IBM Tivoli Software Signature Analyzer User's Guide

IBM Tivoli Software Signature Analyzer User's Guide

Contents

Chapter 1. Overview

The IBM Tivoli Software Signature Analyzer helps you to analyze information that is collected from several computers with the purpose of identifying the installed software products, and selecting suitable files to use as signatures for those products. The information obtained is incremental knowledge that is stored in a local knowledge base file. You can export a catalog of software signatures from the knowledge base to Tivoli Configuration Manager.

You need to run the Software Signature Analyzer harvesting tool to scan the systems you want to analyze, and to provide the IBM Tivoli Software Signature Analyzer with the necessary data to produce signatures. You need to scan all the computers in your environment that are running on a particular operating system, or at least a meaningful subset that adequately represents all the software installed in your environment, including patches.

Product definitions and software signatures are used in system management applications to represent and detect installed software products. As a general rule, a product is said to be installed on a system when a file, whose signature is associated to that product, is found on that system.

The creation of product definitions and software signatures consists of two main phases, described in the following sections.

Reconstruction phase

This is the initial phase, during which you acquire data about the installed applications and process it to create footprints. This phase consists of the following main activities:

- Chapter 3, “Acquiring Data,” on page 11
- Chapter 4, “Creating a New Project,” on page 13
- Chapter 5, “Processing Harvest Data,” on page 17
- Chapter 6, “Recreating the Product Structure,” on page 21
- Chapter 7, “Grouping Footprints,” on page 31

Definition phase

This is the second phase, during which you view the folder structure of installed products and derive product definitions and signatures from that information. This phase consists of the following main activities, described in Chapter 9, “Creating Definitions”:

- “Entering the definition phase” on page 39
- “Selecting the information to save” on page 41
- “Storing signatures in the Knowledge Base” on page 44

Knowledge Base administration

Some other activities related to the administration of content in the knowledge base are described in Chapter 10, “Knowledge Base Administration,” on page 45, and they include:

- “Merging Knowledge Bases” on page 46
- “Creating a baseline” on page 49.

Chapter 2. Concepts

Some fundamental concepts of the IBM Tivoli Software Signature Analyzer and signature creation process are described in the following sections.

The IBM Tivoli Software Signature Analyzer

IBM Tivoli Software Signature Analyzer is an Eclipse rich client platform GUI application. It analyzes specific raw data collected from the scan of several computers (referred to as *harvest data sets*) and produces some knowledge about software products found installed on those computers, namely their identity (name and version) and signatures. Signatures can be used by other Tivoli management applications for an accurate and efficient inventory of installed software.

The IBM Tivoli Software Signature Analyzer itself does not include the scanning tools, but relies on external programs such as the Software Signature Analyzer harvesting tool to run the appropriate scanning operations to collect the raw input data files from several computers and store these files under a single folder accessible to the application.

The IBM Tivoli Software Signature Analyzer uses the raw input data collected to produce information (product definitions and signatures) to populate a *Software Knowledge Base*. This is the primary source of information for inventory and license management applications that employ signature-based discovery of installed software.

Workspace

The workspace is a directory in the local file system where the IBM Tivoli Software Signature Analyzer stores the folders containing the data for each single *project*, as well as the Knowledge Base archive.

The default location for the workspace is the IBM Tivoli Software Signature Analyzer installation directory. However, it is recommended that you specify a different location for the workspace, so that it is not accidentally removed when you remove the application from the system.

The only file that the IBM Tivoli Software Signature Analyzer requires to be present in the workspace is the *Knowledge Base archive*. When you select a workspace folder that does not contain a Knowledge Base, the IBM Tivoli Software Signature Analyzer creates an empty archive in that folder. The IBM Tivoli Software Signature Analyzer uses the Knowledge Base archive to store incremental new information.

You can create one or more projects in the workspace. Data related to each project, including the embedded database that contains normalized data from the harvesting raw data files, is stored under a *project private folder*.

It is recommended that you use different projects to handle data related to:
- Different operating systems
- Different harvesting operations

You can open only one project at a time in the application GUI. If you want to open a different project, you must first close the current one.

You can *manually move* a project from one workspace to another by archiving the project folder and re-creating its content under the target workspace.

Projects

A *project* contains information about a collection of harvest data sets. When you create a new project, you must specify the following information:

- The name of the project
- The path to the folder where the harvest data sets are located

In the GUI, each project contains the following information:

- **Harvest data sets** (one for each scanned computer)
- **Software Signature Analyzer database** storing all the information generated or derived from the harvest data sets
- **Folders** identified by their content across all the scanned computers
- **Footprints** representing the reconstruction of a product installation tree.

You can obtain more detailed information, by drilling-down in views, such as:

- *Files*
- *Folders*
- *File-folder* relationships
- *Footprint-file* relationships
- *Signatures* in the local Knowledge Base
- *Product definitions* in the local Knowledge Base

Project phases

A project consists basically of the following two phases:

Reconstruction Phase
during which the IBM Tivoli Software Signature Analyzer is used to reconstruct the structure of a software product installation tree.

Definition Phase
during which you create definitions of software products that then become *canonical* for any exploiter of the generated Knowledge Base. The IBM Tivoli Software Signature Analyzer helps you to select the correct files to use as signatures for those products. Eventually, you store these definitions in the local Knowledge Base. There is some flexibility and arbitrariness in deriving product definitions and signatures from product footprint information. However, the IBM Tivoli Software Signature Analyzer provides the necessary context information to make informed decisions on which selections are the most appropriate.

Harvest data sets

A harvest data set is a collection of raw data that the Software Signature Analyzer harvesting tool gathers on the scanned computers.

Each harvest data set consists of the following text files:

CIF Computer Information File, which contains data relating to the scanned computer (such as operating system, and hostname).

FIF File Information File, which contains data relating to each file (such as name, size, and checksum.)

PIF Product Information File, which contains data relating to each product (such as version, and installation path) found registered on any of the scanned computers.

FPLF File-Product Link File, which establishes relations between some files and some products.

HIF Header Information File, which contains information (product name, product version, and vendor name), extracted from the header of binary files on Windows computers.

Data Normalization Process

Data normalization is part of the reconstruction phase. This is the process by which the application creates a unique object in the database to represent all instances of the same unique entity (file, folder, or product) that are found in the analyzed data sets. As an example, a single object is stored in the database to hold the key attributes of a file (checksum and size) regardless of how many instances of the same file are found across the harvest data sets.

The time required to create normalized information out of data harvested from a single computer depends on the size of the harvested data files. This size can be reduced by excluding some types of files, such as dynamic libraries, from the harvesting. However, the data processing can still require a long time when the collection of harvested computers is very large.

Database tables

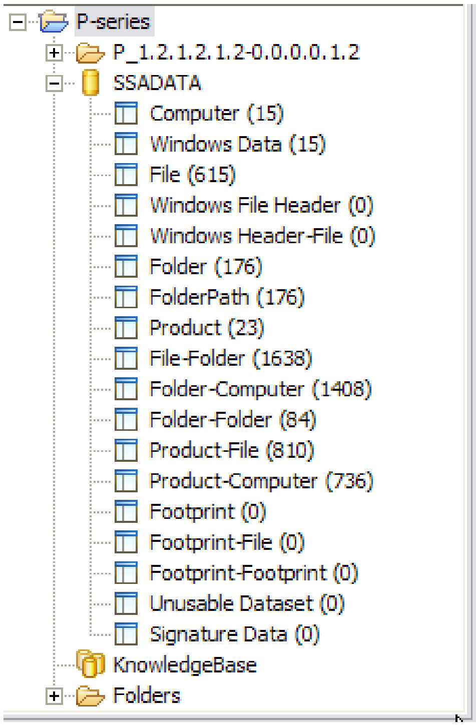
During the reconstruction phase, the IBM Tivoli Software Signature Analyzer processes the generated data and populates tables in the project database. The time required for processing, and the size of the database table, vary depending on the amount of harvested data.

Several tables contain data about files and folders that are *uniquely* identified by their content. Therefore, the size of these tables is likely to grow slowly after most of the items have been created, which might occur after merging the content of only a relatively small percentage of data sets.

By contrast, a few other tables contain *instance data*, such as the drive letter of the actual path of a folder instance on a given computer. These tables significantly affect disk space when merging data from a very large number of analyzed computers.

Table content

The database tables are populated during harvest data processing, and they provide information about the following items:



- P-series
  - P_1.2.1.2.1.2-0.0.0.0.1.2
  - SSADATA
    - Computer (15)
    - Windows Data (15)
    - File (615)
    - Windows File Header (0)
    - Windows Header-File (0)
    - Folder (176)
    - FolderPath (176)
    - Product (23)
    - File-Folder (1638)
    - Folder-Computer (1408)
    - Folder-Folder (84)
    - Product-File (810)
    - Product-Computer (736)
    - Footprint (0)
    - Footprint-File (0)
    - Footprint-Footprint (0)
    - Unusable Dataset (0)
    - Signature Data (0)
  - KnowledgeBase
  - Folders

Computer
One item for each scanned computer.

Windows Data
One item for every Windows operating system scanned.

File Unique files, that is an item for all the instances of the same file.

Windows File Header
One item for each unique set of file header information (product name, version, and vendor).

Windows Header-File
Associates a file to the relating header information.

Folder Unique folders, that is one item for all the instances of the same folder.

FolderPath
Unique paths, that is one item for all the instances of the same path name.

Product
Unique product definitions in the registry. One item for all the instances of the same product.

File-Folder
Associates unique files and unique folders.

Folder-Computer
Associates unique folders and scanned computers.

Folder-Folder
Associates a parent folder with its children (sub-folders).

Product-File
Associates unique products and unique files.

Product-Computer
Associates products and computers.

Footprint
Created footprints.

Footprint-File
Associates footprints with the files they contain.

Footprint-Footprint
Associates a parent footprint with its children.

Unusable Data set
Invalid data sets that cannot be used to create signatures

Signature Data
Temporary storage of created signatures.

Product variability

To identify the sources of variability in the installed files of a product, it is useful to logically separate the product-installed files into the following categories:

Version invariant
Files that are present in *multiple versions* of the same product

Release invariant
Files that are present in *multiple releases* of the same product version

Release specific
Files that are present *only in a specific release* of the product. Most key files in a product fall into this category. However, the possibility of release (and even version) invariants cannot be excluded, which makes the reconstruction significantly more complex.

Moreover, a file can be either affected or unaffected by each of the patches applied to a given installation of the product.

Because the above categories apply both to *core* and to *optional* files, a full specification of the content in a product, or folder can be given by specifying the number of core and optional files that fall in each one of the possible categories.

It is evident from the above considerations that myriads of variants in the installed content can be generated by the combined effect of installation options, product upgrades, and maintenance.

Folder variants

Among all the folders analyzed in the harvested data, the IBM Tivoli Software Signature Analyzer might find some folders that have *almost* the same content, and differ only by the presence or absence of a few files. These folders can be considered as *variants* of the same original folder. An example of a folder variant might be the installation directory of a patched product. In this directory the content of one or more files changes after the installation of the patch. The possible occurrence of *optional* files and patches is an important source of variability in the content of folders.

Files within a folder can be classified as either:

Core files
Files whose presence in a folder *does not depend* on any selectable options that can be specified during the installation, or

Optional files

Files whose presence in a folder *does depend* on a selectable option that must be specified during the installation.

Another source of variability is caused by maintenance, that is patches that can change the content of a given file. The IBM Tivoli Software Signature Analyzer keeps tracks of the file content by means of a checksum, therefore it is possible to make a distinction between folders which contain the *base* version of a product and folders that contain a *patched* version of the same product.

Unique folders

Among all the occurrences of the *same* folder found when analyzing the harvesting data, the IBM Tivoli Software Signature Analyzer uses a single object to uniquely represent all the instances of folders that have identical content.

The application cannot use a folder path to determine the folder identity, instead uses a digest of the folder content. Therefore, two folders that have the same name but different content (for example, because there is a file that is only contained in one of the folders) are identified as different objects by the IBM Tivoli Software Signature Analyzer.

Because the analyzed computers can contain multiple instances of the same folder, the application compares their paths and eliminates any leading part that is instance-dependent. This process keeps only the trailing part of the path string that is common across a majority of instances. The remaining part of the path string is referred to as the folder *canonical path*. A folder that has conflicting names across instances does not have a canonical path. In most cases this means that the folder has not been created by a standard installation process. This might occur when the name of the folder where the product is installed is not meaningful of the product itself, for example, because the installation folder is either very generic, or user-defined.

Footprints

The IBM Tivoli Software Signature Analyzer creates footprints by eliminating files deemed optional from folders that were identified during data normalization.

An accurate identification of optional files requires that the harvest data set collection include both the installation variant of a given product that includes the optional file and the installation variant that does not include the file. Therefore, footprints obtained from the data set collection harvested from a given environment might be safely used to create signatures in that environment but they might not be accurate for another environment. You can obtain a more accurate definition of footprints by merging the information of multiple data set collections harvested from different environments.

User profiles

User Profiles allow multiple users to work on clones of the same project without creating duplicate information.

The IBM Tivoli Software Signature Analyzer automatically creates a file named `UserProfile.xml` on an empty workspace, and whenever at the application startup the file is not found in the specified workspace. When created, the file only contains a sample user definition.

The file can be edited with any standard text editor to add definitions for each of the users who need to work on the application.

When multiple users work on clones of the same project they must coordinate themselves so that each one works under a distinct user profile. This helps the IBM Tivoli Software Signature Analyzer to properly report conflicts when merging definitions into a single Knowledge Base.

File signatures

Signatures are useful to identify what products (versions and releases) are installed on a computer. Therefore, signatures must be as unique and meaningful to a product as possible. So that, each time you scan a computer and find a signature match, you know that the corresponding product is installed.

The IBM Tivoli Software Signature Analyzer currently only supports the creation of file-based signatures and it is assumed that a signature evaluator can establish a match only by considering the file name and size (not the checksum). Before choosing a file as the signature of a product, you should ensure that it meets the following criteria:

- It should be found only within the installation tree of a specific product. A file that is packaged and deployed by several products cannot be used to create a signature.
- The file content should vary across changes of the product version or release. Otherwise it is impossible to associate a matching instance to a specific version or release of the product.

It is possible that folders related to different releases or versions of the same product contain some common files whose content (checksum) is identical. It might be difficult to determine whether a difference in the content of a file is due to a difference of release or maintenance level. Nevertheless, *invariant files* are files whose content is not affected by any maintenance that causes variability in the associated folders; while *common files* are files that are found to be identical across folders that correspond to different releases or versions.

