{{lastupdated_at}} by {{lastupdated_by}}

Continuous Integration

This page summarizes information about the Continuous Integration strategy and procedures that have been implemented for GammaLib.

Continuous Integration Pipelines

Introduction

Continuous integration (CI) is performed in a number of pipelines, each one consisting of a number of integration steps (such as building, testing, documenting, etc.) that will be executed consecutively. Each pipeline step is executed independently of the success of the preceding step. If an integration step fails, a notification is sent.

Below a summary of the integration pipelines that have been implemented. **All CI pipelines run on the devel branch of the Git repository.** Pipelines are either triggered manually or automatically.

Pipeline	Trigger	Task
OS	nightly (00:00)	Validation on various operating systems
Compiler	nightly (02:00)	Validation using various compilers (i386 and x86_64 architectures)
Python	nightly (04:00)	Validation using various Python versions (i386 and x86_64 architectures, gcc and clang compilers)
swig	manually	Validation using various swig versions (for Python 2 and Python 3)

OS pipeline

The Operating System (OS) pipeline performs a CI of GammaLib on various operating systems. For the Linux system, a large number of distributions is tested. In addition to Linux, BSD and Solaris are tested.

The pipeline consists of 3 steps for GammaLib: building (make), unit testing (make check), and creation of the Doxygen documentation (make doxygen).

After the documentation step, additional steps are added for dependent projects (for the moment there is a build and a unit testing step for ctools, resulting in a 5 step OS pipeline).

Compiler pipeline

The compiler pipeline performs a CI of GammaLib using different compiler types and versions. For the moment, gcc and clang are tested. gcc is tested from version 3.2.x on up to the most recent version, clang is tested for version 3.1). The tests are performed on the i386 and x86_64 architectures.

The pipeline consists of 2 steps for GammaLib: building (make) and unit testing (make check).

After the unit testing step, additional steps are added for dependent projects (for the moment there is a build and a unit testing step for ctools, resulting in a 4 step compiler pipeline).

Python pipeline

The Python pipeline performs a CI of GammaLib using different Python versions. For the moment, Python 2.7.3 and Python 3.2.3 are tested. The tests are performed on the i386 and x86_64 architectures and using the gcc 4.7.0 and clang 3.1 compilers.

The pipeline consists of 2 steps for GammaLib: building (make) and unit testing (make check).

After the unit testing step, additional steps are added for dependent projects (for the moment there is a build and a unit testing step

for ctools, resulting in a 4 step Python pipeline).

swig pipeline

The swig pipeline performs a CI of GammaLib using different swig versions. The tests are performed using Python 2.7.3 and Python 3.2.3.

The pipeline consists of 2 steps for GammaLib: building (make) and unit testing (make check).

After the unit testing step, additional steps are added for dependent projects (for the moment there is a build and a unit testing step for ctools, resulting in a 4 step swig pipeline).