Shell 1.1

On this page:

Target audience:

Users of the extension providing **UNIX Shell** support.

Summary: This document provides information about the extension providing **UNIX Shell** support.

What's new?

Please see Shell 1.1 - Release Notes for more information.

Description

This extension provides support for applications written using UNIX shell languages.

In what situation should you install this extension?

If your application contains source code written using UNIX shells and you want to view these object types and their links with other objects, then you should install this extension.

Supported UNIX shells

This version of the extension provides partial support for:

UNIX shell	Supported
Bourne shell (bsh/sh/shell)	✅
Bourne-Again Shell (bash)	✅
C shell (csh)	✅
KornShell (ksh)	✅
Secure Shell (ssh)	✅
Tenex C Shell (tcsh)	✅

Function Point, Quality and Sizing support

This extension provides the following support:

- **Function Points (transactions)**: a green tick indicates that OMG Function Point counting and Transaction Risk Index are supported
- **Quality and Sizing**: a green tick indicates that CAST can measure size and that a minimum set of Quality Rules exist

Function Points (transactions)	✅
Quality and Sizing	✅

CAST AIP compatibility

This extension is compatible with:

CAST AIP release	Extension release	Supported
8.3.x	1.0.1	✅
8.2.x	1.0.1	✅
8.1.x	1.0.1	✅
8.0.x	1.0.1	✅

Supported DBMS servers

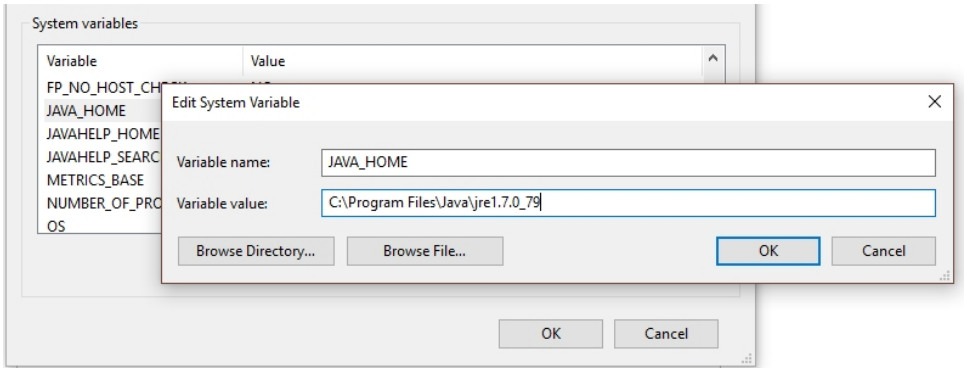
This extension is compatible with the following DBMS servers:

DBMS	Supported
CSS	✅
Oracle	✅
Microsoft SQL Server	❌

Prerequisites

✅	An installation of any compatible release of CAST AIP (see table above)

- The extension requires a **Java JRE** to be installed on the machine: only **Java JRE 1.7** is currently supported.
- The extension requires that a **JAVA_HOME** system environment variable is also present on the machine, pointing to the Java JRE installation folder:



Note that these prerequisites are only applicable if you are using version **1.0.5** of the Shell extension AND **CAST AIP 8.2.0**

Therefore, if you are using **CAST AIP 8.2.1**, you don't need to set a JAVA_HOME. The Shell extension will use the JRE provided with CAST AIP and located in the installation folder.

Download and installation instructions

Please see:

- Download an extension
- Install an extension

The latest release status of this extension can be seen when downloading it from the CAST Extend server.

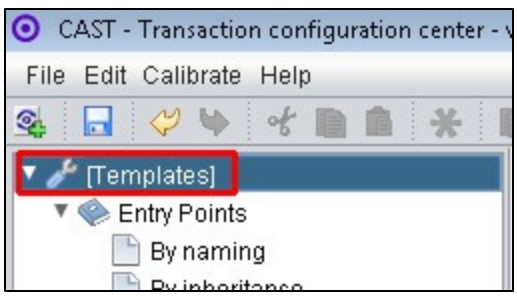
CAST Transaction Configuration Center (TCC) configuration

A set of Shell **Transaction Entry/End Points** for use in the CAST Transaction Configuration Center is delivered in the extension via a .TCCSetup file. Therefore If you are using **Shell 1.0.7**:

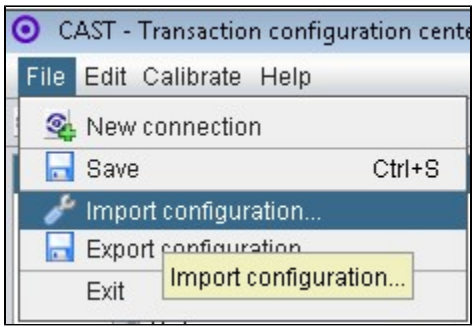
- with **CAST AIP 8.3.x**, there is nothing for you to do: these entry / end points will be automatically imported during the extension installation and will be available in the CAST Transaction Configuration Center under "Entry Points > Free Definition".
- with **CAST AIP 8.2.x**, you can manually import the file **%PROGRAMDATA%\CAST\CAST\Extensions\com.castsoftware.shell.<version>\Configuration\TCC\<name>.TCCSetup** to obtain your entry/end points in the "**Free Definition**" section (see instructions below).

Manual import action for CAST AIP 8.2.x

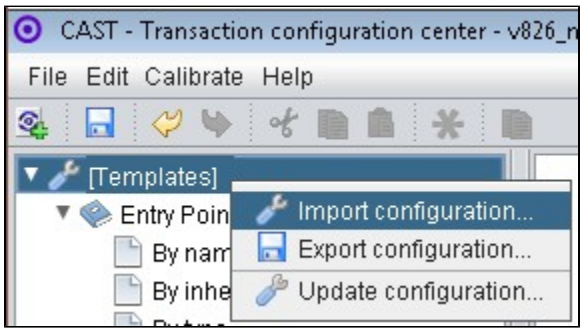
- Locate the .TCCSetup file in the extension folder: **%PROGRAMDATA%\CAST\CAST\Extensions\com.castsoftware.shell.<version>\Configuration\TCC\<name>.TCCSetup**
- In the CAST Transaction Configuration Center, ensure you have selected the **Templates** node:



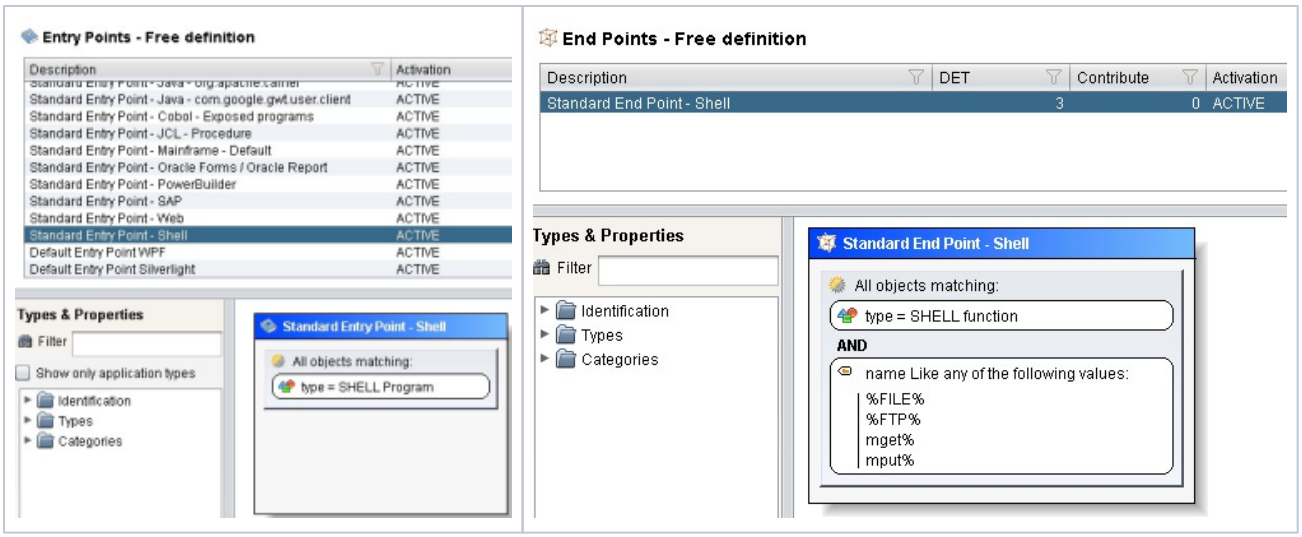
- This .TCCSetup file is to be imported into the CAST Transaction Calibration Center using either the:
  - **File** > **Import Configuration** menu option:



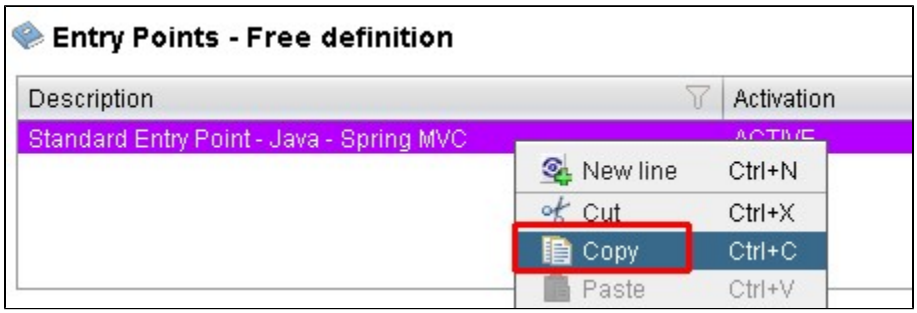
- Or right clicking on the **Template node** and selecting **Import Configuration**:



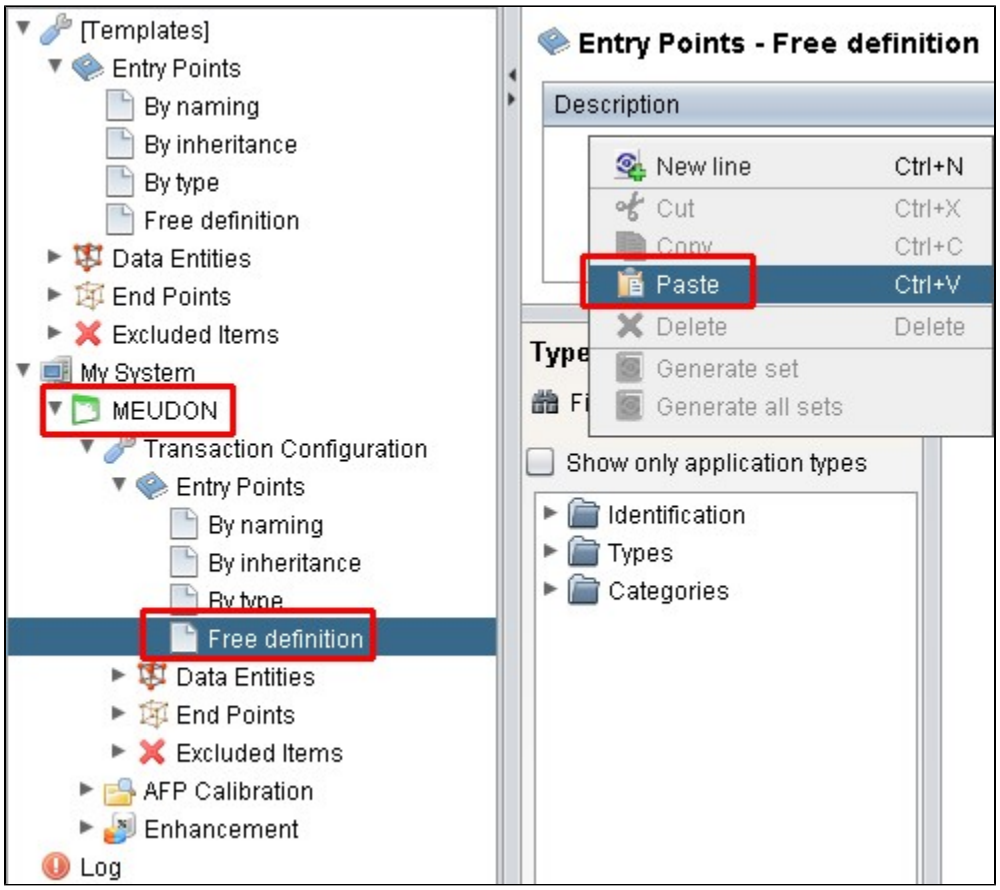
- The import of the "**Base_Shell.TCCSetup**" file will provide you with a sample Transaction Entry and End points in the **Free Definition** nodes under Templates (click to enlarge):



- Now right click the "**Standard Entry Point**" item and select copy:



- Paste the item into the **equivalent node** under the **Application**, for example, below we have copied it into the **Application MEUDON**:



- Repeat for any additional items or generic sets that have been imported from the .TCCSetup file.