Knowledge Base

The local Knowledge Base contains information about software products. In particular, the following elements of information are stored in the Knowledge Base:

- Product information, such as
  - Name
  - Version
  - Manufacturer
- Signature Information such as
  - File name
  - File size

The Knowledge Base schema can handle several elements of information about software products. In particular, it supports the distinction between:

Software components
Entities actually installed on a computer. In the Knowledge Base,

signatures are associated to these components. As a result, the presence of a signature on a computer usually means that the corresponding component is installed.

Software products
Aggregation of multiple components, some of which can be shared among different projects.

The IBM Tivoli Software Signature Analyzer currently only defines file-based signatures, which means signatures where *one key file of the product is identified by name and size*.

Chapter 3. Acquiring Data

Before you can start to produce product definitions and signatures, you have to gather harvesting data from the scanned computers. To do this, use the Software Signature Analyzer harvesting tool (for more information about this tool, see the relevant documentation). The Software Signature Analyzer harvesting tool collects the harvest data sets that you must save in an empty directory.

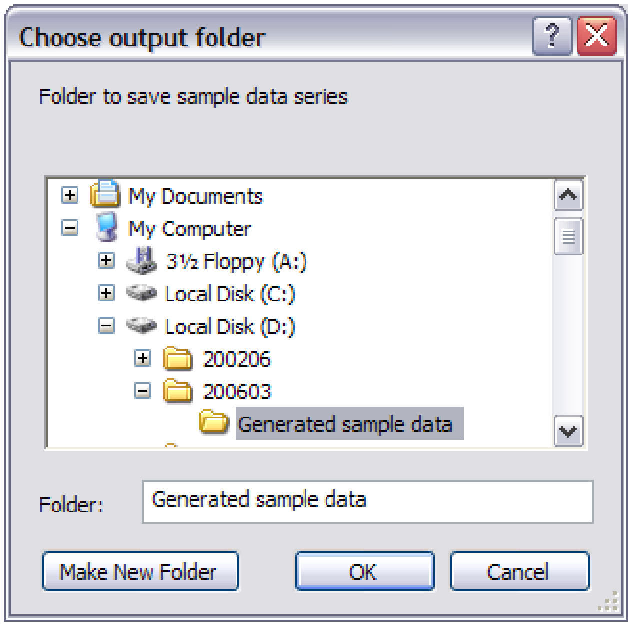
When you start the IBM Tivoli Software Signature Analyzer to create a new project, you choose a directory to be used as a *workspace*. For more information about what a workspace is, see "Workspace" on page 3. The default workspace is the tool installation directory, but it is recommended that you specify a different location.

Generating sample data

In order to get familiar with the application, you can generate sample data and simulate a complete process using the samples.

To start working with the IBM Tivoli Software Signature Analyzer using sample data:

1. From the toolbar, click: to create series of sample data.
   The Choose output folder dialog opens:



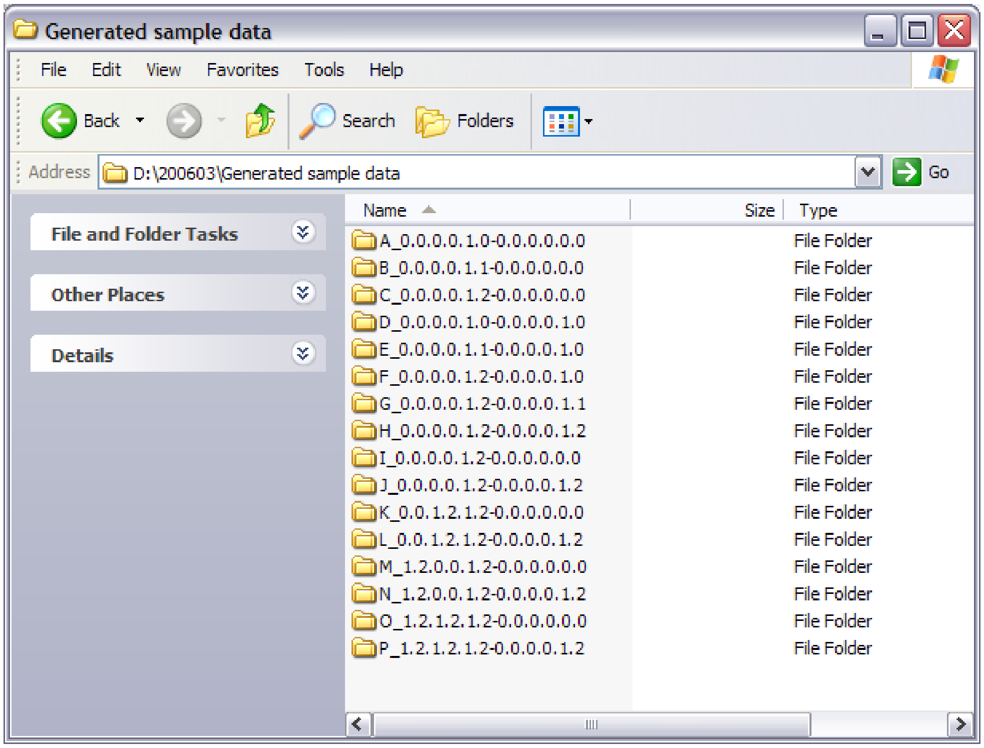
Choose output folder

Folder to save sample data series

Folder: Generated sample data

2. Browse to an **empty** directory where you want to store the data that will be generated.

Our example shows a folder containing 16 sub-folders whose names start with the letters **A** to **P**, as shown in the following dialog:



Name	Size	Type
A_0.0.0.0.1.0-0.0.0.0.0.0		File Folder
B_0.0.0.0.1.1-0.0.0.0.0.0		File Folder
C_0.0.0.0.1.2-0.0.0.0.0.0		File Folder
D_0.0.0.0.1.0-0.0.0.0.1.0		File Folder
E_0.0.0.0.1.1-0.0.0.0.1.0		File Folder
F_0.0.0.0.1.2-0.0.0.0.1.0		File Folder
G_0.0.0.0.1.2-0.0.0.0.1.1		File Folder
H_0.0.0.0.1.2-0.0.0.0.1.2		File Folder
I_0.0.0.0.1.2-0.0.0.0.0.0		File Folder
J_0.0.0.0.1.2-0.0.0.0.1.2		File Folder
K_0.0.1.2.1.2-0.0.0.0.0.0		File Folder
L_0.0.1.2.1.2-0.0.0.0.1.2		File Folder
M_1.2.0.0.1.2-0.0.0.0.0.0		File Folder
N_1.2.0.0.1.2-0.0.0.0.1.2		File Folder
O_1.2.1.2.1.2-0.0.0.0.0.0		File Folder
P_1.2.1.2.1.2-0.0.0.0.1.2		File Folder

The content of each folder simulates the distribution of a set of software products and common components onto 15 computers. Each folder contains the 15 sub-folders corresponding to the simulated harvest data sets.

Data generated for the different series (from A to P) uses the same product names, but assumes a different structure of content for each product. Therefore, do not mix generated data from one series with data from another series into the same collection of analyzed data. Each series must be analyzed separately.

In the examples described in the following sections we only analyze the most complex series (the **P-series**).

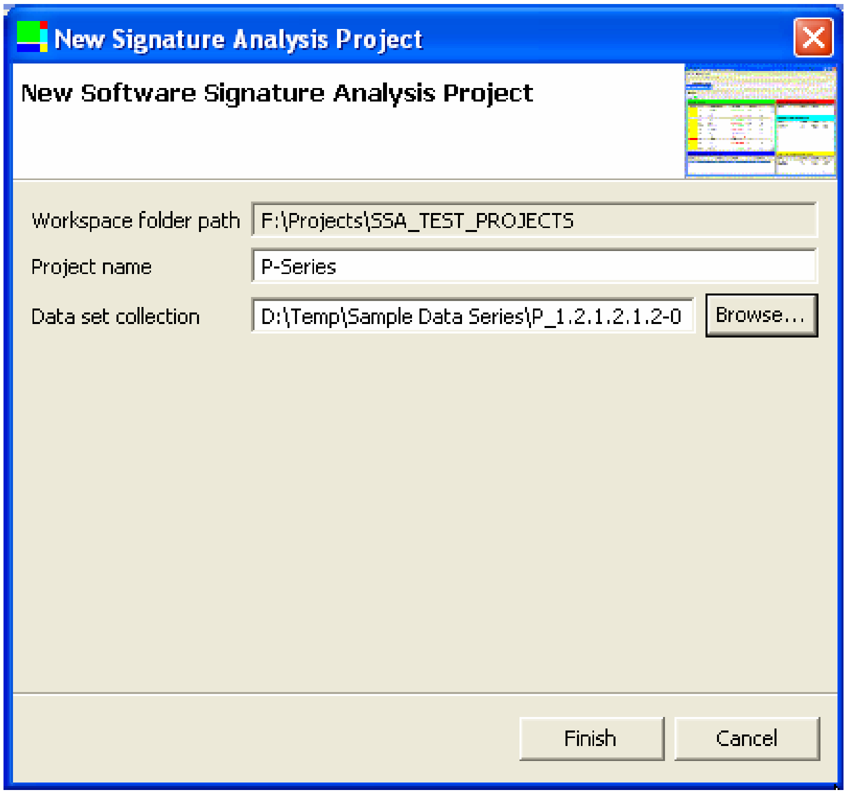
Chapter 4. Creating a New Project

To start the analysis of the data sets, first of all, you have to create a *project* in your workspace. For more information about what a project is, see "Projects" on page 4.

You can have only one open project at a time, therefore, before creating a new one, ensure that no other project is currently open in the workspace.

To create a new project, perform the following steps:

1. From the toolbar, click 🗀.
   The New Signature Analysis Project dialog opens:



New Signature Analysis Project

New Software Signature Analysis Project

Workspace folder path: F:\Projects\SSA_TEST_PROJECTS

Project name: P-Series

Data set collection: D:\Temp\Sample Data Series\P_1.2.1.2.1.2-0

2. Enter the required information in the blank fields:

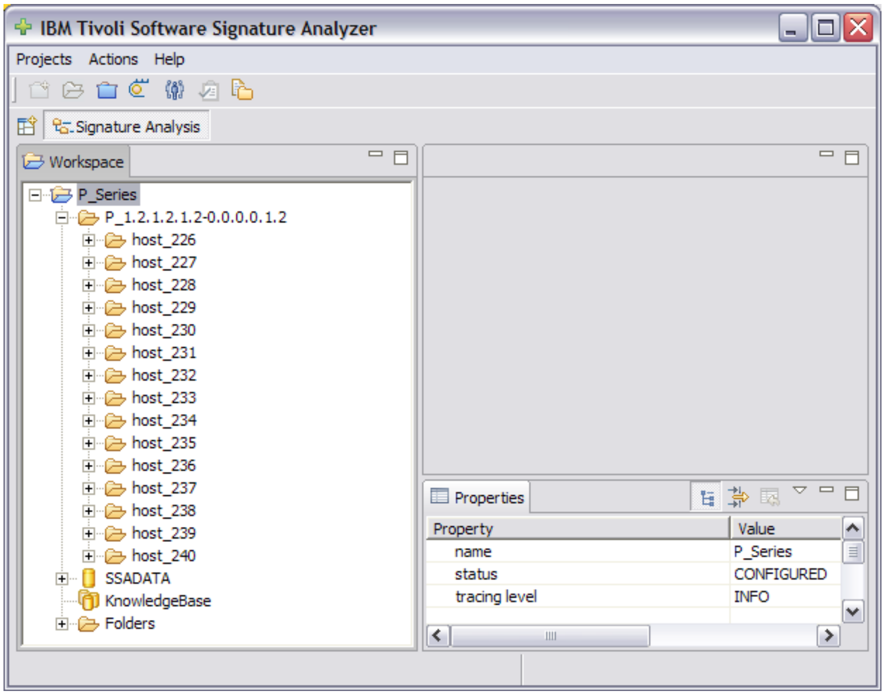
Project name
You can give whatever name you like to your project, provided it is a valid folder name in the underlying operating system. In our example, we use **P-series**, because this project is aimed at analyzing the last and most complex series of generated data.

Data set collection
To fill this field, browse to the parent directory containing all the sub-directories where the data you want to analyze is located. In our example, browse to the P-series folder created in the data generation phase (for more details, see Chapter 3, "Acquiring Data," on page 11.)

Note: This parent folder must contain **only** the sub-folders (one for each scanned computer) that in turn contain the text files produced by the Software Signature Analyzer harvesting tool.

3. Click **Finish** to start the project configuration process.

The IBM Tivoli Software Signature Analyzer now creates the project folder under the current workspace as shown in the following dialog:



Property	Value
name	P_Series
status	CONFIGURED
tracing level	INFO

The tree on the left part of the screen is the **Workspace** view. The following root nodes are displayed in the workspace tree:

P-Series
The name of the project, and the directory where all the results of data analysis are stored.

SSAData
The database that contains a set of tables. At this stage all tables are still empty because no data has been read from the harvest data set collection. For more information about these tables, see: “Database tables” on page 5.

Knowledge Base
The archive that is shared among all projects in the workspace. It is an XML document conforming to a rich and flexible schema. For more information about it, see: “Knowledge Base” on page 9.

Several objects displayed as nodes of the workspace tree have associated properties that are automatically displayed in the **Properties** view located at the bottom-right of the screen when you select the object. In particular, if you select a project node, you can see its state.

In our example, the project P-series is in the CONFIGURED state. This means that links have been persisted about the file system location of the data set collection and an embedded database, represented by the SSADATA object in the project tree, has been created and primed with the database schema.

The only action you can perform on a project that is in CONFIGURED state is the **Process harvest data** action, after which the project is in DATA_NORMALIZED state. For this step, see: Chapter 5, “Processing Harvest Data,” on page 17.

Chapter 5. Processing Harvest Data

After creating a project, you can process the harvest data and populate the project database with normalized data derived from the harvest data set collection.

