Trap Monitor Application - GUI High Level Design

Gustavo Rahal

March 2005

1 The Trap Monitor Application

The Trap Monitor application will be responsible for displaying traps from all accessible Lustre file systems. The tool includes an interface to display all collected traps in a organized and structured manner. For the purpose of this application, traps are messages collected from the managment host that inform about important changes on the managed cluster.

The complete Trap Monitor Application is composed of two parts. The client (GUI) and the Managment Server. The Managment Server HLD will be discussed in a separate document.

2 The Monitor GUI

The Application interface provide information about traps collected from the managment server. The application polls the managment server for current data, being the polling intervals tunable by the user.

The communication between the Monitor GUI and the Management Server will be through a secured, authenticated XML-RPC service on the managment server. Twisted framework will be used to implement both sides of this communication. Apache will be a proxy server taking care of the authentication and directing requests to the XML-RPC server.

The Monitor GUI will be capable of connecting to many different Management and collect traps from different filesystems. Ideally each Management Server should have failover. The GUI will take care of automaticly connecting to the failover server without user interference.

2.1 Traps

The table below shows the different type of traps.

Trap	**Example Error Level**
MDS/OST Free space	$< 15\%$
OST Free Files	$< 15\%$
OSTs status	INACTIVE
MDS status	INACTIVE

2.2 The GUI

2.2.1 Wizard

The first time the program is executed there will be a wizard instructing the user on how to do the basic configuration.

2.2.2 Main Window

The Main Window will have two areas.

- **Incoming traps**: Incoming traps from the whole cluster are displayed here. Traps are displayed in order of severity and time, more sever and older traps are displayed on top. The sorting order and criteria can be changed.
- **Trap Details**: When a trap is selected on the Incoming Traps list the details appear in this area.

Incoming Traps treeview will have five columns:

- **Error Code**: the error code.
- **Node**: the node that has an error
- **Description**: Description of the error
- **Filesystem**: From which filesystem the trap is comming from
- **Date**: Date and time of the trap
- **Dismiss**: A toggle check box. When finished solving the trap or the trap is ignorable, this box should be marked.

Trap Details treeview will have four columns:

- Files Free
- Files total
- Kbytes Free
- Kbytes Total

2.2.3 Authentication Window

Is the first window to appear when the program is executed. User enters username and password for connection.

2.2.4 Preferences Window

This window will provide configuration parameters. It will be composed of two tabs, Management Servers and Tune. The information will be stored in an XML file.

2.3 Future Ideas

2.3.1 Historical Data Tab

This tab would have historical information about traps that were handled. Charts and summarized statistics could be used for better understanding of the historical data set. Clients requirements are very important to determine the relevance of the historical data and the type of data that should be stored.

3 Installation

During installation, the only cases were the user might have to interact is to choose where the files should be installed(if they want to change the default). The installation process on all platform will not make any questions regarding the use of the application since the initial configuration is done by the application wizard.

3.1 Linux

Rpm packages will be available. The basic dependencies check are for python >= 2.3, pygtk >= 2.4, twisted >= 1.3. The initial targeted distros are RedHat and Suse.

3.2 Windows

NSIS (Null Software Install System) and MSI (Microsoft Software Install) are interesting options. The install package will be self contained, including python, pygtk and twisted.

3.3 MacOS

A dmg package could be used. MacOS installation will also be self contained. DropDMG is a good tool to build dmg packages.

4 Functional Specification

4.1 TrapMonitor Class

4.1.1 connectToMngServers(server1, server2, user, password, callback, errback)

Called: When a connection to the Management Servers needs to be stablished

Description: Connects to the pair of servers being server2 the failover server. The connection is done via an authenticated and secure (SSL) tunnel. The connection is asyncronous so the application won't block while waiting for the connection.

