Atacama
Large
Millimeter /
submillimeter
Array

Sampling System GUI

ALMA-BBB-CCC-A-DDD

Version: A
Status: Draft

2009-10-20

Keywords: sampling, trending, properties, java, plotting, sampling system, ssg

Prepared	Cristián Maureira	10/20/2009	
	Name	Date	Signature
Approved	---		
	Name	Date	Signature
Released	---		
	Name	Date	Signature

Change Record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
A	2009-10-20	All	Creation.

Table of Contents

1 Introduction

1.1 Purpose

The following document has been created to describe the usage of the Sampling System GUI (SSG hereafter) tool. It gives a brief explanation of the underlying Sampling System and, in more extension, it explains the concepts involved in the usage and the usage itself, of the SSG tool.

1.2 Scope

This document is intended to be read by all the users of Sampling System GUI (SSG), be them astronomers, operators, engineers or any other.

This document also contains a general overview of the design of SSG as a reference for future developers of the tool.

1.3 About the Sampling System

The ACS Sampling System is a collection of objects designed to easily sample an ACS Component's Property value over time. For more information about ACS Components and Properties, please refer to 1.5.

The Sampling System is composed by two kind of entities: **Sampling Objects** and **Sampling Managers**. A Sampling Object is a distributed object which represents and manages one and only one, sample. A Sampling Object is composed by the following elements:

- The ACS **Characteristic** Component containing the desired property.
- The Property to be sampled on the given Component.
- A period of time between sampling intervals.

Sampling Objects manage the sampling of the given property. They communicate with the property by acquiring a direct CORBA reference of it and sample it with the given frequency. The acquired values are then sent through a Notification Channel (for details refer to 1.5). Sampling Objects allow to start/pause/resume/stop the sampling process by making remote calls over them.

The Sampling Manager is an ACS Component that offers the functionality to create Sampling Objects. This functionality, then, can be accessed by other ACS Components or by ACS Clients. SSG is an example of such clients.

1.4 About SSG

SSG is a client to the Sampling System. It communicates with one or several Sampling Managers to create one or several Sampling Objects, group them as needed, in order to sample one or several properties and, finally, plot the values in user-time (i.e., as they arrive through the Notification Channel). Therefore, it allows an easy, quick visualization of a system's behavior during a period of time or under certain circumstances and gives the possibility of visually correlate the values of different properties of the system.

1.5 Reference Documents

You can find more information about Sampling System GUI in :

[1] ACS Sampling System

[2] ALMA Common Software

[3] Java Component/Container Framework Tutorial

[4] Notification Channel Design - Design, User's Manual and Tutorial

1.6 Abbreviations and Acronyms

ACS	ALMA Common Software
SS	Sampling System
SSG	Sampling System GUI

1.7 Glossary

Component : Represents a piece of hardware and its properties (Example: Clock).

Property : A characteristic of a component, SSG always plot properties (Example: the property temperature of the component CPU).

Sampling group : Is a group of properties to be plotted, each sampling group has its own plot window (Example : clock temperature and lamp brightness are in the sampling group `myGroup`, they will be on the plot window called `myGroup`).

Sampling time : How much time SSG will be plotting the properties of the sampling group

Time window : How many seconds will be visible on the plot window, the width of the plot window in seconds.

Frequency : How often the samples will be updated.

Sample : A single property value at one specific time.

Client : SSG is a client of the sampling manager.

Sampling manager : Returns sampling objects to the client.

Sampling object : It creates a notification channel and send data to the client.

Notification channel : Created by the sampling object for data delivering and to cache the data in order to optimize the network load.

1.8 Stylistic Conventions

In this document the following styles are used :

bold

Is used to highlight important words.

italic

Is used for important phrases.

`teletype`

Is used for commands and filenames.

2 General Overview

The Sampling System GUI (SSG) is a Java graphical tool developed using SWT [2] with the Jchart2D library [1] and its purpose is to show the current values of an ACS Property to the user, by means of an on-line plot. This way, the user can have a quick idea of how the values of the properties are behaving and recognize situations rapidly.

Also, the values may be saved into a comma-separated-values (csv) file for later inspection.

3 Design Overview

Two subsystem where identified. The Monitor Subsystem and the Display Subsystem.

Monitor Subsystem: The monitor subsystem connects to the ACS SS of a given ACS Instance and gathers data from the ACS SS. It also prepares for plotting.

Display Subsystem: The display subsystem is responsible for display data in screen with plots (optional) and save the sampled data in a file.

3.1 Implementation

SSG was developed using Java as programming language.

The classes were separated in two packages: core package 2 contains the Monitor subsystem functionality and SamplingSystemUI packages 1 contains all the GUI necessary for Display Subsystem.