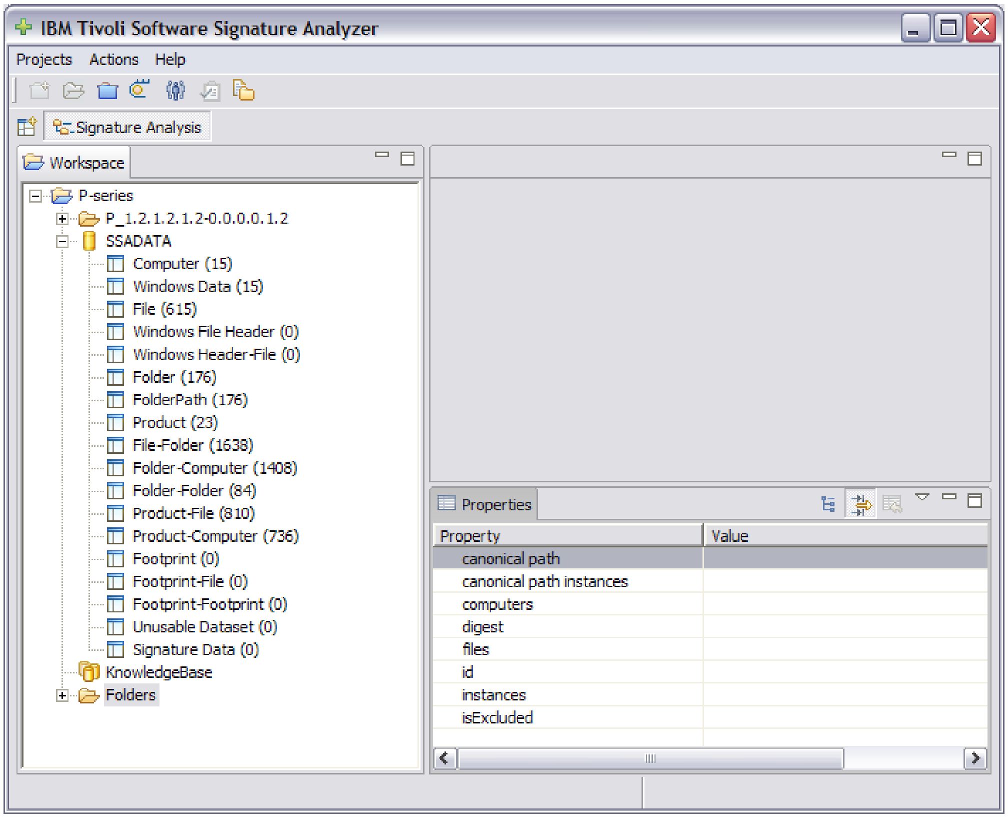
To start data processing, select your project folder (in our example P-Series) and click: from the toolbar.

Verifying the results of data processing

At the end of the process, the folders containing the data sets have changed names to reflect the hostnames of the computers whose harvest data they contain.

Note: If the IBM Tivoli Software Signature Analyzer finds an invalid data set (for example because it contains a corrupted file or because the harvesting process was not successful) it marks it with an error icon: and ignores its content. If this happens, you can investigate the problem by expanding the data set, right-clicking on any contained file, and selecting **View file** to open its content in a text browser window.

The updated tables appear listed under the SSADATA database object, as shown in the following dialog:



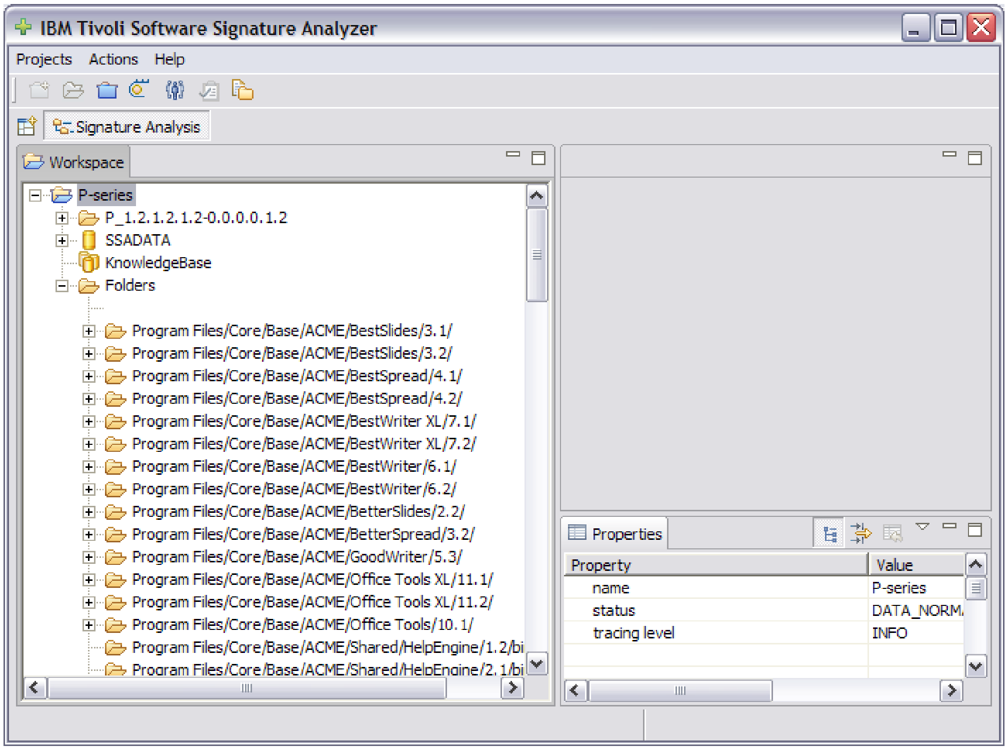
- P-series
  - P_1.2.1.2.1.2-0.0.0.0.1.2
  - SSADATA
    - Computer (15)
    - Windows Data (15)
    - File (615)
    - Windows File Header (0)
    - Windows Header-File (0)
    - Folder (176)
    - FolderPath (176)
    - Product (23)
    - File-Folder (1638)
    - Folder-Computer (1408)
    - Folder-Folder (84)
    - Product-File (810)
    - Product-Computer (736)
    - Footprint (0)
    - Footprint-File (0)
    - Footprint-Footprint (0)
    - Unusable Dataset (0)
    - Signature Data (0)
  - KnowledgeBase
  - Folders

Property	Value
canonical path	
canonical path instances	
computers	
digest	
files	
id	
instances	
isExcluded	

Each counter indicates the number of items contained in the associated table. For more information about these tables, see: “Database tables” on page 5.

Another object appears in the workspace view after the project has reached the DATA_NORMALIZED state, namely the **Folders** collection. Under this object, each *unique* folder identified during data acquisition is represented by a folder icon whose label reflects the folder *canonical path*. For more information about unique folders, see: "Unique folders" on page 8.

The following dialog shows some of the unique folders derived from the analysis of the sample data series:



```
P-series
    P_1.2.1.2.1.2-0.0.0.0.1.2
    SSADATA
    KnowledgeBase
    Folders
        Program Files/Core/Base/ACME/BestSlides/3.1/
        Program Files/Core/Base/ACME/BestSlides/3.2/
        Program Files/Core/Base/ACME/BestSpread/4.1/
        Program Files/Core/Base/ACME/BestSpread/4.2/
        Program Files/Core/Base/ACME/BestWriter XL/7.1/
        Program Files/Core/Base/ACME/BestWriter XL/7.2/
        Program Files/Core/Base/ACME/BestWriter/6.1/
        Program Files/Core/Base/ACME/BestWriter/6.2/
        Program Files/Core/Base/ACME/BetterSlides/2.2/
        Program Files/Core/Base/ACME/BetterSpread/3.2/
        Program Files/Core/Base/ACME/GoodWriter/5.3/
        Program Files/Core/Base/ACME/Office Tools XL/11.1/
        Program Files/Core/Base/ACME/Office Tools XL/11.2/
        Program Files/Core/Base/ACME/Office Tools/10.1/
        Program Files/Core/Base/ACME/Shared/HelpEngine/1.2/bi
        Program Files/Core/Base/ACME/Shared/HelpEngine/2.1/bi
```

Property	Value
name	P-series
status	DATA_NORM.
tracing level	INFO

Excluding spurious folders

Folders identified in the data normalization process are key to the successive analysis phases.

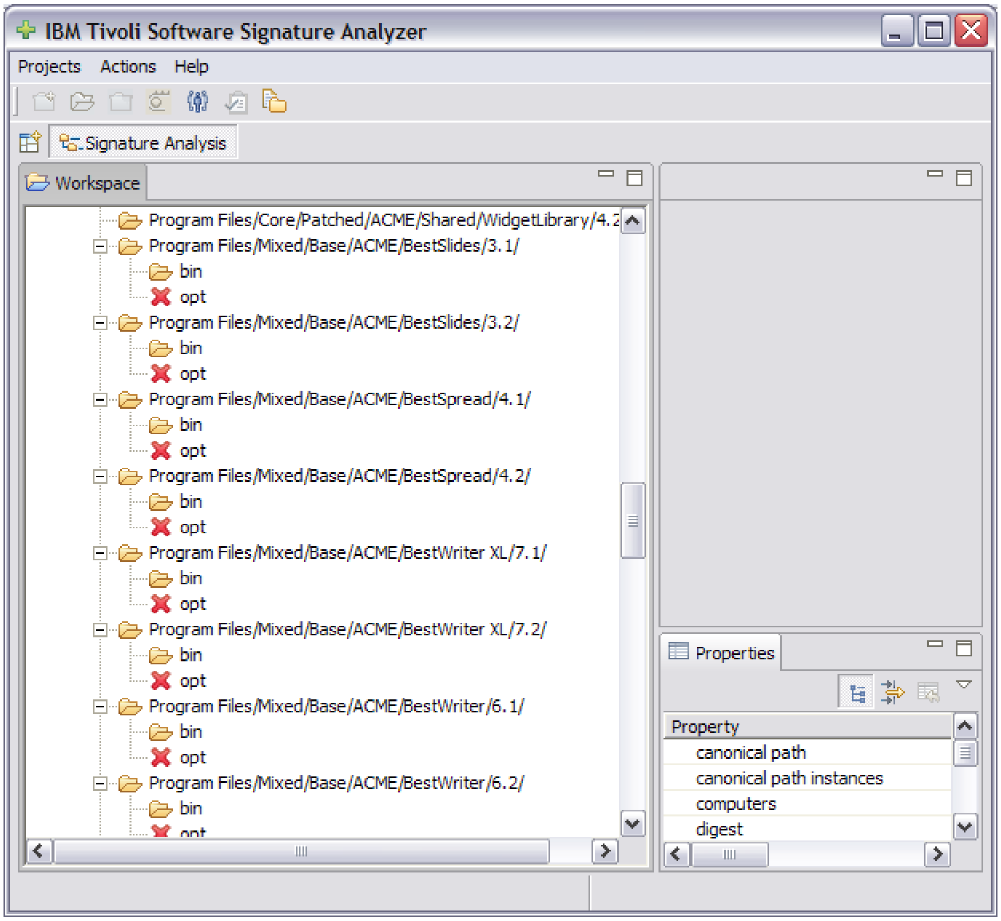
By inspecting the identified folders, you can often determine that a folder is not part of any installed product, on the basis of the folder pathname or number of instances.

In this case, set these folders as *excluded* to simplify the remaining process. As an example, assume that folders containing some optional documentation material have been selected because they contain some executable files needed for the visualization of documents. If you know that these folders are not used to identify the product, for example because they are optional or because their content is actually created after the product installation, you can exclude them.

To exclude one or more folders, right-click on the folder, and select **Set excluded** from the drop-down menu. You can also perform this action against a selection of multiple folders.

As an example, working with the P-Series simulated data, you may want to exclude all the folders whose pathname ends with **opt**.

The following dialog shows the list of folders after some have been excluded:



```
Program Files/Core/Patched/ACME/Shared/WidgetLibrary/4.2
Program Files/Mixed/Base/ACME/BestSlides/3.1/
    bin
    opt
Program Files/Mixed/Base/ACME/BestSlides/3.2/
    bin
    opt
Program Files/Mixed/Base/ACME/BestSpread/4.1/
    bin
    opt
Program Files/Mixed/Base/ACME/BestSpread/4.2/
    bin
    opt
Program Files/Mixed/Base/ACME/BestWriter XL/7.1/
    bin
    opt
Program Files/Mixed/Base/ACME/BestWriter XL/7.2/
    bin
    opt
Program Files/Mixed/Base/ACME/BestWriter/6.1/
    bin
    opt
Program Files/Mixed/Base/ACME/BestWriter/6.2/
    bin
    opt
```

Property
canonical path
canonical path instances
computers
digest

Automatic exclusion of folders

When you analyze a large collection of harvested data sets, the number of unique folders can be very high, up to several thousands.

In this case you can apply some general criteria for the automatic exclusion of folders. Right-click on the root folder object, and select: from the drop-down menu. The IBM Tivoli Software Signature Analyzer examines all the folders and automatically sets a folder in the excluded state when the following conditions are both satisfied:

- There is no product definition associated to any file within the folder, AND
- The folder has no canonical path OR there is one single instance of the folder.

Chapter 6. Recreating the Product Structure

In the reconstruction phase the Software Signature Analyzer re-creates the structure of a product installation tree. The *unique folders* identified during the data normalization process and their *canonical paths* are a good starting point to achieve the goal.

For more information about unique folders and canonical path, see: "Unique folders" on page 8.

From folders to footprints

A footprint is the logical object created by the Software Signature Analyzer to represent the core set of files that are part of one or more unique folders. Other files that are present only in some of these folders, but not in all of them, are classified as *optional* and are not included in the generated footprint. Other folders containing a common set of files with the same names but different content (size or checksum) are classified as *variants*, produced by applying different updates to a common code base. The variability of content that is associated to the core files is retained when generating footprints from unique folders: the footprint is a unique combination of content (size and checksum) associated to the core files of a group of folders. Therefore, folders represented by the same footprint have *exactly* the same content associated to the core files even though they might differ in the optional files, which are not captured by the footprint.

In some cases multiple folders might have the same path trailing part, while differing in the path leading part. This might occur because they are installed under different parent directories. As an example, with the P-Series generated data, some products are installed under: `Program Files/Core/...` while other products are installed under: `Program Files /Mixed/...`. When some of the installed folders have the same *core files* in common (checksum), the Software Signature Analyzer creates a single object (footprint) to represent them all. It removes the path portion that is not common to all the instances, and keeps only the common trailing portion, which becomes the footprint *canonical path*. On the contrary, if the files do not have the same checksum, the folders are variants of the same entity and the application represents them as distinct objects.

