Python 1.4

Summary: This document provides basic information about the extension providing **Python** support.

Extension ID

com.castsoftware.python

What's new?

Please see **Python 1.4 - Release Notes** for more information

Description

This extension provides support for **Python**.

In what situation should you install this extension?

If your application contains **Python** source code (both **.py** and **.jy** extensions are supported) and you want to view these object types and their links with other objects, then you should install this extension.

Files analyzed

Icons	File	Extension	Note
	**Python**	.py,	Python files - standard extension.
	**Jython**	.jy	By convention, Python files to be run in a Java implementation of the Python interpreter.
-	**YAML (YAML Ain't Markup Language)**	*.yml, *.yaml,	Files related to the **YAML** language, commonly used for configuration purposes. Necessary to interpret Amazon Web Services deployment code.

Supported Python versions

The following table displays the supported versions matrix:

Version	Support
3.x	✅
2.x	✅
1.x	❌

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	Security
✅	✅	✅

AIP Core compatibility

This extension is compatible with:

CAST AIP release	Supported
8.3.x	✓

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)

Framework support

Web Service Frameworks (client-side requests and/or server-side operations)	Support
requests	✓
urllib	✓
urllib2	✓
urllib3	✓
httplib	✓
httplib2	✓
http.client	✓
aiohttp	✓
flask	✓
falcon	✓
web2py	✓
Cherrypy	✓
FastAPI	✓

Dependencies with other extensions

Some CAST extensions require the presence of other CAST extensions in order to function correctly. The **Python** extension requires that the following other CAST extensions are also installed:

- **Web Services Linker** (internal technical extension)
- **CAST AIP Internal extension** (internal technical extension)

Note that when using the **CAST Extension Downloader** to download the extension and the **Manage Extensions** interface in **CAST Server Manager** to install the extension, any dependent extensions are **automatically** downloaded and installed for you. You do not need to do anything.

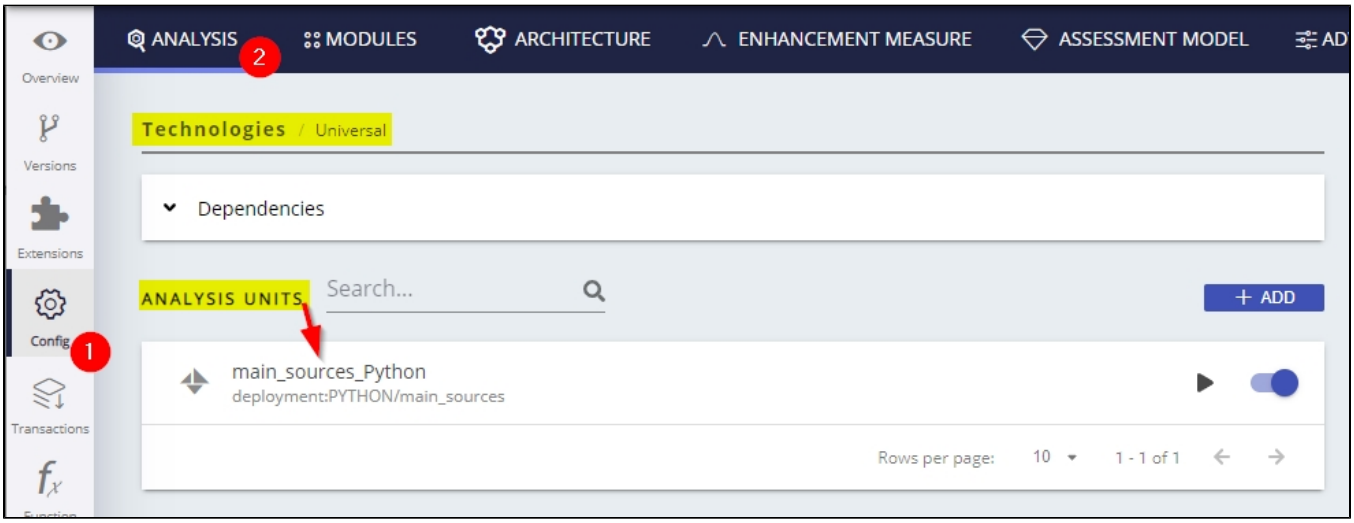
Download and installation instructions

The extension will be automatically downloaded and installed in AIP Console when you deliver Python code. 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**