3.2 Class Diagrams

3.2.1 GUI

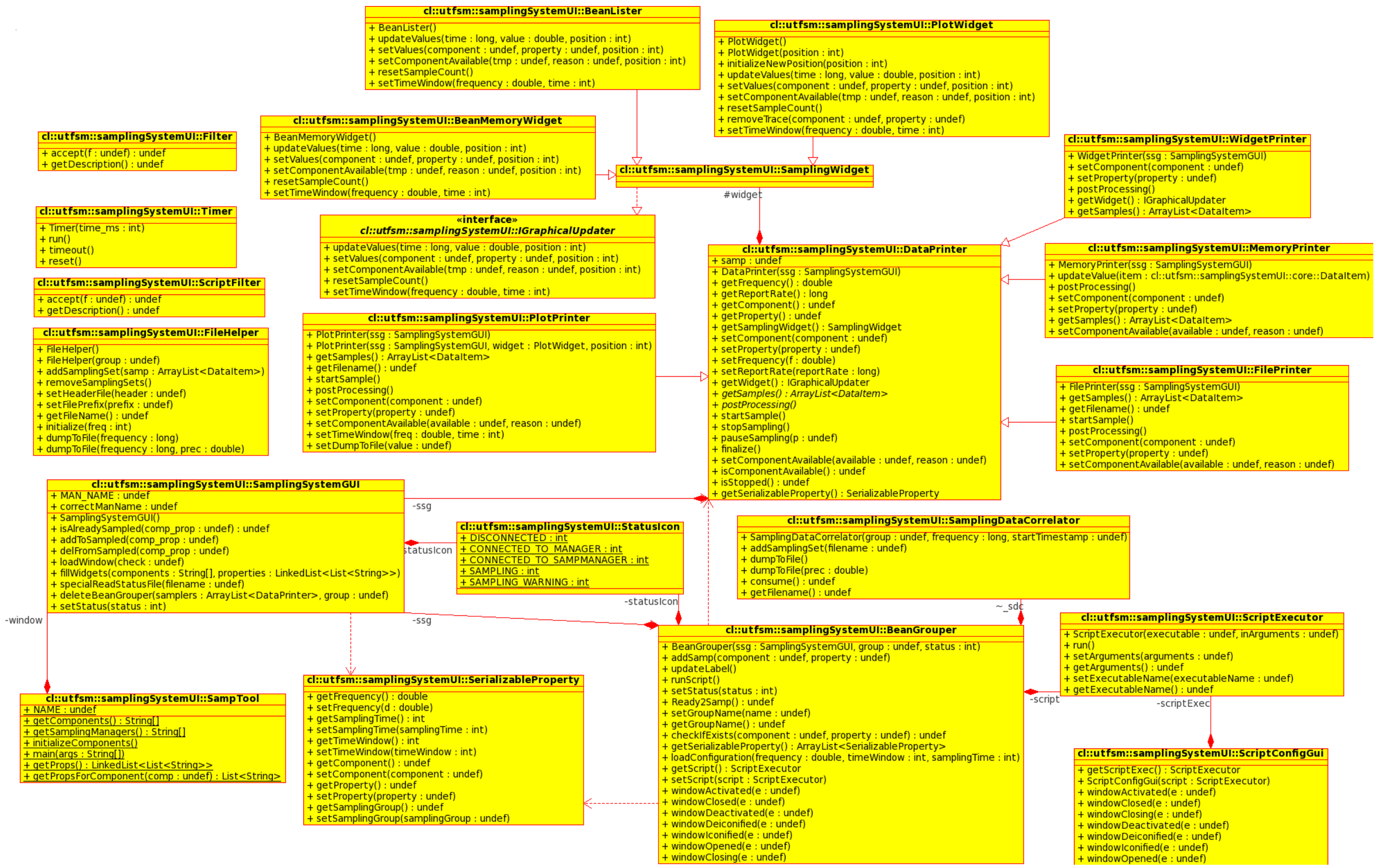
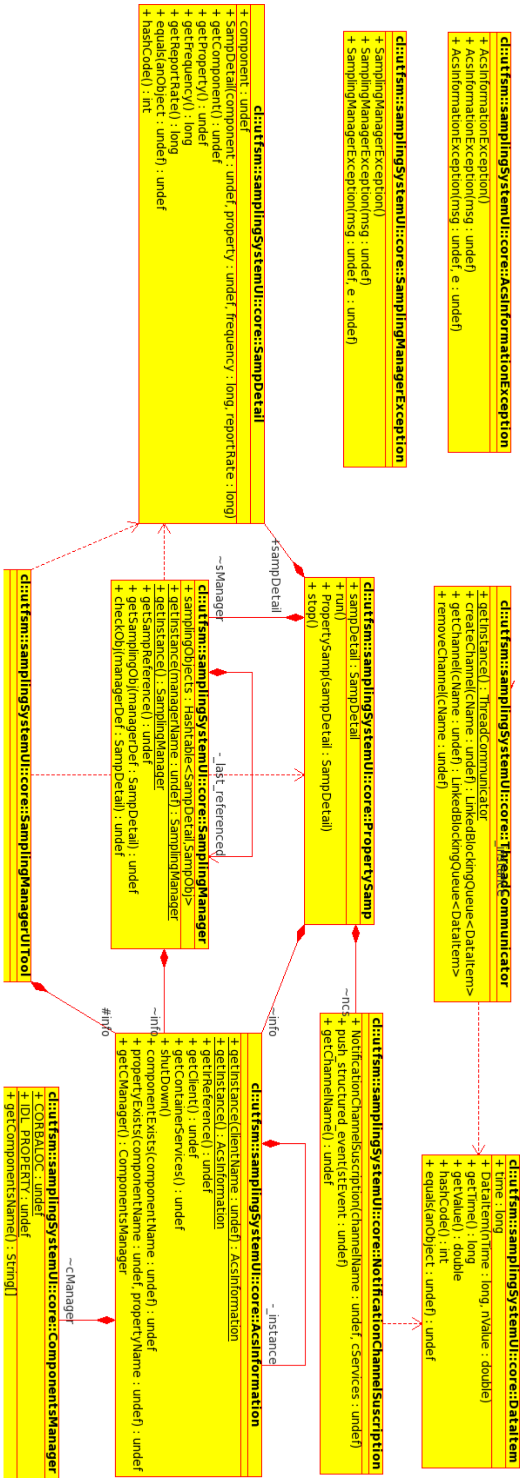