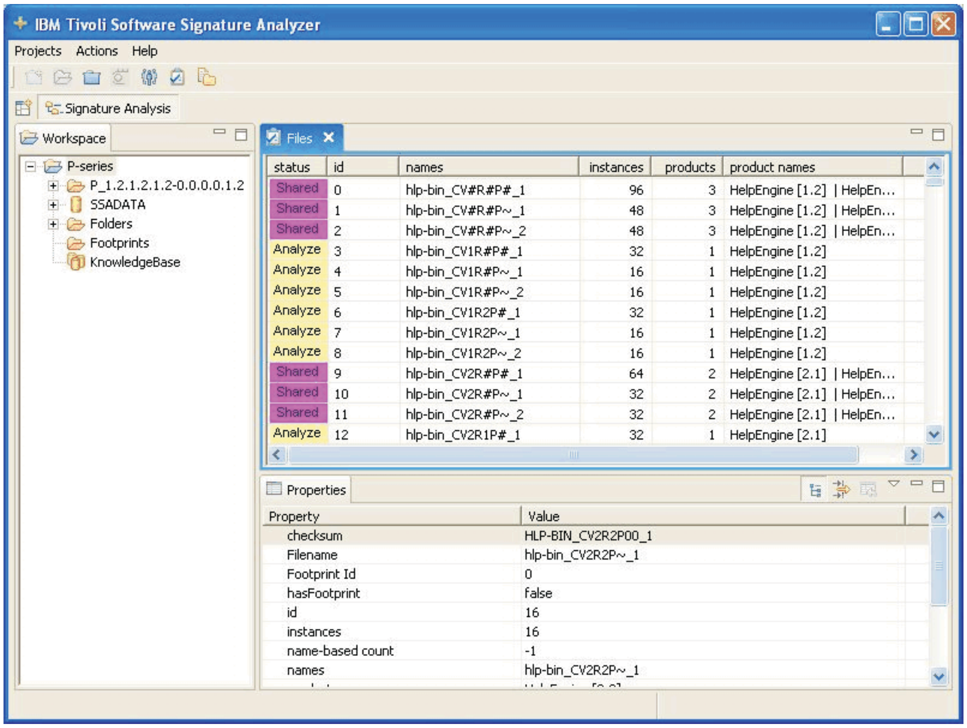
Generating footprints

When the project is in the DATA_NORMALIZED state, you can start generating footprints. For more information about footprints, see "Footprints" on page 8.

To generate footprints perform the following steps:

1. Select your project.
2. From the **Actions** menu, select **Open file list**.

   The following dialog shows the Files view in the upper-right portion of the screen:



status	id	names	instances	products	product names
Shared	0	hlp-bin_CV#R#P#_1	96	3	HelpEngine [1.2] | HelpEn...
Shared	1	hlp-bin_CV#R#P~_1	48	3	HelpEngine [1.2] | HelpEn...
Shared	2	hlp-bin_CV#R#P~_2	48	3	HelpEngine [1.2] | HelpEn...
Analyze	3	hlp-bin_CV1R#P#_1	32	1	HelpEngine [1.2]
Analyze	4	hlp-bin_CV1R#P~_1	16	1	HelpEngine [1.2]
Analyze	5	hlp-bin_CV1R#P~_2	16	1	HelpEngine [1.2]
Analyze	6	hlp-bin_CV1R2P#_1	32	1	HelpEngine [1.2]
Analyze	7	hlp-bin_CV1R2P~_1	16	1	HelpEngine [1.2]
Analyze	8	hlp-bin_CV1R2P~_2	16	1	HelpEngine [1.2]
Shared	9	hlp-bin_CV2R#P#_1	64	2	HelpEngine [2.1] | HelpEn...
Shared	10	hlp-bin_CV2R#P~_1	32	2	HelpEngine [2.1] | HelpEn...
Shared	11	hlp-bin_CV2R#P~_2	32	2	HelpEngine [2.1] | HelpEn...
Analyze	12	hlp-bin_CV2R1P#_1	32	1	HelpEngine [2.1]

Property	Value
checksum	HLP-BIN_CV2R2P00_1
Filename	hlp-bin_CV2R2P~_1
Footprint Id	0
hasFootprint	false
id	16
instances	16
name-based count	-1
names	hlp-bin_CV2R2P~_1

File status

The Files view is a table that displays all the unique folders identified during the normalization process. The files are initially sorted so that the ones with the highest number of instances occur at the top of the list. This is the order in which the files are analyzed during the automatic footprint generation process.

Note: Re-sorting the table changes the order in which these files are displayed, but it does not change the order in which they are automatically processed. Each file is displayed by one table row, which contains the following information

status The status of the file and how suitable it is to generate footprints.

id Identification number of the file.

names All the names the file assumes across all its instances. Names are separated by vertical bars.

instances
How many instances of each file have been found.

products
How many products are associated to that file.

product names
The name of the products that install the file (including product release and version). Names are separated by vertical bars.

The following color codes identify the status of each file. The status can change during footprint generation:

Analyze

The file is candidate to become a *source file* in the generation of a new footprint. It means it is a file that can be used as a guide in the generation process.

Done

The file is already part of one or more footprints.

Excluded

These might be files contained in folders manually excluded before or during the footprint generation process. Alternatively, the application considers them as optional files in some folders whose core files were selected to become part of a footprint. In both cases, these files are excluded from further consideration during the generation process.

Invariant

Marking the file as *invariant* means that the file was found with an identical content across multiple footprints that represent closely related variants of the same folder. This state is set when the file is used as a candidate source file for generating a new footprint.

Save

The file was contained in folders selected during the manual footprint generation process. When one or more files are in this state, you can right-click on the files, and either reset the *Save* status selecting **Reset footprint**, or save the footprint in the database selecting **Save footprint**.

Common

The file is contained in folders with different content, possibly associated to different products or different versions of the same product. For example, it can be a launcher that is used by different applications, or it can be a file included by the application installer to configure or to uninstall the application. A file with these characteristics is marked as *Common* before the footprint generation process. The status may change to *Excluded* or *Done* during the footprint generation process.

Unusable

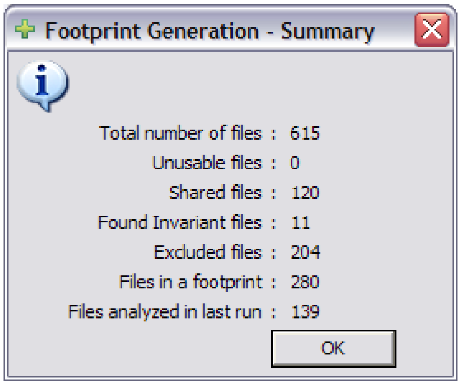
The file cannot be used in the generation process because the Software Signature Analyzer could not find a prevalent name from the list of names that the file assumes across its instances.

By selecting a file from the Files view, you can see its details listed below in the Properties table.

Generating Footprints automatically

To start the automatic footprint generation process, right-click anywhere within the Files view area and select **Generate footprints**.

When the footprint generation process is complete, the following window opens, detailing the number of files and their categories:



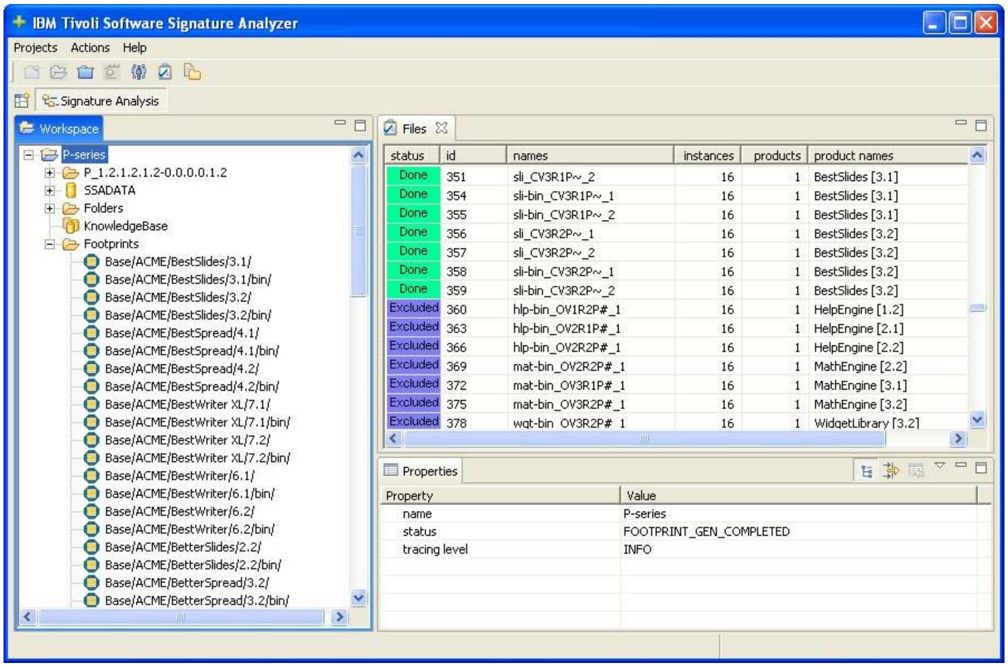
Footprint Generation - Summary

Total number of files :	615
Unusable files :	0
Shared files :	120
Found Invariant files :	11
Excluded files :	204
Files in a footprint :	280
Files analyzed in last run :	139

When the process has completed the files are in one of the following states:

- Done
- Common
- Excluded
- Invariant

The following dialog shows the GUI after the generation is complete:



- P-series
  - P_1.2.1.2.1.2-0.0.0.0.1.2
  - SSADATA
  - Folders
  - KnowledgeBase
  - Footprints
    - Base/ACME/BestSlides/3.1/
    - Base/ACME/BestSlides/3.1/bin/
    - Base/ACME/BestSlides/3.2/
    - Base/ACME/BestSlides/3.2/bin/
    - Base/ACME/BestSpread/4.1/
    - Base/ACME/BestSpread/4.1/bin/
    - Base/ACME/BestSpread/4.2/
    - Base/ACME/BestSpread/4.2/bin/
    - Base/ACME/BestWriter XL/7.1/
    - Base/ACME/BestWriter XL/7.1/bin/
    - Base/ACME/BestWriter XL/7.2/
    - Base/ACME/BestWriter XL/7.2/bin/
    - Base/ACME/BestWriter/6.1/
    - Base/ACME/BestWriter/6.1/bin/
    - Base/ACME/BestWriter/6.2/
    - Base/ACME/BestWriter/6.2/bin/
    - Base/ACME/BetterSlides/2.2/
    - Base/ACME/BetterSlides/2.2/bin/
    - Base/ACME/BetterSpread/3.2/
    - Base/ACME/BetterSpread/3.2/bin/

status	id	names	instances	products	product names
Done	351	sli_CV3R1P~_2	16	1	BestSlides [3.1]
Done	354	sli-bin_CV3R1P~_1	16	1	BestSlides [3.1]
Done	355	sli-bin_CV3R1P~_2	16	1	BestSlides [3.1]
Done	356	sli_CV3R2P~_1	16	1	BestSlides [3.2]
Done	357	sli_CV3R2P~_2	16	1	BestSlides [3.2]
Done	358	sli-bin_CV3R2P~_1	16	1	BestSlides [3.2]
Done	359	sli-bin_CV3R2P~_2	16	1	BestSlides [3.2]
Excluded	360	hlp-bin_OV1R2P#_1	16	1	HelpEngine [1.2]
Excluded	363	hlp-bin_OV2R1P#_1	16	1	HelpEngine [2.1]
Excluded	366	hlp-bin_OV2R2P#_1	16	1	HelpEngine [2.2]
Excluded	369	mat-bin_OV2R2P#_1	16	1	MathEngine [2.2]
Excluded	372	mat-bin_OV3R1P#_1	16	1	MathEngine [3.1]
Excluded	375	mat-bin_OV3R2P#_1	16	1	MathEngine [3.2]
Excluded	378	wgt-bin_OV3R2P#_1	16	1	WidgetLibrary [3.2]

Property	Value
name	P-series
status	FOOTPRINT_GEN_COMPLETED
tracing level	INFO

Now the project is in the FOOTPRINT_GEN_COMPLETED state.

The **Footprints** object in the Workspace view now contains as many children as the footprints created. Footprints inherit their paths from the folders that were used as input in the generation process.

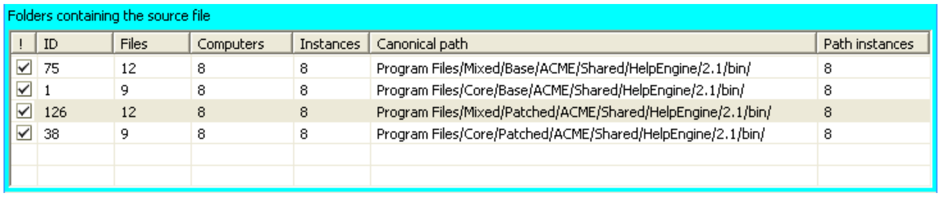
In our example, the leading portion of the folder paths (`Program Files/Core` or `Program Files/Mixed`) is eliminated during the footprint generation, because folders with both the paths contributed to each footprint. In this example the base and the patched versions of each product are installed under different paths (respectively: `.../Base` and `.../Patched`). This path difference is retained in the pathnames of the generated footprints. This situation seldom occurs when you generate footprints from real data, however, when occurs, it can be an obstacle to footprint grouping. You can remove the obstacle by manually editing the path names of the generated footprints, as explained in “Modifying footprint path names” on page 28.

Detecting spurious folders automatically

In this example, the footprint generation process completed automatically without any manual intervention.

Working with real data, it can happen that a file that is included in an installed product folder, is also found in a *spurious folder*, and this can negatively affect the generation of footprints.