Prepare and deliver the source code

Once the extension is downloaded and installed, you can now package your source code and run an analysis. The process of preparing and delivering your source code is described below:

Source code preparation

Only files with following extensions will be analyzed:

- *.bash
- *.bsh
- *.csh
- *.ksh
- *.sh
- *.shell
- *.ssh
- *.tsch

Note that if any files intended for other applications are included in the delivery and which are renamed to supported Shell extensions, the following effects might be observed during an analysis:

- inconsistent objects may get created
- end of string '" not found errors
- if the file is binary: Invalid UTF-8 sequence found in text to be matched or searched for a regular expression

Source code preprocessing

Shell source code needs to be preprocessed so that CAST can understand it and analyze it correctly. In previous releases of the extension, this preprocessing was a **manual action** that needed to be completed **before** the code was analyzed. However, in this release and all future releases, the code preprocessing is **actioned automatically** when an analysis is launched or a snapshot is generated (the code is preprocessed before the analysis starts). In other words you only need to package, deliver and launch an analysis/generate a snapshot for the preprocessing to be completed.

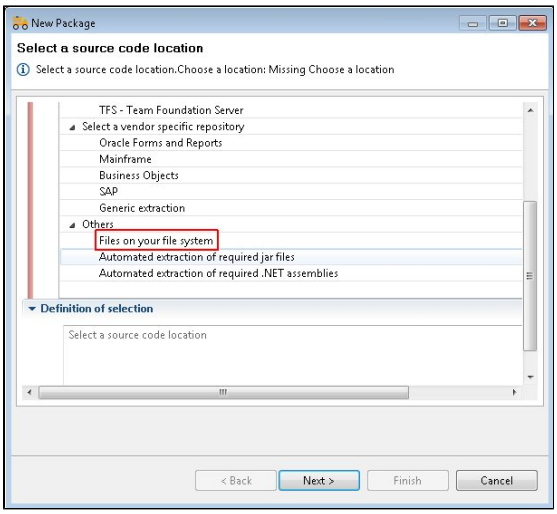
> Note that the CAST Management Studio will use the LISA folder to analyze the preprocessed files (see CAST Management Studio help for more information about this folder).

Deliver the source code

Using the CAST Delivery Manager Tool:

- create a new **Version**
- create a new **Package** for your source code using the **Files on your file system** option and choose the location of your source code:

Click to enlarge



- Run the **Package action**.
- Before delivering the source code, check the **packaging results.**

Discovery

A discoverer is provided together with the extension to automatically detect Shell code. When any file with an extension listed in Source code preparation is discovered, the parent folder of this file will be designated as the project root path. One Analysis Unit is created for each Shell Project discovered.

Analysis configuration and execution

Refer to Analysis Configuration and Execution for more information.

Logging mechanism

Analysis log files

Analysis logs are stored in the default locations used by the CAST Management Studio.

Shell Preprocessor

Shell Preprocessor log files (the preprocessor is launched automatically during an analysis) are stored in the following locations:

CAST AIP release	Location	Log file name

8.2.x	**%PROGRAMDATA%\CAST\CAST\Extensions\<extension_name>\Configuration\Languages\Shell\prepro** Note that the above location is the default, however, if you have modified the CAST_PLUGINS_ROOT_PATH variable in the CastGlobalSettings.ini file, this location may be different.	com.castsoftware.shell.<MAJOR_VERSION.MINOR_VERSION.MAINTENANCE_VERSION>.prepro_<YYYYMMDDHHMMSS>.log
8.3.x	Default location is set to **%PROGRAMDATA%\CAST\CAST\Logs\<unique_application_id>**, but this location can be configured at will in the CAST Management Studio Preferences.	

What results can you expect?