Returns: None

Parameters:

- **server1**: The primary Management Server
- **server2**: The secondary Managment Server
- **user:** username to connect to apache
- **password:** user password
- **callback:** name of a function that will be called when the connection is ready
- **errback:** name of a function that will be called when a connection error occurs

4.1.2 getAppPref()

Called: When application starts

Description: Reads preferences data from a XML file and builds an object with the provided information

Returns: None

4.1.3 pollTraps(connections)

Called: When the connection to the Managements Servers is stablished.

Description: This function is the callback of the connection object. It receives a list with two server connection objects, the primary and secondary server. Then it first tries to connect to the primary management server, if fails tries to connect to the secondary server, if fails again a dialog is displayed informing the user to check the network. After a successful connection to one of the servers, calls incomTraps() to display traps information. The function calls itself every X seconds where X is the polling interval defined in the application preferences window.

Parameters:

- connections: a list with two connection objects, the primary and secondary management servers

Returns: None

4.1.4 buildTrapsTreeview()

Called: As soon as the Main Window opens.

Description: Builds the structure of the treeview that will display the traps.

Return: None

4.1.5 fillIncomTrapsTreeview(traps)

Called: Every time there are new traps available. Called by pollTraps()

Description: Displays a list of traps on the Incoming Traps Treeview

Parameters:

- **traps**: a list of traps

Return: None

4.1.6 fillPrefWindow()

Called: When Preferences Window opens.

Description: Fill the data in Preferences Window available in an XML file.

Return: None

4.1.7 fillTrapDetailsTreeview():

Called: When a trap is selected in the Incoming Traps Treeview.

Description: Displays trap details

Return: None

4.1.8 on_authWindow_okBtn_clicked():

Called: When user clicks the OK button on the Authentication Window

Description: The function will read the list of paired servers (primary, secondary) and connect to them calling connectToMngServers().

Returns: None

4.2 ManagementXMLRPCServer Class

4.2.1 getConnection(primary_server, secondary_server, user, password)

Called: When a connection to the management servers needs to be stablished.

Description: Set internal class variable with username, password, primary and secondary server. Sets a delayed call to connect() method so the connection doesn't block the application

Parameters:

- primary_server: the url of the primary server. If http:// is specified in the url it will be replaced by https:// . If nothing is specified in the port entry 443 will be the default.
- secondary_server: the url of the secondary server. If http:// is specified in the url it will be replaced by https:// . If nothing is specified on the port entry 443 will be the default.
- user: username to connect to apache
- password: the user password

4.2.2 connect()

Called: When a connection needs to be stablished.

Description: Connects to apache via SSL and authenticates.

Return: None

4.3 PollTraps Class

Small class containing two methods: traps() and deffered(). The purpous of this class is to define some callbacks for asyncronous calls to the management server.

4.4 GladeWidgets Class

Reads widgets defined in glade XML file, create widgets objects and attached them to it self.

4.4.1 setControler()

Called: As soon as TrapMonitor object starts.

Description: Connect gtk signals to a given objects.

Parameter:

- controler: object that contains the callbacks that will be used by the gtk widgets

Return: None

4.5 Preferences Class

4.5.1 buildPrefXMLObj()

Called: When Preferences class is instanciated.

Description: Builds a preferences object that holds xml objects pointing to elements in the dom tree

Return: None

4.5.2 buildPrefObj()

Called: When Preferences class is instanciated

Description: Builds a preferences object that holds the preferences values

Return: None

4.5.3 savePrefToFile()

Called: When preferences obj needs to be saved on disc. Usually after Preferences Window closes.

Description: Saves changed information from the Preferences Window to an XML file.

Return: None

4.6 PreferencesObj Class

Class that defines preferences information.