For example, assume that you created a copy of a product installation tree in a different directory to run some tests and that you eliminated and added some files to that tree during the tests. The presence of one or more identical files in both the folders, can mean that the two folders are variants of an identical source. Hence, if the Software Signature Analyzer created one or two footprints from the two folders, errors would be induced in this process by the presence of the corrupted folder. To avoid this problem, the application analyzes the folders associated to any common files, to detect the possible presence of spurious folders. If spurious folders are found, the Software Signature Analyzer interrupts the automatic generation of footprints and displays a view of all the folders associated to the current source file:



Folders containing the source file

!	ID	Files	Computers	Instances	Canonical path	Path instances
☑	75	12	8	8	Program Files/Mixed/Base/ACME/Shared/HelpEngine/2.1/bin/	8
☑	1	9	8	8	Program Files/Core/Base/ACME/Shared/HelpEngine/2.1/bin/	8
☑	126	12	8	8	Program Files/Mixed/Patched/ACME/Shared/HelpEngine/2.1/bin/	8
☑	38	9	8	8	Program Files/Core/Patched/ACME/Shared/HelpEngine/2.1/bin/	8

The folders considered spurious are not selected (excluded from the footprint generation.). You can either confirm or change the suggested exclusion, and then close the dialog. The information created during by the above manual step remains available in the **Files** view, from which you can:

- Click the icon to save the footprints generated from the selected folders (if any).
- Click the icon to discard the entered information.
- Modify your selections, opening again the File-Folders view by selecting the same source file (you need to remember the file ID) and clicking **OK**.

After saving the footprint information generated from your folder selections, you can resume the footprint generation process in automatic mode by selecting the ☑ icon, or continue the manual generation, selecting a new file from the **Files** view.

Working with the file folders view

If you want to manually choose the files to exclude from footprint generation, you can follow the manual process. In this way you can browse and analyze the relationships between files and folders, and view the content of each folder, to determine which files you want to keep as source files in the footprint generation process, and, possibly, identify and exclude spurious folders.

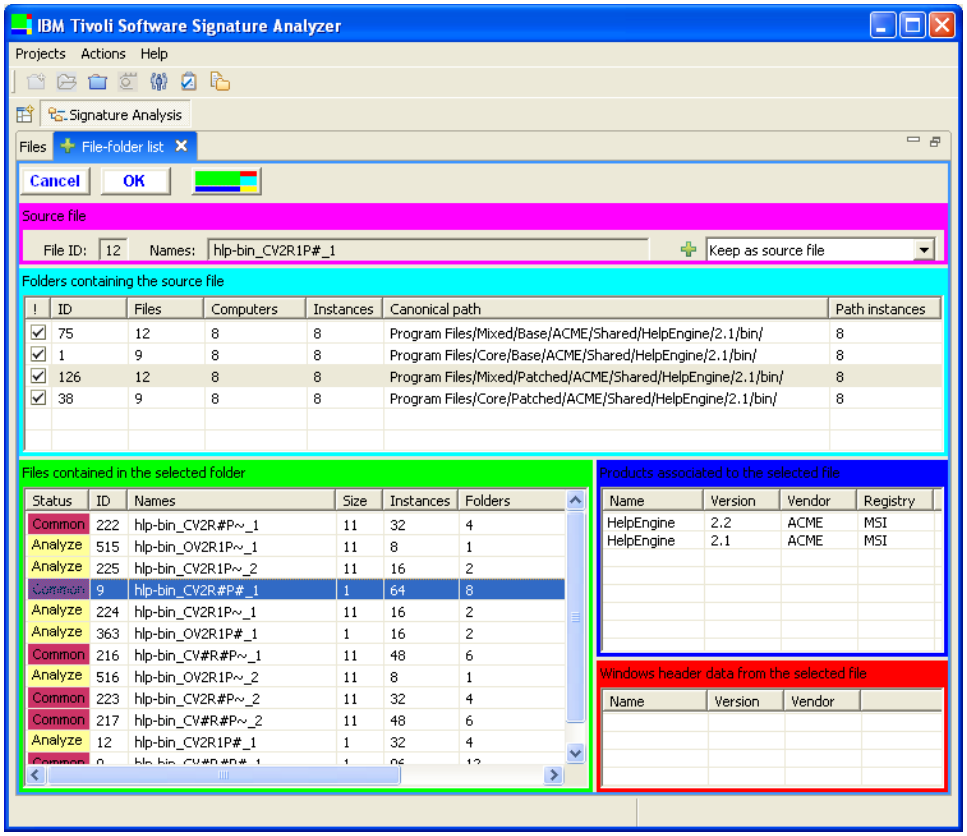
To view the information relating to a source file, perform the following steps:

1. Right-click on your project and select **Open file list**.

   The Files view opens in the upper-right part of the screen.

2. Right-click on a file, and select **Open file-folder list**.

   The view displays all the information available in the database relating to the selected file:



Source file

File ID: 12 Names: hlp-bin_CV2R1P#_1 Keep as source file

Folders containing the source file

!	ID	Files	Computers	Instances	Canonical path	Path instances
☑	75	12	8	8	Program Files/Mixed/Base/ACME/Shared/HelpEngine/2.1/bin/	8
☑	1	9	8	8	Program Files/Core/Base/ACME/Shared/HelpEngine/2.1/bin/	8
☑	126	12	8	8	Program Files/Mixed/Patched/ACME/Shared/HelpEngine/2.1/bin/	8
☑	38	9	8	8	Program Files/Core/Patched/ACME/Shared/HelpEngine/2.1/bin/	8

Files contained in the selected folder

Status	ID	Names	Size	Instances	Folders
Common	222	hlp-bin_CV2R#P~_1	11	32	4
Analyze	515	hlp-bin_OV2R1P~_1	11	8	1
Analyze	225	hlp-bin_CV2R1P~_2	11	16	2
Common	9	hlp-bin_CV2R#P#_1	1	64	8
Analyze	224	hlp-bin_CV2R1P~_1	11	16	2
Analyze	363	hlp-bin_OV2R1P#_1	1	16	2
Common	216	hlp-bin_CV#R#P~_1	11	48	6
Analyze	516	hlp-bin_OV2R1P~_2	11	8	1
Common	223	hlp-bin_CV2R#P~_2	11	32	4
Common	217	hlp-bin_CV#R#P~_2	11	48	6
Analyze	12	hlp-bin_CV2R1P#_1	1	32	4

Products associated to the selected file

Name	Version	Vendor	Registry
HelpEngine	2.2	ACME	MSI
HelpEngine	2.1	ACME	MSI

Windows header data from the selected file

Name	Version	Vendor

Note: The file-folder view can be useful for navigating the database information, even after the footprint generation process has completed. However in this case you can access the information only for browsing, you will not be allowed to change any data stored in the database.
The File-folder view shows the following sections:

Source file section
Contains the IDs and names of the source file (it might have multiple

names across different folders). The drop-down menu allows you to select whether to keep or to discard the file as a source file for the footprint generation.

Folders containing the source file

Contains information about all the folders where the selected file has been found. In particular the table details:

- The folder ID
- The number of files the folder contains, in addition to the selected one
- The number of computers where the folder has been found
- The overall number of folder instances found throughout all the harvested dataset
- The canonical path of the folder.
- The number of folder instances, whose name matches the canonical pathname.

Analyzing the content of a folder

To view the content of a folder, select the folder from the corresponding table. As a result the **Files contained in the selected folder** table populates with all the files contained in that folder (among which, of course, there is also the file you started from). Moreover, when you select a file in this table, additional information is displayed in the adjacent tables about, respectively:

- Any registered product that could be correlated to the selected file
- The product information extracted from the file header (Windows only).

When the purpose of your analysis is to find the best source files for the generation of footprints, you should always select as a source, the first file marked with the Analyze icon.

You can manually adjust the relative sizes of the window sections by dragging their internal borders. If you want to restore the original proportions between the tables, click .

Excluding folders manually

After analyzing the folder content, you can determine that some folders have been accidentally included in the harvest dataset, and you want to exclude them from the footprint generation. In this case, from the **Folder containing the source file** table, clear the check box relating to the undesired folder.

For example: you might find a file that is contained in five folders. Four of them contain 9 to 12 more files, while another folder contains only one more file. Moreover, only one instance is found of the latter folder, while several instances of the other folders are found across the scanned computers. Therefore, this folder is likely to be a spurious one. It might be the remains of an incomplete uninstallation or an old backup. In this case, you can clear the check box relating to the spurious folder to exclude it from the footprint generation.

When you have completed the necessary changes, click **OK** to save them in the Files view. If you want to exit without saving any setting, click **Cancel**.

In the Files view you can see that, as a result of your selections, some files have changed status. Some files are now marked as

Excluded

These files were excluded because they are not contained in all the active folders (folders that were not manually excluded). As an example, a file is excluded if it is found only in a folder that was manually excluded (spurious folder). Or it can be excluded because it is not common to all the folders associated to the current *source file*.

Save

These files are included in the footprint to be generated, because they are common to all the folders associated to the current *source file.*

The information saved in the Files view can be then either discarded or used to generate footprints. Both the options are available from the drop-down menu of the Files view. To perform the footprint generation process in manual mode, repeat the above procedure using as source files the first files marked with the Analyze icon. However, at any time during the manual procedure, you can switch to automatic footprint generation by following the steps described in: “Generating Footprints automatically” on page 23.

Excluding a source file manually

In the footprint generation process, it is very important that files contained in folders that are either loosely correlated (as in the case of folders belonging to different versions of a product) or even totally unrelated (as in the case of utility files deployed by a common installation technology) be detected so that they are not used as source files. Therefore, you can define a file either as **common**, or as **source**, to exclude or include it in the footprint generation process. To define a file, use the drop-down menu located on the upper-right part of the dialog. However, this is an advanced feature that should be used with care.

A file (provided it is associated to a registered product) is marked as **common** if it is found in folders related to different products. For eaxmple: it might happen that a file actually installed by multiple unrelated products be still considered as a suitable candidate source file in the footprint generation process. However, in this case the application detects that the folders containing this file are very different in their content. Under this condition the automatic generation process interrupts, and the application prompts you to analyze the folders that can be spurious. After the analysis, your conclusion might be that there are folders that are effectively unrelated, although there is no evidence of one or more folders being corrupted. In this case, it is inappropriate to exclude these folders. Instead, you should mark the file as **common**, thus excluding it from the use as a source file.

To mark the source file as **common**, use the drop-down menu located in the upper-right part of the dialog.

Note: This action does not leave any pending information in the Files view, because the attribute change to the file is immediately stored in the database.

Modifying footprint path names

Folders usually have an associated *canonical path*. The canonical path is derived by eliminating from the folder pathname the leading tokens that vary across all instances of the same folder. This process also takes place during footprint generation: if the canonical paths of multiple folders relating to a footprint still include different tokens in their leading part, those leading tokens are eliminated from the footprint canonical path. However, it might occur that pathnames attributed to a footprint still carry some leading tokens that reflect the random choice of a parent directory during the installation of products.

This is not a problem when the canonical paths of related footprints that are analyzed together have a common leading part. It would, however, be a problem if footprints relating to the same or similar products have different leading tokens in the pathnames, because these footprints would appear as not-related to each other. This situation is clearly illustrated by the set of footprints obtained after the generation process, where footprints derived from the same products are found under both of the following paths:

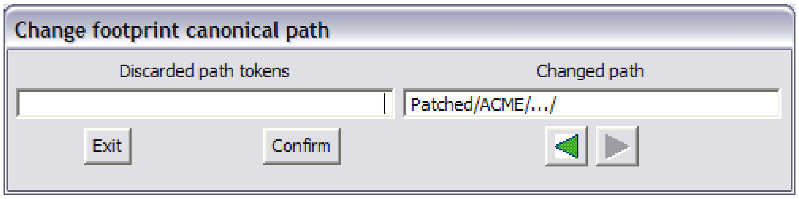
- `Base/ACME`
- `Patched/ACME`

In this case, you need to manually remove some leading tokens to ensure a proper grouping of these footprints (for more details about grouping, see Chapter 7, “Grouping Footprints,” on page 31.)

To modify paths names, perform the following steps:

1. Select all the footprints whose paths need to be edited.
2. Right-click within the workspace view and select **Edit path** from the drop-down menu.

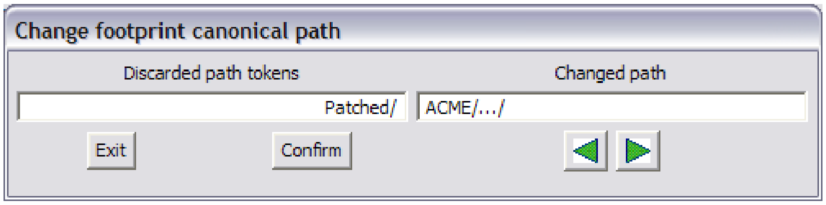
   The following dialog opens, when you select all the footprints having the *Patched/* leading token:

   
   Change footprint canonical path

   Discarded path tokens:

   Changed path: Patched/ACME/.../

3. To eliminate the first token from the path, click the left arrow, and the removed part is displayed in the **Discarded path tokens** field:

   
   Change footprint canonical path

   Discarded path tokens: Patched/

   Changed path: ACME/.../

4. Click the left arrow once for each token you want to eliminate.
5. Click **Confirm** to save the new path name.
6. Perform the same process to eliminate the `Base/` leading token from the remaining footprints.

Undo footprint generation

You can undo footprint generation as follows:

1. From the Workspace view, right-click on **Footprints**
2. From the drop-down menu, select **Undo footprint generation.**

The project resets to the DATA_NORMALIZED state and all footprint-related information is eliminated from the database.

Chapter 7. Grouping Footprints

A footprint captures information related to a *single folder* of the product tree. The tree can include sub-folders that contain executable files critical to the identification of the product. Therefore, by grouping footprints you re-create as much information as possible about the *parent-child* relationships that exist among footprints.

The tree structure resulting from the grouping process is more complex than the tree structure of a specific instance of the associated product, because all the existing variants of a folder (each one corresponding to a different maintenance level and represented by a distinct footprint) need to be represented in the tree.

By combining the information of all the related footprints in the same tree, the IBM Tivoli Software Signature Analyzer helps you to derive product and signature definitions in the next phase. Most of the process is run automatically by the tool, however it might occur that two or more footprints are *siblings* under a common path node, yet there is no corresponding footprint (parent) associated to that node. In these cases, the tool prompts you to make grouping decisions, which may result in the ad-hoc generation of a grouping node for the sibling footprints.