Objects

Icon	Metamodel Name
	SHELL Project
	SHELL Program
	SHELL Function
	SHELL Special Function
	Call to a Java program
	Call to a program

Links

Source Object	Link Type	Target Object
SHELL Program	callProgLink	SHELL Program
SHELL Program	include	SHELL Program
SHELL Program	callLink	SHELL Function/Special Function
SHELL Function/Special Function	callLink	SHELL Program
SHELL Function/Special Function	callLink	SHELL Function/Special Function
SHELL Function/Special Function/Program	use select, update, insert, delete	data functions
SHELL Function/Special Function/Program	callLink	Call to a Java program
SHELL Function/Special Function/Program	callLink	Call to a program

Links to external programs

The following call pattern are supported

- ./exe
- /path/to/exe

- python
- java
- nohup
- eval
- wlst.sh

Shell to Java links

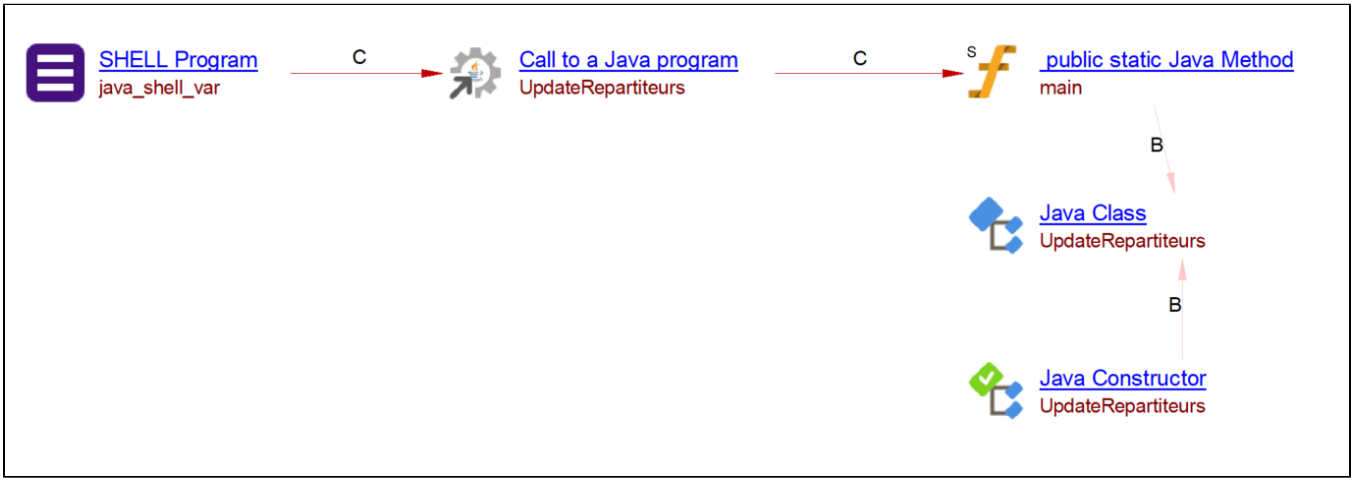
The extension does support links between Shell programs and Java objects (for example methods). Links will be created between these technologies.

The extension manages call to Java classes and .jar files. For .jar files no links will go further as .jar aren't handle by any extensions, the link is purely informative.

Basic Case

```
...
/bin/java UpdateRepartiteurs
...
```

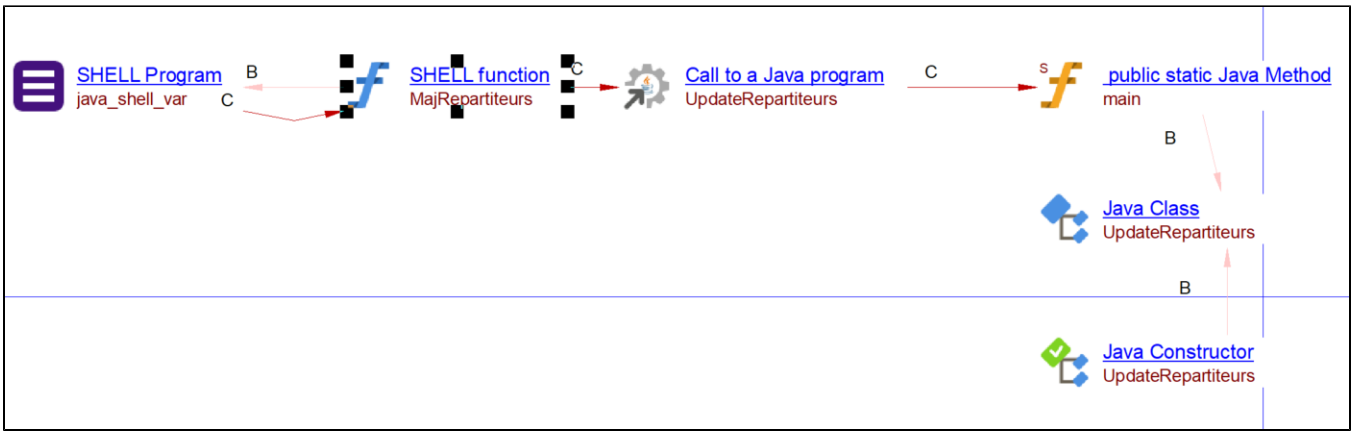
wil generate the following diagram



Function Case

```
...
MajRepartiteurs ()
{
        ...
        java UpdateRepartiteurs
        ...
}
...
```

will generate the following diagram



Shell to COBOL links

The extension does support links between Shell programs and COBOL objects (for example programs). Links will be created between these technologies.