cl::utfsm::samplingSystemUI::BeanLister
+ BeanLister()
+ updateValues(time : long, value : double, position : int)
+ setValues(component : undef, property : undef, position : int)
+ setComponentAvailable(tmp : undef, reason : undef, position : int)
+ resetSampleCount()
+ setTimeWindow(frequency : double, time : int)
cl::utfsm::samplingSystemUI::PlotWidget
+ PlotWidget()
+ PlotWidget(position : int)
+ initializeNewPosition(position : int)
+ updateValues(time : long, value : double, position : int)
+ setValues(component : undef, property : undef, position : int)
+ setComponentAvailable(tmp : undef, reason : undef, position : int)
+ resetSampleCount()
+ removeTrace(component : undef, property : undef)
+ setTimeWindow(frequency : double, time : int)
cl::utfsm::samplingSystemUI::Filter
+ accept(f : undef) : undef
+ getDescription() : undef
cl::utfsm::samplingSystemUI::BeanMemoryWidget
+ BeanMemoryWidget()
+ updateValues(time : long, value : double, position : int)
+ setValues(component : undef, property : undef, position : int)
+ setComponentAvailable(tmp : undef, reason : undef, position : int)
+ resetSampleCount()
+ setTimeWindow(frequency : double, time : int)
cl::utfsm::samplingSystemUI::SamplingWidget
#widget
cl::utfsm::samplingSystemUI::WidgetPrinter
+ WidgetPrinter(ssg : SamplingSystemGUI)
+ setComponent(component : undef)
+ setProperty(property : undef)
+ postProcessing()
+ getWidget() : IGraphicalUpdater
+ getSamples() : ArrayList<DataItem>
cl::utfsm::samplingSystemUI::Timer
+ Timer(time_ms : int)
+ run()
+ timeout()
+ reset()
«interface»
cl::utfsm::samplingSystemUI::IGraphicalUpdater
+ updateValues(time : long, value : double, position : int)
+ setValues(component : undef, property : undef, position : int)
+ setComponentAvailable(tmp : undef, reason : undef, position : int)
+ resetSampleCount()
+ setTimeWindow(frequency : double, time : int)
cl::utfsm::samplingSystemUI::DataPrinter
+ samp : undef
+ DataPrinter(ssg : SamplingSystemGUI)
+ getFrequency() : double
+ getReportRate() : long
+ getComponent() : undef
+ getProperty() : undef
+ getSamplingWidget() : SamplingWidget
+ setComponent(component : undef)
+ setProperty(property : undef)
+ setFrequency(f : double)
+ setReportRate(reportRate : long)
+ getWidget() : IGraphicalUpdater
+ getSamples() : ArrayList<DataItem>
+ postProcessing()
+ startSample()
+ stopSampling()
+ pauseSampling(p : undef)
+ finalize()
+ setComponentAvailable(available : undef, reason : undef)
+ isComponentAvailable() : undef
+ isStopped() : undef
+ getSerializableProperty() : SerializableProperty
cl::utfsm::samplingSystemUI::MemoryPrinter
+ MemoryPrinter(ssg : SamplingSystemGUI)
+ updateValue(item : cl::utfsm::samplingSystemUI::core::DataItem)
+ postProcessing()
+ setComponent(component : undef)
+ setProperty(property : undef)
+ getSamples() : ArrayList<DataItem>
+ setComponentAvailable(available : undef, reason : undef)
cl::utfsm::samplingSystemUI::ScriptFilter
+ accept(f : undef) : undef
+ getDescription() : undef
cl::utfsm::samplingSystemUI::FileHelper
+ FileHelper()
+ FileHelper(group : undef)
+ addSamplingSet(samp : ArrayList<DataItem>)
+ removeSamplingSets()
+ setHeaderFile(header : undef)
+ setFilePrefix(prefix : undef)
+ getFileName() : undef
+ initialize(freq : int)
+ dumpToFile(frequency : long)
+ dumpToFile(frequency : long, prec : double)
cl::utfsm::samplingSystemUI::PlotPrinter
+ PlotPrinter(ssg : SamplingSystemGUI)
+ PlotPrinter(ssg : SamplingSystemGUI, widget : PlotWidget, position : int)
+ getSamples() : ArrayList<DataItem>
+ getFilename() : undef
+ startSample()
+ postProcessing()
+ setComponent(component : undef)
+ setProperty(property : undef)
+ setComponentAvailable(available : undef, reason : undef)
+ setTimeWindow(freq : double, time : int)
+ setDumpToFile(value : undef)
cl::utfsm::samplingSystemUI::FilePrinter
+ FilePrinter(ssg : SamplingSystemGUI)
+ getSamples() : ArrayList<DataItem>
+ getFilename() : undef
+ startSample()
+ postProcessing()
+ setComponent(component : undef)
+ setProperty(property : undef)
+ setComponentAvailable(available : undef, reason : undef)
cl::utfsm::samplingSystemUI::SamplingSystemGUI
+ MAN_NAME : undef
+ correctManName : undef
+ SamplingSystemGUI()
+ isAlreadySampled(comp_prop : undef) : undef
+ addToSampled(comp_prop : undef)
+ delFromSampled(comp_prop : undef)
+ loadWindow(check : undef)
+ fillWidgets(components : String[], properties : LinkedList<List<String>>)
+ specialReadStatusFile(filename : undef)
+ deleteBeanGrouper(samplers : ArrayList<DataPrinter>, group : undef)
+ setStatus(status : int)
-ssg
cl::utfsm::samplingSystemUI::StatusIcon
+ DISCONNECTED : int
+ CONNECTED_TO_MANAGER : int
+ CONNECTED_TO_SAMPMANAGER : int
+ SAMPLING : int
+ SAMPLING_WARNING : int
statusIcon
cl::utfsm::samplingSystemUI::SamplingDataCorrelator
+ SamplingDataCorrelator(group : undef, frequency : long, startTimestamp : undef)
+ addSamplingSet(filename : undef)
+ dumpToFile()
+ dumpToFile(prec : double)
+ consume() : undef
+ getFilename() : undef
-statusIcon
~_sdc
-window
-ssg
cl::utfsm::samplingSystemUI::BeanGrouper
+ BeanGrouper(ssg : SamplingSystemGUI, group : undef, status : int)
+ addSamp(component : undef, property : undef)
+ updateLabel()
+ runScript()
+ setStatus(status : int)
+ Ready2Samp() : undef
+ setGroupName(name : undef)
+ getGroupName() : undef
+ checkIfExists(component : undef, property : undef) : undef
+ getSerializableProperty() : ArrayList<SerializableProperty>
+ loadConfiguration(frequency : double, timeWindow : int, samplingTime : int)
+ getScript() : ScriptExecutor
+ setScript(script : ScriptExecutor)
+ windowActivated(e : undef)
+ windowClosed(e : undef)
+ windowDeactivated(e : undef)
+ windowDeiconified(e : undef)
+ windowIconified(e : undef)
+ windowOpened(e : undef)
+ windowClosing(e : undef)
cl::utfsm::samplingSystemUI::ScriptExecutor
+ ScriptExecutor(executable : undef, inArguments : undef)
+ run()
+ setArguments(arguments : undef)
+ getArguments() : undef
+ setExecutableName(executableName : undef)
+ getExecutableName() : undef
-script
-scriptExec
cl::utfsm::samplingSystemUI::SampTool
+ NAME : undef
+ getComponents() : String[]
+ getSamplingManagers() : String[]
+ initializeComponents()
+ main(args : String[])
+ getProps() : LinkedList<List<String>>
+ getPropsForComponent(comp : undef) : List<String>
cl::utfsm::samplingSystemUI::SerializableProperty
+ getFrequency() : double
+ setFrequency(d : double)
+ getSamplingTime() : int
+ setSamplingTime(samplingTime : int)
+ getTimeWindow() : int
+ setTimeWindow(timeWindow : int)
+ getComponent() : undef
+ setComponent(component : undef)
+ getProperty() : undef
+ setProperty(property : undef)
+ getSamplingGroup() : undef
+ setSamplingGroup(samplingGroup : undef)
cl::utfsm::samplingSystemUI::ScriptConfigGui
+ getScriptExec() : ScriptExecutor
+ ScriptConfigGui(script : ScriptExecutor)
+ windowActivated(e : undef)
+ windowClosed(e : undef)
+ windowClosing(e : undef)
+ windowDeactivated(e : undef)
+ windowDeiconified(e : undef)
+ windowIconified(e : undef)
+ windowOpened(e : undef)