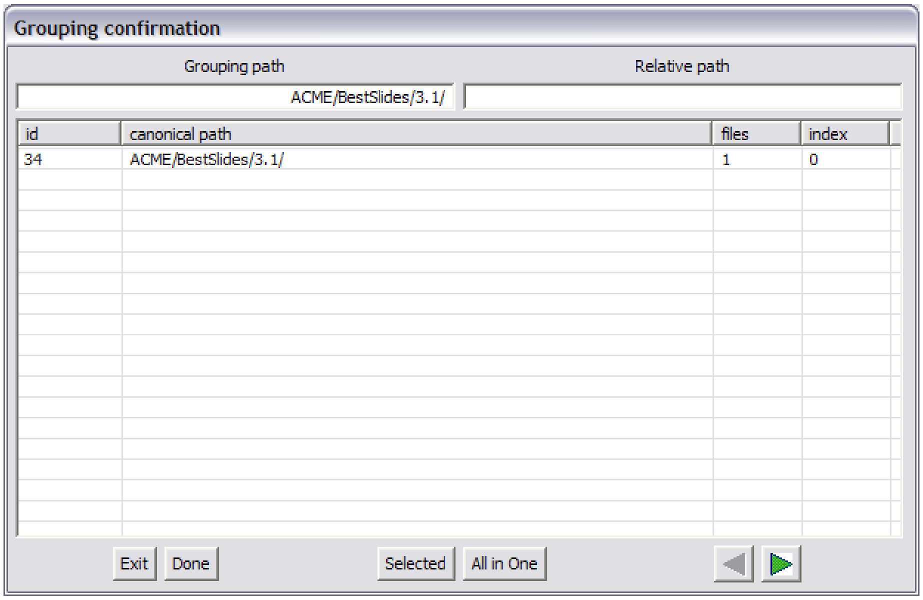
When the project is in the FOOTPRINT_GEN_COMPLETED state, you can start the footprint grouping process.

Start grouping

To start grouping the footprints perform, the following steps:

1. Right-click on **Footprints**, and select **Group footprints** from the drop-down menu.

   The IBM Tivoli Software Signature Analyzer silently reconstructs the parent-child relationships among footprints. It automatically creates some tree nodes to group footprints that have the same canonical path and therefore are variants of the same base content. When two or more footprints are *siblings* under a common path node, the Grouping confirmation dialog opens showing an initial grouping path:


Grouping confirmation

Grouping path: ACME/BestSlides/3.1/

Relative path:

id	canonical path	files	index
34	ACME/BestSlides/3.1/	1	0

In this dialog you can define the path node under which to create the group. The text fields in the upper part of the dialog display, respectively, the path *leading* and *trailing* parts of the folder containing the siblings. While the table below lists all the siblings whose canonical paths start with the path tokens displayed in the Grouping path field.

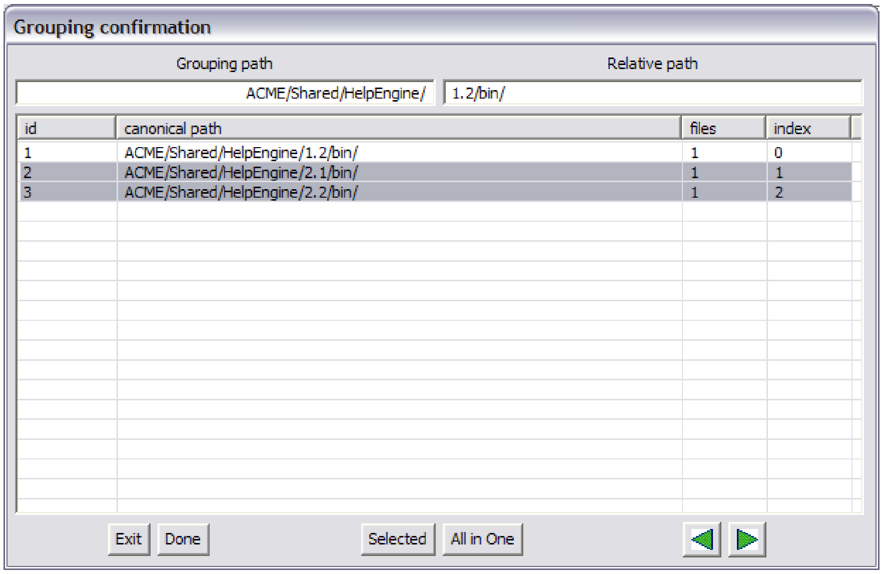
More in detail, the dialog displays different types of paths:

Grouping path
The path portion you leave in the grouping path field must be entirely contained in the leading part of the canonical paths of all the footprints belonging to this group.

Relative path
This field contains the trailing part of the canonical path.

The content of the two path fields at the top of the dialog dynamically change when you click the left or the right arrow at the bottom of the dialog. Initially, when the whole path of the source footprint is displayed in the Grouping path field, only that footprint is shown within the table. When you move one or more tokens from the Grouping path to the Relative path field, the table displays more footprints, whose canonical paths contain the grouping path.

2. Click ▶ until you find a group that has the desired scope. For example: to group footprints that relate to the same version of a product, regardless of the product release, use the arrows at the bottom to select the desired grouping path.
3. In this example, three footprints are displayed in the table when the last token is removed from the Grouping path, and they are all related to the same product version. Click **All in One** to create a group containing all the displayed footprints.
4. In more complex cases, obtaining the desired grouping of footprints is not possible with the above process. For example, the footprints displayed in the table might relate to different versions, while you do not want groups to span across version boundaries. In this case, manually select a subset of the displayed footprints you want to form a new group, and click **Selected**.

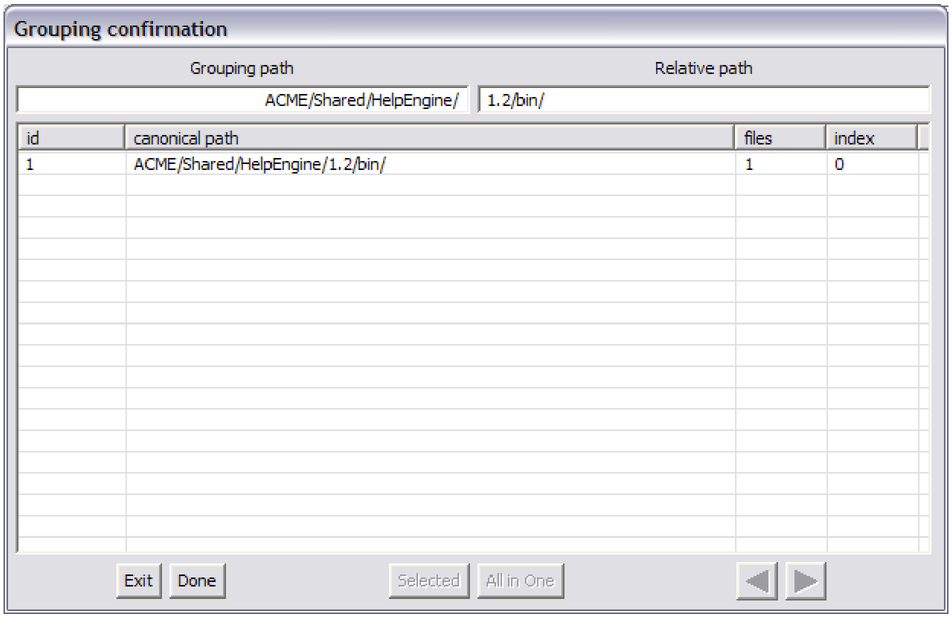


Grouping path: ACME/Shared/HelpEngine/

Relative path: 1.2/bin/

id	canonical path	files	index
1	ACME/Shared/HelpEngine/1.2/bin/	1	0
2	ACME/Shared/HelpEngine/2.1/bin/	1	1
3	ACME/Shared/HelpEngine/2.2/bin/	1	2

The selected footprints are used to form the new group and disappear from the table:



Grouping path: ACME/Shared/HelpEngine/

Relative path: 1.2/bin/

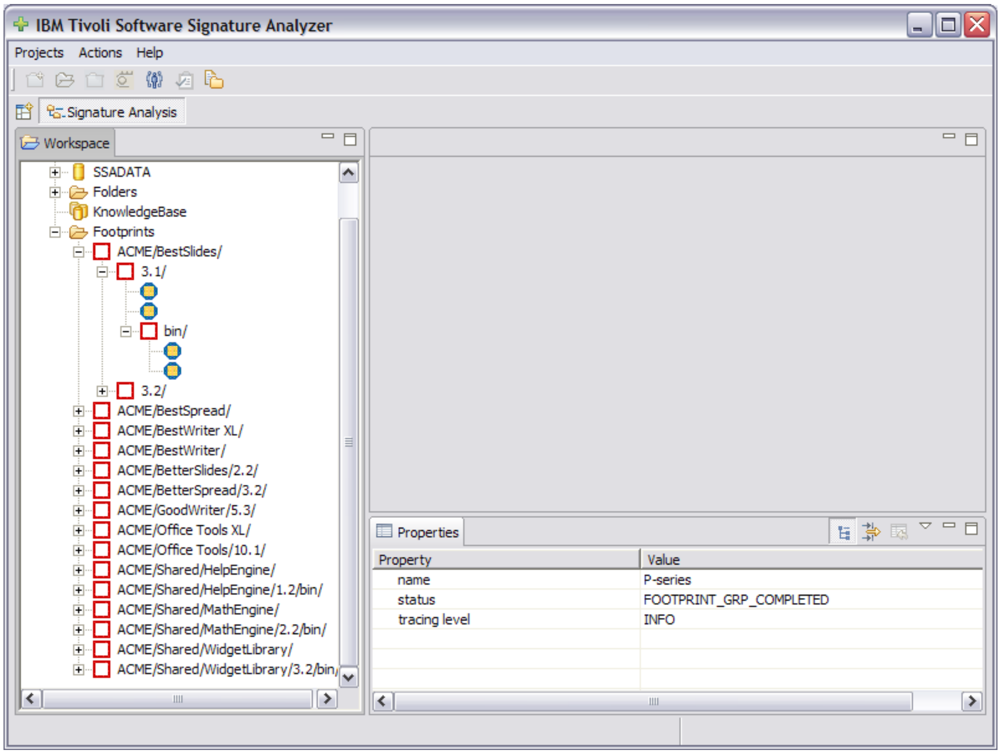
id	canonical path	files	index
1	ACME/Shared/HelpEngine/1.2/bin/	1	0

5. Click **Done** to proceed with new footprints, because you cannot group the remaining one with any other.

Unless you press the **Exit** button, the grouping process continues with the analysis of other footprints and you are prompted again if any of these footprints have siblings.

Note: If you want to interrupt the grouping, click **Exit** at any time during the process. In this case, you have to undo the grouping before starting a new one.

At the end of the process, the project reaches the FOOTPRINT_GRP-COMPLETED state and the final grouping of footprints is displayed if you expand the footprint object:



Property	Value
name	P-series
status	FOOTPRINT_GRP_COMPLETED
tracing level	INFO

In our example, footprints corresponding to different releases of the same product version are in a single group.

However, you can choose to group footprints as you want. The main reason for grouping is to prepare information for the definition phase, so that logically correlated footprints are analyzed and handled altogether.

In the fully expanded group of footprints you find nodes:

- *with label* showing the relative path of the node with respect to its parent, and
- *without labels* that are grouped under a square icon represent a collection of footprints that have the same canonical path but do not contain exactly the same files (for example they have different patch levels). Their path is displayed beside the top square icon

The following icons are used to represent nodes in a footprint group:

▣

This footprint:

- has children (square icon)
- has files (filled core)
- has path (solid colors)

□

This footprint:

- has children (square icon)
- has no files (empty core)
- has path (solid colors)

This footprint:

- has children (square icon)
- is inactive (red-crossed, grayed)

This footprint:

- has children (square icon)
- has files (filled core)
- has no path (pale colors)

This footprint:

- has children (square icon)
- has no files (empty core)
- has no path (pale colors)

This footprint:

- is a leaf node (octagonal icon)
- has path (solid colors)

This footprint:

- is a leaf node (octagonal icon)
- is inactive (red-crossed, grayed)

This footprint:

- is a leaf node (octagonal icon)
- has no path (pale colors)

Undo grouping

To undo the grouping of footprints, right-click on **Footprints** and select **Undo footprint grouping** from the drop-down menu.

The project is reset to the FOOTPRINT_GEN_COMPLETED state and all grouping related information is eliminated from the database.

Chapter 8. Creating User Profiles

After identifying and grouping footprints, you must define the user profile under which you are working to create signatures and then start the definition phase. For more information about user profiles, see “User profiles” on page 8.

When multiple users work on clones of the same project they must coordinate themselves so that each one works under a distinct user profile. This helps the application to properly report conflicts when merging definitions into a single Knowledge Base. For more information about shared projects, see: “Before sharing a project” on page 45.

To create user profiles, perform the following steps:

1. Define all the necessary user profiles in the `UserProfile.xml`file, which is automatically created in an empty workspace, and at the application startup, whenever the file is not found in the specified workspace.
2. Declare an associated user profile before opening the supporting view for the definition phase. To select a profile, on the toolbar click on this icon 👥 , select a profile name, and click **OK**.

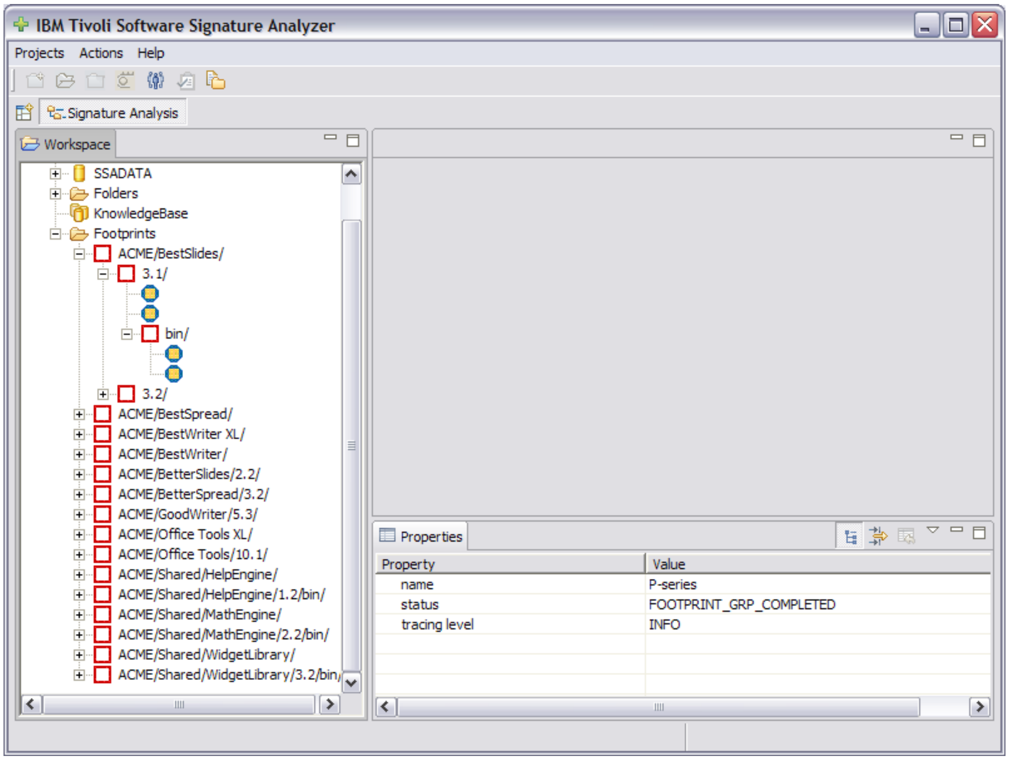
Chapter 9. Creating Definitions

After grouping the footprints, you can start the *definition* phase. Your goal is to create *definitions* in the local Knowledge Base about software products and the related signatures.