Source code discovery

A discoverer is provided with the extension to automatically detect Python code: a Python project will be discovered for the package's root folder when at least one **.py or .jy (jython) file** is detected in the root folder or any sub-folders. For every Python project located, **one Universal Technology Analysis Unit** will be created:



Analysis - Automatic skipping of unit-test code and external libraries

The analyzer skips files that are recognized as forming part of testing code, i.e., in principle, code not pertaining to production code. The reason to avoid inclusion of testing code is that many Quality Rule violations are overrepresented in test code, either because code tends to be of poorer quality (certainly not critical) or prevalence of particular testing patterns. Accounting for test code would negatively impact the total score of the project.

Similarly we skip folders that contain external python libraries. Currently we only skip the canonical folders *site-packages* and *dist-packages* (the latter being used in certain Linux distributions). Not only analyzing external libraries is discouraged, but it can interfere with correct interpretation of supported libraries and frameworks, and have a serious impact in memory consumption and overall analysis performance.

The heuristics used by the analyzer are based on detecting (unit-test) library imports, and file and path naming conventions as summarized in the table below:

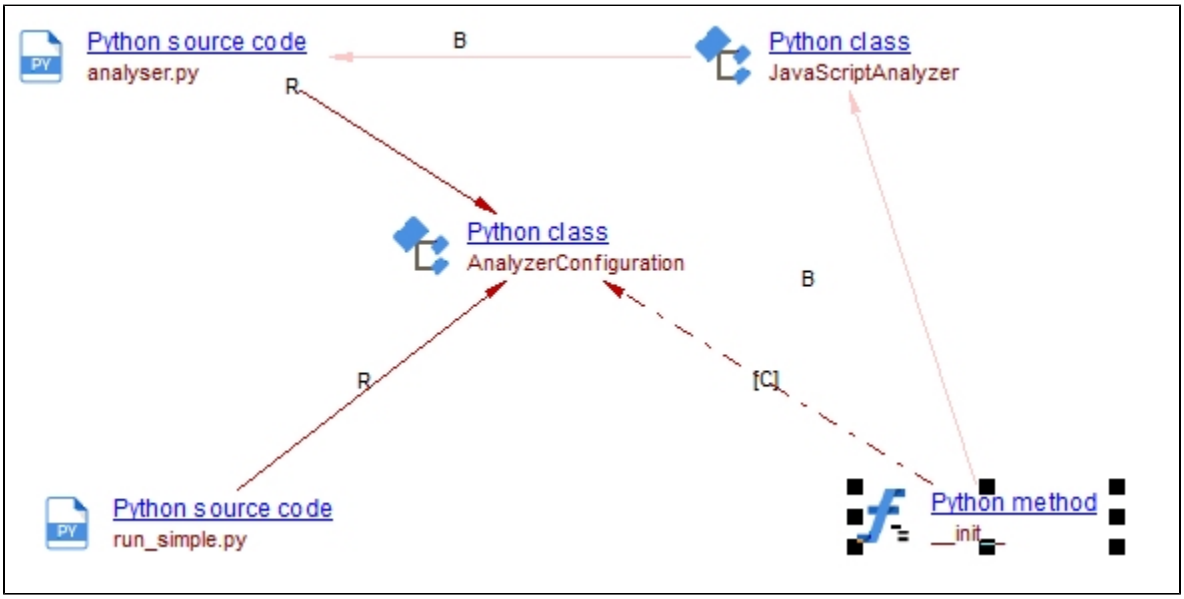
Type	Value	HeaderLines	MinimumCount
FilePath	**/test_*.py		
FilePath	**/*_test.py		
FilePath	**/*_test_*.py		
FilePath	**/test/*.py		
FilePath	**/tests/*.py		
FileContent	import unittest	12	
FileContent	from unittest import	12	
FileContent	from nose.tools import	12	
FileContent	self.assert		2
FilePath	**/site-packages/**		
FilePath	**/dist-packages/**		
FilePath	**/Python*/Lib/**		