Figure 1: GUI Diagram

3.2.2 Core



cl::utfsm::samplingSystemUI::core::AcsInformationException
+ AcsInformationException()
+ AcsInformationException(msg : undef)
+ AcsInformationException(msg : undef, e : undef)
cl::utfsm::samplingSystemUI::core::SamplingManagerException
+ SamplingManagerException()
+ SamplingManagerException(msg : undef)
+ SamplingManagerException(msg : undef, e : undef)
cl::utfsm::samplingSystemUI::core::SampDetail
+ component : undef
+ SampDetail(component : undef, property : undef, frequency : long, reportRate : long)
+ getComponent() : undef
+ getProperty() : undef
+ getFrequency() : long
+ getReportRate() : long
+ equals(anObject : undef) : undef
+ hashCode() : int
cl::utfsm::samplingSystemUI::core::ThreadCommunicator
+ getInstance() : ThreadCommunicator
+ createChannel(cName : undef) : LinkedBlockingQueue<DataItem>
+ getChannel(cName : undef) : LinkedBlockingQueue<DataItem>
+ removeChannel(cName : undef)
_instance
cl::utfsm::samplingSystemUI::core::PropertySamp
+ sampDetail : SampDetail
+ run()
+ PropertySamp(sampDetail : SampDetail)
+ stop()
+sampDetail
~sManager
cl::utfsm::samplingSystemUI::core::SamplingManager
+ samplingObjects : Hashtable<SampDetail,SampObj>
+ getInstance(managerName : undef) : SamplingManager
+ getInstance() : SamplingManager
+ getSampReference() : undef
+ getSamplingObj(managerDef : SampDetail) : undef
+ checkObj(managerDef : SampDetail) : undef
- _last_referenced
cl::utfsm::samplingSystemUI::core::SamplingManagerUITool
~info
~info
#info
~ncs
cl::utfsm::samplingSystemUI::core::NotificationChannelSuscription
+ NotificationChannelSuscription(channelName : undef, cServices : undef)
+ push_structured_event(stEvent : undef)
+ getChannelName() : undef
cl::utfsm::samplingSystemUI::core::DataItem
+ time : long
+ DataItem(nTime : long, nValue : double)
+ getTime() : long
+ getValue() : double
+ hashCode() : int
+ equals(anObject : undef) : undef
cl::utfsm::samplingSystemUI::core::AcsInformation
+ getInstance(clientName : undef) : AcsInformation
+ getInstance() : AcsInformation
+ getIrReference() : undef
+ getClient() : undef
+ getContainerServices() : undef
+ shutDown()
+ componentExists(componentName : undef) : undef
+ propertyExists(componentName : undef, propertyName : undef) : undef
+ getCManager() : ComponentsManager
- _instance
~cManager
cl::utfsm::samplingSystemUI::core::ComponentsManager
+ CORBALOC : undef
+ IDL_PROPERTY : undef
+ getComponentsName() : String[]

Figure 2: CORE Diagram

4 Installation Manual

Actually SSG is supported in ACS 7 and ACS 8.