Entering the definition phase

Before you start creating definitions in the Knowledge Base, ensure that the project is in the FOOTPRINT_GRP-COMPLETED state. This means that footprints exist and are grouped with respect to relationships in their canonical paths (for more details, see: Chapter 7, “Grouping Footprints,” on page 31.)

When you expand the **Footprints** object, you see a structure like the following:

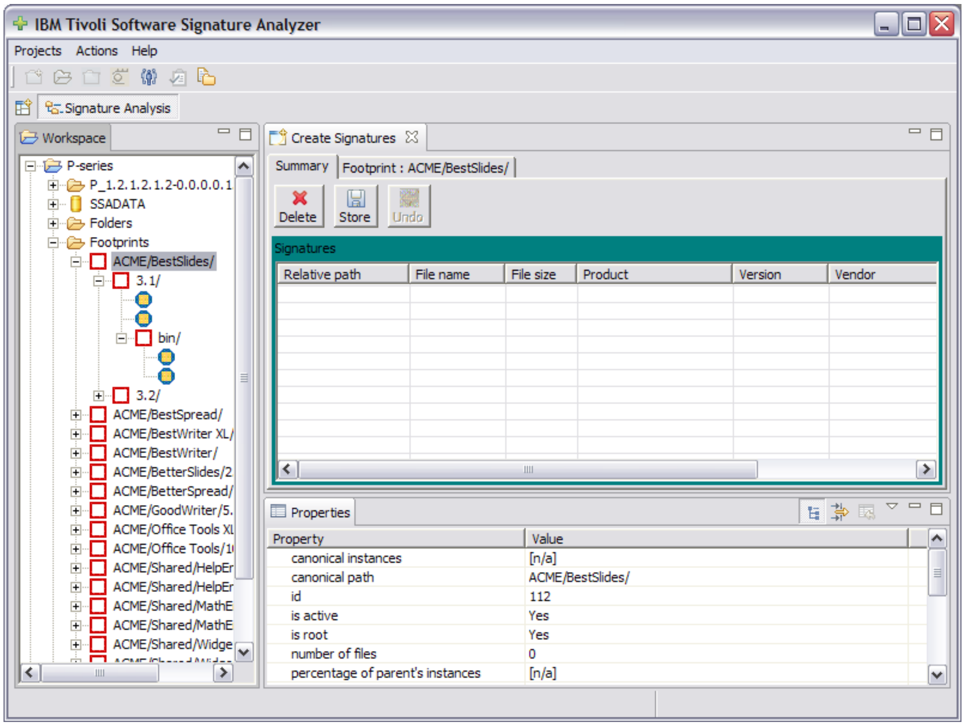


Property	Value
name	P-series
status	FOOTPRINT_GRP_COMPLETED
tracing level	INFO

Now you can select a profile for the current session and start generating product definitions and signatures:

1. To select a user profile, right-click on your project and select **Select user profile** from the drop-down menu.
2. From the drop-down list, choose the profile you want to use.
3. Open the list of footprints within the current project in the workspace view.
4. Right-click on one root footprint node whose content you want to analyze to create the corresponding signatures and product definitions. From the menu list, select **Analyze and create signatures**.

   The following dialog opens, showing the Create Signatures view:



Relative path	File name	File size	Product	Version	Vendor

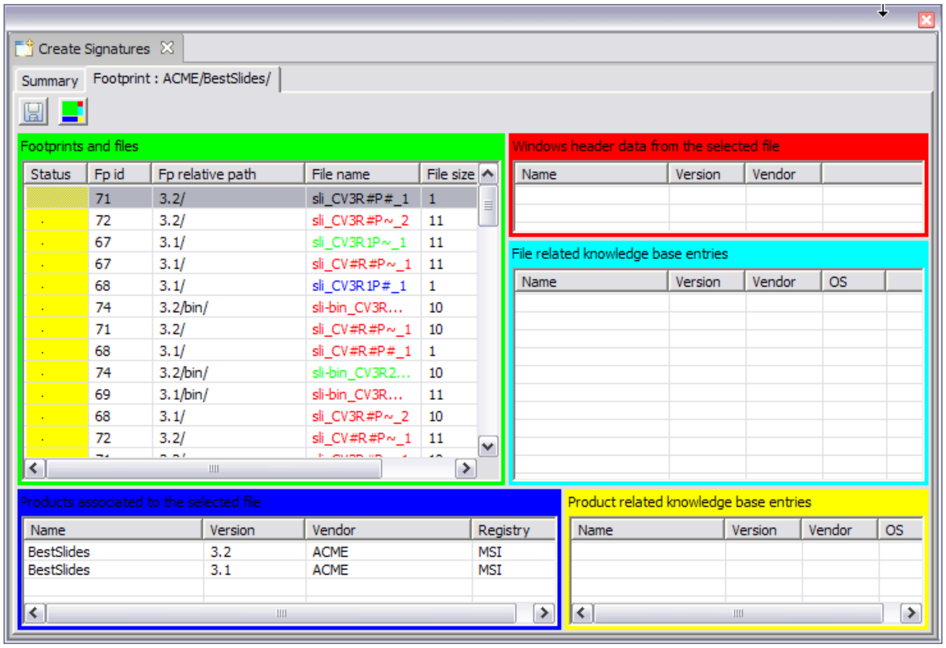
Property	Value
canonical instances	[n/a]
canonical path	ACME/BestSlides/
id	112
is active	Yes
is root	Yes
number of files	0
percentage of parent's instances	[n/a]

The displayed panel shows the following tabs:

Summary
Lists all the definitions you create for the selected footprint (group). When you open the view for the first time, the summary list is empty.

Footprint ***name***
Is associated to the selected footprint node (this might be a group including several related footprints).

5. Detach the view from the underlying window to see all the details about the selected footprint, by right-clicking on Create Signatures and selecting **Detached**.
6. Select the footprint panel and resize the view by moving the table borders. You can maximize the view by double-clicking on the view tab. The following information is displayed.



Footprints and files

Status	Fp id	Fp relative path	File name	File size
	71	3.2/	sli_CV3R#P#_1	1
·	72	3.2/	sli_CV3R#P~_2	11
·	67	3.1/	sli_CV3R1P~_1	11
·	67	3.1/	sli_CV#R#P~_1	11
·	68	3.1/	sli_CV3R1P#_1	1
·	74	3.2/bin/	sli-bin_CV3R...	10
·	71	3.2/	sli_CV#R#P~_1	10
·	68	3.1/	sli_CV#R#P#_1	1
·	74	3.2/bin/	sli-bin_CV3R2...	10
·	69	3.1/bin/	sli-bin_CV3R...	11
·	68	3.1/	sli_CV3R#P~_2	10
·	72	3.2/	sli_CV#R#P~_1	11

Windows header data from the selected file

Name	Version	Vendor

File related knowledge base entries

Name	Version	Vendor	OS

Products associated to the selected file

Name	Version	Vendor	Registry
BestSlides	3.2	ACME	MSI
BestSlides	3.1	ACME	MSI

Product related knowledge base entries

Name	Version	Vendor	OS

Note: After resizing the window, if you want to restore the original proportions between the tables, click the 🟩 icon.

The following tables are displayed in the footprint panel:

- Footprints and Files
- Product associated to the selected file
- Windows header data from the selected file
- Knowledge Base entries associated to the selected file
- Knowledge Base entries associated to the selected product

7. Based on the information contained in the tables, select the files for which you want to create signatures and associate them to products and components in the corresponding tables.

Selecting the information to save

To create a signature, select the appropriate file in the Footprint panel based on the information contained in the displayed tables. For more information about the files to choose, see “File signatures” on page 9.

To select a file and product associated information, perform the following steps:

1. Select a file from the list in the Footprints and Files table (for more information about this table content, see: “Footprints and files” on page 42.)

   The product related information is displayed in the other tables. Namely, all the product tables display names, versions, and vendors of products associated to the selected file. More details about each table are provided in the following sections

2. Select the product definition you choose to associate to that file and store as a signature in the Knowledge Base. Product definition can be:

   - Derived from the file path (see:“Product associated to the selected file” on page 43)

- Derived from the file header (see: "Windows header data from the selected file" on page 43.)
- Found in the Knowledge Base associated to the selected file ("Knowledge Base entries associated to the selected file" on page 43)
- Found in the Knowledge Base associated to the product selected in the product table ("Knowledge Base entries associated to the selected product" on page 43).

If a product definition is already stored in the Knowledge Base for that file, it is recommended that you use it instead of creating a new one.

3. Click the 💾 icon to store signatures in the summary.

   The file is marked red and all the files contained in the same footprints are marked gray. This means you cannot use any of these files to create other signatures. This prevents undesirable duplicate matches.
4. Repeat the whole procedure with all the files and footprints for which you want to create signatures.

   Note: When the 💾 icon appears disabled, it means that you have not selected a file that can be used as a signature.

Footprints and files

This table includes all the files that were found within any footprint of the source group. Clicking on any of the column headers you can sort the files according to the selected characteristic.

Footprint ID

Lists all the footprints by their IDs.

Footprint relative path

Provides the footprint relative paths with respect to the root grouping node. In the case of a root consisting of an isolated footprint, the content of this column is the null path ('.'). This value has a primary importance in determining the location of the associated folder in the product installation tree. If, for example, you find a group whose multiple footprints are in a parent-child relationship, it means that they belong to the same installation tree. In this case, it is highly recommended that you create signatures from footprints that relate to the same folder (relative path). Otherwise you will have a redundancy of signatures that would cause multiple matches for a single instance of the target product.

File name

Lists all the files belonging to the selected footprint or group. Files are identified by the following colors:

GREEN

Identifies files found in a single footprint.

RED

Identifies files found identical across footprints that belong to different products or different versions of the same product.

BLUE

Identifies files found identical across multiple footprints that have the same canonical paths. Therefore, these footprints are likely to relate to variants (with different maintenance levels) of the same product. You can use these files as signatures if you are not interested in keeping track of the different maintenance levels of their footprints.

File size
Is the size of the selected file.

Lists the number of footprints that contain the selected file.

Product associated to the selected file

The content of this table dynamically changes and shows the list of products that install the file selected in the Footprints and Files table. This information is critical in determining the name of the product and its version, because it is not always possible to derive product information from the installation pathnames reflected in the footprint canonical path.

When there is no product information displayed in this table for a given file, determining the product name and version might be impossible, so you might have to resort to external sources of information, such as the Internet, to acquire this data. The lack of product information associated to a file can be due to:

- Harvest data set obtained from computers where the software product owning the file was not registered during installation, or
- On Windows platforms, the product owning the file was registered during installation on at least one computer, but the product installation path could not be determined.

Windows header data from the selected file

The content of this table dynamically changes and shows the product information relating to the file selected in the Footprints and Files table. The information is derived from the file header (only for Windows files).

Knowledge Base entries associated to the selected file

The content of this table dynamically changes and shows all the products that are associated to the selected file in the Knowledge Base. If the Knowledge Base you are working with is already populated and contains one or more product definitions associated to the selected file, they are displayed here.

Knowledge Base entries associated to the selected product

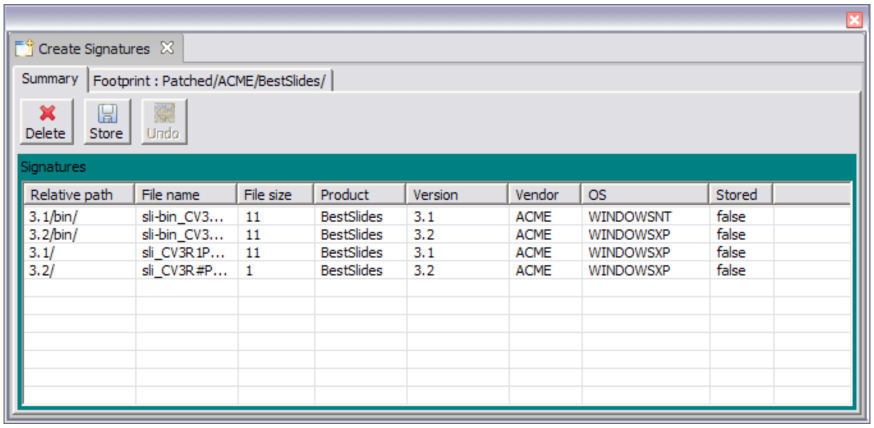
The content of this table is displayed only when you select an entry from the product table.

When you select a file from the Footprints and Files table, the corresponding product information is displayed in the product table. Selecting it you can see how it is defined in the Knowledge Base, if there is already a definition there.

If a product definition already exists in the Knowledge Base, use the existing definition instead of creating a new one, provided that the difference between the two definitions is purely in the description of the *same* product (for example, IBM WAS versus IBM Websphere Application Server.) If the product definitions differ by product version or vendor, it is recommended that you create a new definition, keeping track of the differences.

Storing signatures in the Knowledge Base

All the files and related product definitions you have chosen in the Footprint panel are inserted in the Summary, which is an intermediate step before storing them in the Knowledge Base. Files included in the Summary are displayed in the Signatures table, in the Summary panel:



Relative path	File name	File size	Product	Version	Vendor	OS	Stored
3.1/bin/	sli-bin_CV3...	11	BestSlides	3.1	ACME	WINDOWSNT	false
3.2/bin/	sli-bin_CV3...	11	BestSlides	3.2	ACME	WINDOWSXP	false
3.1/	sli_CV3R1P...	11	BestSlides	3.1	ACME	WINDOWSXP	false
3.2/	sli_CV3R#P...	1	BestSlides	3.2	ACME	WINDOWSXP	false

To store in the Knowledge Base a list of files and product definitions as shown, click the 💾 icon. However, the following icons allow you to:

✖

Delete signatures from the summary before storing them in the Knowledge Base.