FilePath	**/Python*/Scripts/**		
FilePath	**/Python*/Include/**		
FilePath	**/Python*/Bin/**		

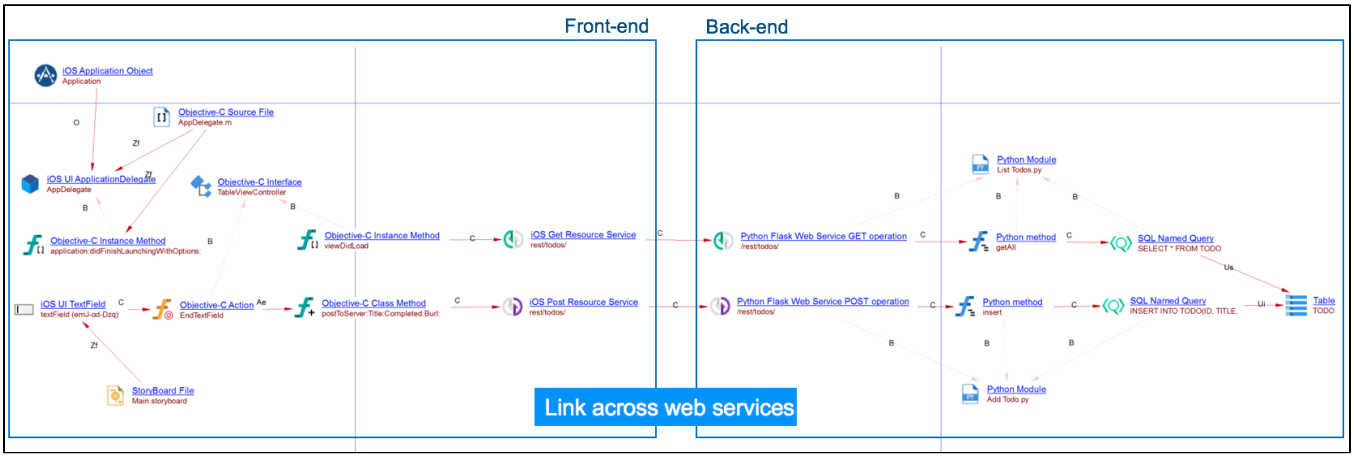
- The ** symbol represents any arbitrary path string, whereas * represents any string without directory slashes.
- The heuristics above should also similarly valid for .jy (jython) files.
- FilePath match is case-insensitive

What results can you expect?

Once the analysis/snapshot generation has completed, you can view the results in the normal manner:



Python Class and method example



iOS Front-end connected to a Python Flask Back-end.

Objects

The following specific objects are displayed in CAST Enlighten:

Icon	Description
	Python Project, Python External Library
	Python Module
	Python Class
	Python Method
	Python Script
	Python GET (*urllib, urllib2, httplib, httplib2, aiohttp*) service Python GET service request Python (*Flask, aiohttp)* Web Service GET operation Python Web Service Get Operation
	Python POST (*urllib, urllib2, httplib, httplib2, aiohttp*) service Python POST service request Python (*Flask, aiohttp)* Web Service POST operation Python Web Service Post Operation
	Python PUT (*urllib, urllib2, httplib, httplib2, aiohttp*) service Python PUT service request Python (*Flask, aiohttp)* Web Service PUT operation Python Web Service Post Operation
	Python DELETE (*urllib, urllib2, httplib, httplib2, aiohttp*) service Python DELETE service request Python *Flask, aiohttp* Web Service DELETE operation Python Web Service Delete Operation
	Python Web Service Any Operation
	Python Query, Python ORM Mapping, Python File Query
	RabbitMQ Python QueueCall ActiveMQ Python QueueCall IBM MQ Python QueueCall
	RabbitMQ Python QueueReceive ActiveMQ Python QueueReceive IBM MQ Python QueueReceive
	Python Call To Java Program
	Python Call To Generic Program
Amazon Web Services	