Basic Case

COBOL file

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
* simple hello world program
PROCEDURE DIVISION.
    DISPLAY 'Hello world!'.
```

SHELL script

```
./hello
```

will generate the following diagram

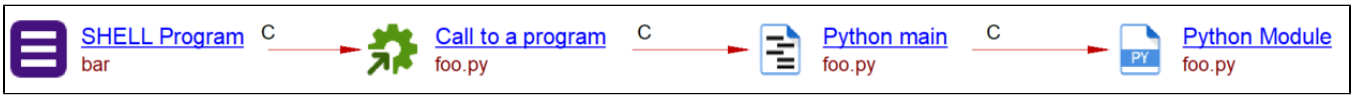
Shell to Python links

The extension does support links between Shell programs and Python objects (for example methods). Links will be created between these technologies.

The following script

```
...
python foo.py
...
```

will generate the following diagram, assuming that foo.py exists and has been analyzed



Structural Rules

The following structural rules are provided:

1.1.6-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.1.6-funcrel

1.1.5-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.1.5-funcrel
1.1.4-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.1.4-funcrel
1.1.3-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.1.3-funcrel
1.1.2-funcrel	Not released.
1.1.1-funcrel	Not released.
1.1.0-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.1.0-funcrel
1.1.0-beta1	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.1.0-beta1

You can also find a global list here:

https://technologies.castsoftware.com/rules?sec=t_1016000&ref=||

Limitations/known issues

Deployment folder path

The deployment folder path should contain only ASCII characters (due to the way the command line text is passed in a Windows operating system). Names of folders and files inside the deployment folder can contain non-ASCII characters.

Links to database objects

When your Shell scripts contain references to database objects in the server side element of an application and these references use fully qualified names for the database object, no links will be created between Shell and the database objects when you have used the SQL Analyzer extension to analyze the SQL if SQL Analyzer does not create database objects with same name. If you have used the SQL analyzers embedded in CAST AIP to analyze the SQL, then links will be created as normal. This is a known issue.

Metrics Assistant (embedded in CAST AIP) limitations

Searches not limited only to embedded SQL

The MA (Metric Assistant) which is used for metric search cannot search only in embedded SQL. Some Shell rules may be affected by this limitation and may produce false violations.

Cannot calculate metric excluding comments

The MA (Metric Assistant) which is used for metric search cannot search correctly while excluding comments especially if comments start or end adjacent to the keyword. If such a condition exists, random false violations may occur.

Shell embedded strings

Shell code allows string to be embedded in strings as shown in the code sample below. Currently, the Shell extension (and other Universal Analyzer type extensions) will consider this as one continuous string. Because we do not have any way to identify perfect end string patterns in this case, we cannot find the end of string and therefore the file will be skipped during the analysis.

```
echo '
Outer string ;
cat '$file03'
Another outer string
'$id'
' | $command
```

KSH: guessing of ending single\double quote

Note that this limitation is no longer applicable to Shell 1.0.10.

KSH supports the *guessing* of ending single/double quotes. The Shell extension supports this when the string is in single line, however, it is not supported when the string is in a multiple line, for example:

```
export OUTPUT=`basename $SOME_VARIABLE | $AWK 'FS="-" {
  i=3
  tmpMachineName= "mach_"$2
  while ( i <= NF){
    tmpMachineName=tmpMachineName"-"$i;
    i++;
  }
  print(tmpMachineName);
}`
```

Multi-line document markers

Note that this limitation is no longer applicable to Shell 1.0.10.

When a document marker is in a multi-line string, the Shell extension will not be able detect that it is in a string, for example:

```
some_multiline_string="a;b;c;d;\
e;f;g;h;\
x;y;z<<;strong text"
```

In both of these cases, the file will be skipped and logs will contain the warning: "File Skipped"