If you want the latest source code, you must checkout from subversion:

```
svn co https://acs.inf.utfsm.cl/repos/software/ssg
```

Or you can checkout the last version in the official ACS CVS:

```
export CVSROOT=:pserver:userName@cvs.alma.cl:/project2/CVS
cvs login
cvs co ACS/LGPL/CommonSoftware/acssampGUI
```

Please note that you need a valid **userName**.[1]

1. First, the user must have a correct configuration of the ACS environment and know that it works correctly, in other words, the user must have a working installation of ACS and the user must verify the follow environment variables and have sourced the `.bash_profile.acs`:

   ```
                   export INTROOT=\$HOME/introot
                   export MODPATH=1
                   source \$HOME/.acs/.bash\_profile.acs
   ```

2. Second, set up the example CDB (under test/CDB-ACS7 or test/CDB-ACS8). If the user use ACS 8 the CDB must be set as follows:

   ```
                   export ACS\_CDB=\$HOME/ssg/trunk/test/CDB-ACS8
   ```

   Please notice that the user needs to configure the CDB properly, specifying the present Sampling Managers.

3. Third, the user can start ACS and the container defined in the CDB called Container:

   ```
                   acsStart
                   acsStartContainer -cpp Container
   ```

4. Fourth, the user can build SSG. In this case, the user can use the trunk version or some tag For example, to build the trunk:

   ```
                   cd ssg/trunk/src
                   make all install
   ```

   This process generates all the jar files and makes the installation in the $INTROOT$ area.

5. Finally, if the user wants to execute SSG:

   ```
                   sampTool
   ```

[1] For more information about the CVS, visit
http://almasw.hq.eso.org/almasw/bin/view/SE/WANdiscoUserInformation

5 User Manual

5.1 Starting SSG

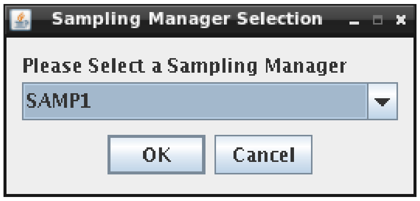
To start SSG and, assuming that the user has already set up his ACS environment, he must execute:

```
sampTool
```

Environment variables like `MANAGER_REFERENCE` are honored by SSG as usual.

5.2 Sampling Manager Selection

When SSG starts, it looks into the CDB for available Sampling Managers and displays them in a list, so the user can select one of them to work with. See the figure 3.

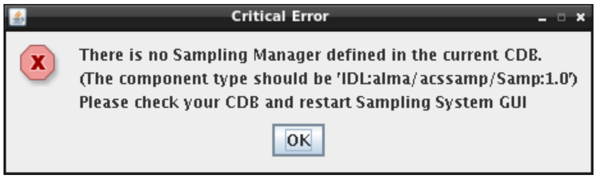


Please Select a Sampling Manager

SAMP1

Figure 3: Sampling Manager Selection

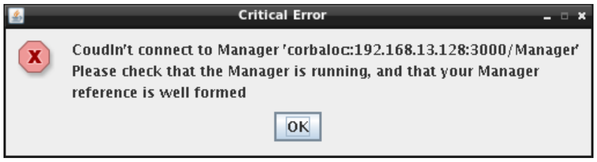
There **must** be at least one Sampling Manager defined in the CDB (not necessarily executing at this time). If there are no Sampling Managers defined in the CDB, SSG will notify you with an error dialog 4 and will fail gracefully.



There is no Sampling Manager defined in the current CDB.
(The component type should be 'IDL:alma/acssamp/Samp:1.0')
Please check your CDB and restart Sampling System GUI

Figure 4: Error without a Sampling Manager defined

If there is no `ACS_MANAGER` the user will receive the error of figure 5. The solution is to verify if an `ACS_INSTANCE` exists and, if not, you need to close SSG and start ACS.

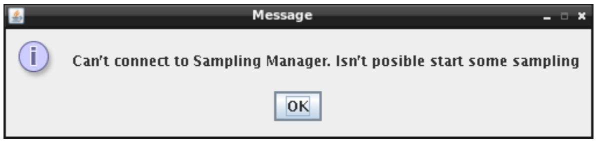


Critical Error

Coudln't connect to Manager 'corbaloc:192.168.13.128:3000/Manager'
Please check that the Manager is running, and that your Manager
reference is well formed

Figure 5: Error caused because there is no `ACS_MANAGER` defined.

If there is no `SAMPLING_MANAGER` the user will receive the error of figure 6. You must check if your **Container** is up and running.

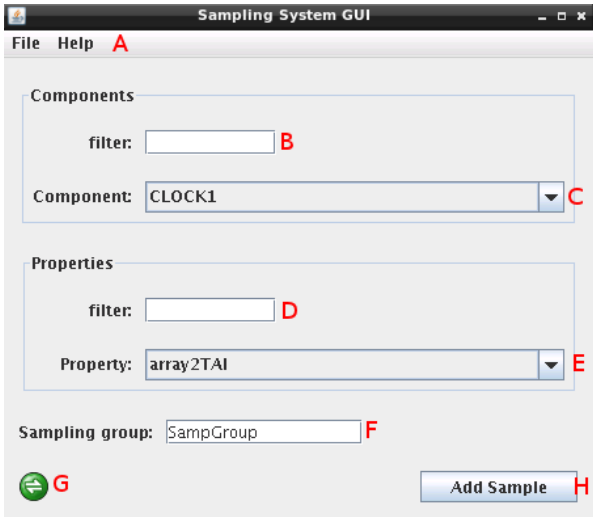


Message

Can't connect to Sampling Manager. Isn't posible start some sampling

Figure 6: Error caused because there is no `SAMPLING_MANAGER` defined

5.3 Main Window



Sampling System GUI

Components

filter:

Component: CLOCK1

Properties

filter:

Property: array2TAI

Sampling group: SampGroup

Figure 7: Main window of SSG.