↩

Undo the last operation.

Note: Signatures that have been stored in the Knowledge Base cannot be removed.

Chapter 10. Knowledge Base Administration

For our purposes, the knowledge base is a repository of definitions located in the workspace and whose content is gradually increased with the information that you produce working on later projects.

An *empty* knowledge base is automatically created on startup, whenever the corresponding file (`kbase.kbxmlzip`) is not found in the specified workspace folder. When you work on a project, you add new definitions to the local knowledge base, thereby increasing its content. When a project is finished, you consolidate this content creating a *baseline* of the knowledge base that can be used as the basis for future projects that might be creatd in the same or different workspaces.

Knowledge bases resulting from the work of different users (each one working on a clone of the same project within a separate workspace) can be merged into a single knowledge base.

When you finish working on a project, you should create a baseline of the knowledge base. You can then export selected content from the knowledge base in a format suitable to be used by external applications (currently the only supported format is a catalog that can be imported by the IBM Tivoli Configuration Manager 4.1.x or later.)

Before sharing a project

To allow multiple users to share their work in the context of a single project, the application supports the concept of project cloning.

The following prerequisites should be met before cloning a project:

- A well-defined level of the knowledge base content must be established as a starting point for the work of all the users among which the work is being split. In the first project this can be an empty knowledge base. In later projects, this should be the baseline created at the end of the last completed project.
- The project you want to clone must be in the FOOTPRINT_GRP-COMPLETED state. When reaching this state, a globally unique identifier (GUID) is assigned to the project and it is used to lock the knowledge base during the first update.

Locking prevents the knowledge base from being inadvertently updated. The knowledge base can only be unlocked by requesting a baseline from a project with the same GUID.

Note: Should the project be inadvertently deleted, it will not be possible to remove the lock. In that situation, you may need to replace the knowledge base file with a backup copy of the last baseline.

Sharing a project

To allow others to work on a clone of your project, perform the following steps:

1. Close the project and the application.
2. Archive the folder of the project you want to clone in a ZIP file.
3. Create in the current workspace a copy of the knowledge base (`Kbase.xmlzip`) that must contain the established baseline.

4. Create in the current workspace a copy of the user information (`UserProfile.xml`) that must contain an entry for each one of the users (including you), to ensure a correct tracking of the activities performed by each user.
5. Provide a copy of the files created on the previous three steps to each other user working on the project.

The other users can now perform the following steps to create their private copy of the project:

1. Create a new empty folder to be used as the application workspace, and copy the `kbase.kbxmlzip` file and the `UserProfile.xml` file into this folder.
2. Re-create the project folder within the new workspace unzipping the archive file created on step 2 on page 45.

Users working on clones of the same project must coordinate their activities so that only one user creates product definitions and signatures related to a given group of footprints. Because the groups are created starting from path names, it might happen that two distinct groups of footprints are actually related to variants of the same product. If that happens, a single user should create definitions from both groups. The application marks the information produced by each user to facilitate the discovery of potential conflicts during the final merge. The applied *marker* is based on the name of the user profile that each user must select before creating new definitions. To avoid jeopardizing the effectiveness of these controls, each user must always select the same unique profile selected the first time. When the work on the project is complete, each user has an updated knowledge base in the workspace, with some specific additional content on top of the common baseline. These knowledge bases must be merged into one that includes all the data produced by each user.

Merging Knowledge Bases

Before the merging, ensure you have a local copy of the knowledge base files created by the other users in their respective workspaces.

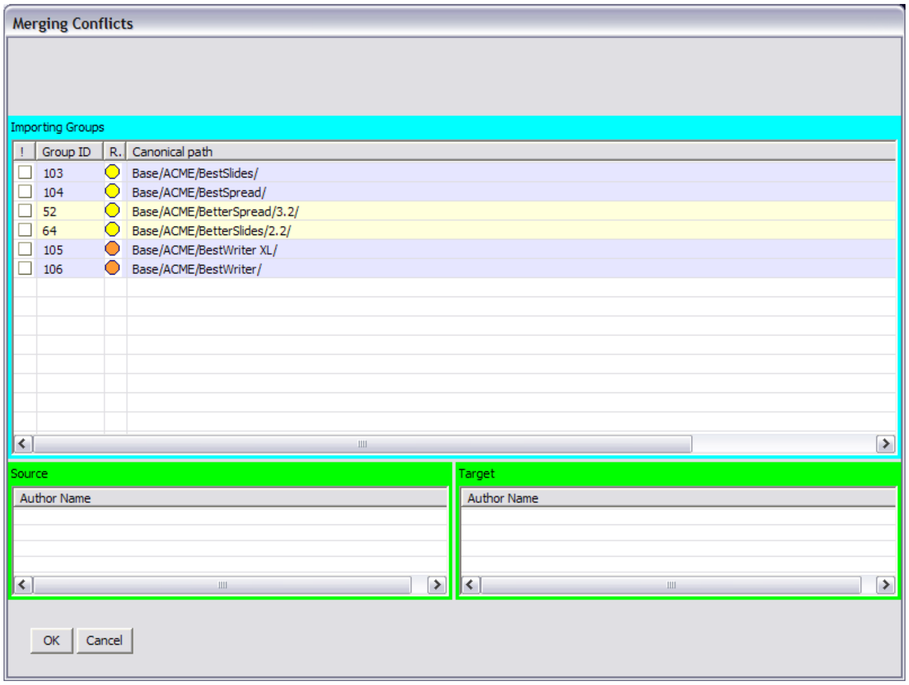
To avoid the inadvertent corruption of the knowledge base in your current workspace, the application only allows the merge when either one of the following conditions is met:

- The knowledge base is a baseline, not locked by any project. This situation should only occur when this is the established common baseline for the current project and no work was done in this particular workspace to modify the baseline.
- The knowledge base is locked by the current project. This situation should occur when some of the project work was done in this particular workspace.

The merge of an external knowledge base file into the current workspace can be performed when the project is in the ANALYZE_SIGNATURE state or in the MERGE_IN_PROGRESS state. The merge process only takes into account definitions contained in the source file which were created by work done in a clone of the same project, as identified by the project GUID. In particular, this mechanism excludes from the merge all those definitions that were already contained in the common baseline established at the beginning of the project. To merge an external knowledge base file into the current workspace, perform the following steps:

1. From the new project tree, right-click on the Knowledge Base object and select **Import knowledge base**.
2. From the displayed dialog, browse to the knowledge base you want to import and click **Open**.

Before actually performing the merge, the application examines the content of both the source and the target knowledge bases. The results of this analysis are presented in a dialog where you can examine the occurrence of possible conflicts in definitions (products and signatures) associated to each given group of footprint:



!	Group ID	R.	Canonical path
□	103	●	Base/ACME/BestSlides/
□	104	●	Base/ACME/BestSpread/
□	52	●	Base/ACME/BetterSpread/3.2/
□	64	●	Base/ACME/BetterSlides/2.2/
□	105	●	Base/ACME/BestWriter XL/
□	106	●	Base/ACME/BestWriter/

The dialog includes the following sections:

Importing Groups

Lists the groups for which new definitions were added in the source knowledge base in the current project. Each row represents a group of footprints. Two footprint groups are considered correlated if they include signatures for the same product definition in either the source or in the target knowledge base. To ensure the consistency of all signatures associated to a given product, the source knowledge base must contain data for a whole set of correlated groups. The table is therefore arranged as a sequence of zones with an alternating background color, each zone including a whole set of correlated footprint groups. Correlated footprints thus belong to the same zone and they can be either all merged-in together or they should be all discarded.

Source Author

Lists the names of users that created content (in the source knowledge base) for the footprint group currently selected in the table above.

Target Author

Lists the names of users that created content (in the target knowledge base) for the footprint group currently selected in the table above. When this table is not empty, some definitions related to the same group of footprints were created in different clones of the project, so there is a risk of redundant or inconsistent information being created by multiple users.

Importing groups table

Each table row represents the root of a footprint group. Two footprint groups are considered correlated if they include signatures for the same product definition in either the source or the target knowledge base. To ensure the consistency of all signatures associated to a given product, the source knowledge base must contain data for a whole set of correlated groups. The table is therefore arranged as a sequence of zones with an alternating background color, each zone including a whole set of correlated footprint groups. Correlated footprints thus belong to the same zone and they can be either all merged-in together or they should be all discarded.

Each row of the table is related to a footprint group and it contains the following fields:

Selection check box

This check box is selected by default when no risk has been detected for importing data related to all footprints in the corresponding zone. You can still decide to clear the check box relating to one footprint group to avoid importing any data associated to any group in the same zone.

Group ID

The ID by which the group is identified within the project;

Risk indicator

Icons useful to understand what action should be taken. The possible values of the risk indicator are:

✚

New footprint. There are no signatures in the target file derived from the same footprint, nor signatures related to the corresponding product definitions.

🟡

Conflicting footprint. This conflict might occur when the target file already contains signatures derived from the same footprint. If you select this footprint, all signatures from the whole set of correlated footprints in the source file are used to replace existing data associated to the same set of footprints in the target file.

🟠

Cross project footprint. The target file contains signatures related to the same product definition created in a different project. Therefore they should have been in the baseline at the beginning of the project. You must decide whether the two sets of signatures are compatible, for example, because they cover additional maintenance levels of the same product, and could be added without causing problems (such as multiple matches for a single instance). A confirmation dialog appears when you select a footprint with this indicator.

✖

Missing footprint. The target file contains signatures derived from multiple footprint groups that are related to the same product definition. However the source file only contains data for a subset of those footprint groups. Footprint groups marked with this risk indicator are the ones for which data is only found in the target file, and is missing in the source file. The presence of one such element in a set of correlated footprints makes the whole set not eligible for being imported, therefore it cannot be selected. If you want to import the existing data in the source file, you must go back to the source project and define signatures also for the missing footprints.

Canonical path
The footprint canonical path (when available).

About group correlation

The following example illustrates the process of detecting correlations among the footprint groups. Different groups of footprints are correlated if they include signatures for the same product definition in the source or in the target knowledge base. Sets of correlated groups thus created on both the source and the target knowledge bases are then merged into one single set, if they share at least one footprint group or any related product definition. For example: In the source file there are signatures for DB2-UDB 8.2 from footprint groups 100 and 120, while in the target file there are signatures for Lotus Notes 6.1 from footprint groups 120 and 130. As a result, the footprint groups 100, 120 and 130 are all correlated.

Or, for example: In the source file there are signatures for DB2-UDB 8.2 from footprint groups 100 and 110, while in the target file there are signatures for DB2-UDB 8.2 from footprint groups 120 and 130. As a result, the footprint groups 100, 110, 120 and 130 are all correlated.

Creating a baseline

When the complete set of definitions produced within the scope of a single project has been consolidated in the knowledge base, you can unlock the project, thus allowing the knowledge base file to be used as the baseline for a new project. You can create a baseline when the project is in the ANALYZE_SIGNATURE state or in the MERGE_IN_PROGRESS state.

To create a baseline, right-click on the Knowledge Base object, and select **Baseline knowledge base**.

Exporting the Knowledge Base

The signature information stored in the knowledge base can be examined, and, possibly, exported at any time during the life of a project. Normally you would export the signature data when you have finished on with a project and your work has been consolidated in a new baseline. These signatures are intended for consumption by an ITCM application, therefore the set can be filtered to avoid including signatures that are already available to the application.

By default, only signatures created by the Software Signature Analyzer tool are exported. By accepting this default you avoid exporting signatures that were already part of the knowledge base

To export the knowledge base content, perform the following steps:

1. Right-click on the Knowledge Base object, and select **Export**.
2. Enter the required information in the Export wizard.

Best Practices to work with the Knowledge Base

To work and manage your projects in the most effective way it is recommended that you:

- Use a new, dedicated workspace (knowledge base) for projects aimed at exploiting a new collection of data harvested from your environment.
- Use different projects to analyze data related to different platforms, unless the intentional objective is to identify multi-platform (such as JAVA) software.
- Backup your knowledge base periodically, consolidating the definitions and creating a *baseline*.
- Start to work with a well-known knowledge base baseline from a repository, **or** use the application-generated empty knowledge base if a baseline has not yet been created.
- When multiple users need to work together to create definitions for the same project, *clone* the project so that they can work separately.
- Save and store each baseline in a versioning repository, that helps you keep track of all previous versions of a knowledge base.

Glossary

C

canonical path. The installation of a software product creates an installation tree of folders whose names are the same across multiple installed instances of the product. The canonical path is the trailing portion of the pathname of the product-installed folders that remains constant across instances, except for the parent folder under which the product installation tree is created

core file. File whose presence in a folder *does not depend* on any selectable options that can be specified during the installation.

F

footprint. The IBM Tivoli Software Signature Analyzer virtual reconstruction of a product installation tree. It is a set of core files uniquely identifying a product. It does not contain optional files.

H

harvest data set. Specific raw data and information collected from the scan of the analyzed computers. Each data set corresponds to a scanned computer.

K

knowledge base. Archive where signatures and product related information are stored. Every project must have one, and only one, knowledge base.

N

normalizing data. Creating unique objects (files or folders) from of all the multiple instances of the same objects across the analyzed data.

O

optional file. File whose presence in a folder *depends* on a selectable option that must be specified during the installation.

S

signature. Specific characteristics of a file (name and size), which, if found on a system, usually means that the product associated to that file is installed on that system.

source file. A file that can be used as a guide in the footprint generation process. Usually, it has the same content and name across its multiple instances in different folders.

spurious folder. Imperfect copy of an original folder. A spurious folder can be the result of an incomplete uninstall or an old backup copy.

T

token. Portion of a string. Folder paths are managed as strings to create footprints. A token is the string portion included between two slashes (that is, a directory). The leading token is the first directory in the path.

U

unique folder. Out of all the occurrences of the *same* folder found across the harvesting data, the IBM Tivoli Software Signature Analyzer isolates a single object to uniquely represent all the instances. To ascertain the folder identity the IBM Tivoli Software Signature Analyzer takes into account the folder content (checksum), not its path.

V

variant folder. A folder is a folder that has almost the same content as the original folder. For example, the installation folders of the same product, but with different maintenance level.

Printed in USA