Shell 1.2

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

Extension ID

com.castsoftware.shell

What's new?

Please see **Shell 1.2 - 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	✓

AIP Core compatibility

This extension is compatible with:

AIP Core release	Extension release	Supported
8.3.x	1.0.1	✓

Supported DBMS servers

This extension is compatible with the following DBMS servers:

DBMS	Supported
CSS / PostgreSQL	✓

Prerequisites

✓	An installation of any compatible release of AIP Core (see table above)

Download and installation instructions

The extension will be automatically downloaded and installed in AIP Console when you deliver Shell code (i.e. any file with an extension listed **below**). You can also **manually install** the extension using the **Application - Extensions** interface. When installed, follow the instructions below to run a new analysis /snapshot to generate new results:

- **Advanced onboarding - run and validate the initial analysis**
- **Advanced onboarding - snapshot generation and validation**

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**

> 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 Shell 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 LISA folder will be used to analyze the preprocessed files.

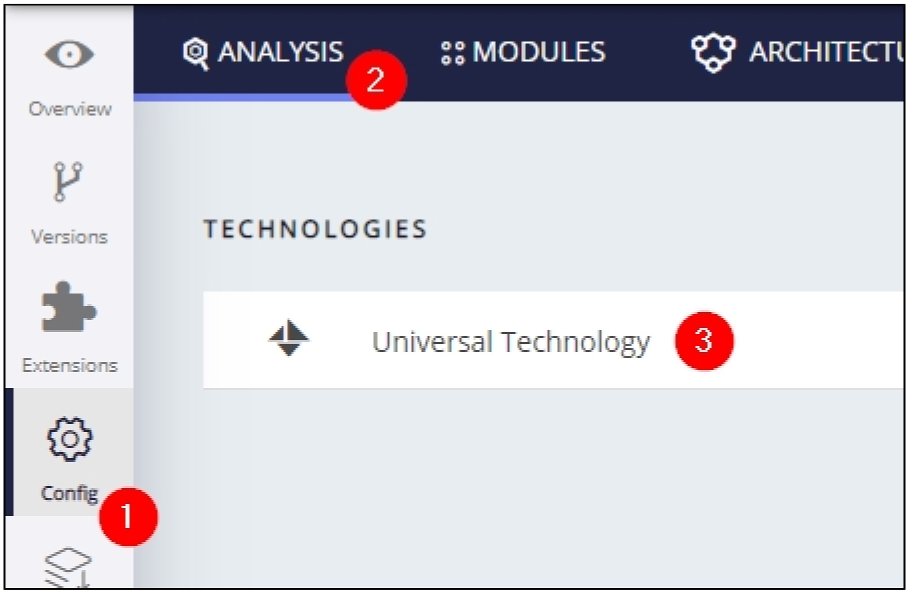
Deliver the source code

- **Advanced onboarding - add a new version and deliver source code**
- **Advanced onboarding - validate and accept the version**