After selecting the Sampling Manager and loaded or not the old status, the user must see the Main window 5.3. This window allows the user to select the properties that need to be sampled.

The Main window contains the following elements:

A: The `File` Menu have three options, Load, Save and Exit, which is explained in the next section.

B: A text-box to filter the list of the components.

C: A component chooser, presenting all the ACS Characteristic Components found in the current instance of ACS.

D: A text-box to filter the list of the properties.

E: A list with the properties of the component chosen in **C**.

F: The Sampling Group where the property will belong (see below).

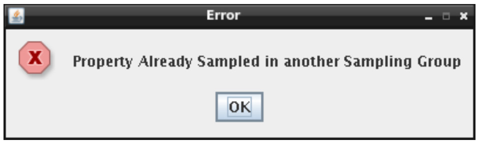
G: A status icon indicating the status of the connection with the Sampling Manager. Green means OK, while other colors represent different kind of failures. Put the mouse over the icon to receive more details about the problem.

H: When the Add Sample Button is pressed the current property is added to the current Sampling Group, if the sampling group does not exists, it is created.

Different samples can be grouped into "Sampling Groups". Sampling Groups will be visually displayed together and should be used to correlate different properties under the same timeline. For example, a user may want to know how the luminosity of a lamp changes while the voltage of the associated power supply changes. Therefore, these two properties must be added into the same Sampling Group. One property can be only in one sampling group at the same time.

In order to create a sample, the user first selects a Component from the first list and the needed Property from the second list. Then, it writes down the name of the Sampling Group where the sample will be added (which will be created in case that it doesn't exist yet). Finally, the user must press the "Add Sample" button.

If the user tried to add a sample that is in another Sampling Group, will receive the following error:



Property Already Sampled in another Sampling Group

Figure 8: Error caused because the property is already in a Sampling Group

5.4 File Menu

A: In this menu 5.4 the user can *Load status*, *Save status* or *Exit* the main window.

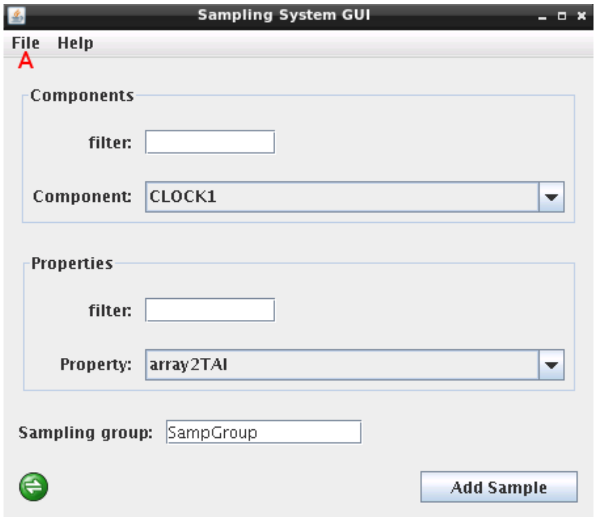


Figure 9: File Menu

5.4.1 Load a GUI status (Optional step)

The user can also, load an existing SSG status file with the extension file `.ssgst`: 5.4.1

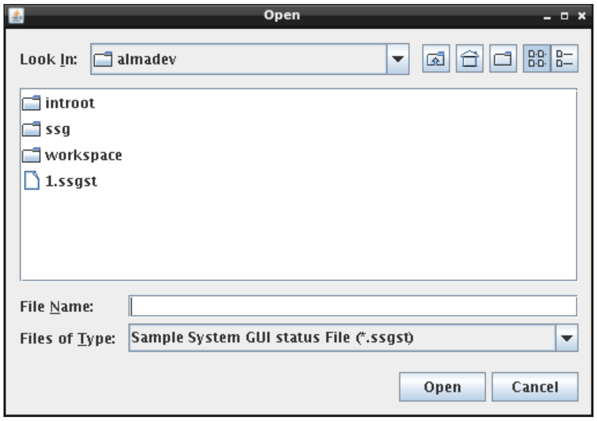
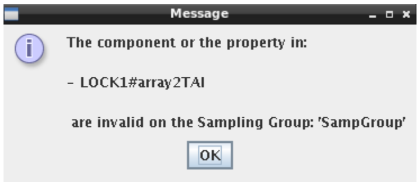


Figure 10: Load previous status dialog.

Also, you can load a status file from the command line, as follows:

```
./sampTool -f <ssgst file>
```

If some **Component** or **Property** is not into the CDB, the Sampling Manager will be unable to load it, so a display message 5.4.1 will appear to show the problematic `Component#Property` couple.



The component or the property in:

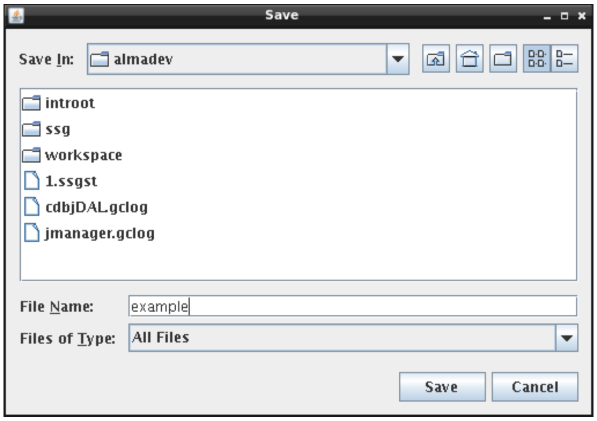
- LOCK1#array2TAI

are invalid on the Sampling Group: 'SampGroup'

Figure 11: Error when the user tried to add an invalid Component or Property.

