSTEM_DISABLED**
- **VIRTUAL_REGISTRY_DISABLED**
- **VIRTUALENV**
- **VM**
- **WORKINGDIR**

I also suggest you visit our online tool The OSD Illustrated.

SOFTPKG Element

The SOFTPKG element identifies the application that is represented by this OSD file.

Usage:

The SOFTPKG element is required. Only one SOFTPKG element is allowed, and it must be the top element in the file.

Attributes:

The SOFTPKG element requires three attributes:

GUID

The GUID attribute is a required attribute. This is the application GUID (a unique number that can be used to identify the application). This is different from the package GUID (which is an attribute of the CODEBASE element).

NAME

The NAME attribute is a required attribute. This is the name of the application. This is the name that will be displayed to the user just above the tray icon bar when the user launches the application. Previously, NAME was limited to 64 characters in length (this limitation may have been lifted?).

VERSION

The VERSION attribute is a required attribute. This is a text string that usually represents the version of the application. The version is also displayed to the user above the tray icon when the user launches the application. Although typically a "number dot number" format, it does not need to follow any special format. When used with the full App-V Management server, it is important that this version string match what is stored in the database for this application (the database is populated from the OSD upon import, so this is a concern only if the string in the OSD is modified after import). A mismatch between what is in the OSD and database will cause this application to not be included in metering reports. The VERSION atttribute is limited to no more than 16 characters.

Contents:

None.

Sub-Elements:

The SOFTPKG supports a number of sub-elements. The most important of these is the IMPLEMENTATION element, followed by the two MGMT_ elements.

The following elements are the known sub-elements of SOFTPKG:

IMPLEMENTATION

MGMT_SHORTCUTLIST

MGMT_FILEASSOCIATIONS

SUITE

TITLE

ABSTRACT

ASSETS

Example:

```
<SOFTPKG GUID="8A1E47F3-B0F5-44A2-83EE-4858C84E4839"
    NAME="Microsoft Visual Basic 6.0"
    VERSION="6.0.81.76"
>
    ...
```

```
</SOFTPKG>
```

Additional Description of this Element:

The NAME and VERSION parameters originally are generated by the sequencer, which uses the properties of the file that generated this application to populate the fields in the sequencing wizard. The NAME and VERSION parameters must be unique for all App-V applications to be used on the same OS.

Parents:

None. SOFTPKG must be the top level element in the OSD file.