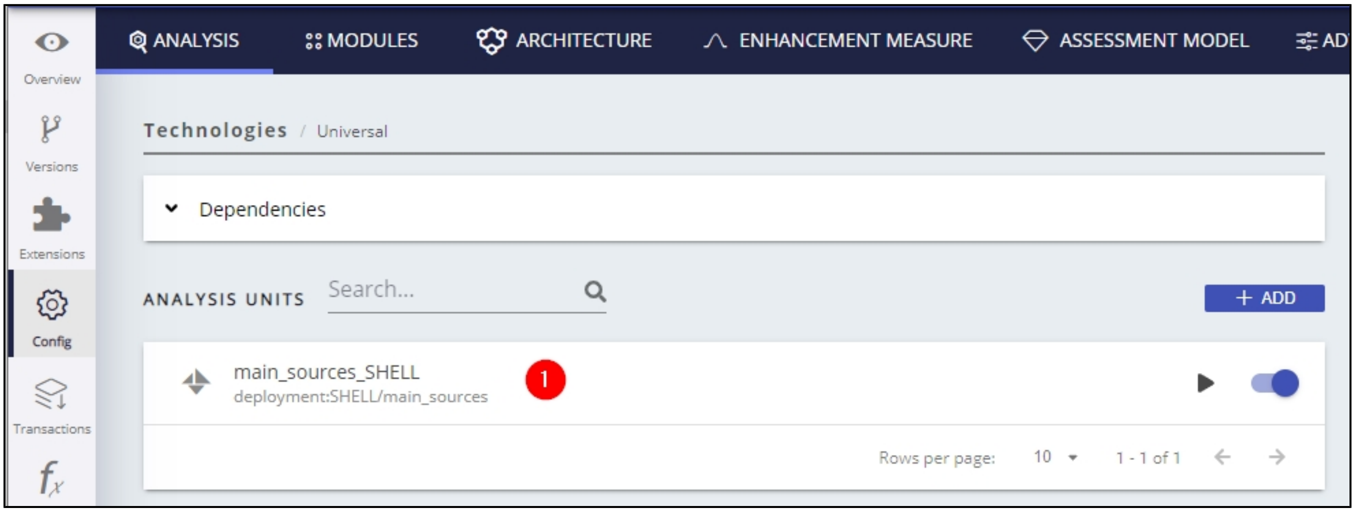
Analysis configuration and execution

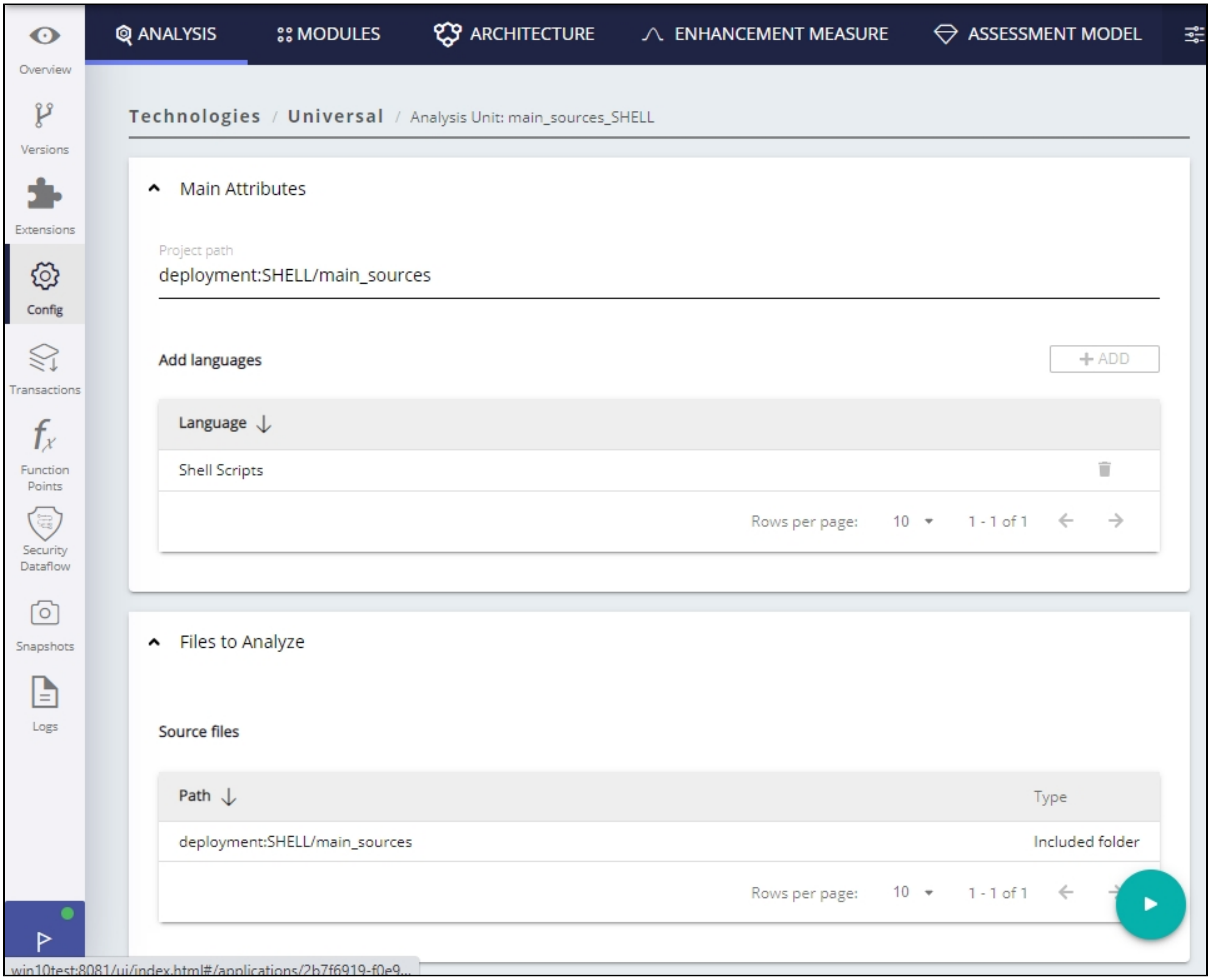
> There are **no analysis/technology configuration options** available for Shell, however you should check that at least one Shell analysis unit has been created as shown below.