5.4.2 Save a GUI status (Optional step)

In the same way as using Load a SSG status, the user can save it in a `ssgst` file at any time. 5.4.2



Save In: almadev

File Name: example

Files of Type: All Files

Figure 12: Save status dialog.

5.5 Help Menu

A: In this menu 5.5 the user can gather information about the development team.

5.6 Sampling Window

While samples are added through the Main window, 5.6 Sampling Groups are defined. When a Sampling Group is first instantiated, a new Sampling Window is created for it.

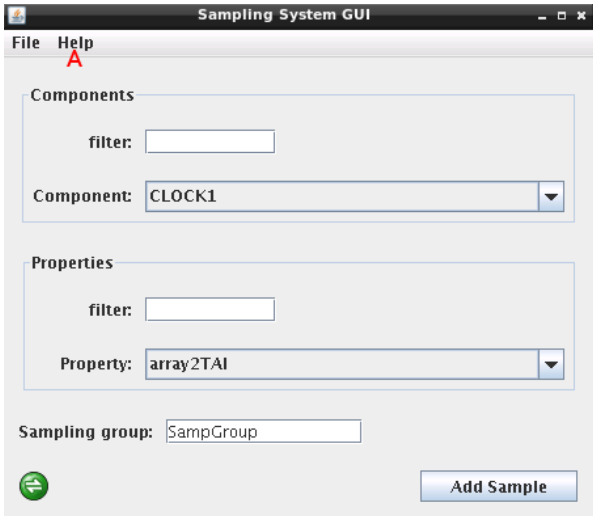
Here is an explanation of the important parts of this window:

A: The data will be plotted here

B: Component#Property : is the legend of the graphic (same color as the plotted data)

C: Play button : starts the plotting.

D: Pause button: pauses the plotting.



filter:

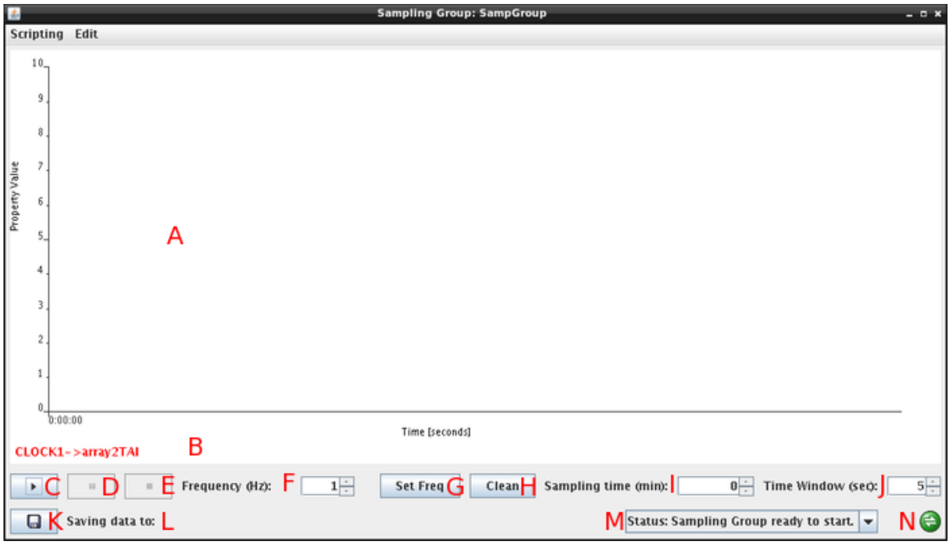
Component: CLOCK1

filter:

Property: array2TAI

Sampling group: SampGroup

Figure 13: Help Menu



Frequency (Hz): 1

Sampling time (min): 0

Time Window (sec): 5

Saving data to:

Status: Sampling Group ready to start.

Figure 14: A Sampling Window

E: Stop button : stops the plotting.

F: The user can type a new frequency here.

G: Change the current frequency to the typed frequency in **F**.

H: Clean the plotted data (all the traces will be removed from the plot).

I: Set the duration time of the plotting.

J: Set the quantity of seconds that will be visible on the window.

K: Save the current data.

L: Name of the file to save the data.

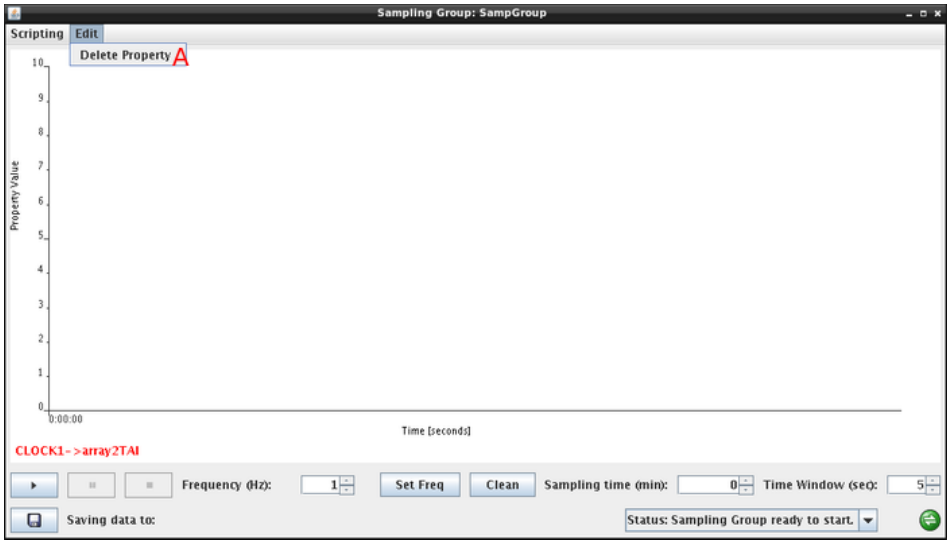
M: Status list of the sampling process for each property.

