Inferapp

IT Discovery Service

Agent Specification for Software Detection

VERSION 3.0

Revision Date: 7 February, 2015

Author: Artur Kornatowski

Table of Contents

1. Files

For all files, including the described later Software Identification Tags, executables and Java / Web Application archives, the following attributes should be collected:

- FileName
- FileSize
- FilePath (with trailing either backslash or slash always included)

Attribute names used in the next sections, i.e. CompanyName, ProductName, ProductVersion, FileVersion and FileDescription, come originally from Microsoft executable version resource spec but are then matched to attributes specific to Java / Web Application archives and Software Identification Tags so as to put them in same fields of scan output Files section. Relevant mapping in brackets.

Section header for files:

<SourceName=files><Fields=FilePath FileName ProductVersion CompanyName ProductName FileDescription FileVersion FileSize>

2. Software Identification Tags

Software Identification Tags (ISO/IEC 19770-2) have either .swidtag or .swtag extensions. Parse the XML file for the following additional attributes (only the first one is ISO/IEC 19770-2, the other ones are older vendor-specific that I've seen so far):

- CompanyName (swid:software_creator.swid:name)
- ProductName (swid:product_title)
- ProductVersion (swid:product_version.swid:name)

In addition to the files section, each tag should be put into a separate swidtags section:

Section header for swidtags (for list fields, concatenate all elements separated by spaces):

<SourceName=swidtags><Fields=entitlement_required_indicator product_title
product_version.name product_version.numeric.major
product_version.numeric.minor product_version.numeric.build
product_version.numeric.review software_creator.name
software_creator.regid software_licensor. name software_licensor.regid
software_id.unique_id software_id.tag_creator_regid tag_creator.name
tag_creator.regid abstract product_family tag_creator_copyright
component_of.software_id.unique_id complex_of.software_id.unique_id
installation_details.location_platform installation_details.location_installation
installation_details.installation_locale installation_details.installation_target_id
elements_owner product_category.UNSPSC_ver
product_category.segment_title product_category.family_title
product_category.class_title product_category.commodity_title
product_category.code license_linkage.activation_status keywords.keyword
extended_information.tag_type package_footprint.primary.file
package_footprint.secondary.file package_footprint.related.file>

Additionally, collect IBM-specific signature files:

- .cmptag (IBM component)
- .fxtag (IBM fix pack)
- .sys (IBM signature file)
- .sys2 (IBM signature file)

3. Executables

On Windows files with the following extensions:

- .exe
- .ocx
- .dll
- .sys

Collection of .dlls will markedly burden the collection process, and is not necessary for most apps, so it may be better to include .dlls in instruction list for specific filenames instead.

On UNIX all executable files, see:

http://unix.stackexchange.com/questions/70668/find-executable-files-recursively

The following attributes should be retrieved from version resource for Windows files:

- CompanyName
- ProductName
- ProductVersion
- FileDescription
- FileVersion

Here's a c++ wrapper:

http://www.codeguru.com/cpp/w-p/files/fileinformation/article.php/c4481/Accessing-a-Files-Version-Resource-Information.htm

For java example see:

http://stackoverflow.com/questions/6918022/get-version-info-for-exe

4. Java / Web Application archives

Java / Web Application archives have .jar, .war or .ear extensions. They should be scanned for the presence of MANIFEST.MF file. The order for ProductVersion below is not a mistake, Implementation-Version should have priority as it is usually more accurate than Specification-Version. For ProductName on the other hand Specification-Title usually represents more user friendly product name.

- CompanyName (Specification-Vendor or Implementation-Vendor or Bundle-Vendor)
- ProductName (Specification-Title or Implementation-Title or Bundle-Name or Extension-Name)
- ProductVersion (Implementation-Version or Specification-Version or Bundle-Version)

Note you should retrieve the values from the section specific to the package rather than the common section, if such exists, which reflects JDK or Ant version that is useless for our purposes, see e.g.

http://www.javaworld.com/article/2074257/core-java/accessing-package-version-information-in-java-code.html

Additionally, each .jar, .war and .ear should be scanned for the presence of files with Software Identification Tag extensions. If some are found, then they should be processed as if they were stand-alone, according to the logic for Software Identification Tags, so there could be more than one output line for one .jar, .war and .ear file in files section.

5. AddRemoves

Generic

For all sub keys of

32 bit entries keys:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

64 bit entries keys:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall
HKEY_CURRENT_USER\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall

collect data of the following value names:

DisplayName
DisplayVersion
Publisher
InstallLocation
UninstallString
SystemComponent

The values should be merged from all the four keys with a unique composite index on DisplayName, DisplayVersion and Publisher fields. Not all value names are always present. Ignore sub keys with no DisplayName. Put 0 for SystemComponent if not present for consistency with SCCM query results.

Section header for addremoves:

<SourceName=addremoves><Fields=DisplayName DisplayVersion Publisher InstallLocation UninstallString SystemComponent>

6. Pkginsts

Parse the output from relevant package managers for the following fields, trying to map fields to those of RPM, whose field names are then used in scan output format. Here's the mapping for Solaris:

RPM (Red Hat): `rpm -aqi`
pkginfo (Solaris): `pkginfo -l`

- RPM Name (pkginfo PKGINST)
- RPM Version (pkginfo VERSION)
- RPM Vendor (pkginfo VENDOR)
- RPM Group (pkginfo CATEGORY)
- RPM Summary (pkginfo NAME)

For Mac OS parse the output of
system_profiler SPApplicationsDataType
There may be a better way for Mac OS, I've found the above at

http://stackoverflow.com/questions/15164132/enumerate-all-installed-applications-on-os-x

Section header for pkginsts:

<SourceName=pkginsts><Fields=Name Version Vendor Group Summary>