AIP Console exposes the technology configuration options once a version has been **accepted/imported**, or an **analysis has been run**. Click **Universal Technology (3)** in the **Config** **(1)** > **Analysis** **(2)** tab to display the available options for your Shell source code:



Then choose the relevant **Analysis Unit (1)** to view the configuration:





Technologies / Universal / Analysis Unit: main_sources_SHELL

Main Attributes

Project path
deployment:SHELL/main_sources

Add languages

Language
Shell Scripts

Files to Analyze

Source files

Path	Type
deployment:SHELL/main_sources	Included folder

win10test:8081/ui/index.html#/applications/2b7f6919-f0e9...

Logging mechanism

Analysis log files

Analysis logs are stored in the default locations.

Shell Preprocessor

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

Location	Log file name
%PROGRAMDATA%\CAST\CAST\Logs\<unique_application_id>\	com.castsoftware.shell.prepro_<ExtensionVersion>_<YYYYMMDDHHMMSS>.log

What results can you expect?

Objects

Icon	Metamodel Name

▣	SHELL Project
☰	SHELL Program
f	SHELL Function
f	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
SHELL Function/Special Function/Program	callLink	SQL Table
SHELL Function/Special Function/Program	callLink	SQL Script

Embedded SQL

Shell allows the use of here-document to write SQL queries, for example:

```
#!/bin/sh
/opt/solid/bin/solsql db db <<abc
select count(*) from table_1;
select count(*) from table_2;
exit;
!;
abc
```

The Shell extension can use these queries to create links.

Call to SQL script

The Shell extension allows links to SQL script when used with a dependency to the SQL analyzer

For example:

script.sh

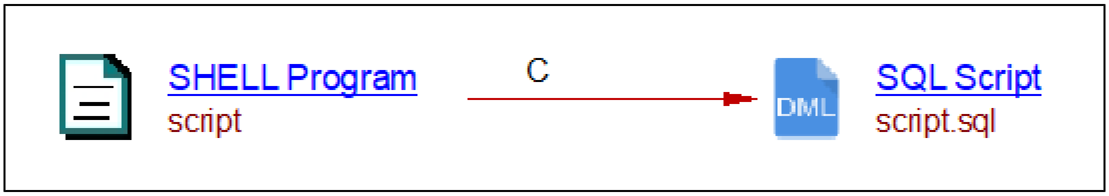
```
#!/bin/env bash

sqlplus script.sql
```

script.sql

```
CREATE TABLE ...
```

In Enlighten



Links to external programs

The following call pattern are supported

- ./exe
- /path/to/exe
- python
- java
- nohup
- eval
- wlst.sh

Shell to Java links

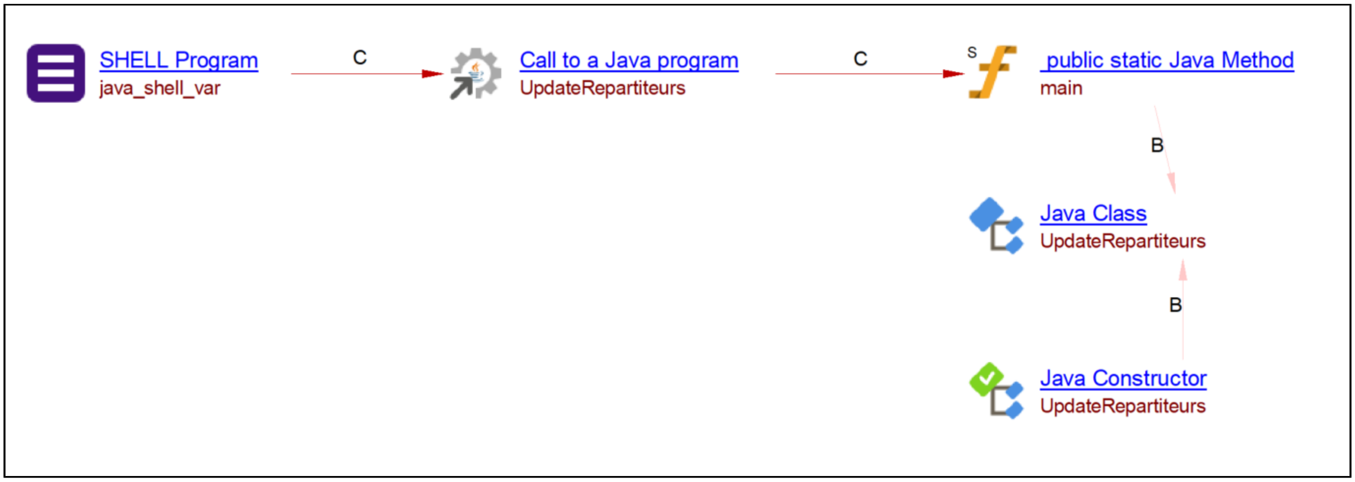
The extension does support links between Shell programs and Java objects (for example methods). Links will be created between these technologies.