N: The status icon indicates the overall status of the sampling processes. Each color represents a different state. Put the mouse over the icon to get more details or refer to the status list (see L) for a per-property status when the sampling has already started.

In this step the user has various options before starting the sampling **(C)** If a user notes the left-inferior corner, he must see a pressed button **(K)**; this means that all the sampling data will be saved into a file. This is the default option, if a user doesn't want to save the information, he should press the button once.

- In **(F)** you can change the frequency to another value (1 Hz is the default value).
- In **(I)** you can change the Sampling time for this sample (like a countdown for sampling).
- In **(J)** you can change the Time window, this means the plotting time that will be shown in the window.
- If you have all this options set, the Status is Sampling Group ready to start **(M)** and the color of **(N)** is blue, you can start the sampling pressing the play button **(C)**.
- When the sampling has began, you can:
  - Pause the plotting to see some details of the plot (the sampling doesn't stop, only the plot)
  - Stop the sampling.
  - Set another frequency to the sampling (G) Note: this implementation doesn't work fine every time. Please wait for a better implementation.

In the menu of the figure 5.6 the user can open a new frame to delete a property of the Sampling Group.



Frequency (Hz): 1

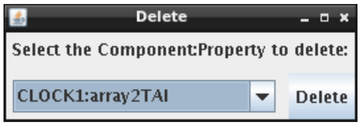
Sampling time (min): 0

Time Window (sec): 5

Status: Sampling Group ready to start.

Figure 15: Edit menu in the Sampling Group

When the user opens the delete property frame 5.6, he can select a Component/Property pair, to remove it from the SamplingGroup.

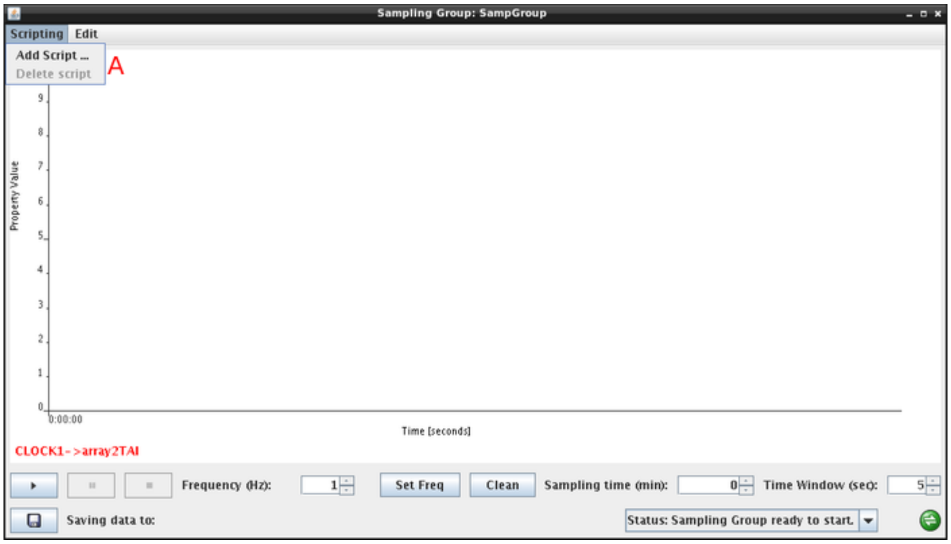


Select the Component:Property to delete:

CLOCK1:array2TAI

Figure 16: Delete Property frame

Another menu of the Sampling Group, the scripting submenu 5.6, is a feature that allows the user to do the **sampling** only in the execution time of the script.



Status: Sampling Group ready to start.

Figure 17: Scripting menu in the Sampling Group

In the Scripting window 5.6 you can select the script location and write some optional parameters, if the script needs them.

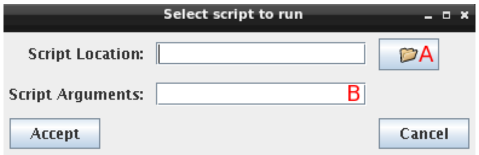
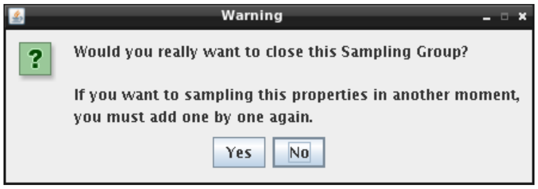


Figure 18: Scripting window

Now the Sampling Group will change the behavior, if you start the sampling the **Sampling Time** will be the execution time of the selected script, the script does not have access to the output data generated by the script.

The idea of this feature is to perform some specific diagnostic routines, for example, to command the mount to different positions with different velocities, while the user monitors some related property values.

When the user tries to close a Sampling Group, he will receive a message 5.6 asking for a confirmation, to avoid stopping a sampling process or erase the actual property list.



Would you really want to close this Sampling Group?

If you want to sampling this properties in another moment, you must add one by one again.

Figure 19: Warning message when try to close a Sampling Group

References

[1] Jchart2d library. `http://jchart2d.sourceforge.net/`.

[2] Swt: The standard widget toolkit. `http://www.eclipse.org/swt/`.