5 Use Cases

5.1 Scenario Number 1 - Wizard

Scenario: Application not running

User Objective: Easily set-up a Trap Monitor Application

Source	**Stop**	**Action**	**Comments**
User	1	./trapmonitor.py	Program executed
Program	2	Displays a welcome window explaining the purpouse of the application and what will be configured next	First Wizard Window
user	3	Clicks NEXT	
Program	4	Displays a window where the user will enter information about the primary and secondary management server, like url and port. The user also chooses the poll interval for traps	Second Wizard Window
User	5	Enter url and port of the management servers. Decides that the default poll interval is reasonable and doesn't change it. Clicks NEXT	
Program	6	Displays a window with Tune parameters like when Errors (traps) should be displayed.	Third Wizard Window
User	7	Doens't change any values. Default are OK. Clicks NEXT	
Program	8	Display an appreciation message. And explains what happens next	Four and last Wizard Window
User	9	Clicks FINISH Button	
Program	10	Application starts	

5.2 Scenario Number 2 - View current traps and exit

Scenario: Application is not running

Source	**Stop**	**Action**	**Comments**
User	1	./trapmonitor.py	Program executed
Program	2	Authentication Window	Preferences settings are loaded
User	3	Enters username and password, clicks OK	
Program	4	Checks username and password	Connects to both servers
Program	5	Authentication Failed. Display a Failure Label and ask user to retry	
User	6	User retype username and password	
Program	7	Authentication succeeds. Authentication Window closes and Main Window appears	
Program	8	Traps are displayed on Incoming Traps treeview	
User	9	User goes to Application Menu and selects quit	
Program	10	Program terminates	

5.3 Scenario Number 3 - Adding Management Servers

Scenario: User has already loged in and traps are being displayed.

Source	**Stop**	**Action**	**Comments**
User	1	Goes to Application Menu and selects Preferences	
Program	2	Display Preferences Window	
User	3	Goes to Management Server Tab	
User	4	Adds a new pair of servers (primary and secondary)	
Program	5	Changes preferences internal dictionary and saves it to an xml file.	
Program	6	Connects to the new pair of servers and start polling traps	
Program	7	Incoming Traps treeview is updated with new traps	

5.4 Scenario Number 4 - Dismiss Traps

Scenario: User has already loged in and traps are being displayed.

Source	Stop	Action	Comments
User	1	On the Incoming Traps display, user clicks on the Dismiss toggle box.	
Program	2	Remove the row representing a trap	
User	3	User can continue dismissing traps	
Program	4	As user clicks on dismiss toggle box rows are removed	

5.5 Scenario Number 5 - Connection Problems

Scenario: User has already loged in and traps are being displayed.

Source	Stop	Action	Comments
Program	1	Connection to one of the servers fail	
Program	2	Connection to the secondary server is attemped	
Program	3	Connection to the secondary server fails	
Program	4	Connection Failed Dialog appears	
User	5	User can Quit, go to Preferences Window or Retry. User clicks RETRY	
Program	6	Trys connecting to servers again	

6 Logic Specification

6.1 TrapMonitor Class

6.1.1 getAppPref()

Description: The application preferences will be store in a XML format file. The function will use python dom module and parse the information mainly calling getElementByTagName method. The dom tree will be converted to a python dictionary to be easily manipulated by other functions

6.1.2 connectToMngServers()

Description: Creates two connection objects (ManagementXMLRPCServer class), one for each management server. This method also sets the call backs of this object. The authentication is done using apache Basic Authentication via a SSL tunnel.

6.1.3 buildTreeview()

Description: Builds a GTKTreeview. Creates the columns using GTKTreeViewColumn and cell renderers using GTKCellRendererText for plain text data and GTKCellRendererToggle to renderer the toggle widgets of Solved/Ignore column

6.1.4 pollTraps()

Description: Try to connect to a primary management server, if it fails connects to the secondary. This function is set as a callback of the connection object created by connectToMngServers(), so when the connection is stablished pollTraps is called. Sends the traps dictionary to fillIncomTrapsTreeview() to display the traps.

6.1.5 fillIncomTrapsTreeview()

Description: Gets the list of traps and displays on the Incoming Traps treeview

6.1.6 fillPrefWindow()

Description: Reads the preferences object and fills the text entries and combo boxes with the data.

6.2 Preferences Class

6.2.1 savePrefToFile()

Description: Reads the preferences object and use DOM replaceWholeText method to change the DOM tree. Then writes the tree to a text file.