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
...
```

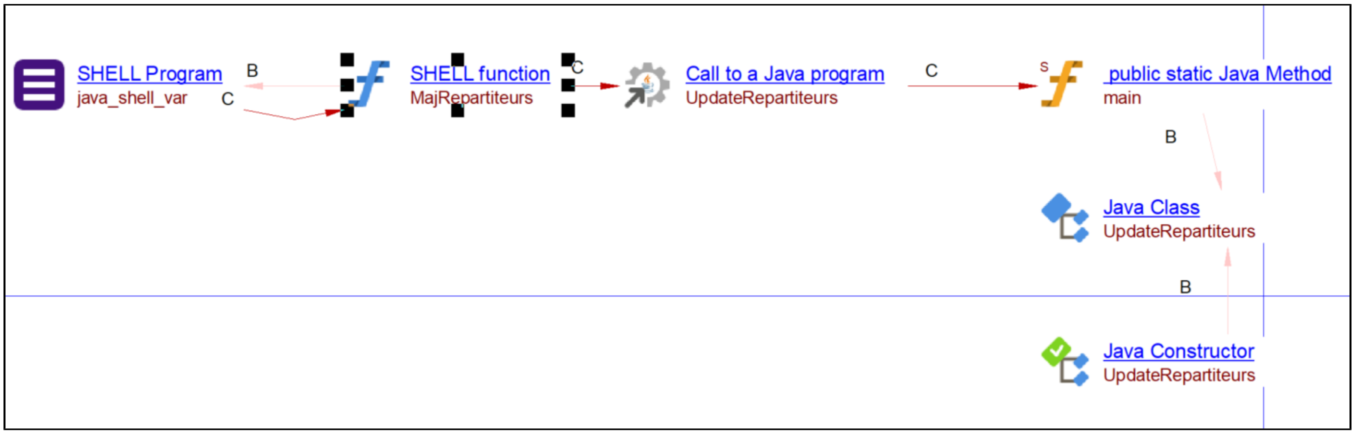
wil generate the following diagram



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

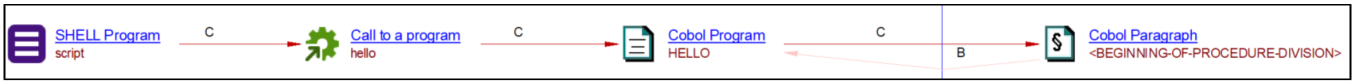
COBOL file

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

SHELL script

```
./hello
```

will generate the following diagram



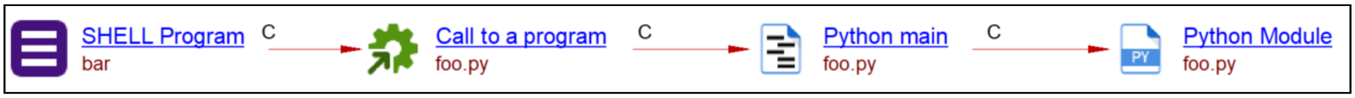
Shell to Python links

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:

1.2.11	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.11
1.2.10	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.10
1.2.9	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.9
1.2.8	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.8
1.2.7	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.7
1.2.6	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.6
1.2.5	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.5
1.2.4	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.4
1.2.3	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.3
1.2.3-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.3-funcrel
1.2.2-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.2-funcrel
1.2.1-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.1-funcrel
1.2.0-funcrel	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.0-funcrel
1.2.0-beta1	https://technologies.castsoftware.com/rules?sec=srs_shell&ref=||1.2.0-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".