	Python Call to AWS Lambda Function
	Python Call to Unknown AWS Lambda Function
	Python AWS Lambda GET Operation
	Python AWS Lambda POST Operation
	Python AWS Lambda PUT Operation
	Python AWS Lambda DELETE Operation
	Python AWS Lambda ANY Operation
	Python AWS SQS Publisher, Python AWS SNS Publisher
	Python AWS SQS Receiver, Python AWS SNS Receiver
	Python AWS SQS Unknown Publisher, Python AWS SNS Unknown Publisher
	Python AWS SQS Unknown Receiver, Python AWS SNS Unknown Receiver
	Python S3 Bucket
	Python Unknown S3 Bucket
	Python DynamoDB Database
	Python DynamoDB Table

	Python Unknown DynamoDB Table
	Python Email, Python SMS

Python callable artifact

Python Script, Python Module and *Python Method* objects form part of Python (callable) artifacts.

Links

The following links are created:

- **call links** between methods
- **inherit link** between hierarchically related classes
- **refer link** from methods to class (constructor call)
- **use link** between modules through import
- **call links** between Python callable artifacts and Python Call objects
- **call links** between Python Call objects and external programs or lambda functions

The following links are created between Python ORM Mapping objects and database table objects:

- **useSelectLink** in case of SELECT operation
- **useDeleteLink** in case of DELETE operation
- **useInsertLink** in case of INSERT operation
- **useUpdateLink** in case of UPDATE operation
- **call links** in case of generic operation on S3 buckets

Structural Rules

The following structural rules are provided:

1.4.0-funcrel	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-funcrel
1.4.0-beta8	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta8
1.4.0-beta7	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta7
1.4.0-beta6	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta6
1.4.0-beta5	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta5
1.4.0-beta4	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta4
1.4.0-beta3	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta3
1.4.0-beta2	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta2
1.4.0-beta1	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-beta1
1.4.0-alpha2	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-alpha2
1.4.0-alpha1	https://technologies.castsoftware.com/rules?sec=srs_python&ref=||1.4.0-alpha1

You can also find a global list here: **https://technologies.castsoftware.com/rules?sec=t_1021000&ref=||**

Expected results for supported frameworks

Refer the following pages for the results that you can expect for each supported framework:

- Python 1.4 - Analysis Results - Web Service calls and operations support
- Python 1.4 - Analysis Results - Access to databases
- Python 1.4 - Analysis Results - File system access functions
- Python 1.4 - Analysis Results - Message Queues support
- Python 1.4 - Analysis Results - Amazon Web Services
- Python 1.4 - Analysis Results - Calls to external program from Python
- Python 1.4 - Analysis Results - Links handled by command line parsers

Limitations

- Not fully supported **Python Decorator function**.
- Quality rules do not apply to code inside the class definition (**class** or **"static" variables**)
- The "Avoid disabling certificate check when requesting secured urls" for 'urllib3' is only partially supported by detecting the call to 'urllib3.disable_warnings'.
- Limited Python resolution that leads to missing links:
  - No support for __**all**__
  - No support for variable of type class, function
- Flask:
  - Objects for other web service operations such as PATCH are not generated.
  - The *endpoint* abstraction layer between functions and annotations is not considered. When using *add_url_rule* the endpoint argument is taken as the calling function name.
- Cherrypy:
  - Only support default request.dispatcher "cherrypy.dispatch.MethodDispatcher()".
- **Django** framework is not supported.
- Java-Python interoperability via **Jython** is not supported. However the files with the specific extension .jy for Jython is analyzed as a regular Python file.
- Message queues
  - To generate queue message objects the queue name has to be initialized explicitly in the code (dynamic naming not supported).